Chasm Reference Manual

Generated by Doxygen 1.3.4

Mon Sep 29 17:57:06 2003

Contents

Chapter 1

Array Descriptor Library

Fortran 90 (and later) compilers pass array-valued parameters via an array descriptor (sometimes called a dope vector). The Chasm library provides C functions to access elements in an array descriptor.

The API for the array descriptor library is divided into several groups:

- Creation and initialization of an array descriptor:
  1. createArrayDesc_F90Vendor()
  2. setArrayDesc_F90Vendor()
  3. resetArrayDesc_F90Vendor()
- Creation and initialization of a temporary, array and hidden descriptor pair, used in a procedure call:
  1. copyToArrayDescAndHidden_F90Vendor()
  2. createArrayDescAndHidden_F90Vendor()
  3. freeArrayDescAndHidden_F90Vendor()
- Access functions:
  1. getArrayBaseAddress_F90Vendor()
  2. getArraySize_F90Vendor()
  3. getArrayLowerBound_F90Vendor()
  4. getArrayExtent_F90Vendor()
  5. getArrayStrideMult_F90Vendor()
- Miscellaneous helper functions:
  1. getArrayDescSize_F90Vendor()

2. hiddenArrayDescType_F90Vendor()
3. verifyArrayDesc_F90Vendor()
4. nullifyArrayDesc_F90Vendor()
5. printArrayDesc_F90Vendor()
6. equalsArrayDesc_F90Vendor()
7. getMangledName_F90Vendor()
8. F90_SetCCFunctions_F90Vendor()

Chapter 2

Chasm Module Index

2.1 Chasm Modules

Here is a list of all modules:

Chapter 3

Chasm Data Structure Index

3.1 Chasm Data Structures

Here are the data structures with brief descriptions:

Chapter 4

Chasm File Index

4.1 Chasm File List

Here is a list of all files with brief descriptions:

Chapter 5

Chasm Module Documentation

5.1 Create and Initialize

Functions in this group are used to create and initialize an array descriptor.

Functions

- void * createArrayDesc_F90Vendor (void *desc, void *hidden, int rank, F90_-DescType desc_type)

  Creates an array descriptor by copying an existing array and hidden descriptor pair.

- int setArrayDesc_F90Vendor (void *desc, void *base_addr, int rank, F90_Desc-Type desc_type, F90_ArrayDataType data_type, unsigned long element_size, const long *lowerBound, const unsigned long *extent, const long *strideMult)

  Sets the elements of a preallocated array descriptor.

- int resetArrayDesc_F90Vendor (void *desc, void *base_addr, int rank, const long *lowerBound, const unsigned long *extent, const long *strideMult)

  Resets the elements of a preallocated array descriptor.

5.1.1 Detailed Description

Functions in this group are used to create and initialize an array descriptor.

Descriptors created by these functions may be permanently stored in C (as long as the array associated with the descriptor has not freed; it is an error to use a descriptor if the associated array has been freed). However, these descriptors cannot be directly used

in a procedure call to Fortran as the call may require a hidden descriptor parameter (depending on the Fortran compiler). Descriptors created by these functions must first be converted to a descriptor and hidden descriptor pair by a function in the **"Procedure Call"** group.

5.1.2 Function Documentation

5.1.2.1 void* createArrayDesc_F90Vendor (void * *desc*, void * *hidden*, int *rank*, F90_DescType *desc_type*)

Creates an array descriptor by copying an existing array and hidden descriptor pair.

This function is used when passing an array from Fortran to C. The array and hidden descriptor input parameters are obtained from the Fortran calling procedure; they must not be otherwise stored or saved.

The returned descriptor cannot be directly used in calls to Fortran because a hidden descriptor may be required by some compilers. The created descriptor may be copied to a descriptor and hidden descriptor pair by either createArrayDescAndHidden_F90Vendor() (for a C call to Fortran) or by copyToArrayDescAndHidden_F90Vendor() (for modifying an array pointer on return from a Fortran call to C). **Note**, it is the callers responsibility to free the returned descriptor.

