**[SVN与TortoiseSVN实战：标签与分支](https://www.cnblogs.com/ym123/p/4326721.html)**

SVN与TortoiseSVN实战系列中很多知识点来源于《版本控制之道》这本书，我写的内容主要结合了TortoiseSVN工具，更多侧重于实际操作。

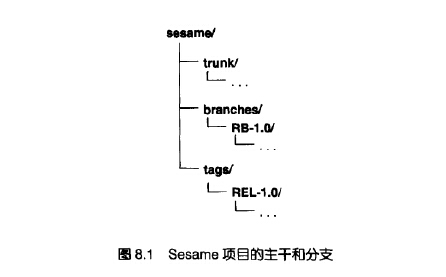
很多项目人员没有正确运用SVN，仅仅作为源代码的团队共享来用，而没有充分利用标签、分支这些强大的特性。

要正确使用SVN的标签和分支特性，就要先建立起合理的项目仓库结构。

**正确的SVN项目仓库结构**

很多项目人员往往在创建项目仓库以后，直接将整个项目的源代码提交到仓库的根目录，这往往是由于对SVN没有系统的了解。

引用书里的图片，正确的项目仓库目录结构如下图：



在根目录下首先建立三个目录：

1、trunk：又称为主干，这个子目录才是应该将项目源代码提交到的位置。

trunk目录代表着开发的主线，大部分的开发工作涉及到的更新、提交都在这个目录里，就跟“原始”的SVN使用一样。

2、branches：这个目录就是分支目录，所有新建的分支都应放在这个目录。

每一个branches下的分支代表了一条并行的开发过程，如果不明白可以继续往后看。

3、tags：这个目录是标签目录，所有建立的标签都应放在tags目录下。

每一个tags下的标签代表了一个项目的时间节点，或者说是事件、里程碑。

这三个目录之所以如此命名，完全是约定俗成，而非SVN规定。

**什么是分支？**

分支的基本概念：一条基于另外一条开发线存在的分支，并共享在分离之前的日志，一个分支就象一个拷贝的存在，并从那里离开，并具有自己的历史。

没有找到关于分支很简洁的解释，我的解释是分支相当于对于开发中的主干进行了复制，从此在分支上后续的所有操作与主干上的所有操作互不影响。

**使用分支前：**

你开发的IOS App已经测试完毕，提交到APPStore审核，在APPStore审核过程中你可能又开发了些功能但还没通过测试，这时APPStore把你的应用退回了，可能是你的Welcome图片不合格需要换一下，很简单的修改可能完全不需要经过测试流程，但这时你只有两种选择：

1、把代码回滚到提交时的状态，修改后提交；

2、把手头的功能开发完成，经过完整的测试、发布流程提交。

**合理使用分支后：**

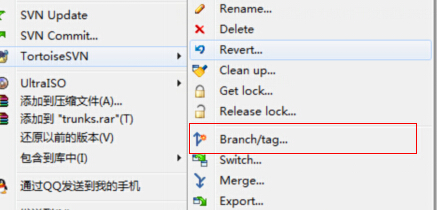
你开发的IOS App已经测试完毕，提交到APPStore审核，根据你提交的版本号建立一个分支，在APPStore审核过程中你可能又开发了些功能但还没通过测试，这时APPStore把你的应用退回了，可能是你的Welcome图片不合格需要换一下，到这里你完全可以轻松应对，只需要把之前建立的那个分支签出到本地，修改后提交。

使用分支后整个过程So easy~

分支里保存的实际上是你建立分支时代码的快照，在分支上进行的修改还可以合并到主干中。

**什么是标签？**

标签和分支的本质是一样的，可以看到TortoiseSVN中，标签和分支实际上在一个菜单：

****

分支与标签的使用也是约定俗称的，混用分支与标签SVN也不会禁止。

标签用来记录项目中的事件或里程碑，可以让你给项目中的事件起一个名字，比如发布2.0.0版本，以后你可以使用这个名字来签出它。

**标签与分支对比**

还是继续上面的例子：

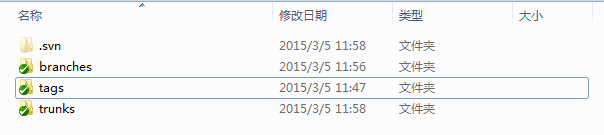
APPStore审核通过，这时你应该为分支建立一个标签，然后将分支的修改合并到主干，然后删除分支。

更直白的说，你可以在分支上进行并行开发，而不应该修改标签里的内容，标签仅应当作为一份代码的快照。

[**SVN与TortoiseSVN实战：TortoiseSVN新建及合并分支**](https://www.cnblogs.com/ym123/p/4331300.html)

