Calling DLL Functions from SLX

Introduction

SLX provides excellent tools for calling functions contained in DLLs. DLLs are almost always generated using C/C++. Although in theory other languages such as Visual Basic could be used, to our knowledge no one has done so. All further discussion presumes the use of C/C++.

The SLX compiler can generate C/C++ ".h" files describing the SLX data to be passed to a C/C++ function. If you're using C/C++ functions to manipulate data stored in SLX objects, you *must* use an SLX-generated ".h" file, since SLX objects map into C/C++ structs in non-obvious ways. (SLX objects contain descriptive information that is hidden from C/C++ code.

Calling Simple DLL Functions

To call a DLL function from an SLX program, you must supply a function prototype that specifies the name of the function, its arguments (if any), the value it returns (if any), and the name of the DLL in which it is contained. The following are examples of DLL function prototypes:

```
procedure concat(string(*) left, string(*) right, out string(*) result)   dll = "johdll";
procedure sum(int i, int j)                      returning int             dll = "johdll";
procedure float_sum(double x, double y)          returning double          dll = "johdll";
procedure compare(int i, int j)                  returning boolean         dll = "johdll";
procedure get_ocount(pointer(obj) oo)            returning int             dll = "johdll";
procedure array_elt(int iarray[*][*], int i, int j)  returning int         dll = "johdll";
procedure csqrt(double x)                        returning double          dll = "johdll";
procedure access_violation()                                               dll = "johdll";
procedure infinite_loop()                                                  dll = "johdll";
```

If you wish to generate a ".h" file for a DLL, compile the SLX program, then click "File, Write DLL Header File." You will be prompted for the name of the file into which the SLX compiler will place the declarations necessary for all the DLL functions prototypes it has compiled. For the source code shown above, SLX will define a C/C++ struct corresponding to the SLX class named "obj", because function get_ocount is passed a pointer to an "obj". If class "obj" contains instances of objects from other classes (sub-objects) or pointers to such objects, definitions for these classes will also be placed in the ".h" file. (The compiler uses a recursive algorithm to traverse "connected" definitions.)

Next, you must write your C/C++ code. Be sure to supply "#include" statements for "slxdll.h" (a standard include file) and for the SLX-generated ".h" file, if any.

Finally, compile your C/C++ code and generate a DLL. Most integrated development environ-ments (IDEs) provide convenient tools for doing this.

SLX Assistance Available within DLL Functions

There are some things that you cannot or should not do from within a DLL function. For example, you can change the value of an SLX control variable, but to have the change properly acted upon within your SLX program, you must "tell" SLX what you've done.

SLX provides two forms of assistance. First, SLX provides a file named "slxdll.cpp" that contains the following functions:

SLX_GetString	Copies an SLX string into a C/C++ string
SLX_PutString	Copies a C/C++ string into an SLX string
SLX_FirstInSet	Returns a pointer to the first member of an SLX set
SLX_LastInSet	Returns a pointer to the last member of an SLX set

Prototypes for the above functions are provided in slxdll.h.

You should consider slxdll.cpp to be a “starter collection” of utility functions. You can copy this code into your files, and over time, you may want to add functions of your own.

The second form of assistance SLX provides to DLL functions is a collection of functions that are internal to SLX, but designed to be called from a DLL. Prototypes for these functions are provided in slxdll.h. To use these functions, you must provide a function named “Connect” in your DLL. Each time SLX loads a DLL, it checks for the presence of a “Connect” function. If SLX finds a “Connect” function, it calls it to initialize the DLL, passing it a pointer to a vector of pointers to functions. You should copy the passed pointer into a static extern named SLXTV. Slxdll.h contains definitions that assume the use of SLXTV as a global pointer.

SLX provides the following functions for your use in C/C++ DLLs:

ChangeControlVariable	Informs SLX that you’ve altered a control variable
ChangePointer	Changes a pointer
PlaceInto	Places an SLX object into an SLX set
RemoveFrom	Removes an SLX object from an SLX set
BuildQueryFromName	Formats a query about an SLX variable
BuildQueryFromPointer	Formats a query about an SLX variable
QuerySLX	Acquires information given a formatted query
SLXprint	Provides C/C++ access to SLX’s “print” stream
SLXStopExecution	Stops execution
SLXMessageBox	Pops up a Windows MessageBox

Prototypes for the above functions are provided in slxdll.h.

Warnings

SLX’s DLL interface gives you access to a great deal of low-level details of an SLX program. As is usually the case in life, great power can lead to disaster if not exercised with great responsibility. Please take note of the following:

1. 32-bit DLLs and 64-bit DLLs are not interchangeable. If you’re running SLX-64, you *must* create 64-bit DLLs. The examples distributed with SLX are 32-bit DLLs. If you wish to use thexe examples with SLX-64, you’ll have to regenerate the .h files (See the next paragraph) and compile the provided source code with a 64-bit C++ compiler.

2. You should always use SLX’s “Write DLL Header File” feature to create a “.h” file describing the data to be passed to your DLL functions. Disagreement over data formats is the number one cause of failure in all cross-language interfaces.

3. SLX NULL pointers are not zero. They point to an internal “NULL object.” This allows SLX to increment and decrement use counts when pointer values are changed, without having to test whether the pointer is NULL. If you assign a C/C++ NULL (zero-valued) pointer to an SLX pointer, disaster will likely ensue.

4. SLX treats float as a synonym for double. There are no 4-byte floating point values in SLX. Even if you say float, you’ll get a double.

5. SLX uses 8-byte (Microsoft standard) alignment for doubles. If your C/C++ compiler, e.g., OenWatcom, assumes 4-byte alignment of doubles in structs, you'll have to use compiler options and/or pragmas to specify 8-byte alignment for compatibility with SLX. 64-bit versions of SLX use 8-byte alignment for pointers as well as doubles.

6. SLX strings are *not* the same as C/C++ strings. SLX strings are passed as pointers to a struct that contains the current and maximum lengths of the string and a pointer to where the string is stored. If you modify an SLX string in a DLL function, you must update the struct. If you change the pointer to where the string is stored, results are unpredictable. At the very least, SLX's checkpoint/restore feature will be compromised, because you have modified an SLX-assigned address. SLX_GetString and SLX_PutString, described above, can handle all the necessary details for you.

7 SLX arrays are passed as pointers to a struct that describes them. This struct is defined in SLXDLL.H, and an example of array access is included in dlltest.cpp.

Distributed Files

The following Files are installed in the SLX\DLLs Folder:

SLXDLL.PDF	This document
SLXDLL.H	A standard header that should always be included in your DLL source code
SLXDLL.CPP	A "starter collection" of C/C++ utility functions
DLLTEST.SLX	An SLX program that calls a number of illustrative DLL functions
DLLTEST.H	The SLX-generated .H file for DLLTEST.SLX
JOHDLL.CPP	Source code for the DLL used by DLLTEST.SLX
JOHDLL.DLL	A compiled version of JOHDLL.C
NEWDLL.SLX	An SLX program that illustrates SLX-assisted DLL functions
NEWDLL.H	The SLX-generated .H file for NEWDLL.SLX
NEWDLL.CPP	Source code for the DLL used by NEWDLL.SLX
NEWDLL.DLL	A compiled version of NEWDLL.C
DLLQUERY.SLX	An SLX program that illustrates the ability to query the internal structure of SLX data from a DLL function
DLLQUERY.H	The SLX-generated .H file for DLLQUERY.SLX
DLLQUERY.CPP	Source code for the DLL used by DLLQUERY.SLX
DLLQUERY.DLL	A compiled version of DLLQUERY.C