Further interoperability features in Fortran 2015

John Reid, ISO Fortran Convener,

JKR Associates and

Rutherford Appleton Laboratory

BCS Fortran Specialist Group

London, 1 October 2015

Introduction

Fortran 2003 contains features for interoperability of Fortran with C, and they are widely implemented as extensions of Fortran 95 compilers.

This provides for interoperability of procedures with non-optional arguments that are scalars, explicit-shape arrays, or assumed-size arrays, but not with arguments that are assumed-shape, allocatable, pointer, or optional.

To fill this gap, WG5 constructed a Technical Specification (ISO/IEC TS 29113:2012). This talk will describe the additional features. They will all be included in Fortran 2015.

C header file

The C header file `ISO_Fortran_binding.h` provides the C programmer with

standardized C structs,

macro definitions, and

C prototypes for C functions

to allow access in C to additional Fortran features.

Type for bounds and strides

The struct type `CFI_dim_t` has components:

`lower_bound`

Lower bound of an array in a given dimension.

`extent`

Extent of an array in a given dimension.

`sm`

Stride multiplier (distance in bytes between successive elements) in a given dimension.

C descriptor

A C descriptor for an object is a struct of type `CFI_cdesc_t` with components:

`base_addr`	C address of the first element of the object. `NULL` if unallocated or not associated.
`elem_len`	The `sizeof()` an element of the object.
`rank`	Rank of the object.
`type`	Code (see next slide) for the type of the object.
`attribute`	Code (see next-but-one slide) to indicate whether the object is allocatable, a pointer, assumed-shape, or otherwise.
`dim[]`	Lower bounds, extents, and stride multipliers.

Macros for type codes

Type codes:

- `CFI_type_struct` : interoperable struct
- `CFI_type_signed_char` : signed char
- `CFI_type_short` : short
- `CFI_type_int` : int
- `CFI_type_float` : float
- `CFI_type_double` : double
- `CFI_type_cptr` : void *
- `CFI_type_cfunptr` : pointer to a function

... Lots more types.

Other macros

`CFI_MAX_RANK` : Largest rank supported.

Attribute codes:

`CFI_attribute_assumed`	assumed-shape
`CFI_attribute_allocatable`	allocatable
`CFI_attribute_pointer`	pointer
`CFI_attribute_unknown_size`	assumed size

The new calling mechanism

A dummy argument in a Fortran interface that is allocatable, assumed-shape, or a pointer may correspond to a formal parameter in a C prototype that is a pointer to a C descriptor.

When calling the C function from Fortran, a suitable C descriptor is provided by the system.

Assumed-rank object

A dummy argument in an interface may be of assumed rank. E.g.

```
interface
   subroutine scale(a)
      real a(..)
   end subroutine scale
end interface
```

It may correspond to a pointer to a C descriptor in a C function prototype.

Allows a C function to accept an allocatable, assumed-shape, or a pointer array of any rank.

Severely restricted in Fortran. Can be passed around, remaining as assumed rank, or passed as the first argument to an inquiry function.

New intrinsic function: `rank(a)`

Assumed-type objects

A dummy argument may be of assumed type. E.g.

```
interface
   subroutine archive(a)
      type(*) :: a
   end subroutine archive
end interface
```

Allows a C function to accept an allocatable, assumed-shape, or a pointer array of any type.

If it is not allocatable, assumed-shape, assumed-rank, or a pointer, it may correspond to a pointer to void in a C function prototype.

Allows a C function to accept a Fortran object of any type. Helpful for calling MPI.

Severely restricted in Fortran. Can be passed around, remaining as assumed type, or passed as the first argument to some inquiry functions.

Constructing C descriptors in C

A C descriptor must not be initialized, updated, or copied other than by calling one of these functions.

- `CFI_allocate` does a Fortran allocation
- `CFI_deallocate` does a Fortran deallocation
- `CFI_establish` establishes a C descriptor
- `CFI_section` updates a C descriptor to describe an ordinary array section
- `CFI_select_part` updates a C descriptor to describe an array section such as `array%part`
- `CFI_setpointer` updates a C descriptor to point to the whole of an object or be disassociated.

No mixing of C and Fortran allocation mechanisms is allowed.

Other functions

`CFI_address` computes the C address of an object described by a C descriptor

`CFI_is_contiguous` tests the contiguity of an array described by a C descriptor

Optional arguments

An absent actual argument in a reference is indicated by a formal parameter with the value NULL.