本篇重点通过例子讲解TortoiseSVN新建及合并分支的实际操作。

我这里已经新建好一个SVN项目仓库，并按照第二篇介绍的标准项目仓库的目录结构新建好trunks、tags、branches三个目录：

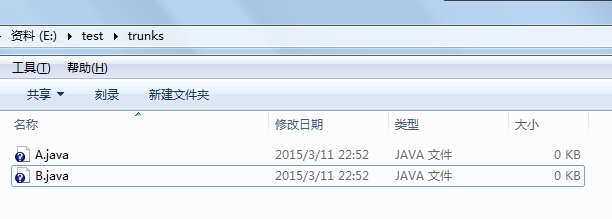


下面开始实际操作。

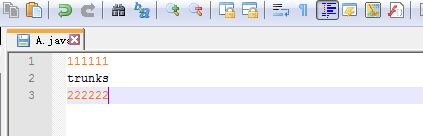
**使用TortoiseSVN新建分支**

由于这三个目录是刚建好的，为了更好体现分支的新建及合并对项目中代码的影响，先伪造点代码文件。

打开trunks目录，在trunks目录下新建两个文本文件A.java，B.java：

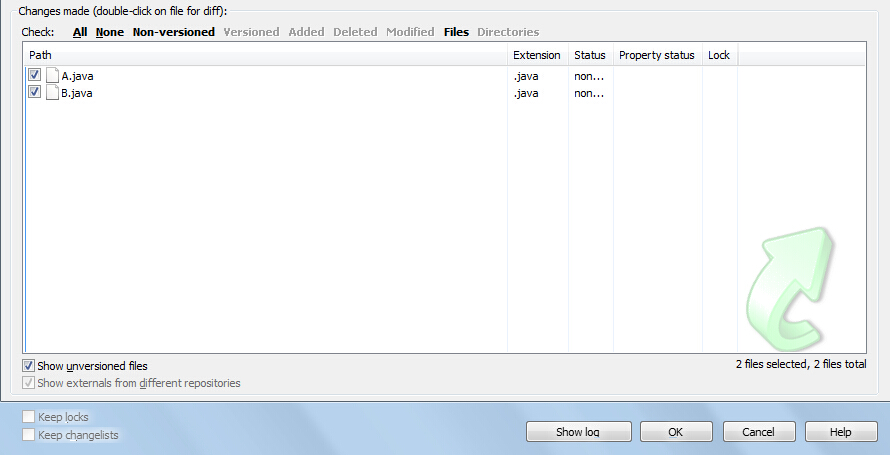


打开A.java输入以下内容：



B.java文件可以随机输入些，本例中主要用于观察后续是否变化。

两个文件编辑完成后使用SVN Commit将更改提交到SVN：



**下面开始创建分支：**

1、在trunks上点击右键，在TortoiseSVN菜单中选择Branch/Tag；

2、在To path输入框中输入新建分支的路径，一般是：/branches/分支名，也就是相当于分支保存的路径名；

3、在下面选择HEAD revision in the repository，为当前SVN中trunks目录下最新的代码建立分支，如果需要为制定的revision建立分支，可以进行选择；

4、点击OK分支建立完成。

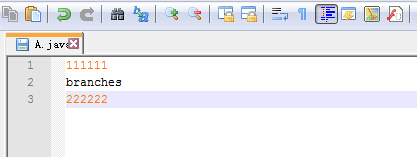
打开branches目录，发现目录依然为空，没有刚才建立的分支，这是因为分支建立的操作是在服务器端完成的，当需要编辑分支时，在branches目录点击Update，刚才建立的分支就会下载下来。

建立分支的过程会非常的快，不会因为主干中的文件多而降低速度，原因后续说明。

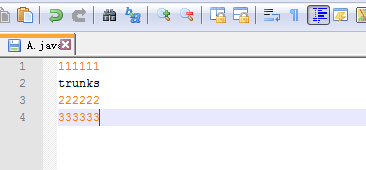
**使用TortoiseSVN合并分支**

第二篇中举过例子，有可能存在主干、分支并行开发的情况，下面我们模拟一下。

将刚才新建的分支Update下来，打开分支中的A.java，对其进行编辑如下：



在打开trunks主干中的A.java，对其进行编辑如下：



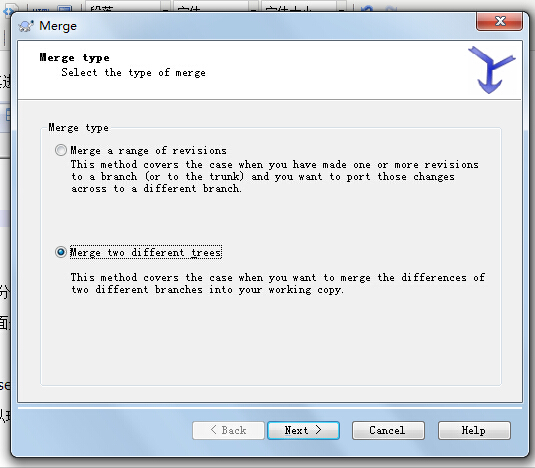
对以上两个文件的改动基本上模拟了分支、主干并行的情况，两个目录下的同一个文件被分别修改。

将以上两个文件的改动Commit，下面进行合并操作。

**下面开始合并分支到主干：**

1、在trunks上点击右键，在TortoiseSVN菜单中选择Merge；

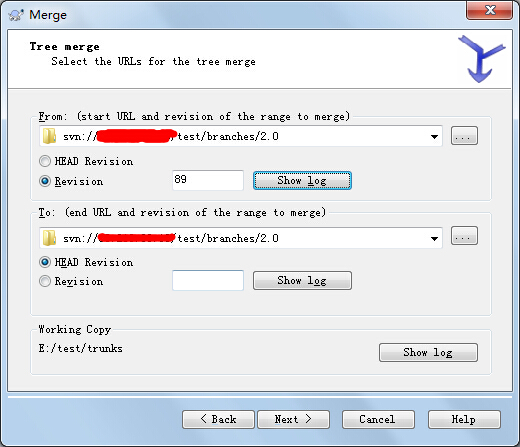
2、在弹出的窗口中选择第二项，可以理解为合并两个树；



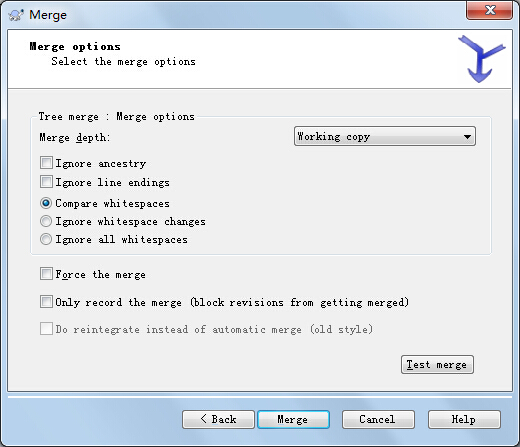
3、在From和To中都选择要合并的分支目录；

4、在From的Revision选择创建分支时的那个Revision，具体就是点击Show log，选择最下面那一条；

5、在To的Revision选择HEAD Revision，也就是最新操作；



6、点击Next，下一个页面使用默认项，点击Merge；

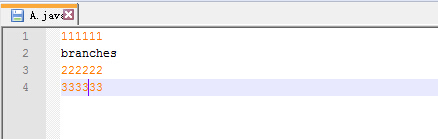


7、合并完成。

这时候会发现trunks目录下的文件已被修改，这时候的合并操作是在本地完成的，并没有提交到SVN，这与分支建立时有区别，请注意。

也就是说如果你在本次合并中发现问题，只需要对trunks目录Revert，放弃本次合并即可。

可以看一下trunks目录合并后的A.java文件，大家与前面对照下，已达到我们想要的合并效果：



**创建分支时发生了什么**

SVN服务器在创建分支时的拷贝是非常有效率的，瞬间就可以完成，并且只需要很少的空间来存储，所以不需要担心分支建立过多导致项目仓库膨胀的问题。

可以理解为SVN在服务器中存储的是文件的引用，而非物理上的拷贝。

**不要多次合并同一个分支到主干**

合并分支有一个原则，就是不要对一个分支多次合并到主干，虽然你有可能在合并后又对分支进行了修改，之所以这样说是有原因的。

上面提到在合并分支时，在From的Revision选项中需要选择合并的其实版本，如果是第一次合并只需要选择最早的那个版本即可，但如果合并过一次，你真的能记得你上次合并时最后的Revision是多少吗？

