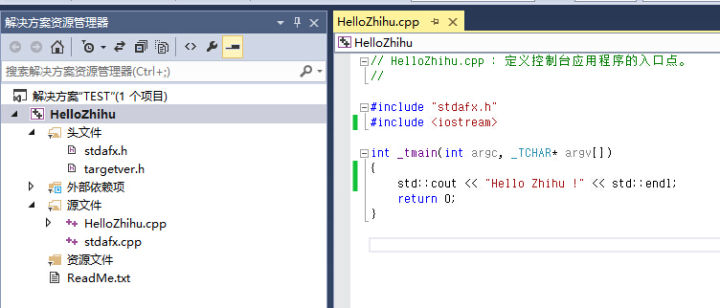
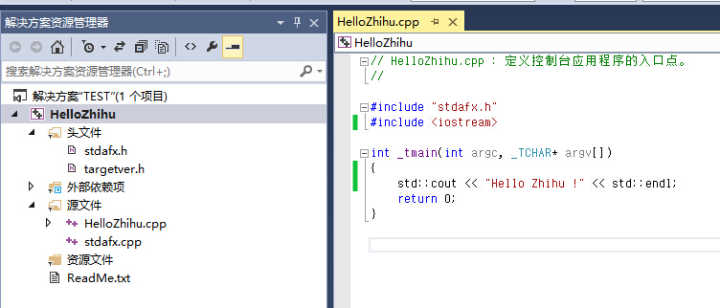
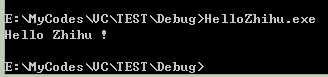
作者：Dr Yao  
链接：https://www.zhihu.com/question/25415940/answer/30803949  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

现在，我们深入探讨一下：  
《如何使用VS 2013发布一个可以在Windows XP中独立运行的可执行文件》。

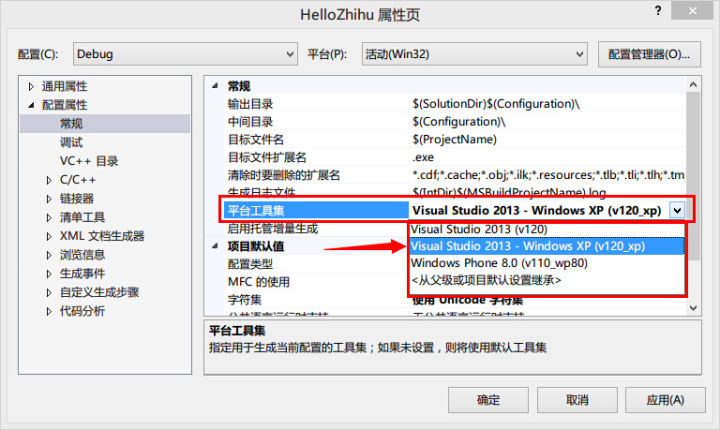
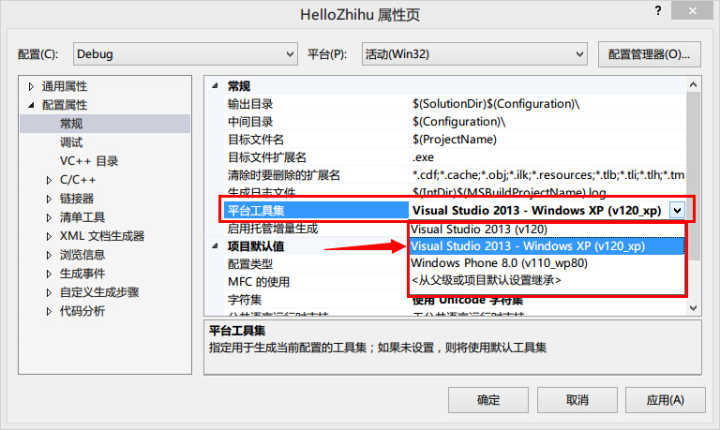
这个问题是比较常见且容易造成初学者困惑的，作为曾经撞了无数次南墙的初级代码狗终于看到了自己能够回答的问题，那么就让我来简单阐述一下造成这个问题的简单原理极其简单解决方法，如有错误纰漏敬请指正。

