Date: 5 February 1997
To: J3
From: Van Snyder
Subject: Access to status error messages

During J3 meeting 140, it was decided to add access to status error messages as a possible minor technical enhancement to Fortran 02 (II?)

The instructions from WG5 are explicit: Work on minor technical enhancements is not adversely to affect work on the major items.

This report discusses several mechanisms to provide access to status error messages. The intent of this report, and successors, is that members of J3 shall have pondered the several mechanisms, chosen one or more, and developed precise specification and description of the desired mechanism(s) *without consuming any time on the floor of J3 committee or subgroup meetings.*

Please take the time to study this report, and send your comments to Van Snyder, `vsnyder@math.jpl.nasa.gov`.

1 Background

When one strives to write a program that responds to error conditions other than by displaying a message and stopping, one uses `ERR=` and `IOSTAT=` (in I/O statements) and `STAT=` (in `ALLOCATE` and `DEALLOCATE` statements). During processing for an error, one may wish to display a message that describes the error. At present the only recourse is either to write a non-portable program that contains a collection of messages selected by the processor-dependent status values, or to print the status value and admonish the user to consult processor-dependent manuals. Neither alternative is satisfactory. A superior alternative is the ability to request the vendor's run-time library either to return or display a message that depends on the error status value. Several vendors provide such a facility.

2 Alternatives

There are at least four alternative methods to provide this functionality.

1. Provide an intrinsic procedure that returns the message corresponding to the status value.

2. Provide an intrinsic procedure that displays the message corresponding to the status value, perhaps with an option to perform the "default" action that would have been performed if IOSTAT= or STAT= had been absent.

3. Provide an intrinsic procedure that can do either or both of the above.

4. Provide separate intrinsic procedures to do each of the above.

2.1 Advantages and disadvantages for alternative 1

If a program can get the message corresponding to a status value into a variable it can:

- Re-format the message and/or combine it with other text.

- Write the message to "*" or any unit.

The disadvantages of getting the message corresponding to the status value into a variable include:

- If the original error occurred on "standard output," attempting to display the error message may simply cause another error.

- Many systems provide for a "standard error" output stream. Fortran has no such concept, and therefore provides no mechanism to display a message at that destination.

- One needs to know the number of characters per line, and the number of lines in the message.

2.2 Summary of possibilities for alternative 1

Some combinations are not meaningful.

1. (a) Standard specifies maximum number of lines (maybe only one).
   (b) Separate intrinsic returns number of lines in specific message, or maximum number of lines in any message, if the message number argument is absent.

2. (a) Standard specifies maximum line length.

   (b) Separate intrinsic returns length of specific line in specific message, or maximum length of any line in the message if the line number argument is absent, or the maximum length of any line in any message if both arguments are absent.

3. Retrieving the message: Starting with line L (absent L means line 1), return K lines (absent K or $K \leq 0$ means all the lines from L to the end of the message) and, optionally, return the number of lines returned (zero means "no line L in this message").

4. Storage layout

   (a) `CHARACTER (LEN=N) C[(L)]`, with N = maximum line length, L = maximum or specific number of lines. N can't be a parameter for ALLOCATE.

   (b) `CHARACTER (LEN=1) C(N[,L])`, N = maximum or specific line length, L = maximum or specific number of lines. N can be a parameter for ALLOCATE, but printing is harder.

   (c) `CHARACTER (LEN=M) C` and `INTEGER E(0:L)` where M is the total number of characters in the message, L is the number of lines in the message, and E(I) specifies the position in C of the end of the I'th line – E(0) == 0. Requires an intrinsic for maximum and/or specific M, instead of line length.

   (d) `CHARACTER (LEN=1) C(M)`, otherwise as immediately above.

5. (a) User declares or allocates (and deallocates) storage.

   (b) Intrinsic allocates storage. User de-allocates it.

2.3 My choices

1. Alternative 2. Simplifies internationalization, too.

2. Alternative 4, with 1(nothing), 2(nothing), 4d, 5b, and a simpler 3 that returns all of the message, and no argument to return the number of lines (`UBOUND(E)` gives that).

3. Alternative 4, with 1b, 2(nothing), 4d, 5a.

4. Alternative 4, with 1b, 2b, 4b, 5a.

If one could declare `CHARACTER (LEN=:), ALLOCATABLE :: C` and specify the LENGTH attribute during allocation, I would prefer 4a to 4b, and 4c to 4d.