良好的操作是合并后新建分支。

**合并主干到分支**

其实合并是个可逆的过程，合并主干到分支与合并分支到主干正好是相反的操作，大家如果会了一种另一种应该非常简单。

[**SVN与TortoiseSVN实战：冲突详解（一）**](https://www.cnblogs.com/ym123/p/4345771.html)

在写到SVN分支合并时，有评论中也提到合并后发生冲突的问题，相信关于冲突的知识也是开发人员的痛点。

关于冲突的知识，重点介绍以下几个方面：

1、什么情况会产生冲突？

2、冲突发生时产生的三个文件是什么含义？

3、怎样使用TortoiseSVN解决冲突？

**什么情况会产生冲突？**

当我第一次使用SVN这类项目管理工具时，心中就有一个疑问：多个人同时编辑一个文件会发生什么情况？

为了模拟这种情况，我通过使用另外一个账号签出上一篇中的项目，来模拟多个用户同时操作的情况（在TortoiseSVN的设置对话框里通过Saved Data可以删除保存的账号信息以便重新登录）。

https://images0.cnblogs.com/blog2015/693632/201503/172213309701230.jpg

现在test目录中新建一个测试文本文件，输入以下内容并签入，然后签出到testother目录中：

[复制代码](javascript:void(0);)

1

2

3

4

5

6

[复制代码](javascript:void(0);)

首先编辑test目录中的测试文本文件，把第一行的1改为11，保存后签入到SVN：

[复制代码](javascript:void(0);)

11

2

3

4

5

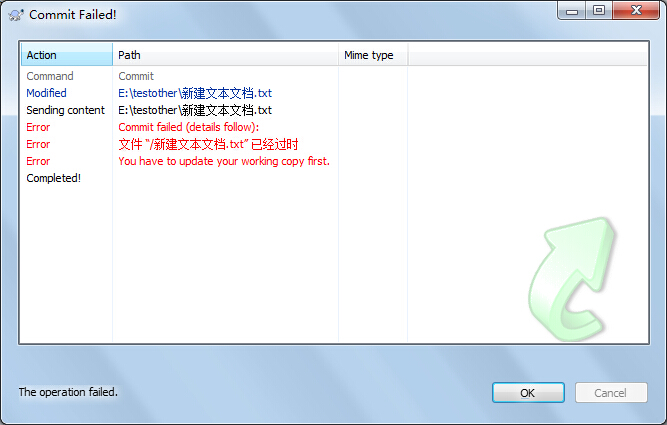
6

[复制代码](javascript:void(0);)

现在编辑testother目录中的测试文本文件，把最后一行6改为66，保存后签入到SVN：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 1  2  3  4  5  66 |

在签入testother的修改时，会提示我们该文件已过期，需要首先签出最新版本：



当我们签出后TortoiseSVN会提示让我们再次尝试签入，这次签入会成功，在这个过程中没有发生冲突，testother目录中的测试文本文件的内容为：

[复制代码](javascript:void(0);)

11

2

3

4

5

66

[复制代码](javascript:void(0);)

**可以看到内容自动合并了来自两个目录下的修改，那什么情况下会产生让人头疼的冲突呢？**

上面的操作之所以没有产生冲突，是因为两个目录中对文件内容的修改没有在同一个位置，如果在testother目录中对第一行进行修改（比如不修改6，而把1改为111），就会产生冲突。

可能有人看到这里就会想到以下这种情况，还是在test目录中修改1为11：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 11  2  3  4  5  6 |

在testother中的第一行前新建一行，修改为：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | 0  1  2  3  4  5  6 |

这种情况会产生冲突吗？

结果是：会。

这是因为SVN并没有那么智能，认为你是新建了一行，SVN对于冲突的判断关键在于对相同位置上内容的修改，所以当你新建一行时就改动了所有后边的位置。

了解了这些大家基本对SVN冲突产生的原理应该认识的非常透彻了。

关于冲突的其他几个问题稍后讲解，一篇没有办法写的太详细。

[**SVN与TortoiseSVN实战：冲突详解（二）**](https://www.cnblogs.com/ym123/p/4352320.html)

硬广：《SVN与TortoiseSVN实战》系列已经写了四篇，第二篇《[SVN与TortoiseSVN实战：标签与分支](http://www.cnblogs.com/ym123/p/4326721.html)》和第三篇《[SVN与TortoiseSVN实战：TortoiseSVN新建及合并分支](http://www.cnblogs.com/ym123/p/4331300.html)》重点介绍了标签和分支的概念及实际操作演示，关注人数较多。

上一篇提到关于冲突的知识，其中已经说明了第1点：

**1、什么情况会产生冲突？**

2、冲突发生时产生的三个文件是什么含义？

3、怎样使用TortoiseSVN解决冲突？

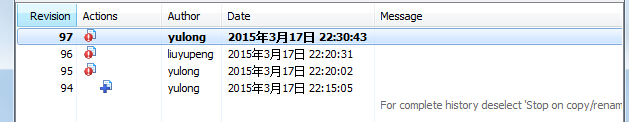
SVN是根据同时对相同位置上内容的修改来判断冲突的，这是核心的一点，下面说明2、3点。

**冲突发生时产生的三个文件是什么含义？**

 在签出文件时，当本地文件的修改与其他人签入的修改产生冲突时，会产生三个冲突文件，如下图：

****

 我们在冲突的文件上点击右键，使用TortoiseSVN的Show log查看下冲突文件的修改日志：



先补充一点，**SVN的Revision是基于整个项目的，而不像有些版本控制程序针对于每个文件有自己的Revison，SVN每一次签入都会递增，一个Revision下可能包含多个文件的修改。**

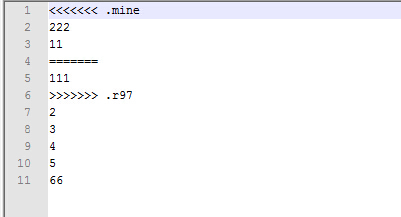
对比这Show log出来日志，我们来说下这三个文件的含义：

新建文本文档.txt.r96：96就是版本号，是你本次修改前的初始版本，也就是说你是基于这个版本修改的，如果你Revert本次修改，那本地会退回到这个文件的内容；

