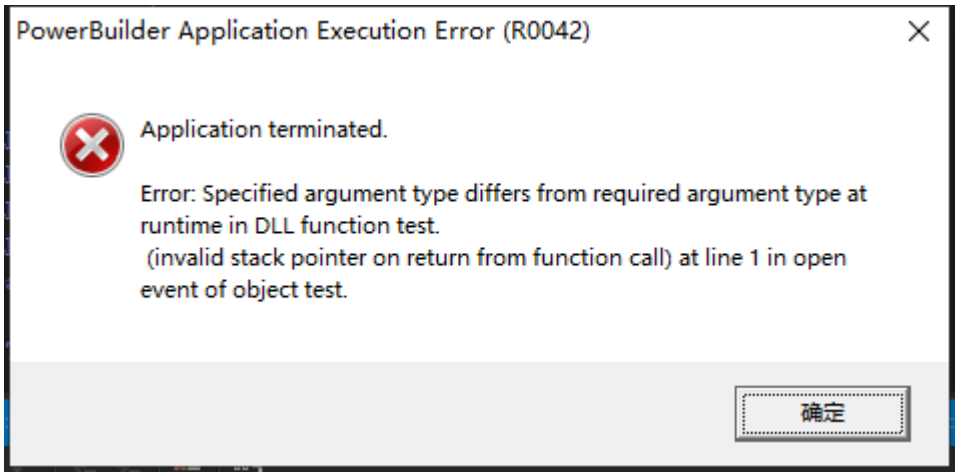
PB 调用 DLL 的常见问题及处理方法

首先 ,为方便描述 ,先假设有一个 DLL 文件 ,名称为 test.dll ,里面有个函数叫 test。

第一类：通用型标准 DLL

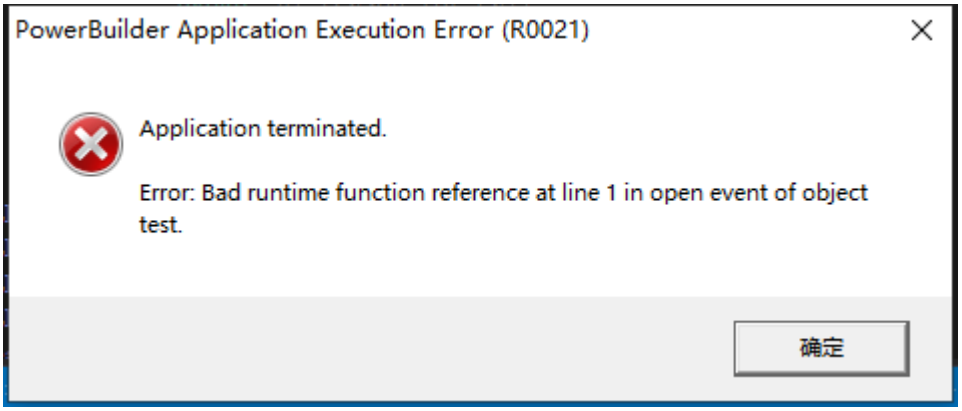
1、调用约定问题。Windows 系统的标准 DLL，通常有 2 种调用约定，即__cdecl 和__stdcall，__stdcall 约定在.h 文件中通常又定义为 WINAPI 和 CALLBACK。咱们的 PB 只能调用__stdcall 约定的 DLL，不能调用__cdecl 约定的 DLL。



如果调用了__cdecl 约定的函数 ,会报以上错误。关键是 “specified argument type deffers from …. “，内容是给定的参数顺序与要求的不一样。

对于这两种约定的区别，有兴趣了解更多的，可以自行百度。

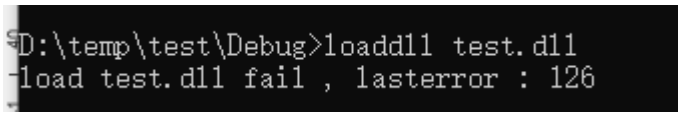
2、DLL 依赖问题。有时因为 test.dll 还需要依赖其他 DLL，导致调用失败。



依赖的 DLL 找不到时，通常会报类似这样的错误，然而这个提示并不明确，所以比较难查。在群共享里可以下载 loaddll.exe 这个小工具


LoadDLL.rar

,使用也比较简单，和 DLL 放在同一个目录，然后进入控制台命令行，输入 loaddll test.dll，观察控制台输出



如果是依赖问题，这里的错误号是 126。可以通过百度搜索 “getlasterror 126” 查找具体情况，如果是其错误号，搜索时使用其他错误号。

3、DLL 路径搜索错误。这个错误通常常是 DLL 不在应用程序当前目录下造成的，错误截图同上。也许你会说“我肯定我的 DLL 就在程序目录下 “，但请注意是” 当前目录 “，当前目录并不是只指应用程序目录，虽然初始时默认是应用程序目录。假设你的程序在 d:\test 目录，你使用 GetOpenFileName 函数选择了 d:\document\work.txt，这时候，你的当前目录就是 d:\document\，而这时的当前目录下当然找不到 test.dll 了，你的 test.dll 在 d:\test 呢。这是许多 PB 程序员容易忽略的原因。解决方法通常有以下几种：

