如何配置SVN服务器

文章摘要：如何快速建立Subversion服务器，并且在项目中使用起来，这是大家最关心的问题，与CVS相比，Subversion有更多的选择，也更加的容易，几个命令就可以建立一套服务器环境，可以使用起来，可。 本文是使用Subversion最快速的教程，在最短的时间里帮助您建

如何快速建立Subversion服务器，并且在项目中使用起来，这是大家最关心的问题，与CVS相比，Subversion有更多的选择，也更加的容易，几个命令就可以建立一套服务器环境，可以使用起来，这里配套有动画教程。

本文是使用Subversion最快速的教程，在最短的时间里帮助您建立起一套可用的服务器环境，只需略加调整就可以应用到实际项目当中。

本教程分为以下几个部门，不仅仅是快速入门，最后我们还有一些高级功能的说明，为了说明简单，教程是在windows下使用的方式，以方便资源有限的项目使用，对于UNIX环境下，区别并不大。

软件下载

服务器和客户端安装

建立版本库（Repository）

配置用户和权限

运行独立服务器

初始化导入

基本客户端操作

1，软件下载

下载Subversion服务器程序。

到官方网站的下载二进制安装文件，来到二进制包下载部分，找到 Windows NT, 2000, XP and 2003部分，然后选择" this directory "，这样我们可以看到许多下载的内容，目前可以下载 svn-1.4.0-setup.exe 。

下载Subversion的Windows客户端TortoiseSVN。

TortoiseSVN是扩展Windows Shell的一套工具，可以看作Windows资源管理器的插件，安装之后Windows就可以识别Subversion的工作目录。

官方网站是TortoiseSVN ，下载方式和前面的svn服务器类似，在Download页面的我们可以选择下载的版本，目前的最高稳定版本的安装文件为TortoiseSVN-1.4.0.7501-win32-svn-1.4.0.msi。

2，服务器和客户端安装

服务器安装，直接运行svn-1.4.0-setup.exe ，根据提示安装即可，这样我们就有了一套服务器可以运行的环境。

安装TortoiseSVN，同样直接运行TortoiseSVN-1.4.0.7501-win32-svn-1.4.0.msi按照提示安装即可，不过最后完成后会提示是否重启，其实重启只是使svn工作拷贝在windows中的特殊样式生效，与所有的实际功能无关，这里为了立刻看到好的效果，还是重新启动机器。

3，建立版本库（Repository）

运行Subversion服务器需要首先要建立一个版本库（Repository），可以看作服务器上存放数据的数据库，在安装了Subversion服务器之后，可以直接运行，如：

svnadmin create E:svndemorepository

就会在目录E:svndemorepository下创建一个版本库。

我们也可以使用TortoiseSVN图形化的完成这一步：

在目录E:svndemorepository下"右键->TortoiseSVN->Create Repository here...“， 然后可以选择版本库模式， 这里使用默认即可， 然后就创建了一系列目录和文件。

4，配置用户和权限

来到E:svndemorepositoryconf目录，修改svnserve.conf：

# [general]

# password-db = passwd

改为：

[general]

password-db = passwd 然后修改同目录的passwd文件，去掉下面三行的注释：

# [users]

# harry = harryssecret

# sally = sallyssecret

最后变成：

[users]

harry = harryssecret

sally = sallyssecret

5，运行独立服务器

在任意目录下运行：

svnserve -d -r E:svndemorepository 我们的服务器程序就已经启动了。注意不要关闭命令行窗口，关闭窗口也会把svnserve停止。

6，初始化导入

来到我们想要导入的项目根目录，在这个例子里是E:svndemoinitproject，目录下有一个readme.txt文件：

右键->TortoiseSVN->Import...

URL of repository输入“svn://localhost/”

ok

完成之后目录没有任何变化，如果没有报错，数据就已经全部导入到了我们刚才定义的版本库中。

需要注意的是，这一步操作可以完全在另一台安装了TortoiseSVN的主机上进行。例如运行svnserve的主机的IP是133.96.121.22，则URL部分输入的内容就是“svn://133.96.121.22/”。

7，基本客户端操作

取出版本库到一个工作拷贝：

来到任意空目录下，在本例中是E:svndemowc1，运行右键->Checkout，在URL of repository中输入svn://localhost/，这样我们就得到了一份工作拷贝。