/\*我们讨论的是非托管的C++程序。\*/

为了方便说明，我们新建一个简单的控制台应用程序项目，直接如图：

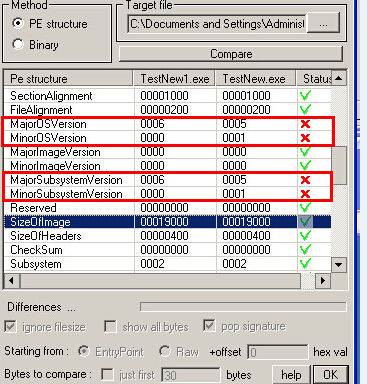
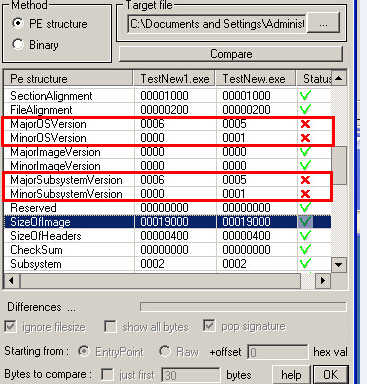
  
非常简单，一个使用了C++标准库的控制台应用程序，在装有开发环境的本机顺利执行出如下效果：  
  
真是一个旷世奇作，我们迫不及待地就此发给XP老哥炫耀，万万没想到：  
  
装逼不成反被XP老哥奚落：“负分滚粗！”

这里我们遇到了题主遇到的问题之一，确实叫人纳闷，不过随便搜索一下就会有解决方法：

  
是的，在项目配置属性中，**将平台工具集选择为"Visual Studio 2013 - Windows XP (v120\_xp)"，即可解决“无效的 Win32 应用程序”问题**。

但是我们还要知其所以然，为什么？  
项目默认的Visual Studio 2013(v120) 与 Visual Studio 2013 - Windows XP(v120\_xp) 生成出来的可执行文件有何区别，以至于前者在XP上执行会出现那样的错误？

最直觉的方法自然是比较一下两版执行文件的区别，我们选用PE （[Portable Executable](https://link.zhihu.com/?target=http%3A//zh.wikipedia.org/wiki/Portable_Executable)：32位或64位Windows操作系统使用的可执行程序或者动态链接库的文件格式）工具 [Stud\_PE](https://link.zhihu.com/?target=http%3A//tools.pediy.com/PE_tools/Editor/stdupe/Stud_PE.zip) 进行PE文件头结构比较，很容易看到区别：

  
看到打红叉的地方，就是两个文件不同之处，其他地方几乎没有区别。  
关于PE文件结构是另外一个话题，我们暂不深入讨论。  
单就这两处我们顾名思义一下就很容易明白：  
MajorOSVersion ，（目标）操作系统主版本号 ，选择默认平台工具集的文件的值是6，后者是5。  
MinorOSVersion ，（目标）操作系统次版本号，前者是0，后者是1。  
MajorSubsystemVersion，（目标）Win32子系统主版本号，前者是6，后者是5。  
MinorSubsystemVersion，（目标）Win32子系统次版本号，前者是0，后者是1。

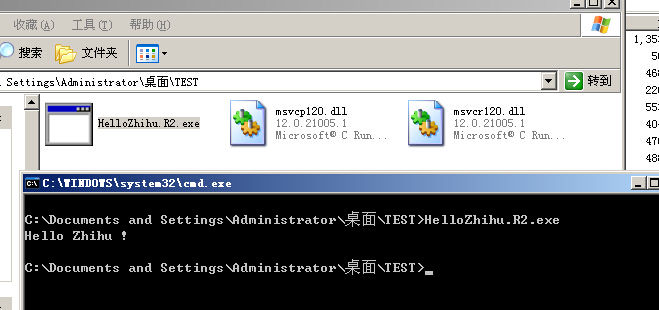
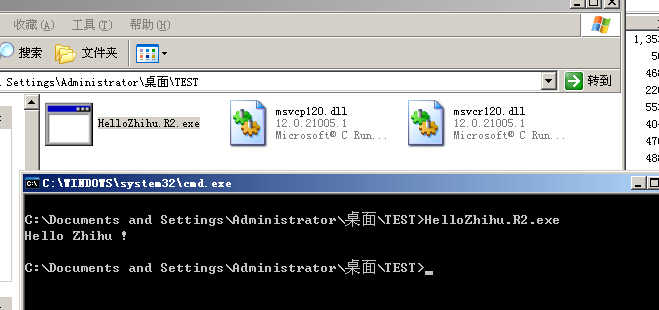
总结一下：一个是6.0 ，一个是5.1 。  
很明显5.1不就是XP的版本号么，6.0就是Vista呗？  
我们是否可以认为，项目默认选择的“平台工具集”生成的可执行文件是不能在6.0以下版本的Windows运行的？