新建文本文档.txt.r97：97也是版本号，是别人在你签入前签入后生成的版本；

新建文本文档.txt.mine：是你修改后准备提交的那个版本，也就是在冲突产生前你修改的新建文本文档.txt的内容；

那这时的新建文本文档.txt的内容是什么呢？



打开后可以看到SVN已经修改了这个文件，使用<<<<<<< >>>>>>> 标明了发生冲突的地方，基于以上对三个文件的理解，大家应该能容易明白SVN的标记的方法了吧：

<<<<<<< .mine 和 ======= 之间是你修改的内容；

======= 和 >>>>>>> .r97 之间是别人最新签入的内容；

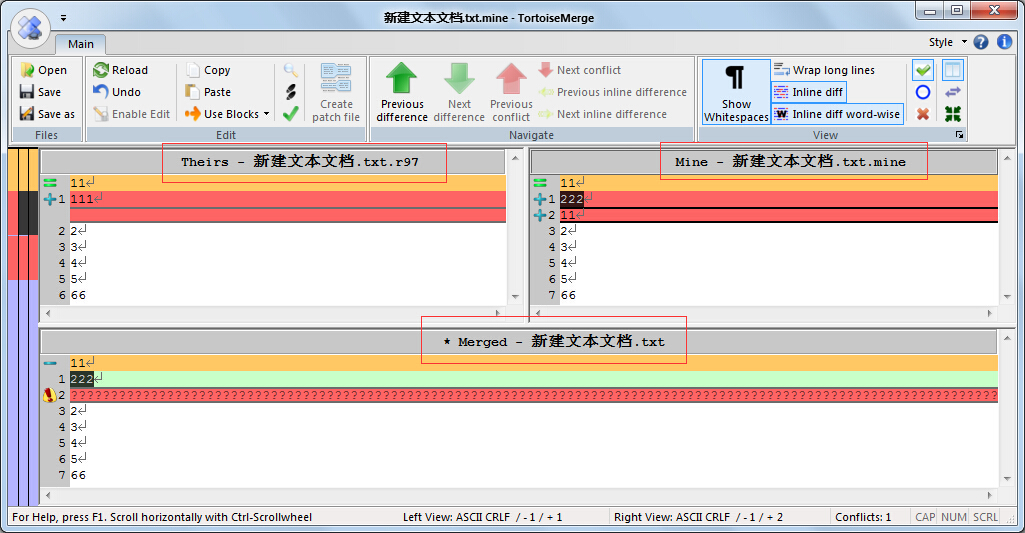
通过Show log查看别人签入时标注的注释，决定采用谁的代码，然后删除掉<<<<<、======、>>>>>和多余的代码来解决冲突。

**最后强调一下 .mine 在SVN中代表自己的修改。**

**怎样使用TortoiseSVN解决冲突？**

 其实理解了SVN的标记方式，在冲突少的情况下手动解决是最方便的。

以下演示在冲突文件上右键，选择TortoiseSVN菜单Edit conficts，使用TortoiseSVN的工具来解决冲突：



相信了解三个冲突文件代表的意义以后，可以很容易了解以上三个区域显示内容的含义了：

左边代表别人的修改，右侧是自己的修改，底部是合并后的内容；

红色标记了冲突的行，在别人与自己的冲突行中选择采用的一行，点击右键Use this text block，就解决了一行冲突。

也可以使用整个文件来解决冲突，右键使用Use this whole file。

所有冲突解决完以后，点击上边的对号按钮，标记解决，这时此前的三个文件消失，签入你的修改。

**额外说点什么**

 每次签入把本次修改的原因与注意事项的写入SVN的备注中是良好的习惯，当产生冲突时可以结合备注来了解修改的原因，对于CodeReview人员来说也是必须的。

SVN服务器可以设置脚本，对于不写备注的签入拒绝签入，设置方法可以自行百度。

[**SVN与TortoiseSVN实战：补丁详解**](https://www.cnblogs.com/ym123/p/4354959.html)

硬广：《SVN与TortoiseSVN实战》系列已经写了五篇，第二篇《[SVN与TortoiseSVN实战：标签与分支](http://www.cnblogs.com/ym123/p/4326721.html)》和第三篇《[SVN与TortoiseSVN实战：TortoiseSVN新建及合并分支](http://www.cnblogs.com/ym123/p/4331300.html)》重点介绍了标签和分支的概念及实际操作演示，关注人数较多，三、四篇《[SVN与TortoiseSVN实战：冲突详解（二）](http://www.cnblogs.com/ym123/p/4352320.html)》详解了SVN如何判定冲突及冲突文件说明和解决冲突方法。

本篇详细SVN中补丁的概念，版本控制之道这本书对于SVN的讲解比较全面了，但书中也没有讲解补丁的概念，关于SVN补丁的介绍也寥寥无几。

其实学习和掌握补丁的用法后，在某些项目环境中还是有用武之地的，主要讲以下两点：

1、SVN补丁的概念；

2、SVN补丁的应用场景；

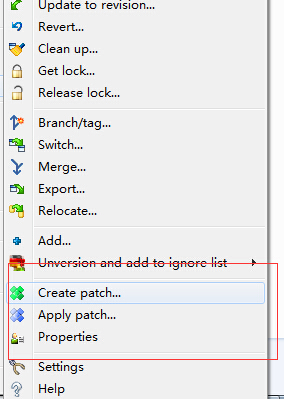
3、SVN补丁的制作及应用；

感谢博友的建议，在每篇前写明本篇所要讲述的重点 ~。~

**1、SVN补丁的概念**

SVN补丁其实并不神秘，只是平时工作中接触较少。

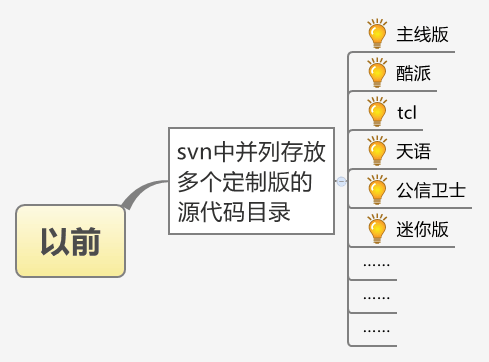
SVN补丁可以理解为一个patch格式的文件，这个patch格式文件可以忠实记录文件变动，而这个patch文件是根据SVN中文件的修改制作的。