在工作拷贝中作出修改并提交：

打开readme.txt，作出修改，然后右键->Commit...，这样我们就把修改提交到了版本库，我们可以运行。

察看所作的修改：

readme.txt上右键->TortoiseSVN->Show Log，这样我们就可以看到我们对这个文件所有的提交。在版本1上右键->Compare with working copy，我们可以比较工作拷贝的文件和版本1的区别。

SVN的安装和配置

工具：

1. Windows 2003 sp1

2. svn-1.3.0-setup.exe SVN服务端软件

3. SVNService.exe 添加到Windows服务需要的工具

4. TortoiseSVN-1.4.0.7501-win32-svn-1.4.0.msi SVN客户端软件

5. LanguagePack-1.4.0.7501-win32-zh\_CN.exe SVN客户端软件的中文语言包

相关下载：

1．Subversion

http://subversion.tigris.org

2．客户端软件TortoiseSVN

http://tortoisesvn.tigris.org

Eclipse 下使用的插件,需要在线更新

更新地址: http://subclipse.tigris.org/update\_1.0.x

安装指南: http://subclipse.tigris.org/install.html

安装SVN服务器

1. 安装服务器软件

2. 将SVN添加到Windows服务

1. 新建一个文件夹用来存放库文件 c:\SVN

2. 运行命令:SVNService -install -d -r c:\svn 安装 Windows 服务

3. 通过Windows服务管理平台启动SVN服务

4. 其它命令

1. 改变服务设置:

运行命令:SVNService -setup -d -r <newsvnroot>

在实际操作中,<newsvnroot>代表与SubVersion服务启动时不同的电子仓库目录.

2. 删除服务:

运行命令:SVNService -remove

安装客户端工具

因为安装TortoiseSVN 需要重启系统，所以在这里略过安装步骤

配置SVN

1. 配置服务器

打开仓库目录下的 conf/svnserve.conf 文件

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[general]

# anon-access = read

# auth-access = write

password-db = passwd

authz-db = authz

--------------------------------------------

去掉 password-db = passwd 和 authz-db = authz 的注释

前两行为匿名用户的权限

后两行为启用配置文件来对用户和权限进行管理

注意：所有配置信息必须顶格

2. 配置用户

打开仓库目录下的 conf/passwd 文件

[users]

admin=admin 等号左边为用户名，右边为密码

新增用户只需要在这里直接增加即可，如： oksonic = 123456

增加好用户以后需要重启SVN服务器

3. 配置工作组

打开仓库目录下的 conf/authz 文件，文件内容如下：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# [groups]

# harry\_and\_sally = harry,sally

# [/foo/bar]

# harry = rw

# \* =

# [repository:/baz/fuz]

# @harry\_and\_sally = rw

# \* = r

--------------------------------------------

因为所有配置信息都被注释掉了，所以所有的用户都有管理员的权限(所有目录的读/写)

[groups] 这里的内容为组的配置

把已经建好的两个用户合并为一个分组，用户名之内以逗号分隔，配置内容如下：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[groups]

admin\_group = admin,oksonic

--------------------------------------------

4. 配置权限

SVN 的权限只有两个，1 是只读，2 是读/写

例：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[/]

# 指定所有的仓库默认只读

\* = r

# 指定 admin\_group 组的用户对库中所有内容具有读/写权限

@admin\_group = rw

--------------------------------------------

[/] ：这里代表的仓库的目录，此为根目录

\* = r : \* 号代表所有的用户， r 为只读

现在需要建一个项目到仓库中

1. 在桌面建一个文件夹，如：SVN测试

2. 在目录中新建几个文件和文件夹以作测试

3. 使用客户端工具把目录中的所有文件加入到SVN服务器中的 test 目录中

补充哈:

SVN1.4.3 安装和配置:

1)下载svn-1.4.3-setup.exe,安装的目录为: D:\Subversion ; 仓库目录: F:\Major\Java\ExerciseSVN

2)由于1.4版本的更新,不需要SVNService.exe(添加到Windows服务需要的工具),先创建SVN服务,命令如下:

D:\>sc create SVNService binpath= "\"D:\Subversion\bin\svnserve.exe\" --service --root \"F:\Major\Java\ExerciseSVN\"" displayname= "Subversion Server" depend= Tcpip start= auto

其中SVNService为服务名, start属性为确定SVNService服务的属性,此为自动启动, 注意=左右边的空格,还有"";

