SVN 分支和合并简介

比较高效的利用 svn 提供的各种功能，需要对 svn 的版本号，分支，合并有所了解。这里主要介绍了这些，以及分支和合并的具体使用方法。

主要内容:

一、基本概念

二、使用举例

三、svn 命令

一、基本概念

高效使用 svn 的关键之一是要理解 svn 中分支和合并的概念。

所谓版本号，是指整个 svn 库的版本号，而不是哪个工作拷贝，那个工作目录的版本号。看完后面会逐渐理解这个概念的。

所谓分支，就是一个拷贝，可以对库内任意子目录进行拷贝，然后在于主线互不影响的前提下对这个拷贝进行任意修改，需要的时候可以把任意两个时间之内的修改合并到主线上。需要注意的是：

（1）客户端检出的时候，可以把检出的分支当做独立的 svn 库来开发，不会影响开发主线。分支和主线为一共享的就是版本号，以及分支之前的开发日志。

（2）分支拷贝，是廉价的拷贝，它只保存创建分支以来修改的内容（却可以当做独立的库来看待）。不用的分支完全可以删除，也可在任何时候找回。

所谓合并，就是把某个分支（或者主干上），某个期间进行的修改动作（即两个版本号之间相对于该分支上的修改），应用到任意分支（或者主线）上，应用的结果变成了新的版本。需要注意的是：

（1）合并的时候，要保证本次合并的内容和其他人合并的内容没有冲突。如果开发前安排每个开发人员所做的修改完全是在他们自己独立的模块中的修改的话，就不会出现交叉修改的情况，也不会有冲突。svn 提供的机制，可以实现冲突可以在合并之前检查出来，也可以在发生冲突之后解决。

（2）合并的时候，最好记录本次应用的是合并操作，这样可以防止多次合并同一个动作。

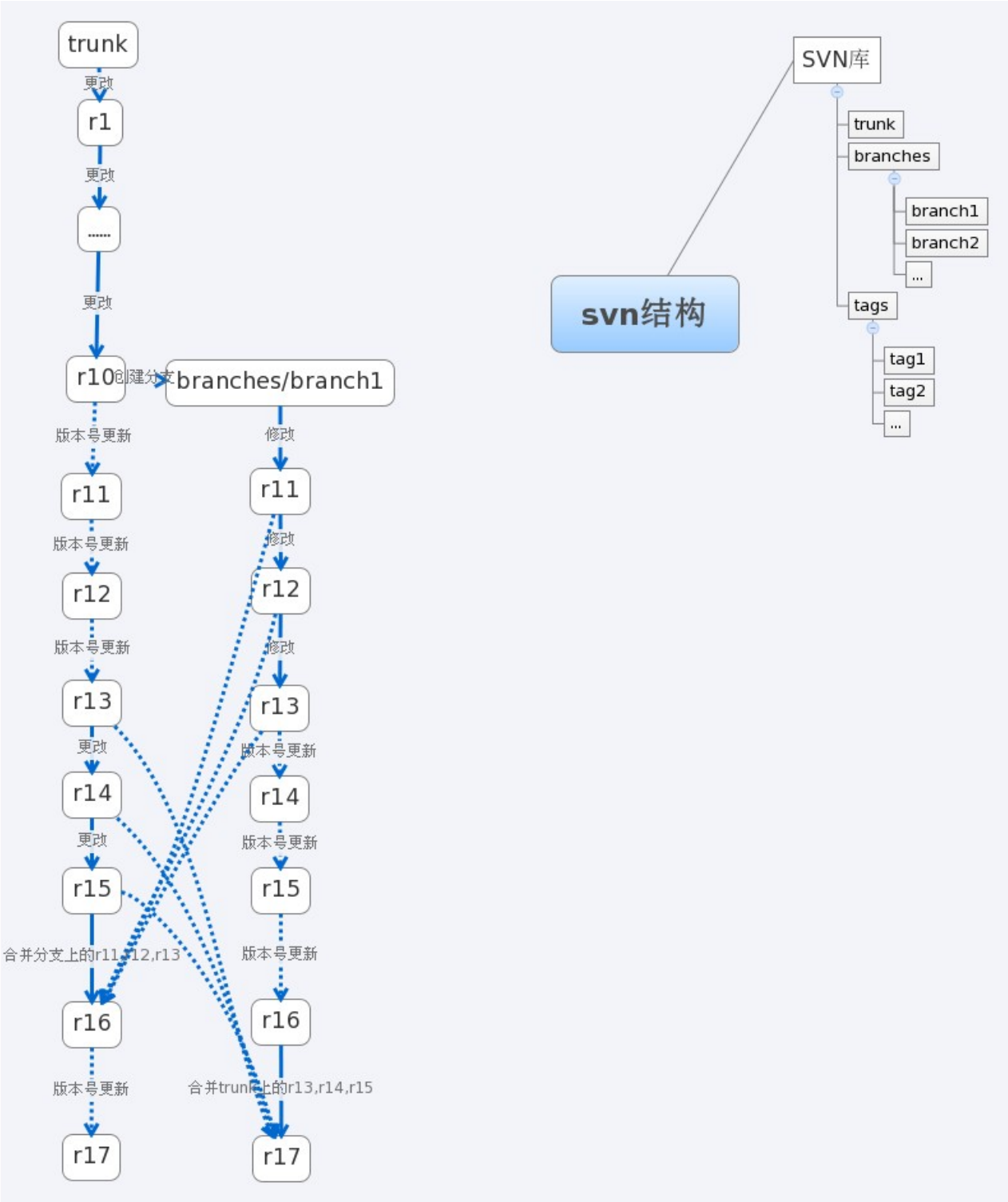
以上具体参见：http://i18n-zh.googlecode.com/svn-history/r734/www/svnbook-1.4/ch04.html