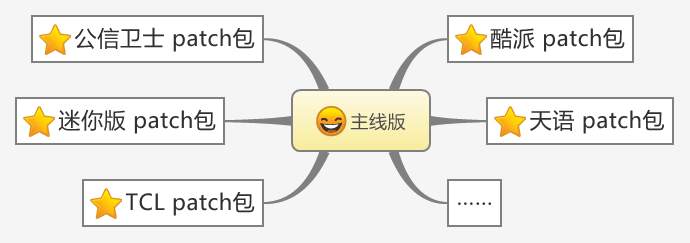


Create patch和Apply patch就是创建和应用补丁，至于Properties为什么在这一组会在属性篇中讲，TortoiseSVN菜单的分组划分是很有设计哲学的，没事时可以仔细研究下。

**2、SVN补丁的应用场景（一）**

 第一种场景看下这两张图：





图一的方式可以用之前讲的主干/分支来解决，每次修改主线版的内容通过合并的方式将修改合并到定制版中，这种方式需要在每个定制版中合并，人工每次操作非常麻烦，还容易出错。

图二是使用补丁的方式，主线版修改后创建补丁，在所有定制版中统一应用一次补丁，其目录结构如下：



patches下是所有定制版的项目目录，这样划分可以在整个patches目录上应用一次补丁，大大简化了操作。

**2、SVN补丁的应用场景（二）**

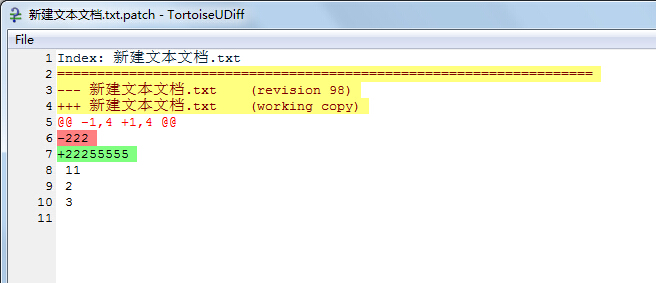
 还有一种是涉及到权限的应用，将某些项目组的成员配置成只有读权限，而没有写权限。

这些只读权限的成员不能直接提交修改，而需要将修改制作成补丁，发给具有写权限的成员，审核通过后应用补丁，提交到SVN中。

这也是基于SVN的一种权限管理，熟悉Git的应该会比较理解。

**3、SVN补丁的制作及应用（一）**

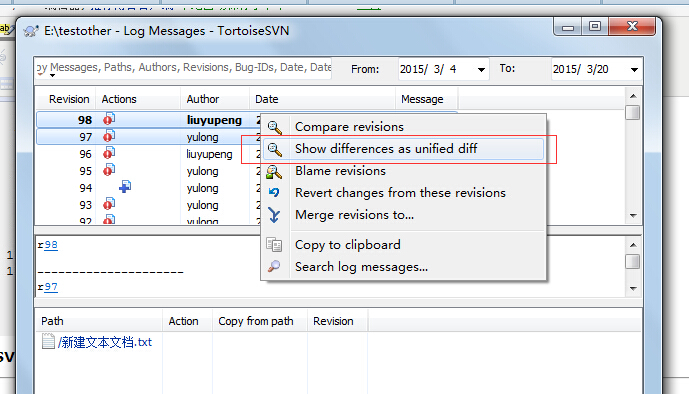
 最简单的方法是基于修改的文件直接创建补丁，在将本地文件修改后，在签入SVN之前（叹号标记时），选中文件，使用TortoiseSVN的Create patch菜单可以直接将文件的修改创建为patch格式的补丁文件：



**3、SVN补丁的制作及应用（二）**

 还有一种方式可以基于已签入的改动来创建补丁，这种方式甚至可以创建降级补丁。

在主干目录上使用TortoiseSVN的Show Log菜单，用Ctrl选中两个版本，右键使用图中标记的菜单：



新窗口中使用File Save存为补丁即可。

补丁的应用非常简单了，就不演示了，选中应用位置使用Apply patch菜单选择应用的补丁即可。

[**SVN与TortoiseSVN实战：属性的奇技淫巧（一）**](https://www.cnblogs.com/ym123/p/4360869.html)

硬广：《SVN与TortoiseSVN实战》系列已经写了六篇，本系列结合TortoiseSVN对SVN中容易被忽视的部分进行了详解，预计再用三、四篇来结束这个系列。

本篇详解一下SVN的属性，属性可以算的上是SVN中的奇技淫巧，像分支、标签、补丁应该在项目中属于标配，而属性则往往应用较少，但是充分的了解对合理使用SVN还是有帮助的。

内容概要：

1、属性的概念和基础操作；

2、属性的关键字展开；

3、使用属性忽略某些文件；

4、设置Mime类型；

5、自动设置属性；

6、引用外部项目；

7、设置签入日志限制；

8、属性冲突；

其实属性还有设置换行、设置可执行标志等用途，但操作简单、用途也较小就不再此详细，有兴趣的可以自行百度。

**1、属性的概念和基础操作**

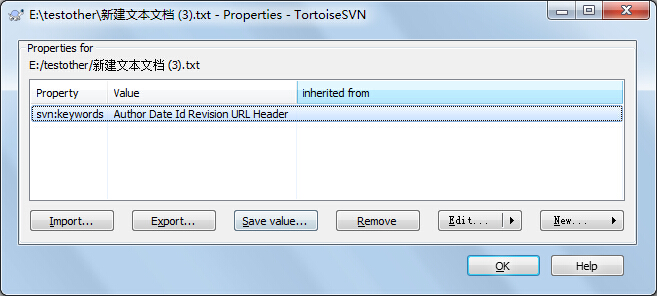
SVN不但可以存储属性，还可以存储与文件相关的元数据，文件的元数据就是属性。

属性的操作与文件类似，也有签入、签出以及冲突。

属性相当于关联到文件KV键值对，键值是字符串，而值的类型可以是任何内容，包括二进制文件。

属性可以分为两种：普通属性和特殊属性。特殊属性以svn开头，TortoiseSVN在SVN的基础上又增加了一些特殊属性，这些属性以：tsvn开头。

对属性的操作在TortoiseSVN的Properties菜单项，会打开属性窗口对属性进行增删改：



另外提一点，文件和目录的可用特殊属性是不同的，大家可以分别在文件和目录中使用Properties菜单，查看New按钮弹出的属性。

**2、属性的关键字展开**

看名字很高大上，也比较抽象难于理解，先简单解释下。

关键字可以理解为SVN内置的一些预定义变量，如：

$Date$ 已知最后提交的日期；  
$Revision$ 已知最后提交的版本；  
$Author$ 已知最后提交的作者；  
$HeadURL$ 此文件在版本库中的 URL；  
$Id$ 上述四个关键字的压缩组合；

