The Super Simple Guide to using a Visual D Project with a Separate D Library

6 June 2014

1. Test that the linker is set to Digital Mars optlink. Open a command window and type link. You should see:  
     
   OPTLINK (R) for Win32 Release 8.00.15  
   Copyright (C) Digital Mars 1989-2013 All rights reserved.
2. I’m going to go through the steps of creating a simple static library, but this should work with existing static D libraries as well.
3. Create the Static Library: Open Visual Studio 2013. Open New Project Dialog (File/New…/Project). In the dialog, under the Installed pane, select “Static Library D”. Enter the Name. For this document, I’m going to assume the Name is StaticLib. Hit OK.
4. The New Project dialog should close and, under the Solution Explorer pane, you should see Solution ‘StaticLib’ (1 project), followed by a drop-down for StaticLib. If you click on the little down arrow to the upper left of StaticLib, it will expand and you should see the name of the automatically generated source file: lib.d. If you double-click on lib.d, it should come up in the main pane for you to edit. The file should contain a single line defining the module name:  
     
   module lib;
5. Now add the following lines so you have:  
     
   module lib;  
   class Bar {  
    double val;  
    this(double value) {  
    val = value;  
    }  
   }  
     
   Under the BUILD menu, select Build or hit F7. In the output pane, you should see:  
     
   ------ Build started: Project: StaticLib, Configuration: Debug Win32 ------  
   Building Debug\StaticLib.lib...  
   ========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
6. OK. Were done with creating the static library. Staticlib.lib should be in your Visual Studio 2013\Projects\StaticLib\Debug directory. lib.d (which you will need for specifying the import and import path to other projects using this library) should be in Visual Studio 2013\Projects\StaticLib.
7. Go to the FILE menu and select New/Project… The New Project dialog menu should pop up again. This time after selecting D under the Installed pane, select Console Application DMD/GDC. Name the application ConsoleApp, and click OK.
8. The New Project dialog should close and, under the Solution Explorer pane, you should see Solution ‘ConsoleApp’ (1 project), followed by a drop-down for ConsoleApp. If you click on the little down arrow to the upper left of ConsoleApp, it will expand and you should see the name of the automatically generated source file: main.d. If you double-click on main.d, it should come up in the main pane for you to edit. The file should contain the following lines:  
     
   import std.stdio;  
     
   int main(string[] argv)  
   {  
    writeln("Hello D-World!");  
    return 0;  
   }
9. Now add lines to import the module you defined in the static library and its type Bar, and to reference that type. You should end up with the following.  
     
   import std.stdio;   
   import lib;  
     
   int main(string[] argv)  
   {  
    auto b = Bar(5.0);  
    writeln("Hello D-World!");  
    return 0;  
   }
10. Now build the project by selecting Build from the BUILD menu or hitting F7. If you see:  
      
    ------ Build started: Project: ConsoleApp, Configuration: Debug DMD Win32 ------  
    Building Debug DMD Win32\ConsoleApp.exe...  
    main.d(5): Error: undefined identifier Bar  
    Building Debug DMD Win32\ConsoleApp.exe failed!  
    Details saved as "file://c:\users\larry\documents\visual studio..."  
    ========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========  
      
    It means you forgot to add lib to the import statement. Do that. Now, You should see:  
      
    ------ Build started: Project: ConsoleApp, Configuration: Debug DMD Win32 ------  
    Building Debug DMD Win32\ConsoleApp.exe...  
    main.d(1): Error: module lib is in file 'lib.d' which cannot be read  
    import path[0] = C:\D\dmd2\windows\bin\..\..\src\phobos  
    import path[1] = C:\D\dmd2\windows\bin\..\..\src\druntime\import  
    import path[2] = C:\D\dmd2\windows\bin\..\..\src  
    Building Debug DMD Win32\ConsoleApp.exe failed!  
    Details saved as "file://c:\users\larry\documents\visual studio 2013... "  
    ========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
11. The reason the build failed this time is because the compiler has no idea where static library lib is. In the Solution Explorer pane, right click on the project name ConsoleApp and select the menu item properties. The ConsoleApp Property Pages dialog should come up. Under the Configuration Properties drop down, select Compiler. You should see the Compiler option Additional Imports. Enter the full path name of to the lib.d file you created when you created the static library. Just the path, not the file name itself. Click OK. Hit F7 to build. Still didn’t work (same error). That’s probably because your path to your static library contained spaces (like the “Visual Studio” part). Go back to the Additional Imports field of the ConsoleApp Property Pages and surround the path name with double quotes. Close the dialog and re-run build. That’s right, it still didn’t work. In the Output pane, you should see:  
      
