

C++ 调用dll的方法_Sakuya__的博客-CSDN博客

成就一亿技术人!

动态链接库

动态链接库英文为**DLL**，是**Dynamic Link Library**的缩写。dll是一个被其他应用程序调用的程序模块，其中封装了可以被调用的资源或函数。它是依附于EXE文件创建的的进程来执行的，不能够单独运行。每个程序都可以通过包含dll使用dll中包含的功能，这有助于避免代码重用和促进内存的有效使用。通过使用 DLL，程序可以实现模块化，由相对独立的组件组成。因为模块是彼此独立的，所以程序的加载速度更快，而且模块只在相应的功能被请求时才加载。

目录

[动态链接库](#)

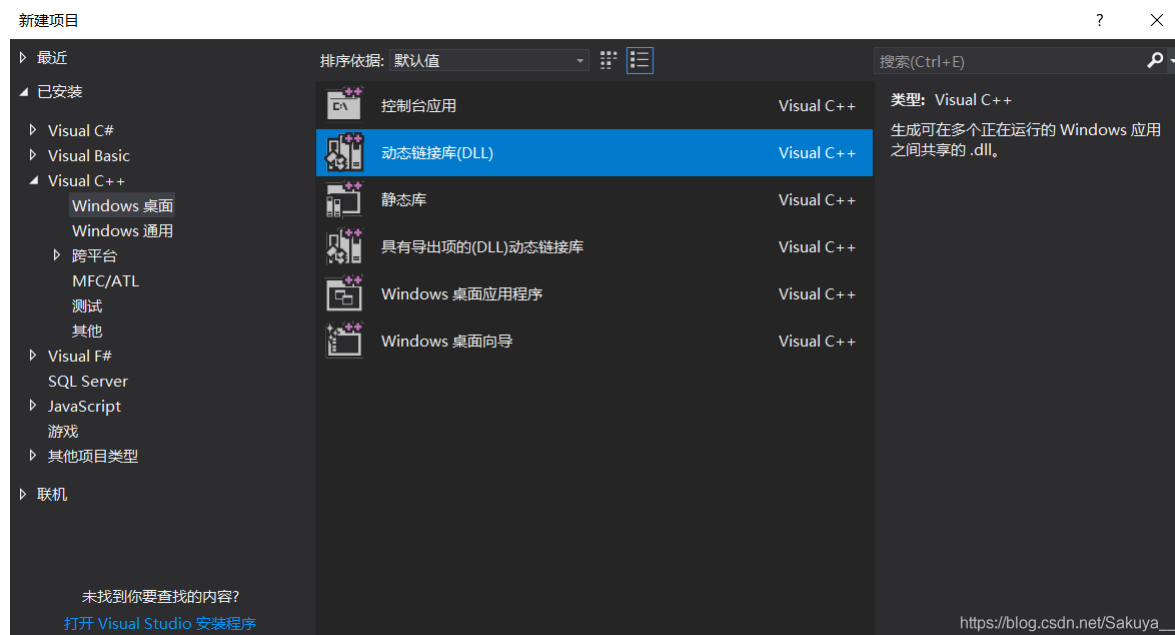
[建立一个DLL](#)

[查看DLL中的函数](#)

[调用DLL](#)

建立一个DLL

打开Visual Studio，新建->项目->Visual C++->动态链接库



使用__declspec(dllexport)关键字导出dll

添加头文件arithmetic.h 在头文件中添加导出函数add函数:

```
#ifndef __cplusplus
extern "C"{
#endif

__declspec(dllexport) int __stdcall add(int a, int b);

__declspec(dllexport) int __stdcall subtract(int a, int b);

#ifdef __cplusplus
}
#endif
```

__cplusplus是c++中的自定义宏，这个宏表示这是一段C++的代码。在C++中，为了支持重载机制，在编译生成的汇编码中，要对函数的名字进行一些处理，加入比如函数的返回类型等等而在C中，只是简单的函数名字而已，不会加入其他的信息.也就是说:C++和C对产生的函数名字的处理是不一样的。这里使用extern "C"就是告诉编译器，以C的方式来链接它括起来的函数。上面的函数的含义就是：如果这是一段C++代码，那么加入extern "C"处理大括号中的代码。