See also:

setArrayDesc_F90Vendor()
resetArrayDesc_F90Vendor()

Parameters:

desc the array descriptor (from Fortran call stack) to copy (in parameter).

hidden hidden descriptor (from Fortran call stack; in parameter).

rank the rank of the array (in parameter).

desc_type type of the source descriptor (in parameter).

Returns:

a newly allocated array descriptor.

5.1.2.2 int resetArrayDesc_F90Vendor (void * *desc*, void * *base_addr*, int *rank*, const long * *lowerBound*, const unsigned long * *extent*, const long * *strideMult*)

Resets the elements of a preallocated array descriptor.

The descriptor must have been previously initialized by either setArrayDesc_F90Vendor() or createArrayDesc_F90Vendor(). The rank, data type and element size

of the array MUST NOT change. **Note**, at least, getArrayDescSize_F90Vendor() bytes must have been allocated for the descriptor.

Parameters:

desc a previously initialized array descriptor (memory created and freed by caller; inout parameter).

base_addr the base address of the array (in parameter).

rank the rank of the array (in parameter).

lowerBound array[rank] of lower bounds for the array (in parameter).

extent array[rank] of extents for the array (in elements; in parameter).

strideMult array[rank] of distances between successive elements (in bytes; in parameter).

Returns:

0 if successful (nonzero on error).

5.1.2.3 int setArrayDesc_F90Vendor (void * *desc*, void * *base_addr*, int *rank*, F90_DescType *desc_type*, F90_ArrayDataType *data_type*, unsigned long *element_size*, const long * *lowerBound*, const unsigned long * *extent*, const long * *strideMult*)

Sets the elements of a preallocated array descriptor.

This function is used when memory for an array has been created in C and the C array is to be subsequently passed to a Fortran procedure.

The initialized descriptor cannot be directly used in calls to Fortran because a hidden descriptor parameter may be required by some compilers. The initialized descriptor may be copied to a descriptor and hidden descriptor pair by either createArrayDescAndHidden_F90Vendor() (for a C call to Fortran) or by copyToArrayDescAndHidden_F90Vendor() (for modifying an array pointer on return from a Fortran call to C).

Memory for the descriptor is allocated and freed by the caller. The size of descriptor must be at least getArrayDescSize_F90Vendor() bytes.

See also:

createArrayDesc_F90Vendor()
resetArrayDesc_F90Vendor()

Parameters:

desc an uninitialized array descriptor (memory created and freed by caller; out parameter).

base_addr the base address of the array (in paramemter).

rank the rank of the array (in parameter).

desc_type type of the descriptor (in parameter).

data_type data type of an array element (in parameter).

element_size size of an array element (in parameter).

lowerBound array[rank] of lower bounds for the array (in parameter).

extent array[rank] of extents for the array (in elements; in parameter).

strideMult array[rank] of distances between successive elements (in bytes; in parameter).

Returns:

0 if successful (nonzero on error).

5.2 Procedure Calls

Functions in this group are used to create (or copy to) a temporary array and hidden descriptor pair used in a procedure call.

Functions

- int copyToArrayDescAndHidden_F90Vendor (void *src, int rank, F90_DescType desc_type, void *desc, void *hidden)

  Copies an array descriptor to an array and hidden descriptor pair.

- int createArrayDescAndHidden_F90Vendor (void *src, int rank, F90_DescType desc_type, void **desc, void **hidden)

  Creates an array and hidden descriptor pair from an existing descriptor.

- int freeArrayDescAndHidden_F90Vendor (F90_DescType desc_type, void *desc, void *hidden)

  Frees an array and hidden descriptor pair created by a call to createArrayDescAndHidden_F90Vendor().

5.2.1 Detailed Description

Functions in this group are used to create (or copy to) a temporary array and hidden descriptor pair used in a procedure call.

Note that descriptors created or intialized by functions in the **"Create and Initialize"** group cannot be used directly in a procedure call, but must first be copied by one of the functions in this group.

5.2.2 Function Documentation

5.2.2.1 int copyToArrayDescAndHidden_F90Vendor (void * *src*, int *rank*, F90_DescType *desc_type*, void * *desc*, void * *hidden*)

Copies an array descriptor to an array and hidden descriptor pair.

