Thoughts along developing myHelper

jdong

February 5, 2017

Abstract

I'm going to write this library for my own use but with a long term plan such that the generality level is no less than one of a shared library. Until 2017-02-05, the rough skeleton is thought to be a CUDA-C/C++ dynamic library compilable on both unix and windows and on windows using Excel as the interactive GUI.

Part I
Top-Down

This part starts with Excel programming, moving downwards the center of an XLL that stores CUDA-C programs.

1 Excel UI

2 Excel's C API and XLL Building

Tools: Excel SDK page on MSDN: `https://msdn.microsoft.com/en-us/library/office/bb687883.aspx` Find the files (dropbox/xl):

1. XLCALL.H(1481), XLCALL.CPP(120);
2. FRAMEWRK.H(71), FRAMEWRK.C(2090)
   (a) If made copies of those include files to the project directory, the angle bracktes should be changed to quotes.
3. MemoryManager.h(58), MemoryManager.cpp(207);
4. MemoryPool.h(34), MemoryPool.cpp(80);
5. Generic.sln(with GENERIC.C, GENERIC.H, GENERIC.DEF, RESOURCE.H).

Create dllmain.cpp

1. Defines the XLL function table Bovey et al. [2009] A close relative in VBA is the Application.MacroOptions Method `https://msdn.microsoft.com/en-us/library/office/ff838997.aspx`. An explanation of the XLL function table can be found in Bovey et al. [2009] pp1036. Following is an example code for just 1 function.

```
#define rgFuncsRows 1
static LPWSTR rgFuncs[1][11] = {
        {L"myFun",      //name: name of the C function
         L"BBB",        //type: <ret><arg1><arg2>
                        //A:Boolean(short int), B:double,
                        //D:ByteString(unsigned char*),
                        //I:short int, J:int, K:Array(FP*)
                        //P:oper*, R:xloper*
         L"myFun",      //xlName: name in Excel
         L"x,y",        //xlArgs: name of args in Excel
         L"1",          //function macro type:
                        //1=WorksheetFunction;
                        //2=XLM macro sheet function;
                        //0=hidden function macro.
         L"myCategory", // function category
         L"^+m",                //shortcut
         L"",           //help topic ID
         L"a sample function",  //description in fx box
         L"describe arg1",      //arg1 description
         L"describe arg2",      //arg2 description
        }
};
```

2. Holds DLLMain, the api entry. This is common for all DLLs. It is used to hold library initialization code. The function is explained in the following:

```
//typedef int BOOL
//#define WINAPI __stdcall
//#define APIENTRY WINAPI
//typedef void* LPVOID
//typedef unsigned long DWORD
//#define DECLARE_HANDLE(name) struct name##__{int unused;}; \
//typedef struct name##__ *name
//## is the preprocessor token concatenation operator
//DECLARE_HANDLE(HINSTANCE);
//typedef HINSTANCE HMODULE
BOOL APIENTRY DllMain(HMODULE hModule,
                DWORD ul_reason_for_call,
                LPVOID lpReserved){
        switch(ul_reason_for_call){
                case DLL_PROCESS_ATTACH:        //1
                case DLL_THREAD_ATTACH:         //2
                case DLL_THREAD_DETACH:         //3
                case DLL_PROCESS_DETACH:        //0
                        break;
        }
        return TRUE;    //#define TRUE 1
}
```

3.

Part II
Bottom-Up

This part starts with CUDA programming, moving upwards the center of a DLL that can be called by Excel's C API.

3 GPU Hardware

4 CUDA Programming

References

Bovey, Wallentin, Bullen, and Green. *Professional Excel Development - The Definitive Guide to Developing Applications using Microsoft Excel, VBA, and .NET*. 2009.