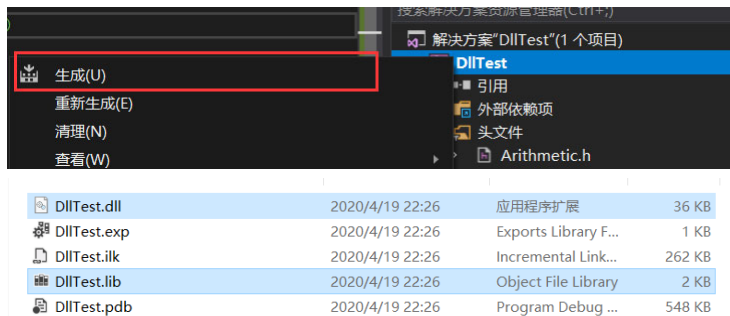
再添加arithmetic.cpp文件实现函数:

```
#include "Arithmetic.h"

int __stdcall add(int a, int b)
{
    return a + b;
}
```

```
int __stdcall subtruct(int a, int b)
{
    return a - b;
}
```

编写好之后，我们直接右键项目名，点生成就可以项目路径下的debug文件夹里找到我们生成的dll和lib文件



也可以使用模块定义文件(.def)导出dll

添加模块定义文件(.def)和__declspec(dllexport)使用一个就可以



模块定义文件(.def)是包含一个或多个描述DLL各种属性的模块语句的文本文件，DEF文件必须至少包含下列模块定义语句：（1）文件中的第一个语句必须是LIBRARY语句。（2）EXPORTS语句列出名称，可能的话还会列出DLL导出函数的序号值。通过在函数名的后面加上@符号和一个数字，给函数分配序号值。当指定序号值时，序号值的范围必须是从1到N，其中N是DLL导出函数的个数。

LIBRARY 库名

EXPORTS

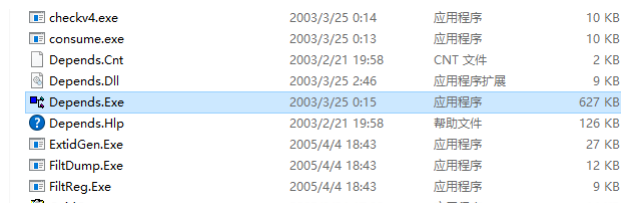
函数名1 @1

函数名2 @2

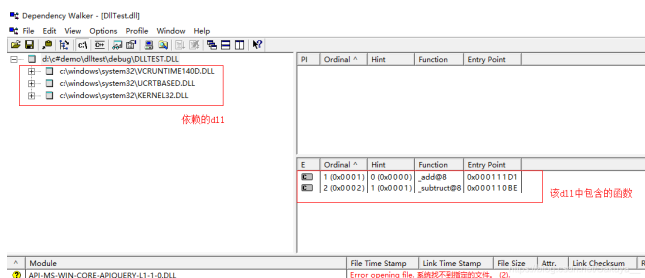
函数名n @n

查看DLL中的函数

有时候，我们需要调用已经写好的dll，但是我们不知道这个dll中有什么函数。我们可以在Visual Studio的安装目录\Common7\Tools\Bin下，找到微软提供的查看已有dll中函数名的工具Dependency Walker



我们只需要把刚才生成的dll文件拖进Dependency中，就可以看到这个dll中所包含的函数名以及它所依赖的dll



调用DLL

一、静态调用

1. 我们测试一下调用这个刚才创建的DLLTest的dll, 首先创建一个工程, 添加如下代码:

```
#include <iostream>

#pragma comment(lib, "DllTest.lib")

extern "C" __declspec(dllimport) int __stdcall add(int a, int b);
extern "C" __declspec(dllimport) int __stdcall subtruct(int a, int b);

int main()
{
    int a = 3;
    int b = 2;


    std::cout << "add result is: " << add(a, b) << std::endl;

    std::cout << "subtruct result is: " << subtruct(a, b) << std::endl;
}
```