This function may be used to modify an array pointer on the return of a Fortran call to C. The source descriptor must have been created by a call to either createArrayDesc_F90Vendor() or setArrayDesc_F90Vendor().

See also:

createArrayDescAndHidden_F90Vendor()

Parameters:

src the source descriptor (in parameter).

rank the rank of the source and destination arrays (in parameter).

desc_type type of the source and destination descriptors (in parameter).

desc the destination array descriptor (from Fortran call stack; out parameter).

hidden the destination hidden descriptor (from Fortran call stack; out parameter).

Returns:

0 if successful (nonzero on error).

5.2.2.2 int createArrayDescAndHidden_F90Vendor (void * *src*, int *rank*, F90_DescType *desc_type*, void ** *desc*, void ** *hidden*)

Creates an array and hidden descriptor pair from an existing descriptor.

This function may used in calling a Fortran procedure from C (with an array-valued parameter). The source descriptor must have been created by a call to either createArrayDesc_F90Vendor() or setArrayDesc_F90Vendor().

Note, the companion function freeArrayDescWithHidden_F90Vendor() must be called to free the parameters desc and hidden after use (lifetime must not exceed that of the source descriptor).

See also:

copyToArrayDescAndHidden_F90Vendor()

Parameters:

src the source descriptor (in parameter).

rank the rank of the source and destination arrays (in parameter).

desc_type type of the source and destination descriptors (in parameter).

desc on return, contains address of created array descriptor (to be passed to Fortran; out parameter).

hidden on return, contains address of hidden form of the array descriptor (to be passed to Fortran; out parameter).

Returns:

0 if successful (nonzero on error).

5.2.2.3 int freeArrayDescAndHidden_F90Vendor (F90_DescType *desc_type*, void * *desc*, void * *hidden*)

Frees an array and hidden descriptor pair created by a call to createArrayDescAndHidden_F90Vendor().

Parameters:

desc_type type of the descriptor (in parameter).

desc address of descriptor to be freed (inout parameter).

hidden address of hidden form of the descriptor to be freed (inout parameter).

Returns:

0 if successful (nonzero on error).

5.3 Accessors

Functions in this group provide access to information contained in an array descriptor obtained by a call to either createArrayDesc_F90Vendor() or setArrayDesc_F90Vendor().

Functions

- void * getArrayBaseAddress_F90Vendor (const void *desc, int rank)
  Returns a pointer to the base address of the array.
- unsigned long getArraySize_F90Vendor (const void *desc, int rank)
  Returns the array size (in elements).
- long getArrayLowerBound_F90Vendor (const void *desc, int rank, int dim)
  Returns the lower bound for the given dimension.
- unsigned long getArrayExtent_F90Vendor (const void *desc, int rank, int dim)
  Returns the extent of the array for the given dimension (in elements).
- long getArrayStrideMult_F90Vendor (const void *desc, int rank, int dim)
  Returns the distance between successive elements (in bytes).

5.3.1 Detailed Description

Functions in this group provide access to information contained in an array descriptor obtained by a call to either createArrayDesc_F90Vendor() or setArrayDesc_F90Vendor().

5.3.2 Function Documentation

5.3.2.1 void* getArrayBaseAddress_F90Vendor (const void * *desc*, int *rank*)

Returns a pointer to the base address of the array.

Parameters:

desc array descriptor (in parameter).

rank the rank of the array (in parameter).

Returns:

the base address of the array.

5.3.2.2 unsigned long getArrayExtent_F90Vendor (const void * *desc*, int *rank*, int *dim*)

Returns the extent of the array for the given dimension (in elements).

Parameters:

desc array descriptor (in parameter).

rank the rank of the array (in parameter).

dim the dimension (in parameter).

Returns:

the array extent for the given dimension (in elements).

5.3.2.3 long getArrayLowerBound_F90Vendor (const void * *desc*, int *rank*, int *dim*)

Returns the lower bound for the given dimension.

Parameters:

desc array descriptor (in parameter).

rank the rank of the array (in parameter).

dim the dimension (in parameter).

