用64位Windows进行开发的五点建议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_ti2020/641/2021_2022__E7_94_A864_E4_BD_8DWi_c100_641627.htm 现在到处都是64位电脑。由于我们要开发VB 应用程序，所以是时候思考对于64位的支持了。 64位运算曾是一个奇特的想法，但是现在已经成为一种现实。那些能够接受大量RAM并运行64位Windows的客户端和服务器无处不在。视频剪辑，图像，三维建模和数据密集型应用程序只是受益于64位操作系统和硬件的其中几个例子。 无论应用程序或组件是否需要64位架构提供额外可设定地址的内存，你都希望能够在64位Windows上使用应用程序。以下是我们给出的五点建议以帮助你用64位Windows开发程序。 1.避免不合格的图像例外 通常，程序员碰到这种情况：“ 出现未处理的类型异常‘ System.BadImageFormatException ’”。如果你要深究该异常的细节信息，可能会发现系统提示：“ 试图加载的程序，格式不正确 ”。 之所以会出现这样的问题是因为64位进程试图加载一个32位组件。虽然你在Windows x64上可以运行64位和32位进程，但是64位代码和32位代码不能在相同进程上运行。你的代码要么全部是64位，要么全部是32位。要加载的组件也要符合这一规律。 VS 2005与.NET 2.0为编译.NET应用程序带来选择，将输入设置为“ Any CPU ”也具备了可选性。“ Any CPU ”是默认平台。如果组件以Any CPU作为平台进行编译，那么它将依据进程加载的情况以32位或64位方式运行。使用Any CPU，相同的组件可以在64位Windows上以32位或64位方式运行：它不是真正的指定了64位的CPU或操作系统，而是一个调用进程。 为

了解决不合格图像异常的问题，要改变到Any CPU的所有组件的目标平台。如果出于某种原因你无法做到这一点或许某组件无法提供来源那么要将所有组件设置成到达同一平台，可以是x86也可以是x64。如果你拥有.NET 1.0或1.1组件，最好是用.NET 2.0对其重新进行编译。如果你不能编译.NET 1.0或1.1组件，那么编译其他代码，设置为到达x86平台，使之兼容。 必须意识到VS 2005和2008，如果你使用的是VB开发设置配置文件， 平台的更改有些小麻烦。出现的问题可能源自VB团队有关用户发现VS中的设置比VB6复杂一些的反馈信息。好的一方面是你可以对其加以控制。对建设和平台配置最简单的修复是采用“ 常规开发设置 ”而不是“ VB开发设置 ”。另外，改变工具/选项菜单中的“ 项目与解决方案 ”选项，以确保“ 显示高级构建配置 ”选项被选定。接下来，自定义工具栏，这样它就会包含“ 解决方案配置 ”和“ 解决方案平台 ”下拉式组合框。 一旦你激活这些功能，就可以从“ 解决方案配置 ”或“ 解决方案平台 ”组合列表中选取“ 配置管理器 ”。这样做会弹出配置管理器窗口，在此窗口中，你可以添加或修改配置。配置可以让你做特殊的架构，如用于测试或只迫使创建某些多项目解决方案中的项目。对于每种配置你还可以支持不同的平台。 为了从配置管理器的“ Active Solution 平台 ”列表中添加一个x86或x64平台，选择“ ”，然后选择x86或x64，从Any CPU中复制设置。稍后，你可以使用项目属性编译标签和高级编译对话，为每个项目单独编辑这些设置。如果你对构建配置还不熟悉，可以看一下VS中的有关“ 建构配置 ”，“ 建构平台 ”和“ 配置管理器对话框 ”的帮助文件。 2.大小问题 64位和32位Windows之间一个重要的

