

揭秘 NTFS

7 个鲜为人知的秘密告诉你文件权限的真实内幕

作者 / Mark Burnett 译者 / 黄思维

本文为您揭示 NTFS 权限鲜为人知的 7 个秘密，让您更深入地了解 Windows 文件系统的工作原理。

在 Windows 系统里控制用户对文件和文件夹的访问是一个复杂的程序，它需要一套完善的机制来保障各方面的安全性。NTFS 安全模型就提供了这种机制，它可以让你对用户如何读取、写入以及以其它方式来操作系统中的文件进行严格的控制。

NTFS 权限的复杂性使它不容易被理解，即使对于经验丰富的管理员。有时你虽然按要求设置了文件或文件夹权限，但是却未能得到书本上所预期的效果。下面就让我们来看看 NTFS 权限的一些鲜为人知的秘密，希望它们可以帮助你更好地发挥权限的作用。

牢牢把握 5 大基本文件权限

NTFS 提供了 14 种权限来精确定义用户以何种方式访问某个文件。这些权限定义得非常细，所以，为了加以简化，NTFS 把这 14 个权限又合并成了 5 个标准（或普通）权限，如表 1 所示。比如，使用标准权限，你可以方便

地把某个文件的“完全控制”权限授予用户而不用逐一授予他 14 种特别权限。

尽管标准权限在大部分时间使用起来非常方便，但更细分的权限在安全性控制方面更得力。设想一下审核日志文件的场景。你或许希望应用程序向日志文件末尾附加新项目而不是覆盖原有内容。对于这种场景，NTFS 特意区分了写入文件和向文件末尾附加数据这两种权限。

在命令行下，你可以测试写入文件和附加数据的区别。写入文件的命令是：

```
C:\>echo "new text" > test.txt
```

如果你用记事本打开 test.txt 文件，你会看到里面的内容是“new text”。

如果要附加数据，你需要使用的命令是：

```
C:\>echo "Line 2" >> test.txt
```

现在再打开 test.txt 文件，你看到的内容就应该是：

```
"new text"
"Line 2"
```

不幸的是，使用这些精确定义的权限并不像你想象的那么简单。不信我就证明给你看。NTFS 理解写入和附加数据之间的区别，所以您可以通过设置权限来允许附加数据而禁止写入。为了实现这个效果，你把 test.txt 文件的权限设成允许所有权限，唯独拒绝“写入数据”权限，如图 1 所示。

这么设置应该可以让你往文件末尾附加数据，同时禁止覆盖

表 1：标准权限和特别权限

特别权限	标准权限				
	完全控制	修改	读取和运行或列出文件夹	读取	写入
遍历文件夹/运行文件	•	•	•		
列出文件夹/读取数据	•	•	•	•	
读取属性	•	•	•	•	
读取扩展属性	•	•	•	•	
创建文件/写入数据	•	•			•
创建文件夹/附加数据	•	•			•
写入属性	•	•			•
写入扩展属性	•	•			•
删除子文件夹和文件	•				
删除	•	•			
读取权限	•	•	•	•	•
更改权限	•				
取得所有权	•				
同步	•	•	•	•	•

原有数据。可是，当你键入下面这条附加数据命令时：

```
C:\>echo "Line 3" >> test.txt
```

你将被拒绝访问。事实上，你现在根本无法从命令行对该文件做任何写入或附加数据操作。此外，如果你试着反过来操作的话，也就是允许“写入数据”而拒绝“附加数据”，那么结果同样也是失败的。换言之，如果你拒绝了其中一项权限，同时也就拒绝了另一个。

乍一看，似乎是权限设计不合逻辑，实则不然。当程序员通过写代码的方式来访问文件的时候，他们需要向Windows系统请求文件操作权限。

勤奋的程序员会根据特定的操作来细化每一条权限，而偷懒的程序员则不管什么样的操作都要求拥有“完全控制”权限。大部分程序员会请求标准的读或写权限，这在大多数情况下是奏效的。不过，这么做也会带来问题，那就是无法实现更细粒度的权限控制。

