Subject: Extensions to C interoperability for optional and assumed-shape dummy arguments
From: Van Snyder
Reference: 03-258r1, section 2.16.1

Number

TBD

Title

Extensions to C interoperability to support optional arguments, assumed-shape arguments, and "fat" pointers.

Submitted By

J3

Status

For consideration.

Basic Functionality

Standardize the interfaces for C functions to create and interpret descriptors for assumed-shape dummy arguments, optional arguments, and pointers that include the same extent and stride information as Fortran pointers. Standardize the function names and names of any types necessary for those functions.

None of these facilities necessarily apply to noninteroperable procedures. Processors can use different descriptors for optional and assumed-shape arguments for interoperable and noninteroperable procedures. Fortran pointers and assumed-shape arguments are intentionally alike. It is reasonable to require that an interoperable descriptor for Fortran-like pointers and an interoperable descriptor for assumed-shape dummy arguments be the same.

Rationale

The usability of Fortran procedures by C functions is reduced because C functions cannot create descriptors for actual arguments to be associated with optional dummy arguments or assumed-shape dummy arguments, or descriptors having the same sort of extent and stride information as Fortran pointers.

If an assumed-shape dummy argument of a Fortran procedure is used as the actual argument of a C function, and the array is not contiguous, a copy will be required because there is no way specified by the standard for a C function to interpret a descriptor for an assumed-shape argument.

If n optional dummy arguments of a Fortran procedure are to be used as actual arguments for a C function, 2^n versions of that C function and a test with 2^n branches will be needed to use the functions.

If a Fortran pointer with rank greater than one needs to be an actual argument for a C function, it needs to be passed through a Fortran interface in which it is associated with an assumed-size or explicit-shape dummy argument, thereby causing a copy if the array is not contiguous.

Estimated Impact

A few subclauses in Section 15. Very little interaction with other portions of the standard, except perhaps to cross off some constraints.

Detailed Specification

A C function can use functions, structs and typedefs described in this subclause to examine or create descriptors for optional or assumed-shape dummy arguments for interoperable procedures. Processors shall provide the header files described here, containing at least the functions, structs and typedefs described here. There is no implication that the procedures described in this section can be used to examine or create descriptors for optional or assumed-shape arguments for noninteroperable procedures. Processors may or may not use the same form of descriptors for interoperable and noninteroperable procedures. Procedures are provided to convert between Fortran pointers and objects of a type that interoperates with the descriptors for assumed-shape arrays described here.

Drafts of subclauses describing the C functions, structs and typedefs follow. An attempt has been made to mimic the format of the 1999 C standard. They will almost certainly need to be polished; substantial revision of the approach may be necessary or desirable. This is merely an illustration of concept.

Optional arguments

The header <`f_optional.h`> declares two functions and a type.

The type is

```
struct f_optional
```

which may contain any members the processor finds necessary.

The `f_setoptional` function

Synopsis

```
#include <f_optional.h>
struct f_optional *f_setoptional(void *arg, _Bool present);
```

Description

The `f_setoptional` function creates a descriptor that may be associated as an actual argument with an optional dummy argument of an interoperable Fortran procedure. It represents a present argument if `present` is `true` and an absent argument otherwise. The `arg` argument shall not be `NULL` if `present` is `true`.

The `f_getoptional` function

Synopsis

```
#include <f_optional.h>
_Bool f_getoptional(struct f_optional *arg);
```

Description

The result of the `f_getoptional` function is `true` if `arg` represents a present optional argument, and `false` if `arg` represents an absent optional argument.

Assumed-shape arguments

The header <`f_assumed_shape.h`> declares three functions, a typedef, and two types.

The typedef is

```
f_shape_t
```

which denotes a standard integer type.

The types are

```
struct f_assumed_shape
```

which may contain any members the processor finds necessary. A pointer to an `f_assumed_shape` struct may be used as an actual argument associated with an assumed-shape dummy argument of an interoperable Fortran procedure, or as the formal parameter of a C function associated with an array actual argument declared in an interoperable Fortran interface.

```
  struct f_extent_stride
```

which contains members that represent the extent and stride of one dimension of an assumed-shape array. The structure shall contain at least the members

```
f_shape_t extent;  // The extent of the dimension
f_shape_t stride;  /* The stride between consecutive elements in the
                      dimension, in units of the array element */
```

The `f_set_extent_stride` function

Synopsis

```
#include <f_assumed_shape.h>
struct f_extent_stride *f_set_extent_stride(f_shape_t extent, f_shape_t stride);
```

Description

The `f_set_extent_stride` function combines the extent and stride for one dimension of an assumed-shape array. The result may be used as an actual parameter for the `f_create_assumed_desc` function.

The `f_create_assumed_desc` function

Synopsis

```
#include <f_assumed_shape.h>
struct f_assumed_shape *f_create_assumed_desc(void *array,
                                              f_extent_stride dim, ... );
```

Description

The `f_create_assumed_desc` function creates a descriptor for an assumed-shape array. The result may be used as an actual argument associated with an assumed-shape dummy argument of an interoperable Fortran procedure.

The `f_get_array_ptr` function

Synopsis

```
#include <f_assumed_shape.h>
void *f_get_array_ptr(f_assumed_shape *desc);
```

Description

The `f_get_array_ptr` function returns the address of the first element of an array described by the `desc` parameter.

The `f_get_extent` function

Synopsis

```
#include <f_assumed_shape.h>
void *f_get_extent(f_assumed_shape *desc, f_shape_t dim);
```

Description

The `f_get_extent` function returns the extent of the dimension given by the `dim` parameter of an array described by the `desc` parameter.

The `f_get_stride` function

Synopsis

```
#include <f_assumed_shape.h>
void *f_get_stride(f_assumed_shape *desc, f_shape_t dim);
```

Description

The `f_get_stride` function returns the distance between consecutive elements in the dimension given by the `dim` parameter of an array described by the `desc` parameter. The units of the result are the size of an array element.

Interoperable type having functionality of a Fortran pointer

Define a derived type in `ISO_C_BINDING`, with private components, that interoperates with the C struct `f_assumed_shape`.

Define procedures that convert Fortran pointers to objects of that type, and vice versa.

History