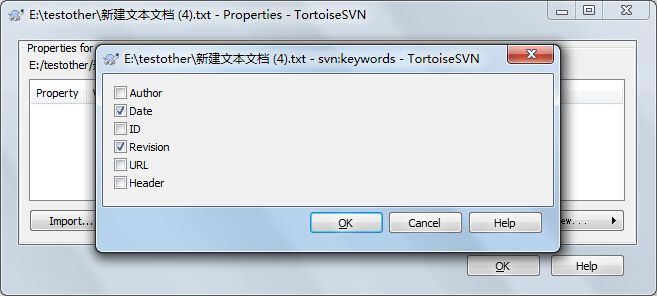
所谓的展开，就是在你设置属性的文件内容中写入这些关键字，当文件在签入时，SVN会将这些关键字替换为实际的内容，下面实际操作下：

1、新建文本文档，输入以下内容：

|  |  |
| --- | --- |
| 1  2 | $Date$  $Revision$ |

2、右键菜单，使用TortoiseSVN的Properties菜单项，在属性窗口中点击New按钮，选择Keywords；

3、勾选上Date和Revision；



4、签入文件，再查看一下文件内容，会被SVN关键字展开为以下内容：

|  |  |
| --- | --- |
| 1  2 | $Date: 2015-03-23 19:41:11 +0800 (周一, 23 三月 2015) $  $Revision: 120 $ |

**3、使用属性忽略某些文件**

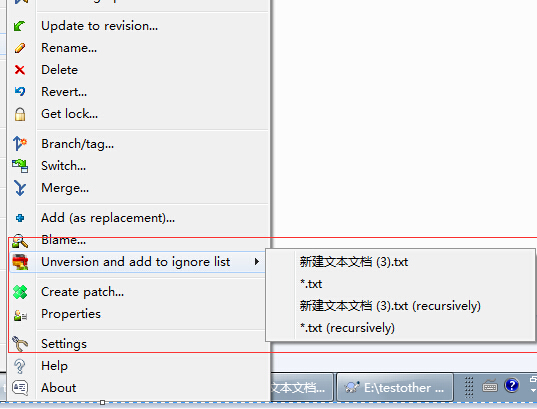
使用SVN中常常遇到这种情况，SVN目录下某些本地生成的文件不想签入到SVN服务器，这些文件仅用于本地操作，如C++项目中的bin和obj目录。

如果你没有把这些目录和文件加入忽略列表，而是签入到SVN，那估计你的项目组成员会遇到满屏的问号。

可以使用忽略属性 svn:ignore 来解决这种问题，本篇介绍的SVN机制有突破一般人理解的地方，请留意。

TortoiseSVN的菜单中已经提供了忽略操作，其实质也是通过忽略属性来实现的，我们来解刨一下TortoiseSVN的操作：

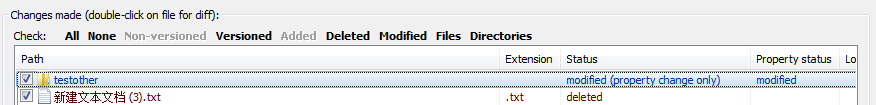
在文件或目录上右键，鼠标移动到图中标示的忽略菜单：



这四个子菜单项分别提供了：

1、将选中的文件加入忽略列表；2、将txt后缀的文件加入到忽略列表；3、将选中的文件和子目录中的相同文件名的文件加入忽略列表；4、不解释，按3推理。

选择第一项，在所在的目录（选择所在目录，不要选择文件）提交签入，查看TortoiseSVN做了哪些修改：



可以看出来进行了两处修改，一处是给文件的所在目录添加了svn:ignore属性，可以右键查看属性设置，发现值为我们添加的文件的文件名。

**还有一处是将刚才的文件删除，如果你这时点OK，将本次操作签入的话，文件将会被删掉**。当你下次再生成这个文件时，不会显示加号，而是会显示减号。

**如果我只想加入忽略列表，而在本地保留这些文件怎么办？**

需要先将文件复制出来，然后提交签入，签入后再将文件复制回来，真的就只有这一种方式。

如果你想加入忽略列表里的文件是刚生成的，这时候直接加入到忽略列表里，可以避免刚才的删除、复制操作。

你也可以手动在目录上添加svn:ignore属性来忽略该目录下的文件，但如果文件没有在SVN上删除，操作依然不会有效。

总结一下：你不可能保持一个添加到忽略列表里的文件在SVN上存在。

篇幅所限，其他几个知识点另起一篇来写。

[**SVN与TortoiseSVN实战：属性的奇技淫巧（二）**](https://www.cnblogs.com/ym123/p/4361473.html)

硬广：《SVN与TortoiseSVN实战》系列已经写了七篇，本系列结合TortoiseSVN对SVN中容易被忽视的部分进行了详解。

关于属性的奇技淫巧较多，分为两篇来写，第一篇详见《[SVN与TortoiseSVN实战：属性的奇技淫巧（一）](http://www.cnblogs.com/ym123/p/4360869.html)》，内容概要：

1、属性的概念和基础操作；

2、属性的关键字展开；

3、使用属性忽略某些文件；

**4、设置Mime类型；**

**5、自动设置属性；**

**6、引用外部项目；**

**7、设置签入日志限制；**

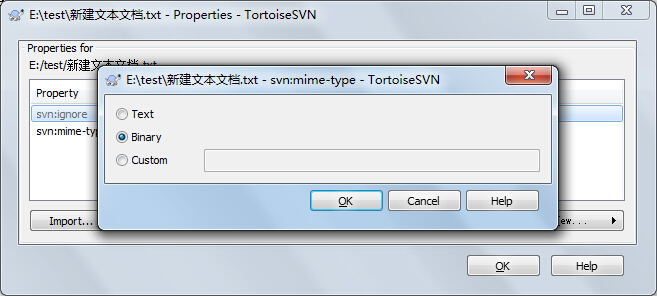
**8、属性冲突；**

**4、设置Mime类型**

SVN将项目仓库中的文件分为两种：文本文件和二进制文件。

由于二进制文件是不可读的，因此SVN对于合并文件、处理冲突时会对二进制文件进行区别对待。

使用TortoiseSVN的Properties菜单打开属性窗口，使用New按钮选择Minme-Type，如下图进行设置：



大家可以试一下设置为二进制后产生冲突与文本类型的差别。

**5、自动设置属性**

自动设置属性其实就是SVN对所有某个类型的文件进行相同的属性自动设置。

比如把所有的.java文件设置一个属性，可以使用如下配置：

|  |  |
| --- | --- |
| 1  2  3  4 | enable-auto-props = yes  [auto-props]  \*.java = svn :keywords =Date Rev Id  \*.txt = svn :keywords =LastChangedDate LastChangedRevision |