在前面的例子里，那些写命令行重定向代码的程序员需要的是标准的“写入”权限，而不是在写入数据和附加数据之间加以区分。如果你看看表1，你就会发现标准的“写入”权限是由六个划分得更细的特别权限组成的。如果用户不同时具有这六个权限，他就无法获得标准的“写入”权限。这就是拒绝追加数据同时也就拒绝写入数据的原因。

所以说，尽管NTFS提供了许多细化的权限，这些权限实际能发挥多大作用却值得怀疑。你可能花了很大力气来设置很详细的文件权限，可到头来却往往是白费功夫，行之有效的只有那五大基本权限。



图1: 设置 test.txt 的权限

当心权限继承的两个例外

NTFS 所使用的继承机制在文件的访问控制上扮演了重要的角色。为了确定权限，Windows 必须查看影响该文件的所有 ACL。任何文件或文件夹都有自己的 ACL，通过它来授予或拒绝特定用户或用户组的访问，此外，每个对象还会从父文件夹及各种上级文件夹继承复杂的权限。所有这些权限都存在相互冲突的潜在可能——一条 ACL 允许用户访问文件，而另一条则明确拒绝访问。此外，一名用户也可能同时是多个用户组的成员，从而拥有不同的权限。

为了解决 ACL 之间的冲突问题，Windows 规定了以下一组规则：

1. 在任一级别上，来自不同用户组的权限将进行组合。
2. 在任一级别上，拒绝权限优先于允许权限。
3. 直接设定在某一对象上的权限优先于该对象继承来的权限。
4. 从近亲文件夹继承来的权限优先于从远亲文件夹继承来的权限。
5. 任何对象都可以通过继承父文件夹的权限而受到保护。

在处理这些规则时，Windows 首先会检查该对象自身所带有的访问控制项目。如果没找到，Windows 就会继续往上查看它的父文件夹，直至找到明确规定允许或拒绝访问该对象的访问控制项目。

你可以阻止某个对象继承父文件夹权限，具体做法是：打开该对象的安全属性高级对话框，找到“权限”页，清除“从父项继承那些可以应用到子对象的权限项目，包括那些在此明确定义的项目”复选框即可，如图

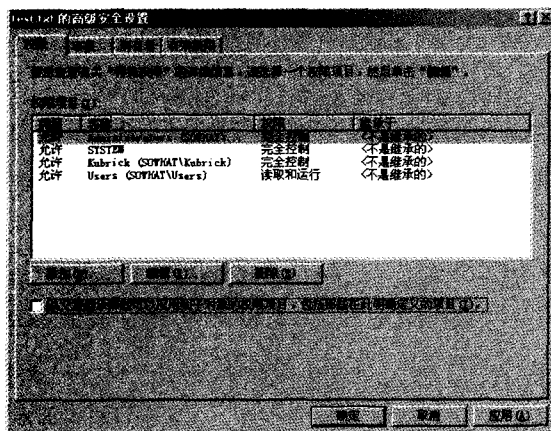


图2: 不允许继承权限

2。如果你这么做了，那么父文件夹的权限将不再影响到当前文件或文件夹以及再下一级文件或文件夹——至少大多数情况下是这样的。

然而，有关权限继承的两个鲜为人知的例外却会给你带来困惑。第一个例外是：如果父文件夹允许列出文件夹的

话，那么对其下文件拒绝“读取属性”权限将不起作用。即使你明确拒绝“读取属性”权限，任何用户只要具有列出文件夹权限就一定能够查看该文件夹下任何文件的属性。这一例外显然违背了规则2和3——它不仅让一条允许权限优先于一条拒绝权限，而且让父文件夹的权限优先于对象自身的权限。这一例外同时也违背了规则4和5，因为即使你通过清除“从父项继承那些可以应用到子对象的权限项目，包括那些在此明确定义的项目”复选框的方式来明确阻止继承权限，父文件夹仍然会对其子对象的权限产生影响。

