Date: 18 July 1998
To: J3
From: Van Snyder
Subject: Typos, miscellaneous remarks on 98-007r2

Page and line numbers to which remarks refer are in the margin.

"4.4" → "4.2" [15:13]

Would be clearer if written: "A structure constructor (4.5.6) may be used to construct a constant value of derived type from an appropriate sequence of constant expressions (7.1.6.2)." As it is written it at first appears to specify "a structure constructor from an appropriate sequence of constant expressions." [29:35-36]

It seems unnecessary to prohibit a nonkind type parameter to be used for initialization. It should be allowed so long as the associated "actual parameter" has the appropriate restrictions. E.g. it shall be an initialization expression in an object declaration, but it could be anything in an allocation or an intent(out) dummy argument if it's value is deferred (see 98-172) in the object declaration. [30:29]

The word "itself" doesn't seem to contribute anything. [40:39]

Are we concerned about split infinitives? [40:45]

"parent type" is not the correct term. Need a term for "the type being defined" or "the type of which the component is a member." [41:31]

"precesed" → "preceded" [41:33]

Need a section **4.5.1.5 Allocatable components**. [44:22+]

Replace by "An extended type has a subobject name that is the same name and has the same type as its parent type. This is not an additional component; it denotes a subobject that has the parent type and that consists of all of the components and subobjects inherited from the parent type. If the parent type is an extended type, the subobject having the parent type name has a subobject having the same name as its parent type." There's related "repair" in 98-179. [48:1-12]

The paragraph should be a note because it's a consequence of the "no duplicate specification" rule. No fixup for grandparents is necessary because there is no subobject having the same name as the grandparent type name – although there is one within the subobject having the same name as the parent type. [48:16-19]

"summy" → "dummy" [54:4]

"sucessful" → "successful" [58:34]

Add "component definition" to the list. [62:46]

Add *procedure-declaration-stmt* and PROCEDURE to the index. [66:14]

Indent [67:26]

Remove the constraint. [69:38]

Replace "components" by "components and subobjects". [84:35]

Add a new paragraph: [86:7+]

For an object of extensible type, the subobject that has the same name as the parent type is accessed as though it were a component.

I couldn't find a syntax rule that defines *designator*. Should this be *data-ref*?	[86:11]
Is there a special reason that "expressions whose primaries are constant expressions" is not simply "constant expressions"?	[106:40]
Is there a special reason that this is not simply "deferred"?	[107:6]
Is there a special reason that "expressions whose primaries are initialization expressions" is not simply "initialization expressions"?	[107:21]
Is there a special reason that this is not simply "deferred"?	[108:4]
This could probably be simplified to: Constraint: The declared type of *target* shall be an extension type of the declared type of *pointer-object*.	[123:16-17]
Is this OK? It would be nice to allow it. It would be nice to allow elemental procedures to be actual arguments so long as the interface is explicit – in that case, the processor can tell whether the dummy procedure is or is not declared to be elemental.	[123:36-37]
"compatable" → "compatible"	[124:25]
What is the relation between character count and user-defined derived-type I/O?	[160:3-10]
WG5 didn't like `w, d, m` arguments of DTIO procedures, and limitations of the `DT` format descriptor. Alternatives discussed included: 1. Allow any number, including zero, of numbers after the letters after `DT`. Replace the `w, d, m` arguments with a single non-optional rank-one argument of assumed shape in which these values are deposited. 2. Allow optional arbitrary text, enclosed between (and), e.g. `DT(foo5bar7.3)`, or as a character string, e.g. `DT"foo5bar7.3"`. Delete the `w, d, m` arguments. 3. Arbitrary text, like item 2, with an arbitrary number of numbers and an assumed shape argument, like item 1.	[167:39ff, 193:34ff]
Would be more precise if "list item list" were replaced by "input or output item list". I don't think a "list item" is formally defined – the term is used elsewhere, with the meaning "an item in the list". "list item list" is therefore "the list of items in the list" which has no meaning.	[168:19]
"must" → "shall"	[168:27]
Delete "," before `(LEN=*)` in seven places	[168, 169]
Belongs in section 10.5.5	[170:38-40]
"and" → "or"	[171:15]
Implies nonadvancing I/O for unformatted files. See 98-174. Nonadvancing I/O would be simpler than described therein if "SIZE =" is prohibited for unformatted nonadvancing I/O.	[171:15-24]
Is it necessary that the value of the variable specified in the ID= specifier shall have been set by a data transfer operation initiated on the same unit as specified in the wait statement?	173:21-24]
"elemnt" → "element"	[208:21]
"theh" → "the"	[208:29]
Doesn't cover functions having result of derived type with nonkind parameters. Should it? If so, does the following work? (e) A result that has a nonkind type parameter value that is not constant and not assumed (character or derived type functions only).	[217:25-26]

"which" → "that".	[218:35]
R311 and R1207 use *defined-operator* but 12.3.2.1.1 is entitled **Defined operations**. Should these be consistent?	[220:23]
"(12.4)" → "(7.1.7.7, 7.3, 12.4)"	[220:25]
"(7.5.1.3)' → "(7.5.1.3, 12.4)"	[221:29]
"disambiguate" → "resolve" (here and elsewhere)?	[221:24]
There is no discussion of the relation between derived-type I/O procedures and extensible or polymorphic types. DTIO procedures evidently cannot be dispatched, i.e. selected at run-time depending on the dynamic type of a polymorphic object.	[222:11-19]
Second "procedure" → "procedure or procedure pointer".	[224:4]
"those" → "values of corresponding type parameters"	[226:1]
Inadequate concerning actual arguments that are dummy procedures associated to dummy procedures. Add the following sentence at 24+ and 27+: In a sequence of associations of dummy procedure actual arguments to dummy procedure dummy arguments, the ultimate actual argument that is not a dummy procedure shall be a function (24+) subroutine (27+).	[228:23-24, 26-27]
"MOLD" → "B"	[252:37]
14.1.2.5 Components, type parameters, and subobject names	[308:33]
[Editor: Replace "component" by "component or subobject".]	[308:34-38]
Yes. "the same scope as the type of which it is a component" lifts it out of the scope of the type definition.	[308:41-42]
Is this true?	[311:7]
Remove "and".	[314:34]
In 14.7.5 and 14.7.6, is it necessary to discuss the part of an actual argument of extensible type that is argument associated to a dummy argument having an ancestor type?	[317-318]
class (5.1.1.8) A class named *N* is the set of types extended from the type named *N*.	[342:28+]
Inadequate in the context of polymorphism. Replace by (1) The *components* of that *type*, including components inherited from ancestor types in the case of extensible types, and	[343:41]
Should "an intrinsic" be "a"?	[345:38]
Should "an intrinsic" be "a"?	[349:14,16]