Returns:

the lower bound of the array for the given dimension.

5.3.2.4 unsigned long getArraySize_F90Vendor (const void * *desc*, int *rank*)

Returns the array size (in elements).

Parameters:

desc array descriptor (in parameter).

rank the rank of the array (in parameter).

Returns:

the array size.

5.3.2.5 long getArrayStrideMult_F90Vendor (const void * *desc*, int *rank*, int *dim*)

Returns the distance between successive elements (in bytes).

Parameters:

desc array descriptor (in parameter).

rank the rank of the array (in parameter).

dim the dimension (in parameter).

Returns:

the array stride for the given dimension (in bytes).

5.4 Miscellaneous

This group of functions contains various miscellaneous helper functions.

Functions

- unsigned long getArrayDescSize_F90Vendor (int rank)

  Returns the size of an array descriptor (in bytes).

- F90_HiddenDescType hiddenArrayDescType_F90Vendor (F90_DescType desc_type)

  Returns the type of hidden descriptor used by the compiler for the given descriptor type.

- int verifyArrayDesc_F90Vendor (const void *desc, int rank)

  Verify an array descriptor.

- int nullifyArrayDesc_F90Vendor (void *desc, int rank)

  Nullify an array descriptor.

- int printArrayDesc_F90Vendor (const void *desc, int rank)

  Prints all fields in the array descriptor.

- int equalsArrayDesc_F90Vendor (const void *desc1, const void *desc2, int rank)

  Determines if two descriptors are equal (equivalent).

- char * getMangledName_F90Vendor (const char *fun_name, const char *mod_name)

  Returns the symbol name of a module procedure, if the module name is not null, otherwise returns the name of the procedure.

- void F90_SetCCFunctions_F90Vendor (F90_CompilerCharacteristics *cc)

  Set CompilerCharacteristics function pointers for a particular compiler.

5.4.1 Detailed Description

This group of functions contains various miscellaneous helper functions.

Note that all descriptor parameters in this function group must have been obtained by a call to either createArrayDesc_F90Vendor() or setArrayDesc_F90Vendor().

5.4.2 Function Documentation

5.4.2.1 int equalsArrayDesc_F90Vendor (const void * *desc1*, const void * *desc2*, int *rank*)

Determines if two descriptors are equal (equivalent).

Parameters:

desc1 first descriptor (in parameter).

desc2 second descriptor (in parameter).

rank the rank of the array (in parameter).

Returns:

1 if equal, 0 otherwise.

5.4.2.2 void F90_SetCCFunctions_F90Vendor (F90_CompilerCharacteristics * *cc*)

Set CompilerCharacteristics function pointers for a particular compiler.

This function is called by the generic (and global) F90_SetCompilerCharacteristics() function to provide a generic interface to the Chasm array-descriptor library.

Parameters:

cc the F90_CompilerCharacteristics struct (out parameter).

5.4.2.3 unsigned long getArrayDescSize_F90Vendor (int *rank*)

Returns the size of an array descriptor (in bytes).

Parameters:

rank the rank of the array (in parameter).

Returns:

the descriptor size (in bytes).

5.4.2.4 char* getMangledName_F90Vendor (const char * *fun_name*, const char * *mod_name*)

Returns the symbol name of a module procedure, if the module name is not null, otherwise returns the name of the procedure.

Note that static memory is used for the return value so it must be copied if retained because the memory is overwritten at each call.

Parameters:

fun_name the name of the procedure (in parameter).

mod_name the module name (in parameter). Should be NULL if a global procedure rather than a module procedure.

Returns:

the symbol name.

5.4.2.5 F90_HiddenDescType hiddenArrayDescType_F90Vendor (F90_DescType *desc_type*)

Returns the type of hidden descriptor used by the compiler for the given descriptor type.

Parameters:

desc_type type of the descriptor (in parameter).

Returns:

the hidden descriptor type.

5.4.2.6 int nullifyArrayDesc_F90Vendor (void * *desc*, int *rank*)

Nullify an array descriptor.

This sets elements of the array descriptor so that the Fortran nullify intrinsic will return false, given the associated array.