启动服务: D:\> net start SVNService 停止:D:\> net stop SVNService

3)创建仓库, D:\>svnadmin create F:\Major\Java\ExerciseSVN

其实见SVN1.4帮助文档

Windows下Subversion配置管理员指南

作者： rocksun 　来源：Subversion

Subversion安装成service

　以前的svnserve要想成为windows服务，必须依赖于svnservice或其他工具。从Subversion1.4开始，Subversion本身就集成Windows服务的工具。

1，安装svnservice

　在Windows NT中（包括Windows XP, Windows 2000, Windows 2003 Server）本身包含了一个安装服务的工具，叫做"Service Control"，也就是sc.exe。

例如我的Subversion安装在"D:\Subversion"，版本库在"D:\svnroot"，而我希望对应的Subversion服务名为svnservice，安装这个svn服务的命令就可以这样写：

sc create svnservice

binpath= "D:\Subversion\bin\svnserve.exe --service -r D:\svnroot"

displayname= "SVNService"

depend= Tcpip

请注意，因为便于察看，上面的命令分为多行，但在实际执行时应该在一行里。另外，在以前启动svnserve时会使用"-d"选项，也就是守护进程模式，在这里不能使用，会导致服务无法启动。同样，"-i"和"-t"选项也不能使用。

在命令行窗口执行完这个命令之后，服务还没有启动，你可以继续运行"net start svnservice"启动这个服务，然后使用"net stop svnservice"停止服务。

另外还有两点需要小心处理。首先，如果路径中包括空格，一定要用“\”处理“"”号，例如上面的例子中如果svnserve.exe在“c:\program files\subversion\”中，则命令应该写为“binpath= "\"c:\program files\subversion\bin\svnserve.exe\"”（“”中的内容），整个命令如下，红色部分是改变部分：

sc create svnservice

binpath= "\"D:\program files\Subversion\bin\svnserve.exe\" --service -r D:\svnroot"

displayname= "SVNService"

depend= Tcpip

其次，sc对选项的格式还有要求，例如“depend= Tcpip”不能写为“depend = Tcpip”或“depend=Tcpip”，也就是“=”前不能有空各，而后面必须有空格。

2，删除服务

　如果服务安装的有问题，你可能需要删除服务。要删除前面添加的服务，只需要运行"net start svnservice"，"svnservice"就是我们创建服务时使用的名字。

3，配置服务是自动启动

　默认情况下安装的服务不会随Windows的启动而启动，为了使svn服务能够随Windows启动而启动，需要修改一下"sc create"命令（首先要删除），增加"start= auto"选项：

sc create svnservice

binpath= "D:\Subversion\bin\svnserve.exe --service -r D:\svnroot"

displayname= "SVNService"

depend= Tcpip

start= auto

当然你也可以使用图形化的工具修改服务的属性，你可以在“开始->运行...”中执行"services.msc"，然后在界面中修改。

Subversion的权限控制

　1，认证（Authentication）和授权（Authorization）

　这两个术语经常一起出现。其中认证的意思就是鉴别用户的身份，最常见的方式就是使用用户名和密码，授权就是判断用户是否具备某种操作的权限，在 Subversion里提供了“authz-db”文件，实现了以路径为基础的授权，也就是判断用户是否有操作对应路径的权限，在Subversion 1.3之后，svnserve和Apache一样都可以使用“authz-db”文件。

2. svnserve下的配置文件

　因为本文是以svnserve为例的，所以先介绍一下版本库目录的结构：

D:\SVNROOT\PROJECT1

├─conf

├─dav

├─db

│ ├─revprops

│ ├─revs

│ └─transactions

├─hooks

└─locks

其中conf下面有三个文件：

authz

passwd

svnserve.conf

其中的“svnserve.conf”是这个版本库的配置文件，当使用svnserve时，这个配置文件决定了使用什么认证和授权文件：

password-db = passwd

authz-db = authz

上面的配置说明使用“svnserve.conf”同目录的passwd和authz，其中的password-db指定了用户密码文件，authz-db是我们的授权文件，也就是我们本文主要介绍的文件。

注意：使用Apache作为服务器时，根本就不会参考“svnserve.conf”文件的内容，而是会参考Apache的配置。

3，基于svnserve的版本库文件布局

　使用svnserve时，为了管理的方便，应该使用相同的认证和授权文件，所以应该让所有版本库的配置文件svnserve.conf指向同一个password-db和authz-db文件。下面是一个多版本库的目录：