试验结果是：当我把6.0手动修改成5.1之后，这个文件立刻可以正常在XP里运行了，事实上Major/MinorOSVersion的值似乎没有起到什么作用，仅仅修改XxxxxSubsystemVersion的值就可以保证程序顺利执行起来了。反之亦然。同时还发现：在XP中，改为5.1可以，5.2及更高就不行。

对于这个问题的认识虽然仍流于表面，但由于知识有限，我们就不再深究PE结构中的这几个值的深层次含义了。当然如有大牛指点一二就更好了。

现在好了！我们不但解决了XP的运行问题，还大致了解了问题的根源。那么让我们继续发布吧！

将新的、XP的、“5.1的”版本发给XP老哥：

  
我勒个去？你等等，我再编译一个release版本…… 试试：  
  
擦？不是说好的“Release"吗？你等等，不就是一个dll文件吗，我这里有！我发给你……  
我从自己的系统（Win 8.1 x64）C:\Wdinwos\system32 文件夹找一个MSVCP120.DLL发过去：  
  
是啊，这不是逗呢？拿64位的dll文件去冒充32位，能行？ 重新去VS目录里扒一个正确的32位msvcp120.dll 补上：  
  
又来，这次叫做MSVCR120.dll ，不仔细看还真没认出来。继续补上：  
  
呵呵呵呵呵，终于得以正确运行了，但是这么狼狈的炫耀怎么能让人高兴起来呢？

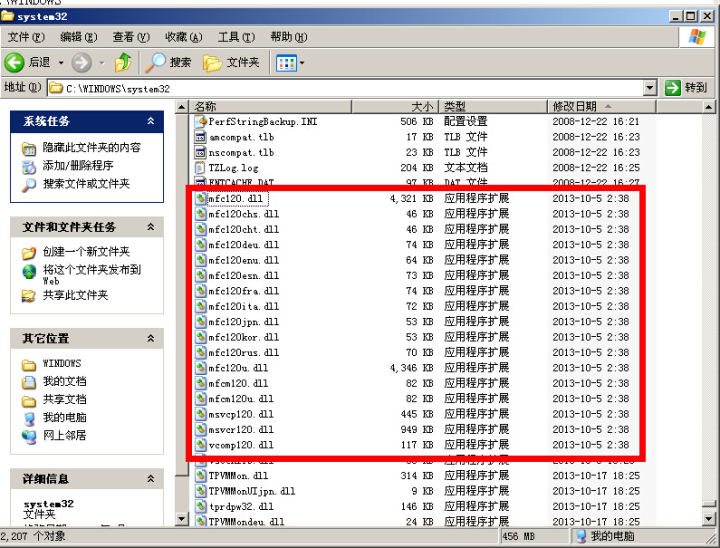
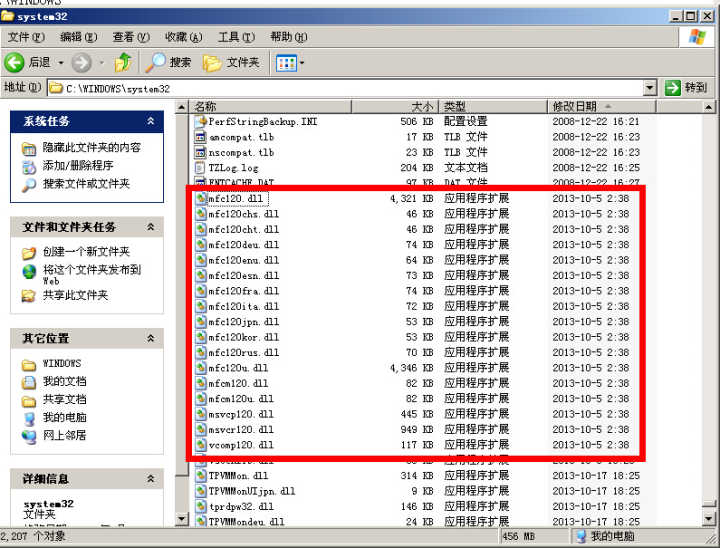
经过一番折腾，好歹知道了是因为缺少文件，那么下次发布程序把这些瓶瓶罐罐DDLL都带上打包不就行了吗？没错，确实是这么个道理，但总感觉很不专业的样子。

