Welcome to Polyglot Programming

To become a successful programer , one needs to be conversant with multiple languages. Mixing C# and C/C++ is a necessary skill for anyone who works with the Microsoft Windows Platform.

The first skill one should acquire is how to write a Windows Dynamic Link Library.

STEP 1 :- Key in the source code into notepad file... (Test.cpp)

```
#include <windows.h>


/////////////////////
//
// extern "C" - Guarantee to the compiler that no overload
// __stdcall - pushes from left to right, callee pops the stack
// __declspec(dllexport) - read MSDN
//

extern "C" __declspec(dllexport) int __stdcall Add(int a , int b )
{
   return a + b;
}


/////////////////////
//
// extern "C" - Guarantee to the compiler that no overload
// __stdcall - pushes from left to right, callee pops the stack
// __declspec(dllexport) - read MSDN
//

extern "C" __declspec(dllexport) int __stdcall Sub(int a , int b )
{
   return a - b;
}


/////////////////////
//
// extern "C" - Guarantee to the compiler that no overload
// __stdcall - pushes from left to right, callee pops the stack
// __declspec(dllexport) - read MSDN
//

extern "C" __declspec(dllexport) int __stdcall Mul(int a , int b )
{
   return a*b;
}
```

```
/////////////////////
//
// extern "C" - Guarantee to the compiler that no overload
// __stdcall - pushes from left to right, callee pops the stack
// __declspec(dllexport) - read MSDN
//

extern "C" __declspec(dllexport) int __stdcall Div(int a , int b )
{
   return a/b;
}
```

Compile and Link the program

```
F:\DOORDIE\DLL>cl /c Test.cpp
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 16.00.30319.01 for 80x86
Copyright (C) Microsoft Corporation.  All rights reserved.

Test.cpp

F:\DOORDIE\DLL>link /DLL /out:test.dll Test.obj
Microsoft (R) Incremental Linker Version 10.00.30319.01
Copyright (C) Microsoft Corporation.  All rights reserved.

   Creating library test.lib and object test.exp

F:\DOORDIE\DLL>
```

The Problem with this approach is the Symbol Exported will be mangled by microsoft compiler.

<u>Dumpbin /EXPORTS test.dll</u>

```
Microsoft (R) COFF/PE Dumper Version 10.00.30319.01
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file Test.dll

File Type: DLL

  Section contains the following exports for test.dll

    00000000 characteristics
    4E06A617 time date stamp Sun Jun 26 08:53:03 2011
        0.00 version
```

```
          1 ordinal base
          4 number of functions
          4 number of names

    ordinal hint RVA      name

          1    0 00001000 _Add@8
          2    1 00001030 _Div@8
          3    2 00001020 _Mul@8
          4    3 00001010 _Sub@8

  Summary

        2000 .data
        2000 .rdata
        1000 .reloc
        5000 .text
```

To avoid Microsoft name mangling , we need to create a module Definition file.

STEP 2 :- Create a Module Definition file. The name of the file should be exactly the name of the DLL. For creating TEST.dll , you have to create TEST.def

```
; Filename :- test.def
;--------- Name of the library
LIBRARY TEST

;-------- List of Exported Functions...
EXPORTS
          Add
          Sub
          Mul
          Div
```

Now try to compile and link using the following command line.

```

F:\DOORDIE\DLL>link /DLL /out:test.dll Test.obj /DEF:test.def
Microsoft (R) Incremental Linker Version 10.00.30319.01
Copyright (C) Microsoft Corporation.  All rights reserved.

   Creating library test.lib and object test.exp

F:\DOORDIE\DLL>dumpbin /EXPORTS test.dll
```

```
Microsoft (R) COFF/PE Dumper Version 10.00.30319.01
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file test.dll

File Type: DLL

  Section contains the following exports for TEST.dll

    00000000 characteristics
    4E06A817 time date stamp Sun Jun 26 09:01:35 2011
        0.00 version
           1 ordinal base
           4 number of functions
           4 number of names

    ordinal hint RVA      name

          1    0 00001000 Add
          2    1 00001030 Div
          3    2 00001020 Mul
          4    3 00001010 Sub

  Summary

        2000 .data
        2000 .rdata
        1000 .reloc
        5000 .text

F:\DOORDIE\DLL>
```

A note about the Output

When you create a DLL , you also get a .LIB file. This file is called Import Library. The Library contains the meta information about the contents of the DLL file. Import Library is used to Link , If you are statically Linking to a DLL.

STEP 3

Create a header file , for the Client Programs

```
/////////////
// Test.h
// Pre-Processor constant _TEST_DOT_H is declared to
// avoid duplicate inclusion.
//
```

```
#ifndef _TEST_DOT_H
#define _TEST_DOT_H

#ifdef __cplusplus
extern "C" {
#endif

int __stdcall Add( int , int );
int __stdcall Mul( int , int );
int __stdcall Div( int , int );
int __stdcall Sub( int , int );

#ifdef __cplusplus
}
#endif



#endif
```

How to Create an SDK for your Library ?

