[**.h（头文件） .lib（库文件） .dll（动态链接库文件） 之间的关系和作用的区分**](https://www.cnblogs.com/azbane/p/7364060.html)

.h头文件是编译时必须的，lib是链接时需要的，dll是运行时需要的。  
  
附加依赖项的是.lib不是.dll，若生成了DLL,则肯定也生成 LIB文件。如果要完成源代码的编译和链接，有头文件和lib就够了。如果也使动态连接的程序运行起来，有dll就够了（放在Debug文件夹里）。在开发和调试阶段，当然最好都有。  
  
.h .lib .dll三者的关系是：  
  
H文件作用是:声明函数接口  
  
DLL文件作用是: 函数可执行代码  
  
当我们在自己的程序中引用了一个H文件里的函数,编链器怎么知道该调用哪个DLL文件呢?这就是LIB文件的作用: 告诉链接器 调用的函数在哪个DLL中，函数执行代码在DLL中的什么位置，这也就是为什么需要附加依赖项 .LIB文件，它起到桥梁的作用。如果生成静态库文件，则没有DLL ，只有lib，这时函数可执行代码部分也在lib文件中  
  
目前以lib后缀的库有两种，一种为静态链接库(Static Libary，以下简称“静态库”)，另一种为动态连接库(DLL，以下简称“动态库”)的导入库(Import Libary，以下简称“导入库”）。静态库是一个或者多个obj文件的打包，所以有人干脆把从obj文件生成lib的过程称为Archive，即合并到一起。比如你链接一个静态库，如果其中有错，它会准确的找到是哪个obj有错，即静态lib只是壳子。动态库一般会有对应的导入库，方便程序静态载入动态链接库，否则你可能就需要自己LoadLibary调入DLL文件，然后再手工GetProcAddress获得对应函数了。有了导入库，你只需要链接导入库后按照头文件函数接口的声明调用函数就可以了。导入库和静态库的区别很大，他们实质是不一样的东西。静态库本身就包含了实际执行代码、符号表等等，而对于导入库而言，其实际的执行代码位于动态库中，导入库只包含了地址符号表等，确保程序找到对应函数的一些基本地址信息。  
  
一般的动态库程序有lib文件和dll文件。lib文件是必须在编译期就连接到应用程序中的，而dll文件是运行期才会被调用的。如果有dll文件，那么对应的lib文件一般是一些索引信息，具体的实现在dll文件中。如果只有lib文件，那么这个lib文件是静态编译出来的，索引和实现都在其中。静态编译的lib文件有好处：给用户安装时就不需要再挂动态库了。但也有缺点，就是导致应用程序比较大，而且失去了动态库的灵活性，在版本升级时，同时要发布新的应用程序才行。在动态库的情况下，有两个文件，而一个是引入库（.LIB）文件，一个是DLL文件，引入库文件包含被DLL导出的函数的名称和位置，DLL包含实际的函数和数据，应用程序使用LIB文件链接到所需要使用的DLL文件，库中的函数和数据并不复制到可执行文件中，因此在应用程序的可执行文件中，存放的不是被调用的函数代码，而是DLL中所要调用的函数的内存地址，这样当一个或多个应用程序运行是再把程序代码和被调用的函数代码链接起来，从而节省了内存资源。从上面的说明可以看出，DLL和.LIB文件必须随应用程序一起发行，否则应用程序将会产生错误。  
  
-------------------------------------------------------------------------------------  
静态链接库(Lib)与动态链接库(DLL)的区别   
  
     静态连接库就是把(lib)文件中用到的函数代码直接链接进目标程序，程序运行的时候不再需要其它的库文件；动态链接就是把调用的函数所在文件模块（DLL）和调用函数在文件中的位置等信息链接进目标程序，程序运行的时候再从DLL中寻找相应函数代码，因此需要相应DLL文件的支持。  
  
静态链接库与动态链接库都是共享代码的方式，如果采用静态链接库，则无论你愿不愿意，lib 中的指令都全部被直接包含在最终生成的 EXE 文件中了。但是若使用 DLL，该 DLL 不必被包含在最终 EXE 文件中，EXE 文件执行时可以“动态”地引用和卸载这个与 EXE 独立的 DLL 文件。静态链接库和动态链接库的另外一个区别在于静态链接库中不能再包含其他的动态链接库或者静态库，而在动态链接库中还可以再包含其他的动态或静态链接库。  
  