Parameters:

desc array descriptor (inout parameter).

rank the rank of the array (in parameter).

Returns:

0 if successful (nonzero on error).

5.4.2.7 int printArrayDesc_F90Vendor (const void * *desc*, int *rank*)

Prints all fields in the array descriptor.

Parameters:

desc array descriptor (in parameter).

rank the rank of the array (in parameter).

Returns:

0 if successful (nonzero on error).

5.4.2.8 int verifyArrayDesc_F90Vendor (const void * *desc*, int *rank*)

Verify an array descriptor.

This function does what it can (given only the limited information in the descriptor) to verify that the elements of the array descriptor are correct (i.e., self consistent).

Parameters:

desc array descriptor (in parameter).

rank the rank of the array (in parameter).

Returns:

0 if the descriptor is valid, nonzero otherwise.

Chapter 6

Chasm Data Structure Documentation

6.1 F90_CompilerCharacteristics Struct Reference

CompilerCharacteristics contains function pointers that provide a generic interface to the array descriptor library.

`#include <CompilerCharacteristics.h>`

Data Fields

- int(∗ setArrayDesc)(void ∗desc, void ∗base_addr, int rank, F90_DescType kind, F90_ArrayDataType data_type, unsigned long element_size, const long ∗lowerBound, const unsigned long ∗extent, const long ∗strideMult)
- int(∗ resetArrayDesc)(void ∗desc, void ∗base_addr, int rank, const long ∗lowerBound, const unsigned long ∗extent, const long ∗strideMult)
- void ∗(∗ createArrayDesc)(void ∗desc, void ∗hidden, int rank, F90_DescType kind)
- int(∗ createArrayDescAndHidden)(void ∗src, int rank, F90_DescType kind, void ∗∗desc, void ∗∗hidden)
- int(∗ freeArrayDescAndHidden)(F90_DescType kind, void ∗desc, void ∗hidden)
- int(∗ copyToArrayDescAndHidden)(void ∗src, int rank, F90_DescType kind, void ∗dest, void ∗hidden)
- void ∗(∗ getArrayBaseAddress)(const void ∗desc, int rank)
- unsigned long(∗ getArraySize)(const void ∗desc, int rank)
- long(∗ getArrayLowerBound)(const void ∗desc, int rank, int dim)

- unsigned long(∗ getArrayExtent)(const void ∗desc, int rank, int dim)
- long(∗ getArrayStrideMult)(const void ∗desc, int rank, int dim)
- unsigned long(∗ getArrayDescSize)(int rank)
- int(∗ nullifyArrayDesc)(void ∗desc, int rank)
- int(∗ verifyArrayDesc)(const void ∗desc, int rank)
- F90_HiddenDescType(∗ hiddenArrayDescType)(F90_DescType kind)
- char ∗(∗ getMangledName)(const char ∗fun_name, const char ∗mod_name)
- int(∗ printArrayDesc)(const void ∗desc, int rank)
- int(∗ equalsArrayDesc)(const void ∗desc2, const void ∗desc1, int rank)

6.1.1 Detailed Description

CompilerCharacteristics contains function pointers that provide a generic interface to the array descriptor library.

This struct contains a pointer for each function that is needed for manipulating fortran array descriptors. The F90_SetCompilerCharacteristics() function is used to initialize the function pointers to the correct vendor-specific function.

See also:
F90_SetCompilerCharacteristics()

6.1.2 Field Documentation

6.1.2.1 int(∗ copyToArrayDescAndHidden)(void∗ src, int rank, F90_DescType kind, void∗ dest, void∗ hidden)

6.1.2.2 void∗(∗ createArrayDesc)(void∗ desc, void∗ hidden, int rank, F90_DescType kind)

6.1.2.3 int(∗ createArrayDescAndHidden)(void∗ src, int rank, F90_DescType kind, void∗∗ desc, void∗∗ hidden)

6.1.2.4 int(∗ equalsArrayDesc)(const void∗ desc2, const void∗ desc1, int rank)

6.1.2.5 int(∗ freeArrayDescAndHidden)(F90_DescType kind, void∗ desc, void∗ hidden)