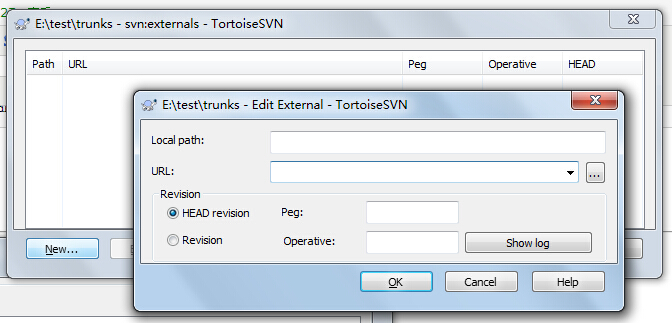
自动设置属性仅在本地有效。

**6、引用外部项目**

有时候，你的SVN中的项目可能依赖了另一个项目仓库中的代码，通常这些共用和通用的代码会有一个独立的项目仓库。

SVN提供了externals属性来把其它项目仓库中的代码纳入到你的本地拷贝中，而这些签出的外部依赖项建议作为只读拷贝，不要对其修改。

新建一个目录，使用TortoiseSVN的Properties菜单打开属性窗口，使用New按钮选择Externals：



在URL中输入外部依赖项的SVN地址即可。

**7、设置签入日志限制**

在团队管理中要求每次提交必须填写提交日志是强烈推荐的做法，可以帮助团队成员了解其他成员的修改情况，也便于做历史追踪。

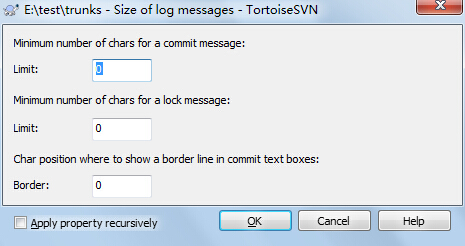
TortoiseSVN通过自有的属性，允许设置提交日志和锁定日志的最小长度。如果你输入的日志短于预设值，提交会被禁止，这两个属性如下：

tsvn:logminsize；

tsvn:lockmsgminsize；

由于是通过TortoiseSVN自有的属性来设置的，因此只有使用TortoiseSVN客户端才有效。

在项目根目录上右键，使用TortoiseSVN的Properties菜单打开属性窗口，使用New按钮选择Log Sizes：



1、2分别是签入和加锁的日志最小字符数。

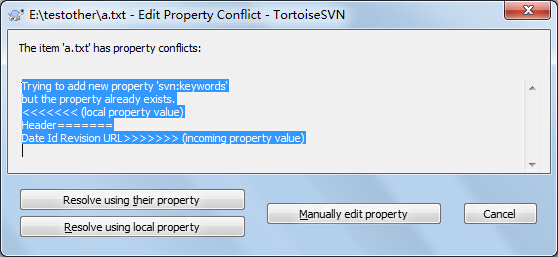
还有一种实现这种限制的方法是在SVN服务端编写钩子，不过稍微麻烦一点。

**8、属性冲突**

属性跟文件一样，不同成员同时对同样位置的修改也会产生冲突，更简单点说是对同一个属性的修改。

如果因为其他用户已经提交了同样的属性，提交时出现冲突，会产生一个 .prej 文件。

冲突的处理窗口如下：



对于冲突处理，可以参考《[SVN与TortoiseSVN实战：冲突详解（一）](http://www.cnblogs.com/ym123/p/4345771.html)》讲解的比较清楚，触类旁通。

[**SVN与TortoiseSVN实战：文件加锁详解**](https://www.cnblogs.com/ym123/p/4375549.html)

硬广：《SVN与TortoiseSVN实战》系列已经写了八篇，本篇是完结篇，整个系列结合TortoiseSVN对SVN中容易被忽视的部分进行了详解，以技巧性为主。

本篇详解使用TortoiseSVN对文件进行加锁和解锁的技巧，其中涉及到了之前介绍的SVN属性知识，关于属性可以翻看《[SVN与TortoiseSVN实战：属性的奇技淫巧（一）](http://www.cnblogs.com/ym123/p/4360869.html)》、《[SVN与TortoiseSVN实战：属性的奇技淫巧（二）](http://www.cnblogs.com/ym123/p/4361473.html)》。

加锁与解锁的操作对于项目中的二进制文件，如图片、声音、动态库等不可合并文件是非常有用的，可以让这些文件防止产生恼人的冲突，但TortoiseSVN中的Get lock的其实并不能满足需要，并不能在编辑前获得对文件的独占访问，这也是本文要结合属性来详解文件加锁的原因。

本篇内容概要：

1、版本控制系统的加锁策略：乐观加锁、严格加锁；

2、使用TortoiseSVN获取锁和释放锁；

3、使用属性，将文件设置为只读；

4、关于锁的建议；

**1、版本控制系统的加锁策略：乐观加锁、严格加锁**

为了避免多个人同一时间对同一个文件改动的相互覆盖，版本控制系统就必须有一套冲突处理机制。

对于解决冲突，有两种处理办法：乐观加锁和严格加锁。

严格加锁：所有签出的文件都是只读的，任何对文件的修改必须要获得文件的锁，如果其他人没有拥有该文件的锁，那么版本控制系统就会授权给你文件的锁，并将文件设置为可编辑的。

乐观加锁：所有签出的文件都是可读写的，对文件的修改不必获得文件的锁，当你修改完文件签入时，会首先要求你更新本地文件，版本控制系统不会覆盖你的本地修改，而是会让你自己合并冲突后签入。

