Windows下DLL编程技术及应用 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/252/2021_2022_Windows_E4_B8_8B_c104_252472.htm 一、引言 由于Windows为微机提供了前所未有的标准用户界面、图形处理能力和简单灵便的操作，绝大多数程序编制人员都已转向或正在转向Windows编程。在许多用户设计的实际应用系统的编程任务中，常常要实现软件对硬件资源和内存资源的访问，例如端口I/O、DMA、中断、直接内存访问等等。若是编制DOS程序，这是轻而易举的事情，但要是编制Windows程序，尤其是WindowsNT环境下的程序，就会显得较困难。因为Windows具有"与设备无关"的特性，不提倡与机器底层的东西打交道，如果直接用Windows的 API函数或I/O读写指令进行访问和操作，程序运行时往往就会产生保护模式错误甚至死机，更严重的情况会导致系统崩溃。那么在Windows下怎样方便地解决上述问题呢？用DLL(Dynamic Link Libraries)技术就是良好途径之一。DLL是Windows最重要的组成要素，Windows中的许多新功能、新特性都是通过DLL来实现的，因此掌握它、应用它是非常重要的。其实 Windows本身就是由许多的DLL组成的，它最基本的三大组成模块Kernel、GDI和User 都是DLL，它所有的库模块也都设计成DLL。凡是以.DLL、.DRV、.FON、.SYS和许多以.EXE为扩展名的系统文件都是DLL，要是打开Windows\System目录，就可以看到许多的DLL模块。尽管DLL在Ring3优先级下运行，仍是实现硬件接口的简便途径。DLL可以有自己的数据段，但没有自己的堆栈，使用与调用它的应用程序相同的堆栈模式，减少了编

程设计上的不便；同时，一个DLL在内存中只有一个实例，使之能高效经济地使用内存；DLL实现的代码封装性，使得程序简洁明晰；此外还有一个最大的特点，即DLL的编制与具体的编程语言及编译器无关，只要遵守DLL的开发规范和编程策略，并安排正确的调用接口，不管用何种编程语言编制的DLL都具有通用性。例如在BC31中编制的DLL程序，可用于BC、VC、VB、Delphi等多种语言环境中。笔者在BC31环境下编译了Windows下直接内存访问和端口I/O两个DLL，用在多个自制系统的应用软件中，运行良好。 二、DLL的建立和调用 DLL的建立及调用方法在许多资料上有详细的介绍，为了节省篇幅，在这里仅作一些主要的概括。 1.DLL的建立 关于DLL的建立，有如下几个方面的要素是不可缺少和必须掌握的： 入口函数LibMain( ) 就象C程序中的WinMain( )一样，Windows每次加载DLL时都要执行LibMain( )函数，主要用来进行一些初始化工作。通常的形式是： int FAR PASCAL LibMain(HINSTANCE hInstance,WORD wDataSeg,WORD wHeapSize,LPSTR lpszCmdLine) { if(wHeapSize!=0) //使局部堆、数据段可移动 UnlockData(0). //解锁数据段 /*此处可进行一些用户必要的初始化工作*/ return 1. //初始化成功 } 出口函数WEP( ) Windows从内存中卸载DLL时，调用相应的出口函数WEP( )，主要做一些清理工作，如释放占用的内存资源；丢弃某些字串、位图等资源；关闭打开的文件等等。 自定义的输出函数 为了让位于不同内存段的应用程序进行远程调用，自定义的输出函数必须定义为远程函数(使用FAR关键字)，以防使用近程指针而得到意外的结果；同时，加上PASCAL关键字可加快程序的运行速度，使代码简单高效，提高程序的

运行速度。 输出函数的引出方法 在DLL的模块定义文件中(.DEF)由EXPORTS语句对输出函数逐一列出。例如：EXPORTS WEP @1 residentname //residentname可提高DLL效率和处理速度 PortIn @2 PortOut @3 //通常对所有输出函数附加系列号 在每个输出函数定义的说明中使用_export关键字来对其引出。 以上两种方法任选其中的一种即可，不可重复。后面的两个实例分别使用了上述两种不同的引出方式，请留意。 2.DLL的调用 加载DLL时，Windows寻找相应DLL的次序如下： 当前工作盘。 Windows目录；GetWindowsDirectory( )函数可提供该目录的路径名。 Windows系统目录，即System子目录；调用GetSystemDiretory( )函数可获得这个目录的路径名。 DOS的PATH命令中罗列的所有目录。 网络中映象的目录列表中的全部目录。 DLL模块中输出函数的调用方法： 不论使用何种语言对编译好的DLL进行调用时，基本上都有两种调用方式，即静态调用方式和动态调用方式。静态调用方式由编译系统完成对DLL的加载和应用程序结束时DLL卸载的编码（如还有其它程序使用该DLL，则Windows对DLL的应用记录减1，直到所有相关程序都结束对该DLL的使用时才释放它），简单实用，但不够灵活，只能满足一般要求。动态调用方式是由编程者用API函数加载和卸载DLL来达到调用DLL的目的，使用上较复杂，但能更加有效地使用内存，是编制大型应用程序时的重要方式。具体来说，可用如下的方法调用： 在应用程序模块定义文件中，用IMPORTS语句列出所要调用DLL的函数名。如： IMPORTS MEMORYDLL.MemoryRead MEMORYDLL.MemoryWrite 让应用程序运行时与DLL模块动态链接 先用LoadLibrary加载DLL，再用GetProcAddress函数检

取其输出函数的地址，获得其指针来调用。如： HANDLE hLibrary. FARPROC lpFunc. int PortValue. M hLibrary=LoadLibrary("PORTDLL.DLL"). //加载DLL if(hLibrary>31) //加载成功 { lpFunc=GetProcAddress(hLibrary,"PortIn"). //检取PortIn函数地址 if(lpFunc!=(FARPROC)NULL) //检取成功则调用 PortValue=(*lpFunc)(port). //读port端口的值 FreeLibrary(hLibrary). //释放占用的内存 } M