二、使用举例

1、使用分支和合并的大体过程和原理

这里先给出一个直观的例子，便于概念上的理解。后面是具体的一个使用例子。

图示如下:



在图中：右面描述常用的 svn 库目录结构；左面描述使用分支和合并的过程。

右边图中，

trunk 目录：其内容就是整个 base 的内容，是创建分支或者快照的源，也是一些项目中做为最终 release 时合并的目标（例如我们的项目）。

branches 目录：其内容是 trunk 的“拷贝”。目的就是作为一个临时修改、提交的工作目录。这里拷贝如前面所述是廉价的拷贝，具体说就是：对 branch 进行 checkout，出来的是一个完整的拷贝，但提交时实质在 svn 库中优化了存储方法，只存放本分支上的修改。 也由于这个特点，我们可以随意创建分支，随意修改，不用担心会影响到其他人的工作拷贝，也

不用担心 svn 库的空间问题（但是注意过于随意会造成管理的混乱^_^）。

tags 目录：实际它就是分支，只是 tag 中的内容是用来方便的将版本库中某个时期的快照取出，而不像 branches 是为了修改；其目的只是为了快速方便（例如某一次 release 时的代码库"快照"）。所以主要应该知道什么是 branch 就行了。

左面图中，
纵向表示时间的延续，用版本号 rN 来表示，
纵向实线相连接的节点表示下面的节点是上面的节点修改生成的，
纵向虚线连接的节点表示下面的节点和上面的节点内容一样，不同的只是一个版本号。
开发主干就是 trunk，分支就是 branch.

这样，从左面图可看出，无论主干还是分支，同一个时间，其最新的版本都是一样的，即整个 svn 库的版本号。 图中描述的具体内容如下：

（1）在初始开发主干上管理代码
最初使用的是 trunk 来开发，
直到版本 10，都是直接将修改提交到 trunk 上。
期间涉及到的 svn 命令可能包括：checkout,update,add,delete,status,commit.

（2）创建分支
从 r10 开始，创建了一个 branch1 分支。
期间涉及到的 svn 命令可能包括：copy,status,commit

（3）在分支上进行修改
r11,r12,r13 都是在 branch1 上面的提交产生，
同时 trunk 上面也相应产生 r11,r12,r13 但是 trunk 上的 r11,r12,r13 内容都和其 r10 一样只是号码不同。branch1 上面的修改并没有影响到 trunk 上面.
期间涉及到的 svn 命令可能包括：update,add,delete,status,commit.

（4）同时在开发主干上也进行修改
r14,r15 都是在 trunk 上面的提交产生的
同时 branch1 上面也相应产生 r14,r15 但是 branch1 的 r14,r15 都和其 r13 一样只是号码不同。trunk 上面的修改并没有影响到 branch1 上面。
期间涉及到的 svn 命令可能包括：update,add,delete,status,commit.