（1）在调用 GetOpenFileName GetSaveFileName 等函数前，先 GetCurrentDirector 函数保存当前目录，完成后，再 ChangeDiredtory 切换回原当前目录。

（2）假设你的程序在 d:\test 目录，你可以直接设置系统的 PATH 环境变量指向这

个目录。当然这个给实施人员带来一些小不便。

（3）程序初始化时，使用系统函数 `BOOL SetEnvironmentVariable(LPCTSTR lpName,LPCTSTR lpValue)`把程序所在目录添加到进程环境变量里去，让 DLL 可以被搜索到。一个更加智能的方案是使用 pbidea 提供的设置函数

```
//!!!!!!!!!!!!!!!!!!!!! AddAppPathToEnv 函数可以解决经常报各种找不到DLL的错误
//将当前目录及其子目录，一起添加到PATH环境变量中去，避免因改变目录什么的导致DLL找不到
uo_file file
file = create uo_file
file.AddAppPathToEnv(true,"..\sharedll","c:\windows","c:\windows\system32")
destroy file
```

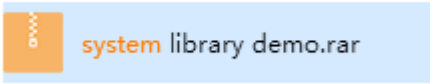
可以确保当前目录、当前目录下的子目录，以及指定的其他目录里存放的 DLL 都可以被 搜索到。

特别值得一提的是，使用 oracle 数据库时，它的客户端在连接时会改变当前路径。

4、各种函数声明错误。这种错误就比较复杂了，涉及面也比较广。

可以参考另一篇博客 https://blog.csdn.net/lxbin2003/article/details/121527801，提升对 DLL 及函数的理解 ，从而正确声明和使用函数 。

特别注意，一般情况下声明函数时 library “test.dll” ，而不是 system library “test.dll” ，这里的”system”不是随便加的。只有按照 PB 特定写法的 DLL 才可以添加这个 system 关键字。如果对 C/C++写 PB 可调用的 DLL 有兴趣的话，可以到群共享下载



，看看 system library 方式是如何实现的。

5、函数使用错误。DLL 函数都是按照 C 约定开放接口。我们知道 C 的指针，总是让人不得不打起十二分精神来对待。你认为 PB 里不涉及到指针，你就错了。比如 test 函数

function int test(string a1,ref string a2,long a3) library "test.dll"

这个函数的 C 原型是这样的：

```
extern "C" __declspec(dllexport) int __stdcall test(char* a1, char* a2, int a3)
```

PB 里的 string，对应到 C 的原因，实际上就是指针。如果是输入型参数倒也没太大问题，如果是输出型参数，那就必须要预分配足够的内存空间。通常 PB 里预分配空间，使用 space 函数。

String ls_a2

ls_a2 = space(10000)

这样 ls_a2 这个变量就有 10000 个字符的空间，注意，如果 是 PB10 以上版本，它实际上有了 20000 个字符的空间，因为一个 unicode 字符占用 2 个 char 空间。

如果没预分配内存，或者预分配内存大小不够，程序也许能运行一小会，但后面不知道会在什么位置莫名其妙崩溃。

6、DLL 开发者错误。许多人临时需要某个功能，操起 VC 就开始写 DLL，但也因为对 c/c++掌握程度问题，会导致一些 BUG，这些只能 PB 运行不正常时，自己排查规律，如果确定是 DLL 的原因，得找 DLL 开发人员处理了。

第二类：COM 类 DLL

曾经 COM 是非常热门的一个技术，它也是微软应用与操作系统捆绑的一个典范。然而，在 PB 中，其实并不建议使用 COM 方式的 DLL。

目前 COM 方式的 DLL，一个是历史遗留下来的古老应用，二是有些人图方便使用 C#写 DLL 给 PB 用，三是有些 PB 版本（比如 PB9）自己也支持写 COM 给其他应用使用。

COM 方式的 DLL，它首先需要注册，这个比较麻烦，在 win7 及以前版本中，还可以使用 run 方式进行自注册，但在 WIN10 及其后版本中，自注册就比较困难了，因为注册时需要管理员权限。一两台电脑注册 COM 可以接受，如果类似 HIS 这样的应用，一个医院动辄布署几百台电脑，需要一个一个去注册 COM，那就呵呵了。尤其是 C#写的 COM，

在不同操作系统上，可能依赖的.net framework 还不一样，还需要安装这些额外的内容。

所以，建议 PB 尽量远离 COM 类的 DLL。

第三类：PB 自身的 DLL

上面刚才提到 PB 有些本可以写 COM。另外，PB 自己的 PBL 也可以编译成 DLL。如果是 PB 编译出来的 DLL 库，可以直接当作 PBL 一样添加到库列表中使用。如果给其他人提供 PB 的业务接口，这也不失为一个方案。只不过它有个限制：只能是同一个 PB 版本使用。

欢迎加入 pbidea 群，共同探讨 PB 技术。群号：624409252

大自在(QQ:781770213)

2022/5/8