区别是句柄的大小。如果你的代码中具备任意Windows API调用，那么你要确保你的说明对于64位Windows是正确的。如果代码是从VB6升级而来，那么你不能区分句柄和32位整数的概念，因此你必须寻找源文件或标头文件。识别哪些参数和域是句柄，使用这些类型的IntPtr。通常情况下，文件将以INT_PTR或LONG_PTR将其识别或者用名称只是某类句柄，如HWND。用前缀定义的参数，如PTR或LPTR通常是一些指示器，因此需被视为IntPtr或使用ByRef编组。要确定你运行的64位还是32位，可以在运行时检查IntPtr.Size的值。 3.COM可以是64位 普遍存在的误解是认为COM和ActiveX仅限于32位，但是使用VB.NET，你可以使用64位COM和ActiveX控件。Windows没有按照64位编译的控件，但是其中一个值得注意的例外是sysmon.ocx ActiveX 控件，该控件可以让你创建系统监测图。 你可以使用Windows.Forms应用程序中的64位ActiveX控件，但是当涉及汇编的时候就存在缓冲。VS当前是32位的应用程序，它在汇编的时候需要32位的ActiveX控件以便创建运行时可随时启用的COM组件包装。为了在VS中进行编译，你需要32位的版本。一旦程序被编译，就只需要被选中的平台。 在你没有32位版本或类似版本的情况下，你可以使用命令行来运行64位工具以便创建一个用于ActiveX控件的Interop程序集。见Tlblmp.exe文件以了解详情。 4.意识到问题 32位应用程序运行于Windows 64中时，它与模拟器一起运行。模拟器被称为WOW64，是Windows On Windows64的缩写，可以让32位应用程序按照32位操作系统进行查看。WOW64模拟器加载了一个ntdll.dll的x86版本，提供了切入点和替换程序，还会截取最重要的注册表和文件系统操作。

WOW64模拟器也暴露出不同的适合32位应用程序的环境型变量。因此，32位应用程序将ProgramFiles环境型变量为ProgramFiles(x86)。以64位运行的时候，如果你写出类似MsgBox(Environ(“ ProgramFiles ”))的代码，会获得"C:\Program Files"，如果你32位运行，则获得"C:\Program Files (x86)"。 模拟器所做的事情与注册表和系统目录类似。出于兼容性和性能的考虑，%windir%\System32 文件夹是64位文件夹。也就是说，64位应用程序中不存在任何截取。重新定向的32位操作文件夹默认名为%windir%\SysWOW64。名称SysWOW64指明它要被32位模拟器WOW64使用。这或许有些令人困惑。 WOW64之下的注册表将重定向与反射结合使用。重定向存在于HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node等Key之下。至少注册表中的命名式样更好。映射是32位进程和64位进程都可以编辑共同Key，如文件关联。另一方面讲，重定向是为了确保将32位从64位隔离开。 关键是执行不能影响你除非你打算从64位程序中获取32位程序的详细信息，反之亦然。通常汇编到同一目标要简单得多。例如，如果你想使用VS，将程序锁定到x86，你不需要担心文件或注册表的模拟过程。如果你要以Any CPU运行程序，那么就取决于它是如何启动的，或许你要使用重定向，和类似GetSystemWow64Directory的API调用，并且使用RegOpenKeyEx API，包括KEY_WOW64_32KEY的访问假象。它可能变成混乱而复杂的测试。这是目前为止对相同平台和锁定信息最好的汇编方式。 其中笔者偶然发现的一个问题是注册表反射和Windows Vista UAC的结合。在用于写入许可的HKLM蜂巢中笔者开放了一个Key，API成功了，但是当笔

者向Key写入时，却失败了。应用程序以32位方式运行于Windows 64上，因此有问题的Key会被映射。笔者浪费了大量时间用于确定所发生的事情，因为注册表API不会向往常一样运作。最后，笔者通过往应用程序中包含运载单使之正常运行。即便requestedExecutionLevel对asInvoker有级别设置，这一漏洞也可以被修复。 UAC提供了注册表虚拟化，笔者认为WOW64重定向或反射与UAC虚拟化的结合太多了。现在笔者通常会确保应用程序中具备运载单。在VS 2008中，你可以从项目属性对话框中的程序标签中获取运载单。确保你包含了requestedExecutionLevel选项。 5.激活Edit和Continue 虽然，你可以调试64位应用程序，那么就不能在调试期间使用Edit和Continue。这意味着你不能在调试的时候改变源代码，相反你要停止，应用更改，重新编译并启动调试。但是你可以用32位程序使用Edit和Continue，即便是在64位Windows上。这是另一种手动构建配置的示例。 创建一个x86构建配置，并且在开发的时候使用这一配置，如此你就可以使用Edit和Continue。然后将配置切换称x64或Any CPU用于测试。 使用Windows 32或Windows 64不如VB 8或VB 9中那样难。使用VS并且构建配置，任务会变得简单。 