D:\SVNROOT

├─project1

│ ├─conf

│ ├─dav

│ ├─db

│ │ ├─revprops

│ │ ├─revs

│ │ └─transactions

│ ├─hooks

│ └─locks

└─project2

├─conf

├─dav

├─db

│ ├─revprops

│ ├─revs

│ └─transactions

├─hooks

└─locks

D:\SVNROOT下有两个目录project1和project2，都已经创建了版本库，所以我们修改每个conf目录下的svnserve.conf，使之指向同一个password-db和authz-db文件。

password-db = ..\..\passwd

　authz-db = ..\..\authz这样，D:\SVNROOT\passwd和D:\SVNROOT\authz就控制了所有版本库的svnserve访问。另外在后面的操作中要关闭匿名访问，应该去掉“anon-access = none”前的“#”号，保证只有认证用户可以访问。

注意：还有一点需要注意，那就是svnserve的“realm”的值，在上面的设置下，应该保证所有的版本库使用相同的realm值，这样，对版本库的密码缓存可以在多个版本库之间共享，更多细节见客户端凭证缓存。

4，测试用户和组说明

　版本库禁止任何匿名用户的访问，只对认证用户有效。

root:配置管理管理员，对版本库有完全的管理权限。

p1\_admin1:project1的管理员，对project1有完全权限。

　p1\_d1:project1的开发者，对project1的trunk有完全的权限，但是对其中的/trunk/admin目录没有任何权限。

　p1\_t1:project1的测试者，对project1的trunk有完全的读权限，但是对其中的/trunk/admin目录没有任何权限。

p2\_admin1:project2的管理员，对project2有完全权限。

　p2\_d1:project2的开发者，对project2的trunk有完全的权限，但是对其中的/trunk/admin目录没有任何权限。

　p2\_t1:project2的测试者，对project2的trunk有完全的读权限，但是对其中的/trunk/admin目录没有任何权限。

对应的组及组的用户：

p1\_group\_a:p1\_admin1

p1\_group\_d:p1\_d1

p1\_group\_t:p1\_t1

p2\_group\_a:p2\_admin1

p2\_group\_d:p2\_d1

p2\_group\_t:p2\_t1

5，修改D:\SVNROOT\passwd文件

前面已经说过了，用户和密码文件应该是在D:\SVNROOT\passwd，所以我们为每一位用户设置权限，文件内容如下：

[users]

p1\_admin1 = p1\_admin1

p1\_d1 = p1\_d1

p1\_t1 = p1\_t1

p2\_admin1 = p2\_admin1

p2\_d1 = p2\_d1

p2\_t1 = p2\_t1为了便于验证，所有密码和用户名一致，如果你使用的是其他认证方式，这一步可能不同，但是用户名应该都是一样的。

6，配置授权，修改D:\SVNROOT\authz

[groups]

# 定义组信息

p1\_group\_a = p1\_admin1

p1\_group\_d = p1\_d1

p1\_group\_t = p1\_t1

p2\_group\_a = p2\_admin1

p2\_group\_d = p2\_d1

p2\_group\_t = p2\_t1

[/]

# 指定所有的版本库默认只读，root可读写

\* = r

root = rw

[project1:/]

# 指定对版本库project1根目录的权限

@p1\_group\_a = rw

@p1\_group\_d = rw

@p1\_group\_t = r

[project1:/trunk/admin]

# 指定对版本库project1的/trunk/admin根目录的权限，

# p1\_group\_a读写，p1\_group\_d和p1\_group\_t没有任何权限。

@p1\_group\_a = rw

@p1\_group\_d =

@p1\_group\_t =

[project2:/]

# 指定对版本库project2根目录的权限

@p2\_group\_a = rw

@p2\_group\_d = rw

@p2\_group\_t = r

[project2:/trunk/admin]

# 指定对版本库project1的/trunk/admin根目录的权限

@p2\_group\_a = rw

@p2\_group\_d =

@p2\_group\_t =

经过以上设置以后，你会发现一些有趣的事情。当使用用户“p1\_d1”，检出project1的trunk时，目录是空的，好像admin目录根本不存在一样，当使用p1\_d1用户浏览版本库时，能够看到admin目录，但是其中的内容却无法看到。

