Subject: Another reason for a READONLY attribute
From: Van Snyder

There has been sporadic discussion of the utility of a READONLY attribute for a public variable of a module.

Malcolm has pointed out that at present, if a nonallocatable nonpointer INTENT(IN) dummy argument has the TARGET attribute, one can take a pointer to it, and then use the pointer in a variable definition context, or as an actual argument in association with a dummy argument that does not have the INTENT(IN) attribute (which isn't necessarily a variable definition context). This allows the INTENT(IN) dummy argument to be changed. This situation is not prohibited by the standard, and cannot be detected without global dataflow analysis. A READONLY attribute could prevent this.

It is necessary to specify whether the READONLY attribute of a module variable applies only if a variable with that attribute is accessed by use association, or within the module as well. Suppose we spell the former READONLY(USE), and the latter simply READONLY.

It may also seem desirable to specify whether the READONLY(USE) attribute applies to the pointer association status, allocation status, or target value. Providing this distinction may be of marginal utility: How often would a module like to allow a variable accessed from it by use association to have its value changed, but not its pointer association or allocation status? My inclination for READONLY(USE) is that it applies to every aspect of a variable that has that attribute. The utility of READONLY(USE) has been discussed already, and will not be further discussed here.

A READONLY attribute for a nonmodule variable would only make sense for pointers because otherwise there would be no way for the target to get a value. For pointers, the target gets a value when the pointer becomes associated with a target that has a value.

The relevant constraints become:

- If the *target* in a pointer assignment statement has the READONLY attribute or is a nonallocatable nonpointer dummy argument with the INTENT(IN) attribute the *pointer-object* shall have the READONLY attribute.
- A variable with the READONLY attribute shall not appear in a variable definition context (14.7.7). More strongly, we could add
- If a variable with the READONLY attribute is used as an actual argument, the associated dummy argument shall have the INTENT(IN) attribute and shall not have the POINTER attribute.

The reason for the "nonallocatable nonpointer" part is that INTENT(IN) for an allocatable or pointer dummy argument applies to the allocation status or pointer association status of the dummy argument, not to the target.

A READONLY attribute for a nonmodule pointer variable would plug a hole in the chain of protection of INTENT(IN) dummy arguments. It is incompatible with Fortran 95, but perhaps worth considering anyway. Any program that takes a pointer to an INTENT(IN) dummy argument and then uses that pointer as an actual argument in association with a dummy argument that does not have the INTENT(IN) attribute is probably in error. There are, of course, other legitimate reasons to take a pointer to a dummy argument that has the INTENT(IN) attribute, but the number of program units in which it occurs is almost certainly small.