Subject: Comments on Section 6
From: Van Snyder

1 Edits

Edits refer to 02-007. 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 associated text, while a page and line number followed by + (-) indicates that associated text is to be inserted after (before) the indicated line. Remarks are noted in the margin, or appear between [and] in the text.

[Editor: Delete "when ... and"; "certain events" ⇒ "events described in 16.5.5"; Delete "(16.5)".] 97:5

[Editor: Move Note 6.1 to here – it has nothing to do with constants.] 97:24+

[Same ¶– Editor: Insert "A constant (3.2.2) is a literal constant or a named constant." before "A".] 97:25

["associated" is the wrong term. Editor: Delete "been associated with"; second "that" ⇒ "; the name".] 97:26

[Editor: The comment "`See`" on third line of Note 6.1 is a dangling xref to Note 4.20.] 97:28+4

[There is no specification of the type and type parameters of a *data-ref*. Editor: "structure component" ⇒ "*data-ref*"; then move this line to [99:13], making it a separate paragraph. It then still covers the case it currently covers.] 99:21

[Editor: "allocatable ... allocated" ⇒ "unallocated allocatable variable".] 100:9-10

[Editor: Move paragraph to [102:23+].] 103:3

[Editor: "If" ⇒ "When" for consistency with [102:24].] 103:4

[Editor: Delete the first "section".] 103:9

[Editor: "creation" ⇒ "allocation" because every other event in this sentence is described using "Fortran" terms.] 104:2

[Editor: "For detailed" ⇒ "Detailed"; "see" ⇒ "are in".] 104:8+2

[A *type-spec* is prohibited by C630 if SOURCE= appears, but something is needed to get values for deferred type parameters. Editor: "*type-spec*" ⇒ "either *type-spec* or SOURCE=".] 104:26

C624 (R623) If any *allocate-object* is unlimited polymorphic, either *type-spec* or SOURCE= shall appear. 104:30

[Editor: Move "An allocatable ... **allocated**" to [106:26], replacing "Currently allocated".] 106:11-12

[Editor: Delete "and is definable." because it's said at [106:26].] 106:12

[Editor: Delete "Allocating ... statement." because it's said at [106:27].] 106:12-13

[Editor: Delete "At the ... allocated" because something equivalent will be inserted soon.] 106:13-14

[Editor: Delete "and are not definable." because it's said at [106:22-23].] 106:14-15

[Editor: Delete "The intrinsic ... allocated." because it's said by a conspiracy of [106:24-25] and [106:27-28].] 106:15-16

[Editor: Move paragraph to [107:2+].] 106:17-18

[Editor: Delete – did you notice that the previous paragraph completely evaporated?] 106:19

[Editor: "Not currently allocated" ⇒ "An allocatable variable that is not currently allocated is **unallocated**".] 106:22

[As promised above for [106:13-14] – Editor: Insert "At the beginning of execution of a program, allocatable variables are unallocated." before "Deallocating".] 106:23

[Editor: Insert "(13.7.9)" after "ALLOCATED" twice.] 106:24,28

[Editor: Exchange the two list items.] 106:22-28

[Editor: Exchange "Following . . . define the target" and "Allocation . . . attribute".] 107:4-6

[Editor: Delete "it ... by" and "that".] 107:12

[Editor: Insert "that does not have undefined association status" after "pointer".] 107:13

[Run-on sentence – Editor: "and" ⇒ ";".] 108:19

[Abused semicolon; Editor: "nor shall it" ⇒ "it also shall not".] 108:27

[Abused semicolon; Editor: "nor shall they" ⇒ "they also shall not".] 108:30

[Editor: "is present" ⇒ "appears".] 108:32

[Editor: Delete "An . . . pointer." because it's said in 6.3.3.2 (where it belongs).] 109:8-9

[Editor: Insert "without a *dealloc-opt*" after "statement".] 110:4