关于中文目录，也是没有问题的，只是注意要把authz文件转化为UTF-8格式，在我的WINXP的UltraEdit里显示的文件格式为U8-DOS，具体的做法是用UltraEdit打开authz文件，然后选择“文件->转换->ASCII转UTF-8”，然后保存。

再复杂的情况也不过如此，在实际的工作中要首先规划好权限，只赋给用户最小的权限，保证以最小的配置实现最复杂的权限控制。

Subversion备份

　版本控制最关键的一件事是保证数据的安全性，不能因为磁盘损坏，程序故障造成版本库无可挽回的错误，为此必须制定较完备的备份策略。在Subversion中，我们有三种备份方式：完全备份，增量备份和同步版本库。

1, 完全备份

　最常见和简单的备份就是直接使用拷贝命令，将版本库目录拷贝到备份目录上，就可以了。但是这样不是很安全的方式，因为如果在拷贝时版本库发生变化，将会造成备份的结果不够准确，失去备份的作用，为此Subversion提供了“svnadmin hotcopy”命令，可以防止这种问题。

还记得我们的版本库目录吗？

D:\SVNROOT

├─project1

│ ├─conf

│ ├─dav

│ ├─db

│ │ ├─revprops

│ │ ├─revs

│ │ └─transactions

│ ├─hooks

│ └─locks

└─project2

├─conf

├─dav

├─db

│ ├─revprops

│ ├─revs

│ └─transactions

├─hooks

└─locks

如果要把project1备份到d:\svnrootbak目录下，只需要运行：

svnadmin hotcopy d:\svnroot\project1 d:\svnrootbak\project1

但是我们作为配置管理员，必须想办法优化这个过程，如果我们这个目录下有许多版本库，需要为每个版本库写这样一条语句备份，为此我写了下面的脚本，实现备份一个目录下的所有版本库。我们在D:\SVNROOT下创建了两个文件，simpleBackup.bat：

@echo 正在备份版本库%1......

　@%SVN\_HOME%\bin\svnadmin hotcopy %1 %BACKUP\_DIRECTORY%\%2

　@echo 版本库%1成功备份到了%2！

这个文件仅仅是对“svnadmin hotcopy”的包装，然后是backup.bat：

echo off

rem Subversion的安装目录

set SVN\_HOME="D:\Subversion"

rem 所有版本库的父目录

set SVN\_ROOT=D:\svnroot

rem 备份的目录

set BACKUP\_SVN\_ROOT=D:\svnrootbak

set BACKUP\_DIRECTORY=%BACKUP\_SVN\_ROOT%\%date:~0,10%

if exist %BACKUP\_DIRECTORY% goto checkBack

echo 建立备份目录%BACKUP\_DIRECTORY%>>%SVN\_ROOT%/backup.log

mkdir %BACKUP\_DIRECTORY%

rem 验证目录是否为版本库，如果是则取出名称备份

for /r %SVN\_ROOT% %%I in (.) do @if exist "%%I\conf\svnserve.conf" %SVN\_ROOT%\simpleBackup.bat "%%~fI" %%~nI

goto end

:checkBack

echo 备份目录%BACKUP\_DIRECTORY%已经存在，请清空。

goto end

:end

你在使用的时候，只需要修改backup.bat开头的三个路径，将两个脚本拷贝到“SVN\_ROOT”下就可以了。根据以上的配置，你只需要运行 backup.bat，就可以把“SVN\_ROOT”下的版本库都备份到“BACKUP\_SVN\_ROOT”里，并且存放在备份所在日的目录里，例如 “D:\svnrootbak\2006-10-22”。

虽然这部分工作很简单，可是必须有人定时地去执行这个操作（例如每周一凌晨），为了避免发生遗忘的情况，我们可以将这个操作加入到系统的at任务当中去，例如还是上面的环境，为了安装at任务，我们运行：

at 1:00 /every:M D:\svnroot\backup.bat这样在每周一凌晨1:00都会执行这个备份过程。当然备份在本机也是不安全的，你也许需要上传到别的机器，这个就要靠你自己去实现了。

2, 增量备份

　尽管完全备份非常简单，但是也是有代价的，当版本库非常巨大时，经常进行完全备份是不现实的，也并不必要，但是一旦版本库在备份之间发生问题，该如何呢，这里我们就用到了增量备份。