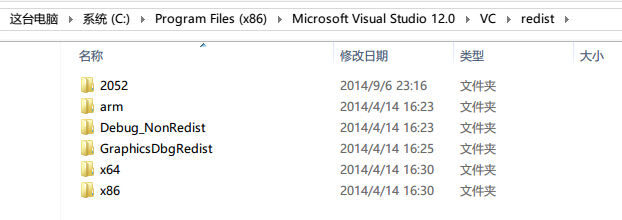
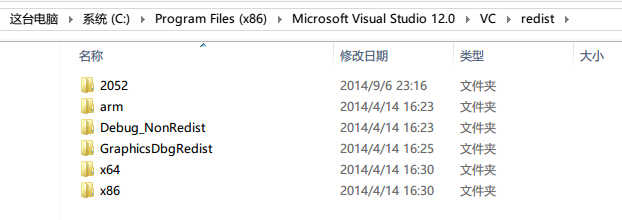
所以一个正常的解决方案就是和其他答案中所说的那样，让目标机器安装   
[VisualC++Redistributable Packages forVisualStudio2013](https://link.zhihu.com/?target=http%3A//www.microsoft.com/zh-cn/download/details.aspx%3Fid%3D40784) 。

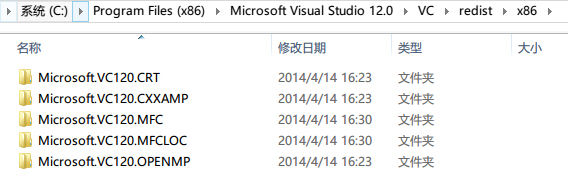
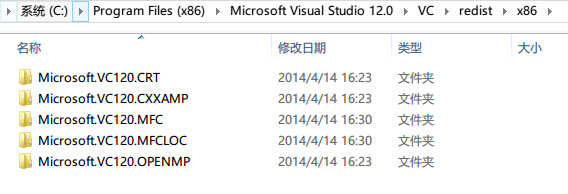
这个东西的作用就是：

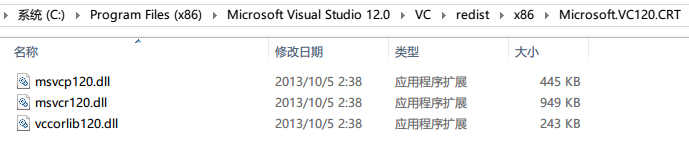
安装运行使用 **VisualStudio2013** 生成的 **C++** 应用程序时所需的运行时组件。

