Preparation

- **Log in to mirage**

```
$ ssh -l login mirage[0-2].ucar.edu
```

  Use CryptoCard or Yubikey

- **Copy the example source files**

```
$ cp –r /glade/home/dnagle/Fortran-II .
```

Modern Fortran II for Computational Scientists

Consulting Services Group
Sidd Ghosh & Dan Nagle (Presenter)
May 23, 2012

Computational & Information Systems Laboratory

NCAR

Outline

- Arrays
- Pointers
- Modules
- Procedures
- Intrinsics
- Generics
- Operators and Assignment

01-Arrays

- arrays have constructors
- the rank is the number of indexes
- the extent of a dimension is its size
- extent = max((last - first + inc) / inc, 0)
- shape is the vector of extents
- size of the array is the product of the extents (which may be zero)

02-Array

- an array slice is specified by a triple: (first: last: inc)
- a missing first is the lower bound
- a missing last is the upper bound
- a missing inc is 1
- the right-hand side is evaluated completely before assignment

03-Array

- array element order is column major
- arrays are conformable when they have the same shape
  - the bounds may differ
  - the strides of slices may differ

04-Array

- arrays expressions can differ in their bounds and skip increment
- the number of elements is what matters when computing the size of an extent
- the shape is the ordered set of extents
- the shapes must match
- a scalar matches any shape

05-Array

- a rank-1 integer array can serve as an index array giving indirect addressing
- this is called a vector subscript
- an array with a vector subscript cannot be a pointer target

06-Array_Intrinsic

- cshift is an array circular shift
- it shifts one extent of an array
- can choose the extent
- can choose the shift count
- can shift positive or negative

07-Array_Intrinsic

- eoshift is an array-based end-off shift
- it shifts one extent of an array
- can choose the extent
- can choose the shift count
- can shift positive or negative
- can choose the boundary value

08-Array_Intrinsic

- bit-level reduction procedures operate on integer arrays
- iall - reduce by and
- iany - reduce by or
- iparity - reduce by xor
- logical reduction procedure
- parity - reduce by logical xor

09-Array_Intrinsic

- vector merge is an element-by-element merge of two arrays
- merge is controlled by an array logical mask
- there is a true stream
- and a false stream
- all must be conformable

10-Array_Intrinsic

- norm2 is a Euclidean norm
- the array must be real but can be of any rank
- the optional dim selects an extent along which the operation goes

11-Array_Intrinsic

- pack produces a vector from an array where a condition is true
- unpack reverses the effects of pack
- together they can create a buffer for transmission or computation

12-Array_Intrinsic

- spread copies a lower-rank array across a higher rank array
- choose the direction of copy

13-Array_Intrinsic

- move_alloc is used to enlarge an array allocation with minimal memory traffic
- this allows the data copy operation to be completely controlled by the program

14-Pointers

- pointers are aliases
- they can reference
  - a whole variable such as an array
  - a slice of an array
  - a whole derived type
  - a component of a derived type

15-Pointers

- derived type components can be referenced by their component names
- or by a pointer reference

16-Pointers

- a pointer can alias a whole array or just a slice of an array
- pointers to discontiguous targets may execute less efficiently

17-Pointers

- calling a procedure without an explicit interface and with a pointer array with a non-unit stride causes a copy

18-Pointers

- pointer arrays take the bounds of their targets
- a slice is different than the whole array, even of it covers the whole array
- dummy arguments are specified by their declarations

19-Pointers

- pointers can be used to remap array rank and/or bounds
- the remapped pointer must cover the entire target

20-Modules

- modules can be sued to share data
- the module is referenced by the use statement

21-Modules

- an only clause on the use statement limits the names imported

22-Modules

- public and private statements within the module control export of names

23-Modules

- modules can share procedures as well as data
- module procedures have explicit interfaces which allows the compiler to check usage

24-Modules

- a rename clause on a use statement allows control of name collisions

25-Procedures

- a procedure may be a function or a subroutine
- the function supplies a value
- the subroutine is referenced via a call statement

26-Procedures

- a function may have a result clause which names the result variable

27-Procedures

- a subroutine can return results via its argument list
- a subroutine can also manipulate host associated data, but this should be done carefully!

28-procedures

- an internal procedure has an explicit interface so the compiler can check usage
- an internal procedure accesses data from its host via host association

29-Procedures

- with an explicit interface an array may be passed as an assumed shape array

30-Procedures

- a program or an external subprogram may have internal procedures
- a module may have module procedures

31-Intrinsics

- Fortran 2008 enlarged the math library
- most compilers have implemented most of it

32-Intrinsics

- reductions produce a single value from an array
- the value may be of derived type for user-defined procedures, but the intrinsic procedures operate on intrinsic types

33-Intrinsics

- some reduction intrinsics have an optional dim argument
- it selects the dimension over which the reduction operates

34-Intrinsics

- the loc intrinsics return the location (that is, the indexes) of the value queried
- they also take an optional dim argument

35-Intrinsics

- the shift intrinsics can be used in calculations such as the Jacobi iteration

36-Intrinsics

- the merge intrinsic produces a single result from two streams and a logical mask

Modern Fortran II for Computational Scientists

Thanks for Attending!

May 23, 2012

CISL Computational & Information Systems Laboratory

NCAR