增量备份通常要与完全备份结合使用，就像oracle数据库的归档日志，记录着每次Subversion提交的变化，然后在需要恢复时能够回到最新的可用状态。在我们这个例子中我们使用的是，svnadmin dump命令进行增量的备份，使用方法是：

svnadmin dump project1 --revision 15 --incremental > dumpfile2

上面的命令实现了对修订版本15进行增量的备份，其中的输出文件dumpfile2只保存了修订版本15更改的内容。

为了记录每次提交的结果，我们需要使用一项Subversion的特性--钩子（hook），看看我们的project1目录：

├─project1

│ ├─conf

│ ├─dav

│ ├─db

│ │ ├─revprops

│ │ ├─revs

│ │ └─transactions

│ ├─hooks

│ └─locks

其中的hooks目录里存放的就是钩子脚本，我们在此处只使用post-commit钩子，这个钩子会在每次提交之后执行，为了实现我们的备份功能，我们在hooks下建立一个文件post-commit.bat，内容如下：

echo off

set SVN\_HOME="C:\Program Files\Subversion"

set SVN\_ROOT=D:\svnroot

set UNIX\_SVN\_ROOT=D:/svnroot

set DELTA\_BACKUP\_SVN\_ROOT=D:\svnrootbak\delta

set LOG\_FILE=%1\backup.log

echo backup revision %2 >> %LOG\_FILE%

for /r %SVN\_ROOT% %%I in (.) do if D:/svnroot/%%~nI == %1 %SVN\_ROOT%\%%~nI\hooks\deltaBackup.bat %%~nI %2

goto end

:end

通过这个脚本，可以实现D:\svnroot下的版本库提交时自动增量备份到D:\svnrootbak\delta（确定这个目录存在），其中使用的deltaBackup.bat其实可以放在任何地方，只是对脚本的svnadmin dump的包装，内容如下：

@echo 正在备份版本库%2......

　%SVN\_HOME%\bin\svnadmin dump %SVN\_ROOT%\%1 --incremental --revision %2 >> %DELTA\_BACKUP\_SVN\_ROOT%\%1.dump

　@echo 版本库%2成功备份到了%3！

以上两个脚本可以直接拷贝到project2的hooks目录下，不需要修改就可以实现project2的自动备份。

以上的操作已经OK了，现在需要做的是将完全备份和增量备份结合起来，也就是在完全备份后清理增量备份的结果，使之只保存完全备份后的结果。

当果真出现版本库的故障，就要求我们实现版本库的恢复操作了，这是用要使用svnadmin load命令，同时也需要上次的完全备份例如要把上次完全备份backuprepo，和之后的增量备份dumpfile：

svnadmin load backuprepo < dumpfile

最后的结果，可以下载svnroot.rar，将之解压缩到d:\下，然后修改几个bat文件的SVN\_HOME就可以使用了。

3, 版本库同步

Subversion 1.4增加了同步机制，可以实现一个版本库同另一个版本库的同步（但好像只是单向的），我们可以用来实现版本库的备份或镜像。

3.1. 对目标库初始化

　svnsync init svn://localhost/project2 svn://localhost/project1

　其中project2是目标的版本库，而project1是源版本库。其中的目标版本库必须为空，而且必须允许修订版本属性的修改，也就是在目标的版本库的hooks目录里添加一个文件pre-revprop-change.bat，内容为空即可。

3.2. 同步project2到project1

　svnsync sync svn://localhost/project2

　这时候你update一下你的project2的一个工作拷贝，就会发现有了project1的所有内容。如果project1又有提交，这时候 project2的版本库无法看到最新的变化，还需要再运行一遍sync操作，这样才能将最新的变化同步。需要注意的是，目标版本库只能做成只读的，如果目标版本库发生了变更，则无法继续同步了。

3.3. 同步历史属性的修改

　因为同步不会更新对历史属性的修改，所以svnsync还有子命令copy-revprops，可以同步某个版本的属性。

3.4. 钩子自动同步

　希望在每次提交时同步，则需要在源版本库增加post-commit脚本，内容如下：

echo off

set SVN\_HOME="D:\Subversion"

%SVN\_HOME%\bin\svnsync sync --non-interactive svn://localhost/project2

把以上内容存放为post-commit.bat，然后放到版本库project1下的hooks目录下，这样project1每次提交，都会引起project2的同步。