SDK stands for Software Development Kit. When you distribute your kit , you need to include

a) DLL files
b) LIB files (Import Libraries for static linking)
Header files for Function prototypes

STEP 4

How do I statically link the DLL to a main program ?

Write a main program.

```
///////////////////
// TestClient.cpp
// Following C/C++ Program calls the exported functions
// from Test.dll
//
// At the Visual studio command line
// ----------------------------------
// >cl /EHsc Testclient.cpp Test.lib
// >Testclient.exe
//
//
#include <windows.h>
#include <iostream>
```

```
using namespace std;

#include "test.h"

int main( int argc , char **argv )
{
     int addans = Add(3,4);
     int divans = Div(3,4);
     int mulans = Mul(4,4);
     int subans = Sub(15,2);

     cout << addans << '\t' << divans <<  '\t' << mulans <<
          '\t' << subans << endl;



}
```

Compile the TestClient.cpp and give Test.lib (import library) at the command line .

```
F:\DOORDIE\DLL>cl /EHsc TestClient.cpp Test.lib
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 16.00.30319.01 for 80x86
Copyright (C) Microsoft Corporation.  All rights reserved.

TestClient.cpp
Microsoft (R) Incremental Linker Version 10.00.30319.01
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:TestClient.exe
TestClient.obj
Test.lib

F:\DOORDIE\DLL>TestClient
7       0       16      13

F:\DOORDIE\DLL>
```

STEP 5

How do I dynamically load a DLL ?

To understand this , one needs to know Function Pointer. The sample program given below illustrates the idea of function Pointer.

```
////////////////
// FuncPointer.cpp
// Function Pointer Tutorial
//
// cl FuncPointer.cpp
// FuncPointer.exe

#include <stdio.h>
#include <stdlib.h>

extern "C" int __stdcall Add(double a, double b )
{
     return (int)(a+b);

}



extern "C" double __cdecl FileTest(double a, char *t ) {

    double n = a + (double)atof(t);
    return n;

}

int main( int argc , char **argv )
{

  /////////////////
  //
  // Calling a Function through function pointer...
  //
  int (__stdcall *  AddFunc)(double,double) = (int (  __stdcall *) (double,double))Add;
  int c = (*AddFunc)(2,3);
  printf("%d\n",c);

  ///////////////////////////////////////////////
  //
  // Typedefed call

  typedef int (__stdcall *ADD_FUNC)(double , double );
  ADD_FUNC ac = (ADD_FUNC)Add;
  printf("%d\n", ac(10,10));  // (*ac)(10,10);

  /////////////////////////////////////////////////
  //
  //
  double (__cdecl *San )(double , char *) = FileTest;
  double nt = (*San)(10,"10");
  printf("%g\n",nt);

  ////////////////////////////////////////////
  //
  //
  typedef double (__cdecl *SAN)(double , char *);
  SAN cr = (SAN)FileTest;
  printf("%g\n",(*cr)(17,"21"));

}
```

The Following C/C++ Program shows how you can dynamically load a DLL and execute a exported function from a DLL through name resolution and function pointers.

```
//////////////////////////
// DynClient.cpp
//
//
// cl DynClient.cpp user32.lib kernel32.lib gdi32.lib
//

#include <stdio.h>
#include <windows.h>


typedef int (__stdcall *BIN_FUNC)(int , int );

int main( int argc , char **argv )
{
  HMODULE ht = (HMODULE)LoadLibrary("Test.dll");

  if ( ht == 0 || ht == INVALID_HANDLE_VALUE )
  {
      printf("Failed to Load DLL\n");
      return(-1);
  }

  ///////////////////
  //
  //   Get Proc Address does the name resolution..
  //
  BIN_FUNC addfn = (BIN_FUNC)GetProcAddress(ht,"Add");

  int nc = (*addfn)(10,10);

  printf("%d\n",nc );

  FreeLibrary(ht);



}
```

How do I call the DLL from a C# program ?

```
/////////////////////
// TestCaller.cs
// The following C# program demonstrates P/Invoke
//
// csc TestCaller.cs
//
//
using System;
using System.Runtime.InteropServices;

namespace Test
{


  class TestCaller
  {
       [DllImport("Test.dll", EntryPoint="Add")]
       static extern int Add(int a , int b );

       [DllImport("Test.dll", EntryPoint="Sub")]
       static extern int Sub(int a , int b );

       public static void Main(String [] args ) {

          int n = Add(10,10)/Sub(12,10);
          Console.WriteLine("Value is {0} ", n );
       }

  }



}
```