系统技巧:WinMain函数Microsoft认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/623/2021_2022__E7_B3_BB_E7_BB_9F_E6_8A_80_E5_c100_623973.htm WinMain函数的原型声明如下： int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine,int nCmdShow). 参数: WinMain函数接收4个参数，这些参数都是在系统调用WinMain函数时，传递给应用程序的。 hInstance: 表示该程序当前运行的实例句柄，是一个数值标识。当程序在Windows下运行时，它唯一标识运行中的实例（注意，只有运行中的程序实例，才有实例句柄）。一个应用程序可以运行多个实例，每运行一个实例，系统都会给该实例分配一个实例句柄，并通过hInstance参数传递给WinMain函数。 hPrevInstance: 表示当前实例的前一个实例的句柄。通过查看MSDN我们可以知道，在Win32环境下，这个参数总是NULL，即在Win32环境下，这个参数不再起作用。 lpCmdLine: 是一个以空字符结尾的字符串，内容为命令行的参数。 nCmdShow： 指定程序的窗口应该如何显示，例如最大化、最小化、隐藏等。这个参数的值由该程序的调用者所指定，应用程序通常不需要去理会这个参数的值。 WinMain函数前的修饰符WINAPI，其实就是__stdcall。 DllMain函数 DllMain只是在Windows系统里注册的一个回调函数（call back） 早期的SDK版本中，DllMain是叫做DllEntryPoint DllMain是Dll 的缺省入口函数，DLLMain 负责Dll装载时的初始化及卸载的收尾工作，每当一个新的进程或者该进程的新的线程访问 DLL 或访问DLL 的每一个进程或线程不再使用DLL时，

都会调用 DLLMain。 使用 TerminateProcess 或 TerminateThread 结束进程或者线程，不会调用DLLMain。 有些DLL并没有提供DllMain函数，也能成功引用DLL，这是因为Windows在找不到DllMain的时候，系统会引入一个缺省DllMain函数版本。 缺省的DllMain函数在 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs 键值中。 DllMain 函数的原型 BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved). BOOL APIENTRY DLLMain(HANDLE hModule,DWORD ul_reason_for_call,LPVOID lpReserved) { switch(ul_reason_for_call) { case DLL_PROCESS_ATTACH: ..... case DLL_THREAD_ATTACH: ..... case DLL_THREAD_DETACH: ..... case DLL_PROCESS_DETACH: ..... } return TRUE. } 参数： hModule： 是动态库被调用时所传递来的一个指向自己的句柄(实际上，它是指向_DGROUP 段的一个选择符)； ul_reason_for_call： 是一个说明动态库被调原因的标志。当进程或线程装入或卸载动态连接库的时候，操作系统调用入口函数，并说明动态连接 DLL_PROCESS_ATTACH 进程被调用 DLL_THREAD_ATTACH 线程被调用 DLL_PROCESS_DETACH 进程被停止 DLL_THREAD_DETACH 线程被停止 lpReserved：是一个被系统所保留的参数； DLL_PROCESS_ATTACH 每个进程第一次调用DLL文件被映射到进程的地址空间时，传递的fdwReason参数为DLL_PROCESS_ATTACH。这进程再次调用操作系统

只会增加DLL的使用次数。 DLL_THREAD_ATTACH 进程中的每次建立线程，都会用值DLL_THREAD_ATTACH调用DllMain函数，哪怕是线程中建立线程也一样。 DLL_PROCESS_DETACH 当DLL被从进程的地址空间解除映射时，系统调用了它的DllMain，传递的fdwReason值是DLL_PROCESS_DETACH。 ◆FreeLibrary解除DLL映射（有几个LoadLibrary，就要有几个FreeLibrary） ◆进程结束而解除DLL映射，在进程结束前还没有解除DLL的映射，进程结束后会解除DLL映射。 用TerminateProcess终结进程，系统就不会用DLL_PROCESS_DETACH来调用DLL的DllMain函数。 注意：当用DLL_PROCESS_ATTACH调用DLL的DllMain函数时，如果返回FALSE，说明没有初始化成功，系统仍会用DLL_PROCESS_DETACH调用DLL的DllMain函数。 DLL_THREAD_DETACH 线程调用ExitThread来结束线程（线程函数返回时，系统也会自动调用ExitThread），用DLL_THREAD_DETACH来调用DllMain函数。 注意：用TerminateThread来结束线程，系统就不会用值DLL_THREAD_DETACH来调DLL的DllMain函数。 早期的SDK版本中，DllMain是叫做DllEntryPoint。其实Dll的入口函数名是可以自己定义的。 ServiceMain函数 当一个服务控制程序请求开启一个新的服务时，SCM（服务控制管理器）开启服务的同时，向控制调度器发送一个开始请求。控制调度器为服务创建一个新的线程来运行ServiceMain函数。 ServiceMain函数应该执行下面的这些任务： 1、立刻为服务调用RegisterServiceCtrlHandlerEx函数，用来注册一个处理控制请求的HandlerEx函数。 RegisterServiceCtrlHandlerEx函数的返

回值是一个service status handle（服务状态句柄），稍后向SCM通知当前服务状态会用到。 2、执行初始化。如果初始化的时间非常短（小于一秒），可以直接在ServiceMain函数中完成。 如果预期初始化的时间会比一秒长，调用SetServiceStatus函数，在SERVICE_STATUS结构中指明等待间隔，以及当前的状态是SERVICE_START_PENDING。 当初始化继续时，服务应该再调用SetServiceStatus函数来报告当前状态。多次调用SetServiceStatus函数对调试服务很有用。 3、初始化结束后，调用SetServiceStatus函数，在SERVICE_STATUS结构中指明SERVICE_RUNNING状态。 4、执行服务的任务，或者已经没有未完成的任务，那就返回。服务状态有任何改变，都要保证调用SetServiceStatus函数来报告新的状态。 5、如果在初始化或者运行的过程中遇到错误并且清理的过程会很长，服务应该调用SetServiceStatus函数，在SERVICE_STATUS结构中指定SERVICE_STOP_PENDING状态。 一旦清理结束，在最后中止的线程中调用SetServiceStatus函数，在SERVICE_STATUS结构中指定SERVICE_STOPPED状态。确保向SERVICE_STATUS结构里的 dwServiceSpecificExitCode和dwWin32ExitCode赋值，来标志错误。 更多优质资料尽在百考试题论坛 百考试题在线题库 微软认证更多详细资料 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com