Subject: More comments on Section 13
From: Van Snyder

1 Edits

Edits refer to 01-007. Page and line numbers are displayed in the margin. Absent other instructions, a page and line number or line number range implies all of the indicated text is to be replaced by immediately following text, while a page and line number followed by + (-) indicates that immediately following text is to be inserted after (before) the indicated line. Remarks are noted in the margin, or appear between [and] in the text.

[There is more gratuitous repetition in section 13 than in other sections. Whenever something is duplicated, it's an opportunity to get it wrong (twice). Editor: Delete all subclauses from 13.4 through the end of 13.13, except 13.5.7, 13.7.1, 13.8.0, 13.8.1, 13.8.2 and 13.13.0. Lift 13.5.7 to be 13.5.0, and 13.7.1 to be 13.7.0. The subclauses proposed to be deleted contribute nothing, either in content or organization, that is not done better elsewhere. 13.4-13.13

There is some value in 13.14 and 13.15 summarizing by category, but it would not be a tragedy to delete them, too (but either keep them both or delete them both).] 13.14-13.15

[Editor: The right-hand column wraps back to the left margin of the left-hand column. Presumably, it was intended to wrap back to the left-hand margin of the right-hand column.] 282:19-27

J3 internal note 288:24+

Unresolved issue xxx
Between Fortran 90 and Fortran 95, the "Result Type and Type Parameters" clause was replaced by "Result Characteristics." In itself, there's no problem with this, but it sometimes says "Same as <one of the arguments>." The introduction to the section is nebulous about whether the descriptions of the arguments apply to actual arguments or dummy arguments. If the specifications apply to the dummy arguments, it is not entirely appropriate to say that the characteristics of the result are the same as the characteristics of an argument – for example, INTENT, VOLATILE and optionality are characteristics. If the specifications apply to the actual arguments, it is not entirely appropriate to say that the characteristics of the result are the same as the characteristics of an argument – for example, POINTER, ALLOCATABLE and ASYNCHRONOUS are characteristics. The introduction should specify whether the descriptions apply to actual or dummy arguments. The "Results Characteristics" paragraphs should say "The type and type parameters are the same as those of <one of the arguments>; the other characteristics are ..." instead of just "Same as <one of the arguments>."

[This description doesn't work after we changed subclause 7.5.2 to allow higher-rank views of rank-one objects. It is also more complicated than necessary. Editor: Replace "shall ... POINTER." by "shall be allowable as the target in a pointer assignment statement (7.5.2) in which POINTER is the pointer object."] 292:11-16

[This is not correct for unlimited polymorphic objects, and ignores allocatable objects.] 303:7-8
The dynamic type of an unlimited polymorphic object that is an unallocated allocatable object or a disassociated pointer is undefined. Otherwise, if either A or MOLD is an unallocated allocatable object or a disassociated pointer, the dynamic type is the same as the declared type.

[The summary at [284:39] says "processor number" but here it says "machine representable 322:23

number". These two terms are probably intended to be the same, and neither one is used anywhere else. Editor: "machine representable number" ⇒ "processor number".]

NOTE 13.16$\frac{1}{2}$ 322:26+

NEAREST is the only intrinsic function that deals explicitly with processor numbers. On some systems there are processor numbers, usually less than TINY(X), that are not model numbers. Therefore it is not necessarily true, for example, that NEAREST(X,1.0) == X + SPACING(X).

[Broken concerning type parameters, because deferred type parameters of disassociated pointers are not specified to be undefined. Don't bother with this if section 2 passes,] 323:15-16

Result Characteristics. The result is a disassociated pointer or an unallocated allocatable. The array bounds of the result are undefined. If MOLD is not present the result characteristics, including whether the result is a pointer, procedure pointer or unallocated allocatable, are determined from context (7.1.4.1). Otherwise if MOLD is a pointer that is associated with a target, or is a procedure pointer that is associated with a procedure, the result is a pointer and its characteristics are the same as MOLD. Otherwise the result characteristics other than deferred type parameters are the same as MOLD, and deferred type parameters are undefined.

[No other function result kind is described by writing "the kind type parameter is the processor-dependent kind type parameter...." This is repetitive and redundant, and it's not necessary to say it twice, or even more than once. Editor: "the processor-dependent kind type parameter for the" ⇒ "that of".] 327:14-15

[This is not correct for unlimited polymorphic objects, and ignores allocatable objects.] 329:5

The dynamic type of an unlimited polymorphic object that is an unallocated allocatable object or a disassociated pointer is undefined. Otherwise, the dynamic type of an unallocated allocatable object or a disassociated pointer is the same as the declared type.

2 NULL is weird already...

[Is it really necessary to require MOLD to be a pointer? NULL is weird already; why not make it easier to use, e.g. why not allow "NULL((/ CHARACTER(LEN=5) :: /))"?] 323:12-14

Argument. MOLD may be of any type or may be a procedure or procedure pointer. If it is a pointer its association status may be undefined, disassociated or associated; if it is associated, the target need not be defined with a value. If it is allocatable it need not be allocated; if it is allocated it need not be defined with a value.

[Broken concerning type parameters, because deferred type parameters of disassociated pointers or of unallocated allocatables are not specified to be undefined.] 323:15-16

Result Characteristics. The result is a disassociated pointer or an unallocated allocatable. The array bounds of the result are undefined. If MOLD is not present the result characteristics, including whether the result is a pointer, procedure pointer or allocatable, are determined from context (7.1.4.1). Otherwise if MOLD is not a pointer or an allocatable, is a pointer that is associated with a target, or is a procedure pointer that is associated with a procedure, the result is a pointer and its characteristics are the same as MOLD. Otherwise if MOLD is an allocated allocatable the result is an unallocated allocatable and its characteristics are the same as MOLD. Otherwise the result characteristics other than deferred type parameters are the same as MOLD, and deferred type parameters are undefined.

3 COMMAND_ARGUMENT_COUNT

According to [275:29-30], "all standard intrinsic functions are pure." A careful reading of the definition of purity reveals that it means side effects are not caused. One might also expect, although unreasonably, that pure functions do not depend on side effects.

The COMMAND_ARGUMENT_COUNT function has no arguments, but does not return the same result value every time the program is executed. Therefore, a processor cannot, at compile time, replace the reference by its result. Although the function is technically pure, it doesn't smell pure. Replace it with an argument to GET_COMMAND_ARGUMENT.

[Editor: Delete subclause 13.17.21.] 295:18-26

[Editor: After "STATUS" insert ", COUNT" 303:28

COUNT (optional) shall be scalar and of type default integer. It is an INTENT(OUT) argument. It is assigned a value equal to the number of command arguments available. If there are no command arguments available or if the processor does not support command arguments, then the value is 0. If the processor has a concept of a command name, the command name does not count as one of the command arguments. 304:4+