Other features

Coarray components

A derived type may have allocatable coarray components:

```
type glob
   real, allocatable :: a(:)[:], b(:)[:]
end type glob
type (glob),save :: dat
allocate (dat%a(n)[*], dat%b(n)[*])
```

An object with a coarray component must not be an array or a coarray.

Coarray subobjects

A subobject of a coarray is a coarray if it has no cosubscripts, vector subscripts, or allocatable component selection.

Has the corank and cobounds of the whole coarray.

```
real,save :: a(10)[*] ! A coarray
type comp
  integer size
  real, allocatable :: comp(:)
end type
type(comp),save :: var[*] ! A coarray
var%size ! A coarray
a(1:10:2) ! A coarray
```

Coarray subobjects that are not coarrays

```
integer :: index(5)=[7,5,3,1,4]
real,save :: a(10)[*] ! A coarray
type comp
  integer size
  real, allocatable :: comp(:)
end type
type(comp),save :: var[*] ! A coarray
a(:)[11] ! Cosubscript
a(index) ! Vector subscript
var%comp ! Allocatable component
```

Actual argument corresponding to coarray dummy argument

- Must be a coarray
- If the dummy is of explicit shape, e.g.

```
real :: a(m)[*]
```

the actual must be simply contiguous (can be seen at compile time always to be contiguous).

Coindexed actual argument

The dummy must not be a coarray, of course.

Example:

```
real :: a(m)[*], b(m)
b(:) = sin(a(:)[i])
```

Almost certainly, a local copy is made and passed to the procedure.

Assignment statement corrected – for clarity, whole array notation not allowed with a cosubscript.

Functions with coarray results

Functions are not allowed to have coarray results.

This is because they would create temporary coarrays and synchronization would be needed within a statement.

Number of subscripts and cosubscripts

Total number of subscripts and cosubscripts is limited to 15.

Pointer components

```
integer, target :: i
type comp
  integer, pointer :: p
end type comp
type (comp),save :: a[*]
a%p => i  ! OK
a[7] = a ! a[7]%p becomes undefined
```

More on locks

```
use :: iso_fortran_env
logical :: success
integer :: st
type(lock_type),save :: lock_var[*]
lock(lock_var[6],acquired_lock=success,stat=st)
if (st/=0) error stop
if (success) then
   p[6] = p[6] + 1
   unlock(lock_var[6])
else
   : ! Do something else
```

Atomics

`call atomic_define(atom, value)`

Defines **`atom`** atomically with the value of **`value`**

`call atomic_ref (value, atom)`

Defines **`value`** atomically with the value of **`atom`**

`atom` must be integer or logical with kind **`atomic_int_kind`** or **`atomic_logical_kind`** from the intrinsic module **`iso_fortran_env`** and **`value`** must be of the same type but need not be of same kind.

Sync memory

`sync_memory` is a local statement that subdivides a segment into two smaller segments.

The compiler is aware that the ordering of the parts may differ. For example, must not keep remote data in registers.

Typical use is for the spin-wait loop.

Spin-wait loop

```
use iso_fortran_env
logical(atomic_logical_kind),save :: atom[*]=.true.
logical :: value=.true.
```

Image **i**

```
: ! Segment A
sync memory
call atomic_define &
     (atom[j], .false.)
```

Image **j**

```
do while (value)
   call atomic_ref &
              (value, atom)
end do
sync memory
  : ! Segment B
    ! Follows segment A
```

Advantages of coarrays

- References to local data are obvious as such.
- Easy to maintain code - more concise than MPI and easy to see what is happening
- Integrated with Fortran - type checking, type conversion on assignment, ...
- The compiler can optimize communication
- Local optimizations still available
- Does not make severe demands on the compiler, e.g. for coherency.

References

ISO/IEC (2010). Information technology - Programming languages - Fortran -Part 1: Base language. ISO/IEC 1539-1:2010(E), ISO, Geneva.

Metcalf, M., Reid, J., and Cohen, M. (2011). Modern Fortran Explained. OUP, Oxford.

The sides and the codes associated with these lectures are available on the web site :

http://www.nd.edu/~dbalsara/Numerical-PDE-Course