  
“每一个lib文件就是若干函数（假设只有函数）的定义”   
lib库有两种，一种是包含了函数所在DLL文件和文件中函数位置的信息，称为导出库；一种是包含函数代码本身，一般现有的DLL，用的是前一种库；以前在DOS下的TC/BC等，是后一种库。包含函数原型声明的，是头文件（.h）。   
  
“通过#include包含这些函数声明的头文件后，我们的应用程序就可以使用lib文件中的函数”  
  
还要指定编译器链接相应的库文件。在IDE环境下，一般是一次指定所有用到的库文件，编译器自己寻找每个模块需要的库；在命令行编译环境下，需要指定每个模块调用的库。   
  
“那他和直接给出那个函数定义的文件，比如.cpp文件，和头文件有什么区别，静态链接库有什么用”   
cpp文件是源代码，库文件是编译后的二进制代码，比如你可以调用Windows的API，但是不能看到其源代码一样。   
  
“还有不明白的是，静态链接库中的lib文件只要用到，则整个lib文件的内容都放进了exe文件中，那它是被编译进去还是链接的时候连接进去的呢?”   
是在链接的时候将lib链接到目标代码中。  
  
静态链接库(Lib)  
在VC++6.0中new一个名称为libTest的static library工程，  
  
并新建lib.h和lib.cpp两个文件，lib.h和lib.cpp的源代码如下：  
  
//文件：lib.h  
#ifndef LIB\_H  
#define LIB\_H  
extern "C" int add(int x,int y);　　　//声明为C编译、连接方式的外部函数  
#endif  
  
//文件：lib.cpp  
#include "lib.h"  
int add(int x,int y)  
{  
return x + y;  
}  
  
  
　　编译这个工程就得到了一个.lib文件，这个文件就是一个函数库，它提供了add的功能。将头文件和.lib文件提交给用户后，用户就可以直接使用其中的add函数了。  
  
　　标准Turbo C2.0中的C库函数（我们用来的scanf、printf、memcpy、strcpy等）就来自这种静态库。  
  
下面来看看怎么使用这个库，在libTest工程所在的工作区内new一个libCall工程。libCall工程仅包含一个main.cpp文件，它演示了静态链接库的调用方法，其源代码如下：  
  
#include <stdio.h>  
#include "..\lib.h"//不可丢失  
#pragma comment( lib, "..\\debug\\libTest.lib" ) 　//指定与静态库一起连接  
int main(int argc, char\* argv[])  
{  
     printf( "2 + 3 = %d", add( 2, 3 ) );  
}  
　　静态链接库的调用就是这么简单，或许我们每天都在用，可是我们没有明白这个概念。代码中#pragma comment( lib , "..\\debug\\libTest.lib" )的意思是指本文件生成的.obj文件应与libTest.lib一起连接  
  
-------------------------------------------------------------------------------------------  
用VC++生成静态库文件  
今天闲着没事做,自己写了一点小笔记,不知道对于新手有没用,高手就不用看了,作为新手的我斗胆来发表一个笔记,就是静态库文件的封装过程,使用VC++6.0编写,下面是正文,也许我的用语并不专业  
  
以前我们写C/C++源文件的时候，都是先将各个写好的源文件编译，编译生成的是目标文件机器码，即.obj文件.（目标文件的扩展名不一定是.obj文件）.  
  
我们调用的标准C/C++函数机器码实际被封装于标准C/C++静态库文件中的.即那些扩展名为.lib的文件中.  
  
最后链接器将我们编译的各个目标文件里的机器码和静态库（标准C/C++库）中的函数机器码链接到一起形成一个扩展名为.exe的可执行文件模块.  
  
在这里我们叙述将C/C++源文件编译链接成一个静态库文件，但它不是可执行模块，它体内含有可执行机器码  
  
静态库文件就像一个仓库或者容器，里面封装了一些可执行机器码.这些机器码是我们用程序设计语言，比如C/C++源文件编译后生成的机器码.  
  
一.下面将讨论将C/C++源文件编译并链接成一个静态库文件的过程，  
  
在VC++6.0中选择File-New-Win32 Static Library，写好工程名创建好工作空间后再选择菜单中New-File来为工程添加C或者C++ 源文件.  
  