要把生成的lib文件放到和工程代码的统一路径下, 然后将dll文件放到和程序要生成的EXE文件同一路径下:

名称	修改日期	类型	大小
 DllTest.lib	2020/4/19 22:26	Object File Library	2 KB
 Mian.cpp	2020/5/1 1:27	C++ Source file	2 KB
 Mian.vcxproj	2020/5/1 1:17	VC++ Project	8 KB
 Mian.vcxproj.filters	2020/5/1 1:17	VC++ Project Fil...	1 KB
 Mian.vcxproj.user	2020/5/1 1:06	Per-User Project...	1 KB
https://blog.csdn.net/Sakuya__			
 DllTest.dll	2020/4/19 22:26	应用程序扩展	36 KB
 Mian.exe	2020/5/1 1:27	应用程序	38 KB
 Mian.ilc	2020/5/1 1:27	Incremental Link...	341 KB
 Mian.pdb	2020/5/1 1:27	Program Debug ...	604 KB

运行程序, 可以看到调用成功:

 Microsoft Visual Studio 调试控制台

```
add result is: 5
subtruct result is: 1
```

2. 当我们要导入好几个dll, 且这些dll中有很多函数时, 可以为每个dll新建一个头文件来声明导入的函数, 如我们新建一个Arithmetic.h的头文件, 把DllTest.dll中所有的函数声明都写在这里:

```
#ifndef __cplusplus
extern "C" {
#endif

    __declspec(dllimport) int __stdcall add(int a, int b);
    __declspec(dllimport) int __stdcall subtruct(int a, int b);

#ifdef __cplusplus
}
#endif
```