6.1.2.6 void∗(∗ getArrayBaseAddress)(const void∗ desc, int rank)

6.1.2.7 unsigned long(∗ getArrayDescSize)(int rank)

6.1.2.8 unsigned long(∗ getArrayExtent)(const void∗ desc, int rank, int dim)

6.1.2.9 long(∗ getArrayLowerBound)(const void∗ desc, int rank, int dim)

6.1.2.10 unsigned long(∗ getArraySize)(const void∗ desc, int rank)

6.1.2.11 long(∗ getArrayStrideMult)(const void∗ desc, int rank, int dim)

6.1.2.12 char∗(∗ getMangledName)(const char∗ fun_name, const char∗ mod_name)

6.1.2.13 F90_HiddenDescType(∗ hiddenArrayDescType)(F90_DescType kind)

6.1.2.14 int(∗ nullifyArrayDesc)(void∗ desc, int rank)

6.1.2.15 int(∗ printArrayDesc)(const void∗ desc, int rank)

6.1.2.16 int(∗ resetArrayDesc)(void∗ desc, void∗ base_addr, int rank, const long∗ lowerBound, const unsigned long∗ extent, const long∗ strideMult)

6.1.2.17 int(∗ setArrayDesc)(void∗ desc, void∗ base_addr, int rank, F90_DescType kind, F90_ArrayDataType data_type, unsigned long element_size, const long∗ lowerBound, const unsigned long∗ extent, const long∗ strideMult)

6.1.2.18 int(∗ verifyArrayDesc)(const void∗ desc, int rank)

Chapter 7

Chasm File Documentation

7.1 CompilerCharacteristics.h File Reference

```
#include "F90ArrayDataType.h"
```

Data Structures

- struct F90_CompilerCharacteristics

  CompilerCharacteristics contains function pointers that provide a generic interface to the array descriptor library.

Enumerations

- enum F90_DescType {

  F90_Array, F90_Pointer, F90_ArrayPointer, F90_ArrayPointerInDerived,

  F90_DerivedPointer, F90_NonArray }

  F90_DescType describes the type of the array descriptor.

- enum F90_HiddenDescType { F90_Hidden, F90_NoHidden, F90_PointerWith-HiddenDesc }

  F90_HiddenDescType describes the types of hidden descriptor arguments used by a compiler.

Functions

- int F90_SetCompilerCharacteristics (F90_CompilerCharacteristics *cc, const char *compiler)

  Sets the CompilerCharacteristics function pointers for the given compiler.

7.1.1 Enumeration Type Documentation

7.1.1.1 enum F90_DescType

F90_DescType describes the type of the array descriptor.

The descriptor layout can depend on how the array-valued paramater is declared.

Enumeration values:

F90_Array an array (e.g., integer :: a(:))

F90_Pointer a pointer (e.g., integer, pointer :: p)

F90_ArrayPointer an array pointer (e.g., integer, pointer :: pa(:))

F90_ArrayPointerInDerived an array pointer within a derived type

F90_DerivedPointer pointer to a derived type

F90_NonArray not an array or pointer type

7.1.1.2 enum F90_HiddenDescType

F90_HiddenDescType describes the types of hidden descriptor arguments used by a compiler.

Hidden descriptors arguments may be passed at the end of the formal parameter list by some compilers. Most compilers pass explicitly-defined array parameters via a descriptor (F90_NoHidden). However, in some instances, a compiler may place a pointer to the base of the array in the formal parameter slot and pass a hidden descriptor argument after the formal parameters (F90_PointerWithHiddenDesc).

Enumeration values:

F90_Hidden has a hidden param

F90_NoHidden has no hidden parameter

F90_PointerWithHiddenDesc array pointer passed with a hidden descriptor argument

7.1.2 Function Documentation

7.1.2.1 int F90_SetCompilerCharacteristics (F90_CompilerCharacteristics * *cc*, const char * *compiler*)

Sets the CompilerCharacteristics function pointers for the given compiler.