第二个例外是：如果父文件夹允许“删除子文件夹及文件”权限的话，那么对其下文件或文件夹拒绝“删除”权限将不起作用。因为大多数文件夹都默认允许“删除子文件夹及文件”权限，所以对其下文件拒绝“删除”权限鲜有效果。这个例外也会带来一些困惑，因为你可能需要禁止某用户删除某个文件，但是当你拒绝他的“删除”权限后你却发现他仍然可以删除文件。这一例外同样也违背了多条继承规则，而且即使你明确阻止从父文件夹继承权限也无济于事。

更令人不解的是，你或许认为允许“删除子文件夹及文件”权限的话，用户就能够删除子文件夹及文件，而拒绝该权限就意味着不允许用户删除子文件夹及文件。然而事实却并非如此。对某个文件夹拒绝“删除子文件夹及文件”权限实际上所起的效果是对其下的所有子对象开启了“删除”权限。你不能单独依靠“删除子文件夹及文件”和“删除”这两个权限中的任何一个来阻止用户删除文件。防止文件被删除需要两个步骤——首先，对该文件本身拒绝“删除”权限，然后再对其父文件夹拒绝“删除子文件夹及文件”权限。

如果你发现上述内容需要读好几遍才能理解的话，那么你一定和我一样都觉得权限继承及其例外情况实在太复杂了。

“运行文件”权限很有用

前面我说过，那些细分的权限往往起不到应有的作用，不过有一个例外，那就是“运行文件”权限。基本权限中有一个叫做“读取和运行”，不过实际上用户并非必须具有读取权限才能运行文件。大多数情况下，用户运

行文件所需的全部权限是“运行文件”和“读取属性”。此外，一般情况下你都不需要特意授予用户“读取属性”权限，因为该权限往往可以从父文件夹那儿继承来。

需要特别注意的是：有些程序可能需要读取数据或读取扩展属性的权限才能运行可执行文件，不过这样的程序毕竟不多见。取消“读取”权限的另一个负面影响还在于用户将无法查看该文件的任何版本信息或其它属性，Windows资源管理器也将无法正常显示程序图标，因为它无法从文件中读取图标信息。

然而，有时候某个可执行文件也会包含一些敏感信息（比如：数据库连接字符串），你显然不希望这样的信息被他人通过16位进制编辑器来读取，这时，取消“读取”权限就显得很有必要了。此外，还有一个好处是：用户在没有“读取”权限的情况下是无法将可执行文件拷贝到另一个地方的。

你可以通过灵活运用“运行文件”权限来实现不同级别的访问控制。然而，需要牢记的是：脚本和批处理文件的情况正好相反——运行这类文件，用户必须具有“读取”权限，而不一定需要“运行文件”权限。还有一点非常有趣：如果对一个DLL或OLE自定义(OCX)ActiveX组件拒绝“运行文件”权限的话，用户将无法使用regsvr32.exe来注册该组件。

有些文件权限不如另一些管用

有些文件权限不如另一些管用，原因仅仅在于其它默认的系统特权将它们覆盖了。比如，如果用户具有“还原文件和目录”系统特权的话，你就无法拒绝他的“取得所有权”权限，默认情况下“Administrators”和“Backup Operators”用户组成员都属于这类用户，因为他们具有“取得文件或其它对象所有权”这一特权。你虽然可以剥夺管理员所具有的这一系统特权，但管理员毕竟是管理员，他们也可以把特权重新夺回来。

“取得所有权”权限在很大程度上是没有意义的，因为默认情况下只有“Administrators”或“Backup Operators”组的成员才有权取得某个文件的所有权，而这两个组中任意一组的成员已经具有了“取得文件或其它对象所有权”这一系统特权。唯一用的上这一权限的情形

是：如果某一用户或用户组不属于“Administrators”组，不具有“还原文件和目录”系统特权，但是又具有“取得文件或其它对象所有权”特权，而你希望拒绝该用户或用户组取得某些文件的所有权。只有在这种场合，“取得所有权”权限才会像你期待的那样发挥作用，否则就不要在它身上浪费时间了。