然后在所有用到这个dll中函数的cpp文件中包含这个头文件, 就可以使用dll中的函数了

```
#include <iostream>
#include "Arithmetic.h"

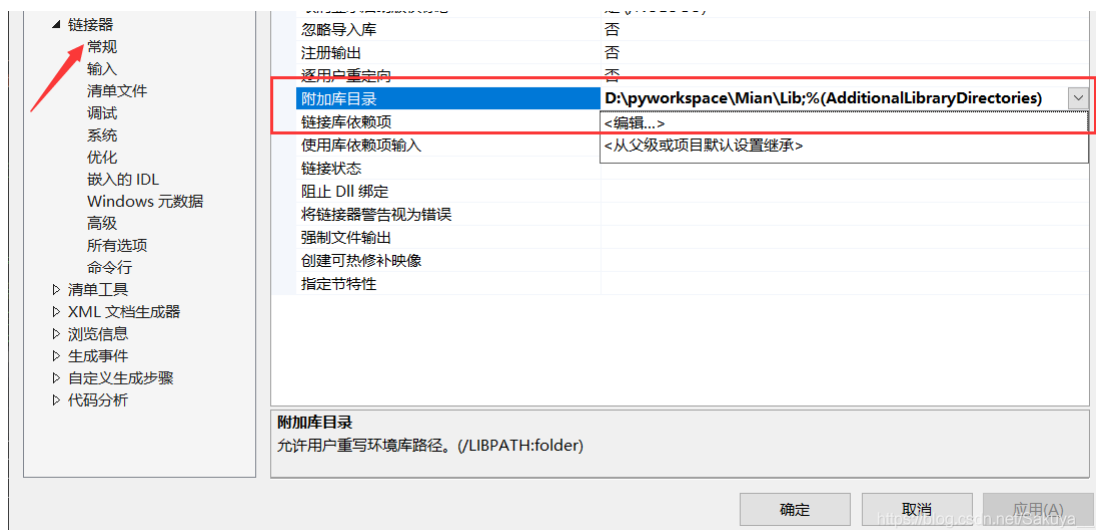
int main()
{
    int a = 3;
    int b = 2;

    std::cout << "add result is: " << add(a, b) << std::endl;

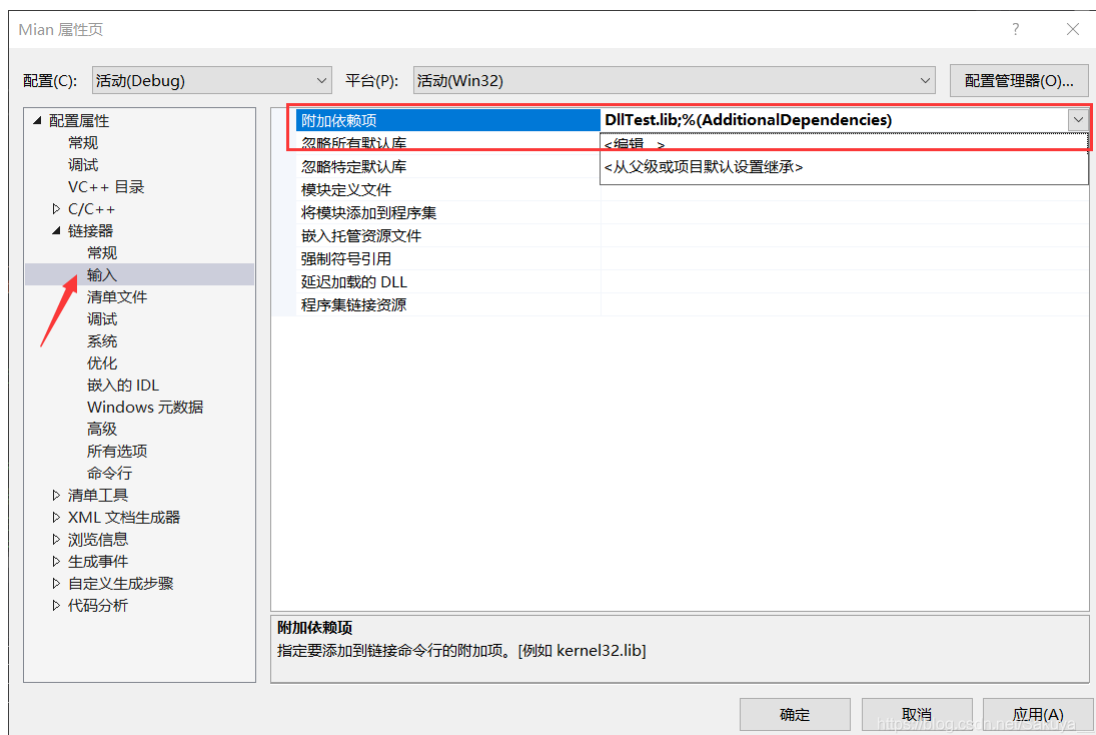
    std::cout << "subtruct result is: " << subtruct(a, b) << std::endl;
}
```

然后我们还需要添加lib库, 可以新建一个名为Lib的文件夹, 把所有的dll的lib文都放在里面。然后直接右键项目, 选择属性, 在属性配置→链接器→常规→附加库目录里添加这个Lib文件夹的路径





然后在链接器→输入→附加依赖项中添加lib文件的名字，最后一样要把dll文件放在程序要生成的EXE文件同一路径下



运行程序，可以得到和上面一样的结果。

二、动态调用

动态调用不是链接时完成的，而是在运行时完成的，动态调用不会在可执行文件中写入DLL相关的信息，而是直接调用dll中的函数。

```
void DynamicUse()
{
    HMODULE module = LoadLibraryA("DLLTest1.dll");
    if (module == NULL)
    {
        printf("加载DLLTest1.dll动态库失败\n");
        return;
    }
    typedef int(*AddFunc)(int, int);
    AddFunc add;

    add = (AddFunc)GetProcAddress(module, "add");

    int sum = add(100, 200);
    printf("动态调用, sum = %d\n", sum);
}
```



一般使用动态调用，都是调用一些系统dll中的几个函数函数时使用这种方法，因为调用我们自己写的dll时，很多函数都会经常用到，动态调用就太麻烦了。