Subject: boost::multi_array
Posted by Robbie on Thu, 11 Oct 2007 08:29:11 GMT
View Forum Message <> Reply to Message

Does anyone have examples of interfacing IDL with the
boost::multi_array C++ library? I am trying to write a set of type
casting methods using templates. I'm hoping to use template resolution
of assist the writing
of wrappers.

Examples and notes below

Thanks,
Robbie

==========

```
// Takes in an integer and returns an integer
IDL_VPTR some_function_wrapper(int argc, IDL_VTPR argv[]) {
 f = some_function(idl_cast_in<int>(argv[0]));
 return(idl_cast_out<int>(f));
}

// The first argument is the 3d array input, the second arguement is
the 3d array output
// Returns an integer
IDL_VPTR some_function_wrapper(int argc, IDL_VTPR argv[]) {
 multi_array<int, 3> m();
 f = some_function(idl_array_cast_in<int, 3>(argv[0]),m);
 argv[1] = idl_array_cast_out(m,argv[1]);
 return(idl_cast_out<int>(f));
}

// The first argument is the 3d array input and output
// Returns an integer
IDL_VPTR some_function_wrapper(int argc, IDL_VTPR argv[]) {
 f = some_function(idl_array_cast_in_out<int, 3>(argv[0]));
 return(idl_cast_out<int>(f));
}


// Casting methods
idl_array_cast_in
 Turn an IDL variable into a const_multi_array_ref with the given
signature.
 The array can't be altered from within C++.
idl_array_cast_out
 Create a multi_array and copy into an IDL variable after returning.
```

idl_array_cast_in_out
 Turn an IDL variable into a multi_array_ref with the given signature.
 The array can be altered and returned to IDL
idl_cast_in
 Turn an IDL variable into a constant scalar.
 The scalar can't be altered from within C++
idl_cast_out
 Turn a scalar into an IDL variable.
idl_test
 Returns true if a variable matches the signature
idl_array_test
 Returns true if an array matches the signature

// Template definitions of casting methods
template<class IDL_TYPE, const unsigned int IDL_VNDIM>
 const_multi_array_ref<IDL_TYPE, IDL_VNDIM> idl_array_cast_in(IDL_VPTR
v);
template<class IDL_TYPE, const unsigned int IDL_VNDIM>
 IDL_VPTR idl_array_cast_out(multi_array<IDL_TYPE, IDL_VNDIM> a);
template<class IDL_TYPE, const unsigned int IDL_VNDIM>
 IDL_VPTR idl_array_cast_out(multi_array<IDL_TYPE, IDL_VNDIM> a,
IDL_VPTR v);
template<class IDL_TYPE, const unsigned int IDL_VNDIM>
 multi_array_ref<IDL_TYPE,IDL_VNDIM>idl_array_cast_in_out(IDL_VPTR v);
template<class IDL_TYPE>
 const IDL_TYPE idl_cast_in(IDL_VPTR v);
template<class IDL_TYPE>
 IDL_VPTR idl_cast_out(IDL_TYPE c);
template<class IDL_TYPE>
 IDL_VPTR idl_cast_out(IDL_TYPE c, IDL_VPTR v);
template<class IDL_TYPE>
 bool idl_test(IDL_VPTR v);
template<class IDL_TYPE, const unsigned int IDL_VNDIM>
 bool idl_array_test(IDL_VPTR v);

// Type conversions
Most type conversions are inferred from export.h.
uchar   BYTE
short int INT
long int LONG
float   FLOAT
double   DOUBLE
unsigned short int UINT
unsigned long int ULONG
long long LONG64
unsigned long long ULONG64

Other type conversions

```
std::complex<float> COMPLEX
std::complex<double> DCOMPLEX
std::string  STRING
bool   BYTE

Types not supported
POINTER
OBJECT
UNDEFINED

// Structures will be supported via their own casting methods.
// Structures are always array variables.
// The structure definition is passed as a template variable
idl_struct_cast_in
 Turn an IDL variable into a const_multi_array_ref with the given
signature.
 The array can't be altered from within C++.
idl_struct_cast_out
 Create a multi_array and copy into an IDL variable after returning.
idl_struct_cast_in_out
 Turn an IDL variable into a multi_array_ref with the given signature.
 The array can be altered and returned to IDL
idl_struct_test
 Returns true if a variable matches the signature

// Template definitions of casting methods
template<class IDL_S, const unsigned int IDL_VNDIM>
 const_multi_array_ref<IDL_S, IDL_VNDIM> idl_struct_cast_in(IDL_VPTR
v);
template<class IDL_S, const unsigned int IDL_VNDIM>
 IDL_VPTR idl_array_struct_out(multi_array<IDL_S, IDL_VNDIM> a);
template<class IDL_S, const unsigned int IDL_VNDIM>
 IDL_VPTR idl_array_struct_out(multi_array<IDL_S, IDL_VNDIM> a,
IDL_VPTR v);
template<class IDL_S, const unsigned int IDL_VNDIM>
 multi_array_ref<IDL_S,IDL_VNDIM> idl_struct_cast_in_out(IDL_VPTR v);
template<class IDL_S, const unsigned int IDL_VNDIM>
 bool idl_struct_test(IDL_VPTR v);
```