CS 404: Lab 4: Creating Dynamically-Linked Libraries

Getting Started

1. We will be doing this lab in Windows, so boot to that operating system and log in.

2. Open a web browser and go to the course website[1]. Download DLL-test.tar.

3. Double clicking on the tar file will launch WinZip (it may launch the WinZip set-up application first, go ahead and set it up). Click "Extract" and place the contents in the S drive.

Building a DLL

1. Open VisualC++ 6.0 from the Startup menu. You need to create two projects: one to build the DLL and the other to build the program. To create the DLL project, select "New" from the "File" menu. Click on the "Projects" tab to display a selection of several project types. Select "Win32 Dynamic-Link Library." Then, give the project a name (I used testDLL), and choose the location where the project will be placed (I selected the DLLtest directory). Click OK. You will then be asked what kind of DLL you'd like to create. Answer "Empty" (the others will give you sample code and will set the compiler options to do Windows things, none of which we want). You will then see a summary of what you've selected, click OK if everything looks good.

2. We need to add the source for the DLL to the project. From the Project menu, select "Add to Project" and then "Files." Navigate to the source directory and select dlltest.c and dlltest.h (you can select both by holding down the shift key when you click on them). VizStudio has now added theses files to the project and organized them somewhat. Click on the "File View" tab at the bottom of the screen. Then click on the + symbol beside "testDLL files" to view the files in this project. Expanding the Source Files folder will reveal testdll.c, while testdll.h

[1]www.cs.cornell.edu/Course/cs404/2002sp

is in the “Header Files” folder. These folders are just VizStudio’s way of organizing your code–they don’t refer to actual folders (the files are still in the source directory). Double click on dlltest.h. How is PrintArray defined? Compare this to the prototypes for Error and RandomLocalFunction.

3. To build the DLL, select “Build testdll.dll” from the “Build” menu. This will compile the code and create several files in the Debug directory under testDLL. Look at the contents of the Debug directory. The three interesting files are the DLL (testDLL.dll), the list of exported functions (testDLL.exp), and the library (testDLL.lib). In windows terminology, a DLL is like a shared-object library (.so file) on UNIX, while a regular library (.lib file) is a static library (.a file). Why do you think we have both a .lib file and a DLL?

4. We now need to build our main program. Create a new VizStudio project, but this time, select “Win32 console application.” I created my project in DLLtest and called it “TestIt.” You will then be asked what kind of console app you want to create; choose and empty one. Add the source file testit.c. Take a look at this file. It is a simple program whose only purpose is to test our DLL. It asks for a file name, creates a length 10 array, fills it with numbers, and then calls PrintArray (in the DLL) to print it out. Try building this program by selecting “Build TestIt” from the Build menu. What happens?

5. Although the file compiles, it doesn’t link correctly because we haven’t told it where to find the object code for PrintArray. To fix this, we need to do the Windows-equivalent of augmenting our library search path with the -L flag and linking to testDLL.lib. Select “Settings” from the “Project” menu. This will bring up a window with several tabs and menus. The settings panel is how you control how your project gets built–it is a lot like setting compiler flags in UNIX. Since we’re having trouble linking, click on the “Link” tab. Select “Input” from the “Category” menu. The fields in the window will change. In particular, a field called “Additional library path” will appear, second from the bottom. This is like the input for -L. Unfortunately, Microsoft didn’t bother to make this easy by providing a browse option, you must enter the path by hand. I entered:

   S:\DLLtest\testDLL\Debug

We then need to add "\t.lib" to the list of "Object/library modules" near the top. Just type it in at the end of the list. Click OK and try building again. Let me know if it doesn't work.

6. We're now ready to run our program. Our program is designed to run from a console. Open up the Command Prompt application, it is found under the Start menu, in the Accessories folder. An intimidating black window will open up. Type "S:" at the prompt to change to the S drive. Then cd to DLLtest\TestIt\Debug, where TestIt.exe is. To run the program, type TestIt at the command prompt. What happens?

7. Even though we linked to the DLL (or its library), we haven't told the system where to find testDLL.dll. To do this, we need to set a system variable called PATH to include the directory containing our DLL. To see the current value of PATH, type PATH on the command line. To add DLLtest\testDLL\Debug to your path, enter:

   ```
   set PATH=%PATH%;S:\DLLtest\testDLL\Debug
   ```

   Be careful, PATH is case sensitive, so make sure that the path you enter is exactly correct! Type PATH again to check if this directory has been added. Try running TestIt again. If it doesn't work, double check that you entered the path correctly.

8. Now we'll demonstrate the real utility of DLL's. Go back to VizStudio and select TestDLL from the "Recent Workspaces" menu from the "File" menu. Change something inside PrintArray (I added fprintf(FIL,"%% A matlab comment line\n"); just be fore the for loop. Then, rebuild the DLL and try running TestIt. Did your change show up?