SVN使用的是乐观加锁策略，而像TFS一类的版本控制系统允许你对加锁策略进行选择。

相比于严格加锁，乐观加锁可能会有较高的产生冲突的概率，但允许团队成员任何时间对文件进行编辑，拥有极大的协作效率。

对于像不可合并的二进制文件或想在某些文件和目录上采取严格加锁策略，SVN也提供了严格加锁的办法，就是对文件的加锁解锁操作。

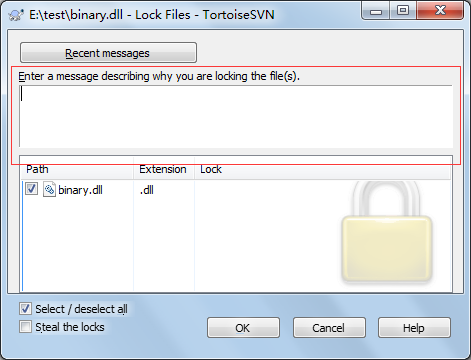
**2、使用TortoiseSVN获取锁和释放锁；**

SVN默认情况下，所有的文件都没有锁定，只要有提交权限的人都可以在任何时候提交任何的文件。其他人会定时更新他们的工作目录，在库中的改变的东西都会与本地合并。

如果你对一个文件获得锁定，那么只有你可以提交这个文件。其他用户的提交都会被拒绝，直到你释放了这个锁。

**获取锁：**

选择你想要获取锁定的文件，然后右键菜单点击TortoiseSVN Get lock，将会弹出锁定窗口：



一个良好的习惯是每次获取锁定时，在红色标注位置输入获取锁的原因。

**释放锁：**

有两种情况会释放锁：

1、选择你想要释放锁定的文件，然后右键菜单点击TortoiseSVN Release lock；

2、TortoiseSVN为了确保你不会忘记释放锁，在提交对话框中，总是会显示锁定的文件，并总是默认被选中。如果你继续提交，选中的文件中的锁就被移除了，就算你从没有修改过。可以在锁定文件的目录上使用Commit，打开提交对话框验证一下。

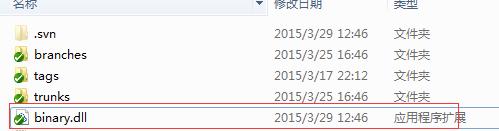
如果你不希望释放文件的锁，就必须取消选中它。

**问题是什么？**

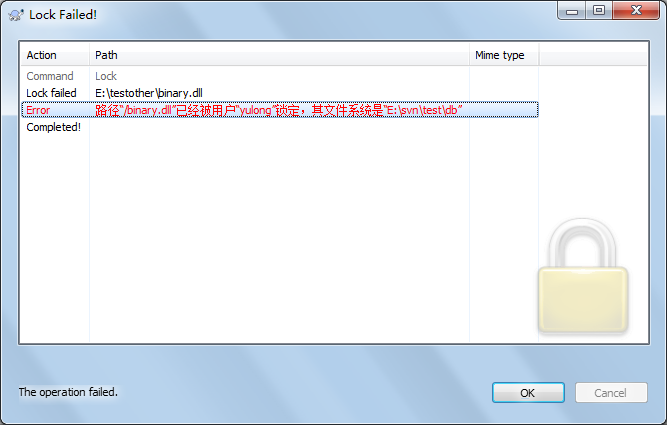
如果实际操作过，使用过TortoiseSVN 的 Get lock来加锁，就会发现并不如你所预期的，对文件实现了严格加锁，本篇也就不必要写了。

问题在哪？使用TortoiseSVN 的 Get lock锁定文件后，其他用户并不知道你对这个文件已经锁定，即时他们签出最新的版本，依然能够对文件在本地进行修改，文件依然是可读写的状态。

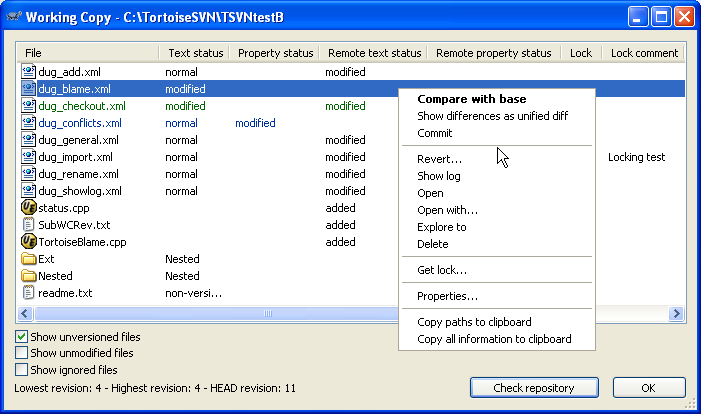
这是一个在其他成员电脑上已经Get lock的文件，可以看到与其他文件在状态上没有任何区别：



这样其实是没有意义的，别人依然会对你锁定的文件进行修改，只有当他们在签入时才会发现文件已经被锁定：



这种加锁操作下，团队成员需要定期的检查文件的锁定状态，可以使用TortoiseSVN的check for modifications：



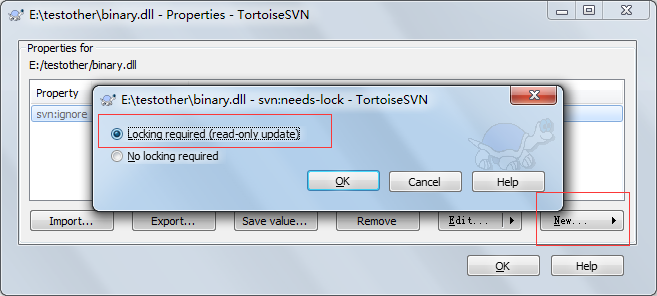
**3、使用属性，将文件设置为只读；**

之所以存在被锁定后其他成员依然可以在本地修改，是因为SVN所采取的是乐观加锁的策略。

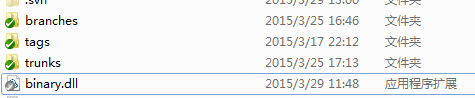
如果我们想对某个文件或某个目录采取严格加锁策略，更明确的说是：所有文件默认为只读，在文件修改前需要先获取锁，只有其他人不拥有锁时才可以锁定文件并修改。

要实现严格加锁，就需要使用SVN的svn:needs-lock属性。

在想要采取严格加锁的文件或目录上点击右键，使用TortoiseSVN Properties属性菜单，点击News新建属性，选择Needs-lock：



设置为Locking required，签入属性修改操作，可以发现文件的图标状态改变了，并变为只读：



当设置了svn:needs-lock属性时，实际上对文件就采取严格加锁策略，对文件的任何修改都必须先使用TortoiseSVN 的 Get lock锁定文件。

**4、关于锁的建议**

对于项目中的二进制文件，如图片、声音、动态库等，可以在目录上统一设置svn:needs-lock属性，避免在这些文件上产生冲突，浪费解决冲突的时间。