

Published: 05/26/67  
(Supersedes: BY.11.02, 10/6/66)

### Identification

Geterr, geterr\_complete - procedures to examine a user's error segment and return error information to the user.  
D. Widrig, K.J. Martin

### Purpose

To provide a convenient method for examining error descriptions which were placed in the user's error segment, the procedures geterr and geterr\_complete are used. It is expected that the user will invoke these procedures whenever he wishes to examine all or part of the last complete error description found in the error segment.

### Usage of geterr

Geterr returns any of seven items of character information. Bit information is not included. The call and geterr's declarations are:

```
call geterr (select_array, return_array, empty_bit);
```

```
dcl select_array (*) fixed bin (17),
```

```
/* an array indicating (by number) which items of  
the following are wanted:
```

<u>number</u>	<u>item</u>
1	time error occurred
2	date error occurred
3	caller of offended procedure
4	location of error
5	error code
6	descriptive information
7	extra character information

```
any numbers other than 1 - 7 are ignored*/
```

```

return_array (*) char (*) varying,

/* an array with the same number of items as select_array;
return_array (n) contains the item specified by
select_array (n) */

empty_bit bit (1);

/* if the error segment is empty (or logically empty -
see below) geterr sets the bit to "1"b */

```

### Implementation of geterr

Geterr calls the procedure geterr\_complete with n = 1 and the skip\_if\_deleted switch set to "1"b (see below for usage and implementation of geterr\_complete) to obtain the items of the most recent error structure. If geterr\_complete returns with empty = "1"b, (i.e., the error segment is empty), geterr sets empty\_bit = "1"b and returns. Otherwise geterr takes each element of select\_array in turn, determines which item of the error structure is wanted and stores it in the corresponding element of return\_array. When all desired items have been placed in return\_array, geterr returns.

Errors which may occur in geterr, and actions taken are:

- 1) An element of select\_array is not one of the digits 1-7. The corresponding element of return\_array is not filled.
- 2) Return\_array has fewer elements than select\_array. Return\_array is filled; geterr calls seterr (BY.11.01) to record the error, then signals the condition, geterr\_err.

### Usage of geterr\_complete

The call and geterr\_complete's declarations are:

```

call geterr_complete (n, skip_if_deleted, time, date,
call_loc, error_loc, error_code, error_info,
extra_bit_info, extra_char_info, attempted_delete,
empty);

```

```

dcl n fixed bin (17), skip_if_deleted bit (1),

```

```

/* use only the n-th (> 0) error description which
fits this description: If skip_if_deleted = "1"b,
it has not been logically deleted (see below).
If skip_if_deleted = "0"b, it may or may not be
logically deleted. */

```

```

dc1 time char (9), date char (6),
    /* time and date that the error was recorded
    by seterr (BY.11.01). */
(call_loc, error_loc) char (38) var,
    /* the caller of the procedure which recorded the
    error and the location of the error in that
    procedure */
(error_code, error_info, extra_char_info) char (*) var,
    /* the error code, descriptive information and extra
    helpful character-string information */
extra_bit_info bit (*) var,
    /* extra helpful bit-string information */
attempted_delete bit (1),
    /* bit switch to indicate if on ("1"b) that the
    error description has been logically deleted (see
    below). It will always be returned off ("0"b)
    whenever skip_if_deleted is set on. */
empty bit (1);
    /* bit switch to indicate if on ("1"b) that the
    error segment contains no n-th error description
    structure which has not been logically deleted
    (see below) */

```

#### Implementation of geterr complete

Upon being called, `geterr_complete` uses the relative pointer `err_ptr→error_out.recent` to access the error-description structure (described in BY.11.01) which was most recently placed in `error_out`. If `err_ptr→error_out.recent` is zero the segment is empty, so `geterr_complete` sets empty to "1"b and returns.

The structure element `eptr→error.attempted_delete` is a switch which if on ("1"b) indicates that an unsuccessful attempt was made to delete this structure. (An unsuccessful delete occurs if the user's `no_error_delete` option is on. A user might turn the option on during debugging runs to avoid losing useful error information.) If the

eptr→error.attempted\_delete switch is on, geterr\_complete considers that the structure is logically deleted and, if the skip\_if\_deleted argument is on, goes on to the next most recent error-description structure. The reason for going to the next most recent structure in this case is that at the time geterr\_complete is called, the user presumably expects the error description in question to have been deleted. If skip\_if\_deleted is off ("0"b), whether or not an error description is logically deleted is of no importance.

Error-description structures are checked in order of recentness (the most recent is checked first) until the n-th is found which fits the desired description. If the segment is exhausted because skip\_if\_deleted is on and no n-th "undeleted" structure is found, it is considered to be empty; geterr\_complete sets empty to "1"b and returns.

Assuming that an appropriate error-description structure is found, geterr\_complete sets empty to "0"b and stores the structure items into return arguments. The arguments call\_loc, error\_loc, error\_code, error\_info, extra\_bit\_info and extra\_char\_info are to be assigned fixed-length data; an assignment statement is sufficient to convert the structure item to varying form for the arguments. The following assignments are made:

<u>structure item</u> (BY.11.01) assigned to	<u>argument</u>
eptr→error.time	time
eptr→error.date	date
eptr→error.call_loc.data	call_loc
eptr→error.error_loc.data	error_loc
eptr→error.error_code.data	error_code
eptr→error.error_info.data	error_info
eptr→error.extra_bit_info.data	extra_bit_info
eptr→error.extra_char_info.data	extra_char_info
eptr→error.attempted_delete	attempted_delete

When appropriate assignments have been made, `geterr_complete` returns. All errors (e.g. errors occurring in the file system) encountered in the course of `geterr_complete` result in a call to `seterr` and signalling the error `geterr_complete_err`. Note, however that an empty or missing error segment is not considered as an error, but simply causes empty to be set to "1"b, followed by a return.