假如我们为该工程添加了一个名为lib\_c.c和一个名为lib\_cpp.cpp的源文件  
  
//lib\_c.c中的内容  
  
extern int Add(int x,int y) //该函数是一个外部函数，任何文件都可以访问它  
  
{  
　　　　return x+y;  
  
}  
  
extern int data\_c  
//这是一个外部全局变量，任何文件可以访问它  
  
//lib\_cpp.cpp中的内容  
  
extern “C” int  
        reduce(int x,int y)//这里加了个”C”表示允许C源文件访问这个C++函数代码  
  
{  
　　　　return x-y;  
  
}  
  
extern “C” int data\_cpp=2;  
  
注意以下几点  
  
(1)当“extern”关键字修饰在函数或全局变量的定义中时，表示该函数或全局变量任何文件可以访问，“extern”关键字可以省略不写，缺省下就是”extern”  
  
  当“extern”关键字修饰在函数声明或全局变量声明中时，表示限定当前文件只能引用用“extern”关键字修饰定义的函数或全局变量.  
  
(2)当”static”关键字修饰在函数或全局变量的定义中时，表示该函数或全局变量只能由本文件中加了”static”关键字修饰的函数声明或全局变量声明来引用.  
  
  当”static”关键字修饰在函数声明或全局变量声明中时，表示限定当前文件只能引用用“static”关键字修饰定义的函数或全局变量.  
  
(3)在CPP源文件的函数和全局变量定义中加了个”C”表示允许C源文件访问该函数和全局变量.如果是C++源文件访它们的话则可加可不加.注意这”C”要大写.  
  
接下来就要将写好的C/C++源文件进行编译和链接，最后会生成一个扩展名为.lib的文件.该文件就是静态库文件了，该静态库文件是不能直接运行的，我们所编译的C/C++源文件的机器码就已经被封装进这个用VC++6.0创建的静态库文件里面去了.  
  
二.如何将编写好的静态库文件像使用C/C++标准库那样使用，下面将继续讨论  
  
1.用VC++6.0新建一个工程名为TEST，添加一个名为TEST.c的源文件到该工程，因为我们将测试一下，将我们编写的库文件里的函数或者全局变量的机器码链接到我们这个TEST.c源文件中去，假设我们生成的库文件名为TEST.lib,先拷贝如下范例代码到TEST.c中  
  
//TEST.c  
  
#include <stdio.h>  
  
extern int   
Add(int x,int y); //当前文件只能访问“extern”关键字修饰定义的Add函数  
  
extern int  
reduce(int x,int y);// //当前文件只能访问“extern”关键字修饰定义的reduce函数  
  
#pragma comment(lib,"TEST.lib") //指示链接器到字符串所表示的文件路径中去找库文件  
  
int main()  
  
{  
　　　　printf("%d\n",Add(2,3));  
　　　　printf("%d\n",reduce(3,2));  
　　　　return 0;  
  
}  
  
这里我们要声明静态库中已知的函数或全局变量的声明  
  
#pragma comment(lib,"TEST.lib")这条指令告诉链接器到字符串所表示的路径下去找库文件，这里我将库文件放到了当前工程目录下.也可以不写这句.  
  
还有一种方法，可以直接在VC++6.0中设置依次选择tools、options、directories、library files菜单或选项，填入库文件路径（只键入库文件所在目录路径而不能输入库文件名），这只是告诉链接器库文件所在目录的路径，还没告诉链接器库文件名，方法是VC++6.0中设置依次选择project-settings-link 在object/library modules: 这栏输入库文件名字然后就OK了  
  
2.当用C++源文件的目标文件和库文件的代码链接时有一点小改变，这里就不浪费口舌了，假设我们新建了一个工程并添加了一个名为TEST.CPP的源文件，拷贝如下范例代码到TEST.CPP中  
  
//TEST.cpp  
  
#include <stdio.h>  
  
extern “C” int   
       Add(int x,int y); //表示引用的是C函数代码  
  
extern int  
      reduce(int x,int y);  
  
#pragma comment(lib,"TEST.lib")  
  
int main()  
  
{  
　　　　printf("%d\n",Add(2,3));  
　　　　printf("%d\n",reduce(3,2));  
　　　　return 0;  
  
}  
  
在这个C++源文件里引用C函数代码同样要加个”C”,但是在C源文件引用C++函数代码不能加”C++”,编译会报错，只能在C++文件函数定义中加”C”.  
  
只有C++才支持这种引用方式，也许因为只有C++兼容C而没有C兼容C++这一原则.

