Subject: Protected types
From: Van Snyder
Reference: 03-258r1, section 2.2.7

Number

TBD

Title

Protected types.

Submitted By

J3

Status

For consideration.

Basic Functionality

Provide an attribute for types that has the effect of applying the PROTECTED attribute to every object of its type.

Rationale

Fortran 2003 has protected variables, but if they have the target attribute, their values aren't protected. A PROTECTED attribute should be provided for types, to indicate that objects of such types cannot be allocated or deallocated, or their values or pointer association status changed, except by procedures within the module where they are declared or the type is defined. This would allow one to have a local pointer step through a linked list of protected objects in a different module without being able to change them.

Protected types could also be used to allow a pure procedure to associate a pointer with a global variable, with the proviso that a pointer of protected type cannot be used in a variable-definition context within a pure procedure.

Estimated Impact

Trivial to minor.

Detailed Specification

Add a PROTECTED attribute for types. Specify that

- Nonpointer nonallocatable variables of the type can be changed only within the module where the type is defined or one of its submodules, or within the scoping unit where the variable is declared or one that accesses that scoping unit by host association. Dummy argument declarations don't count.
- A target of a pointer of the type can allocated, deallocated, or its value changed by way of the pointer, only within the module where the type is defined or one of its submodules. The association status can be changed within the module where the type is defined or one of its submodules, or within the scoping unit where the pointer is declared or one that accesses that scoping unit by host association.

- The association status of a pointer of the type can be changed within a pure procedure if it's declared there or if the procedure is defined in the module where the type is defined or one of its submodules, but it cannot be allocated or deallocated, nor can it appear in a variable-definition context, within a pure procedure.
- An allocatable variable of the type can be allocated or deallocated, or its value changed, only within the module where the type is defined or one of its submodules, or within the scoping unit where the variable is declared or one that accesses that scoping unit by host association. Dummy argument declarations don't count.

History