简单观察安装后系统中多出了哪些文件：

  
这样一看，“运行时组件”就变得直观和具体了。  
它们都是什么呢？我们先去VS的安装目录中看一下：

通过路径很容易理解，这是有关VC的redist（再发行）的东西，我们进x86看一下：

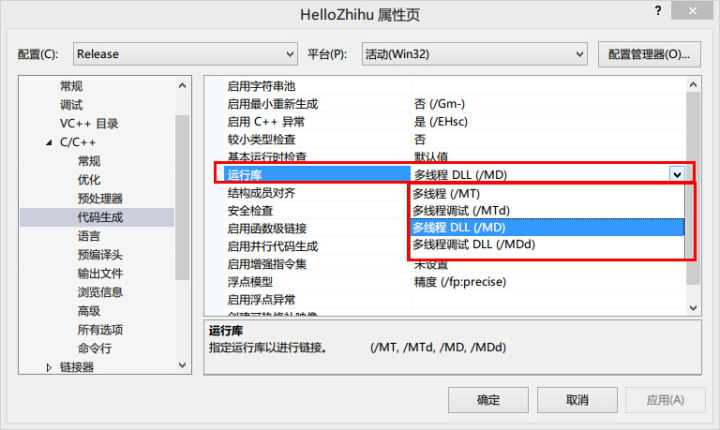
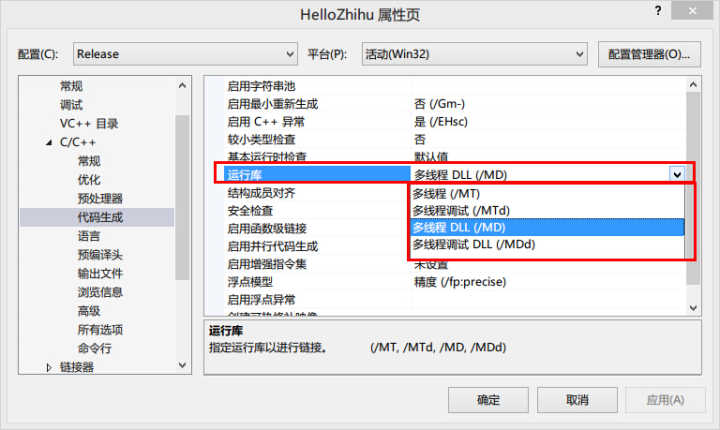
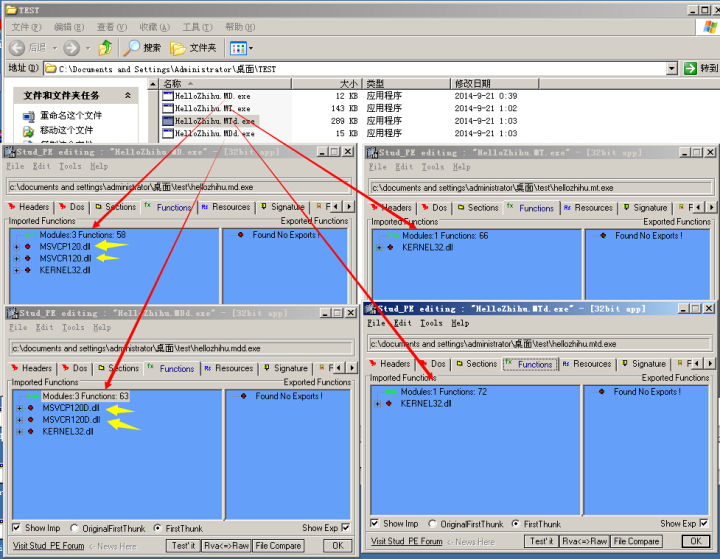
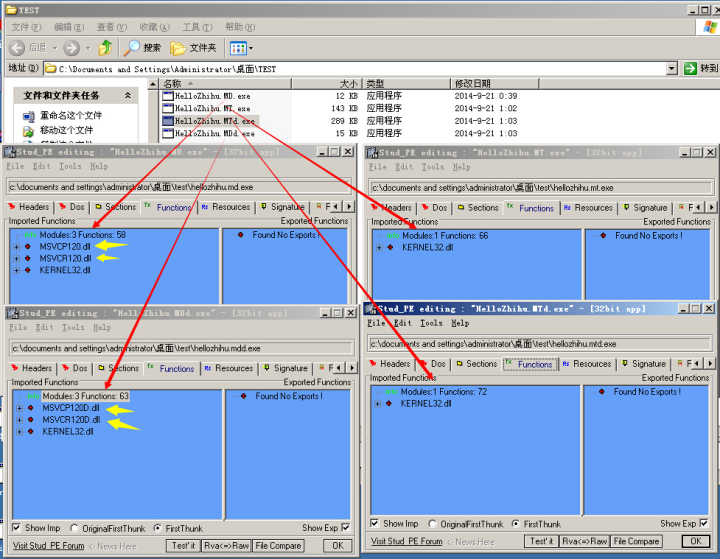
  
有关CRT（运行时库），MFC，MFCLOC(MFC的本地化文件）等等，我们看看CRT里面：

看到了眼熟的这两个dll了，实际上你参考前面那个XP多出文件的图片，那些dll都能在这里找到。  
这就是 Visual C++ Redistributable 包括的东西，每个VS版本都不一样。VS2013对应的就是120。