.h用于编译阶段的审核，如在math.h中有函数声明：  
int abs(int);  
但是在使用中写为  
#include <math.h>  
...abs(3,5);  
编译器阶段就会检测出错误。  
  
.dll用于运行阶段，如调用SetWindowText()函数等，需要在user32.dll中找到该函数。DLL可以简单认为是一种包含供别人调用的函数和资源的可执行文件。  
  
.lib用于链接阶段，在链接各部分目标文件（通常为.obj）到可执行文件（通常为.exe）过程中，需要在.lib文件中查找动态调用函数（一般为DLL中的函数）的地址信息，此时需要在lib文件中查找，如查找SetWindowText()函数的地址偏移就需要查找user32.lib文件。（.lib也可用于静态链接的内嵌代码）  
  
   
  
lib和dll文件的区别和联系      
     
  .dll是在你的程序运行的时候才连接的文件，因此它是一种比较小的可执行文件格式，.dll还有其他的文件格式如.ocx等，所有的.dll文件都是可执行。    
     
  .lib是在你的程序编译连接的时候就连接的文件，因此你必须告知编译器连接的lib文件在那里。一般来说，与动态连接文件相对比，lib文件也被称为是静态连接库。当你把代码编译成这几种格式的文件时，在以后他们就不可能再被更改。如果你想使用lib文件，就必须：    
  1?   包含一个对应的头文件告知编译器lib文件里面的具体内容    
  2?   设置lib文件允许编译器去查找已经编译好的二进制代码    
     
  如果你想从你的代码分离一个dll文件出来代替静态连接库，仍然需要一个lib文件。这个lib文件将被连接到程序告诉操作系统在运行的时候你想用到什么dll文件，一般情况下，lib文件里有相应的dll文件的名字和一个指明dll输出函数入口的顺序表。如果不想用lib文件或者是没有lib文件，可以用WIN32   API函数LoadLibrary、GetProcAddress。事实上，我们可以在Visual   C++   IDE中以二进制形式打开lib文件，大多情况下会看到ASCII码格式的C++函数或一些重载操作的函数名字。    
     
  一般我们最主要的关于lib文件的麻烦就是出现unresolved   symble   这类错误，这就是lib文件连接错误或者没有包含.c、.cpp文件到工程里，关键是如果在C++工程里用了C语言写的lib文件，就必需要这样包含：    
  extern   "C"    
  {    
  #include   "myheader.h"    
  }    
  这是因为C语言写的lib文件没有C++所必须的名字破坏，C函数不能被重载，因此连接器会出错。  
  
   
  
C语言中有一些函数不需要进行编译，有一些函数也可以在多个文件中使用。一般来说，这些函数都会执行一些标准任务，如数据库输入/输出操作或屏幕控制等。可以事先对这些函数进行编译，然后将它们放置在一些特殊的目标代码文件中，这些目标代码文件就称为库。库文件中的函数可以通过连接程序与应用程序进行连接。这样就不必在每次开发程序时都对这些通用的函数进行编译了。      
     
  　　不同类型的应用程序将会使用不同的函数库。例如：libdbm库中组包含了对数据库文件进行访问的dbm函数，需要对数据库进行操作的程序就会与该库进行连接。数学应用程序将使用数学库libm，X-Windows应用程序将使用Xlib库，libX11。另外，所有的程序都将使用标准的C函数库。libc，该库中包含了诸好内存管理或输入输出操作的基本函数，这些库都存放在/usr/lib这些系统公用的目录中，系统中的任何用户都可以利用这些库。当然用户也可以建立自己专用的库函数，供自己或其它指定的人员使用。      
     
  　　库可以有三种使用的形式：静态、共享和动态。静态库的代码在编译时就已连接到开发人员开发的应用程序中，而共享库只是在程序开始运行时才载入，在编译时，只是简单地指定需要使用的库函数。动态库则是共享库的另一种变化形式。动态库也是在程序运行时载入，但与共享库不同的是，使用的库函数不是在程序运行开始，而是在程序中的语句需要使用该函数时才载入。动态库可以在程序运行期间释放动态库所占用的内存，腾出空间供其它程序使用。由于共享库和动态库并没有在程序中包括库函数的内容，只是包含了对库函数的引用，因此代码的规模比较小。  
  
   
  
lib是静态库，dll一般是动态链接库（也有可能是别的）  
  
比如要编译个exe，lib在编译的时候就会被编译到exe里，作为程序的一部分  
  
而dll是不被编译进去，是运行的时候才调入的（可能是exe刚运行就调入，也可能运行了一半才调入）  
  
用法，lib需要个.lib文件和一个.h文件，程序正常使用.h的函数，在链接选项里加入.lib文件就ok  
  
dll用法有2种，一是 .h + .lib + .dll的，用法和前面一样，中间的lib是个中转，运行的时候会调用dll  
二是：直接用dll，需要知道dll的函数定义，用LoadLibrary和GetProcAddress把函数指针取出来，看msdn的例子吧