    ------ Build started: Project: ConsoleApp, Configuration: Debug DMD Win32 ------  
    Building Debug DMD Win32\ConsoleApp.exe...  
    main.d(5): Error: no property 'opCall' for type 'lib.Bar'  
    Building Debug DMD Win32\ConsoleApp.exe failed!  
    Details saved as "file://c:\users\larry\documents\visual studio 2013... "  
    ========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
12. So what is that strange opCall thing? I’ll let you look that up yourself. I have to say this is a terrible error message (that I’ve seen at least one other person complain about). The simple problem is that we just didn’t put the new keyword in front of the request for the Bar object. The line should read:  
      
     auto b = new Bar(5.0);  
      
    OK, let’s rebuild. Hit F7.
13. Not out of the woods yet. Now we get:  
      
    ------ Build started: Project: ConsoleApp, Configuration: Debug DMD Win32 ------  
    Building Debug DMD Win32\ConsoleApp.exe...  
    OPTLINK (R) for Win32 Release 8.00.15  
    Copyright (C) Digital Mars 1989-2013 All rights reserved.  
    <http://www.digitalmars.com/ctg/optlink.html>  
    Debug DMD Win32\ConsoleApp.obj(ConsoleApp)   
    Error 42: Symbol Undefined \_D3lib3Bar7\_\_ClassZ  
    Debug DMD Win32\ConsoleApp.obj(ConsoleApp)   
    Error 42: Symbol Undefined \_D3lib3Bar6\_\_ctorMFdZC3lib3Bar (lib.Bar…)   
    Building Debug DMD Win32\ConsoleApp.exe failed!  
    Details saved as "file://c:\users\larry\documents\visual studio 2013... "  
    ========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========  
      
    The problem is that while the compiler knows where the lib.d file is in order to read class definitions for the import, the linker has no idea where the actual compiled code is for those definitions. In this context, the import statement is much like the include of a header file in C++. The linker knows nothing about it except that you want to use the compiled code for Bar. Once again, in the Solution Explorer pane, right click on ConsoleApp and select Properties. The ConsoleApp Property Pages dialog should come up. Select Linker. In the Library Files field, enter the library name: StaticLib.lib. In the Library Paths field, enter the fully qualified path to the directory containing StaticLib.lib. DON’T FORGET TO PUT DOUBLE QUOTES AROUND THE PATH NAME IF THERE ARE ANY SPACES IN IT!!!”. You won’t get an error message about this. Now, hit OK, and re-build by hitting F7. You should now see:  
      
    ------ Build started: Project: ConsoleApp, Configuration: Debug DMD Win32 ------  
    Building Debug DMD Win32\ConsoleApp.exe...  
    Converting debug information...  
    ========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========  
      
    In other words, WE’RE DONE. Congratulations!
14. An alternative and somewhat easier method of specify the linker dependencies was suggested by Rainer Schuetze. First, remove the linker dependencies you added in step 14. Then add the StaticLib project to the ConsoleApp solution by right clicking on the solution in the Solution Explorer pane, and selecting Add->Existing Project… When the Add Existing Project dialog comes up, navigate to StaticLib/StaticLib and select StaticLib.visualdproj, and click Open. You should now see the StaticLib project under the ConsoleApp project in the Solution Explorer pane. Now, go to the PROJECT menu and select Project Dependencies… When the Project Dependencies dialog comes up, you should see ConsoleApp in the Projects dropdown, and a checkbox for StaticLib in the Depends On box. Check the checkbox. Click OK. Now build the solution again. You should see the same correct result as in step 14.

Note: While I have tried to make this as detailed and accurate as I have time for, I’m a programmer, so the odds that there are bugs in this are probably 100%. Please let me know if you find any at lel486@gmail.com.