那管它是VS2013还是2012还是2008，对应的发行包给装上不就完了。

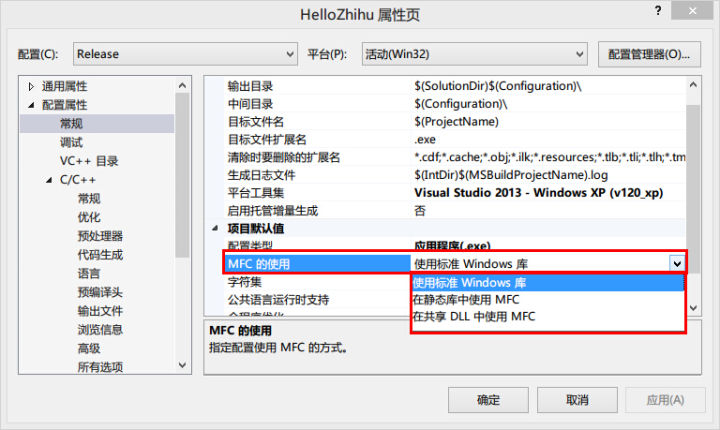
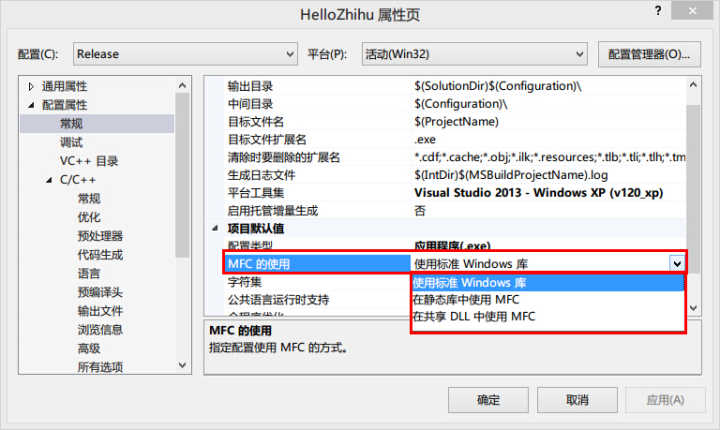
没错。但是我们还要继续研究一下，至少，研究一下如何让一个可执行文件“独立”运行在XP上。

回到项目配置，如下图：

  
我们看到，运行库这一项，包含4种选择。  
废话不多说，我们简单粗暴干脆每一种都生成一个进行比较：  
  
四种版本，分别起了对应的名称，多线程（MT），多线程DLL（MD），多线程调试（MTd），多线程调试DLL（MDd）。  
利用 [Stud\_PE](https://link.zhihu.com/?target=http%3A//tools.pediy.com/PE_tools/Editor/stdupe/Stud_PE.zip) 观察比较它们的 函数导入表 ，发现：  
1. 多线程DLL （MD）和 多线程调试DLL （MDd）  
两者都导入了2个MSVCxxxx.dll （黄箭头所指），但细看又不同，调试版本（MDd）导入的是MSVCP120D.dll和MSVCR120D.dll，比非调试（MD）的那个都多一个字母D。很明显这是配套给调试版的运行时库。而我们之前安装的发行包所部署的都是不带D的版本，是给Release版的程序配套使用的。  
顺便一提MSVCP代表MicroSoft Visual C++(Plus) ，MSVCR则代表MicroSoft Visual C（没有+）Runtime。 一个是C++运行时库一个是C运行时库。

2.多线程（MT） 与 多线程调试（MTd）貌似一样，都没有MSVCP和MSVCR函数导入，只有Kernel32.dll。同时观察这两个文件的体积，都比MD或MDd大了很多，这正是它们不需要导入运行时库DLL函数的原因，因为它们把运行时库静态编译到自己的文件中去了。这也代表着它们运行的时候不会再依赖外部的运行时库DLL文件。

所以答案就在这里，想要你的exe独立运行在XP中：  
1.**将平台工具集选择为"Visual Studio 2013 - Windows XP (v120\_xp)"。**  
**2.将运行库选择为 【多线程 /MT 】或【多线程调试 /MTd】。**  
**3.当然如果使用了MFC，同理的要设置【在静态库中使用MFC】：**



事情很简单，大致就是这样了。

https://www.zhihu.com/question/25415940