“读取权限”和“更改权限”这两个权限对文件的所有者不起作用，即使你明确拒绝这两个权限。因此，这些权限对于任何具有“取得文件或其它对象所有权”特权的用户也不起作用。如果你要拒绝“读取权限”和“更改权限”，你必须同时拒绝“取得所有权”权限。

注意，尽管“读取权限”和“更改权限”没有明确地允许或限制对审核设置（即：系统访问控制列表——SACL）的访问，大多数查看权限的方法（比如：在Windows资源管理器里查看对象属性对话框的安全页）在权限（即：自由访问控制列表——DACL）不可用的情况下都不会显示SACL。

最后一点，你大可不必在“遍历文件夹”这个权限上浪费时间，因为系统已经默认授予所有用户“绕过遍历检查”特权，这使得该权限形同虚设。此外，微软也一直建议不要强制执行遍历检查，因为某些应用程序无法正确处理目录遍历错误，从而会导致一些问题。

共享权限只有补充作用而无法取代NTFS权限

当你在某个共享文件夹上设置权限时，这些共享权限会以相对独立的方式与NTFS权限一同作用于相关的文件和文件夹。Windows会兼顾这两方面权限，取其更严格者实施。

在这两组权限中，NTFS权限的重要性更高。尽管共享权限可以控制用户通过网络共享来访问文件，但它却不能阻止用户从本地系统或通过终端服务来直接访问这些文件。在控制文件访问方面，值得信赖的永远都是NTFS权限，而共享权限只能作为一种辅助手段来对网络用户的访问控制做进一步的限制。

另外，需要注意的是，各共享文件夹的权限只对该共享本身起作用，这些权限不会继承也不会传播给其它共享，即使在同一物理目录结构中是上下级关系。举个例子，假设你为所有用户建了一个共享，映射到“D：

\users”。所有用户都可以访问这个共享以及其中的所有对象。现在，你又专门为管理员建了一个共享“D:\users\admin”。就第二个共享而言，尽管它寄宿在第一个共享内部，但在权限上却与第一个共享毫无关联。就算你拒绝普通用户访问“D:\users\admin”共享，他们还是可以先访问“D:\users”共享然后再进入“admin”目录。要阻止用户访问“admin”目录，你必须设置相应的NTFS权限。所以，为了避免此类情况发生，最好不要把一个共享寄宿在另一个共享内部。

结合内置用户组来设置权限

当你为文件和文件夹设置权限时，不要忘了利用各种内置的用户组，Windows会在用户登录系统时自动将内置用户组的权限分配给用户。比如，对于“Remote Desktop Users”这个用户组。你可以根据登录类型来设置非常详细的NTFS权限。比方说，你可能希望只允许管理员或那些实际登录到本地控制台的访问某些敏感文件。这时，你把访问权只授予“Administrators”和“Interactive Logon Users”这两个组就可以了。你还可以做进一步的限制，比如只允许作为服务登录的帐号访问相关文件。有关Windows内置安全主体的完整列表，请查看“[Well-known security identifiers in Windows operating systems.](http://support.microsoft.com/kb/243330)”

当心“Everyone”

我的最后一条建议是：当心“Everyone”用户组。在XP以前的Windows版本里，“Everyone”用户组中默认包含匿名用户，所以那时的忠告总是不要授予“Everyone”组任何访问权。而在XP、Windows Server 2003以及Windows Vista里，“Everyone”组默认不包含匿名用户，所以根据你的特定权限设置，拒绝“Everyone”组的访问权限并不一定意味着匿名用户的访问也被拒绝。要阻止匿名用户以及其他不希望的用户访问资源，最保险的方法不是拒绝某些用户组的权限，而是把允许访问权限只授予那些需要访问的用户和用户组。

NTFS权限无疑是复杂的，希望本文所揭示的一些秘密可以让你在面对一些复杂局面时少几分困惑。