Delegates the actual setting of function pointers to the vendor-specific function F90_SetCCFunctions_F90Vendor(). The current set of supported compilers is {"Absoft", "Alpha", "IBMXL", "Intel", "Lahey", "MIPSpro", "NAG", "SUNWspro"}.

Parameters:

cc the CompilerCharacteristics struct

compiler the name of the compiler

Returns:

0 if successful (nonzero on error)

7.2 F90Vendor.h File Reference

Function declarations for the Chasm array-descriptor library.

```
#include <stdio.h>
#include <CompilerCharacteristics.h>
```

Functions

- void * createArrayDesc_F90Vendor (void *desc, void *hidden, int rank, F90_DescType desc_type)

  Creates an array descriptor by copying an existing array and hidden descriptor pair.

- int setArrayDesc_F90Vendor (void *desc, void *base_addr, int rank, F90_DescType desc_type, F90_ArrayDataType data_type, unsigned long element_size, const long *lowerBound, const unsigned long *extent, const long *strideMult)

  Sets the elements of a preallocated array descriptor.

- int resetArrayDesc_F90Vendor (void *desc, void *base_addr, int rank, const long *lowerBound, const unsigned long *extent, const long *strideMult)

  Resets the elements of a preallocated array descriptor.

- int copyToArrayDescAndHidden_F90Vendor (void *src, int rank, F90_DescType desc_type, void *desc, void *hidden)

  Copies an array descriptor to an array and hidden descriptor pair.

- int createArrayDescAndHidden_F90Vendor (void *src, int rank, F90_DescType desc_type, void **desc, void **hidden)

  Creates an array and hidden descriptor pair from an existing descriptor.

- int freeArrayDescAndHidden_F90Vendor (F90_DescType desc_type, void *desc, void *hidden)

  Frees an array and hidden descriptor pair created by a call to createArrayDescAndHidden_F90Vendor().

- void * getArrayBaseAddress_F90Vendor (const void *desc, int rank)

  Returns a pointer to the base address of the array.

- unsigned long getArraySize_F90Vendor (const void *desc, int rank)

  Returns the array size (in elements).

- long getArrayLowerBound_F90Vendor (const void *desc, int rank, int dim)

  Returns the lower bound for the given dimension.

- unsigned long getArrayExtent_F90Vendor (const void *desc, int rank, int dim)

  Returns the extent of the array for the given dimension (in elements).

- long getArrayStrideMult_F90Vendor (const void *desc, int rank, int dim)

  Returns the distance between successive elements (in bytes).

- unsigned long getArrayDescSize_F90Vendor (int rank)

  Returns the size of an array descriptor (in bytes).

- F90_HiddenDescType hiddenArrayDescType_F90Vendor (F90_DescType desc_type)

  Returns the type of hidden descriptor used by the compiler for the given descriptor type.

- int verifyArrayDesc_F90Vendor (const void *desc, int rank)

  Verify an array descriptor.

- int nullifyArrayDesc_F90Vendor (void *desc, int rank)

  Nullify an array descriptor.

- int printArrayDesc_F90Vendor (const void *desc, int rank)

  Prints all fields in the array descriptor.

- int equalsArrayDesc_F90Vendor (const void *desc1, const void *desc2, int rank)

  Determines if two descriptors are equal (equivalent).

- char * getMangledName_F90Vendor (const char *fun_name, const char *mod_name)

  Returns the symbol name of a module procedure, if the module name is not null, otherwise returns the name of the procedure.

- void F90_SetCCFunctions_F90Vendor (F90_CompilerCharacteristics *cc)

  Set CompilerCharacteristics function pointers for a particular compiler.

7.2.1 Detailed Description

Function declarations for the Chasm array-descriptor library.

Fortran 90 (and later) compilers pass array-valued parameters via an array descriptor (sometimes called a dope vector). The Chasm library provides C functions to get and set array-descriptor elements. This file declares the API for the array-descriptor library.

Note that separate global functions are provided for each Fortran compiler with vendor name appended to the generic function name. In this file, F90Vendor is used in place of a specific compiler vendor. A generic interface is available via function pointers in the F90_CompilerCharacteristics struct. These function pointers are set with the F90_SetCompilerCharacteristics() function.

Index