（5）合并分支上的修改到开发主干上
过了一段时间之后，为了让 trunk 上面包含所有分支上的修改，将 branch1 上面的 r11,r12,r13 期间做的修改应用到 trunk 上面，同时提交生成 r16.
这时候，trunk 就包含了以前在 branch1 上面 r11~r13 期间做过的修改。
期间涉及到的 svn 命令可能包括：update,log,merge,commit.

（6）合并最新主干上的更改
过了一段时间之后，为了让 branch1 上的内容不和 trunk 偏差太多，应该让 branch1 包含 trunk 上面有而 branch1 上没有的修改；将 trunk 上面的 r14,r15 修改应用到 branch1 上面，同时生成 r17.
注意，这里生成的是 r17,因为 r16 已经在之前 branch1 向 trunk 合并的时候生成了。
期间涉及到的 svn 命令可能包括：update,log,merge,commit.

总结，
上面图中，所描述的就是一般使用 branch 和 trunk 的思路。虽然例子只有一个分支，看不出什么，但是不难想像，如果有多个分支的话，肯定比只在一个主干上进行开发要好的多。相对优点包含一下：

（1）无论是 trunk 还是个个 branch，它们之间都相互独立，没有任何依赖关系。

（2）每个 branch 和 trunk 共享同一套版本号体系，这个版本号准确地说应该是 svn 库的版本号。

（3）每个 branch 和 trunk，都可以将自己某段时间的修改动作（批量）应用到其他

branch 或者 trunk 上。
（4）可以在任何时刻取出任何分支或者主干任何版本的工作拷贝。
（5）可以查看任何时刻任何分支（包括已删除的分支）或者主干上的全部修改日志。

2、一个使用分支和合并的具体例子
（1）查看服务器上面的 svn 库信息：

```
$cd /home/quietheart/svn_repos/test_repos
$du -sh
1.8M
$cd $_
```

（2）查看当前 svn 库最新 checkout 的工作拷贝目录信息

```
$cd /home/quietheart/test/svn_study/test
$tree
.
|-- branches
|-- tag
`-- trunk
    |-- busybox-1.4.2.tar.bz2
    `-- test_char_driver
        `-- 00_widget_dev
            |-- Makefile
            |-- readme
            |-- testDemo
            |   `-- main.c
            |-- widgetdrv_api.h
            |-- widgetdrv_simple.c
            `-- widgetdrv_simple.h
$du -sh
2.0M
```

（3）创建一个分支
创建一个分支，需要切换到某个工作拷贝目录中，具体过程和例子如下：

```
$pwd
/home/quietheart/test/svn_study/test
$ svn copy trunk/ branches/quietheart_branch
A         branches/quietheart_branch
$ svn status
A  +   branches/quietheart_branch     <===这里'+'表示是用 svn copy 创建的一个工作拷贝。
$ tree
.
|-- branches
|   `-- quietheart_branch
|       |-- busybox-1.4.2.tar.bz2
|       `-- test_char_driver
|           `-- 00_widget_dev
|               |-- Makefile
|               |-- readme
|               |-- testDemo
|               |   `-- main.c
|               |-- widgetdrv_api.h
|               |-- widgetdrv_simple.c
```

```
|           `-- widgetdrv_simple.h
|-- tag
`-- trunk
    |-- busybox-1.4.2.tar.bz2
    `-- test_char_driver
        `-- 00_widget_dev
            |-- Makefile
            |-- readme
            |-- testDemo
            |   `-- main.c
            |-- widgetdrv_api.h
            |-- widgetdrv_simple.c
            `-- widgetdrv_simple.h
$du -sh
4.0M
$cd /home/quietheart/svn_repos/test_repos
$du -sh
1.8M
$cd $_
```

由上可知，创建的分支，实际就是一个 trunk 上的拷贝，

虽然在 checkout 分支之后，本地工作拷贝中，test 目录中的大小变化了(增大了一倍多)，但实际上，服务器上面 svn 库"test_repos"中大小却没有多大变化。所以 svn 的创建分支是“廉价”的拷贝。

（4）合并主干的内容到分支

创建分支之后，我们可以只将分支 checkout 出来，然后在分支上面修改，提交，这样完全不会影响主干上面的内容，可以和主干上面的内容相互独立地并行更新。

当分支之间，或者主干和分支之间，需要对方最新修改的时候，只需要将对方中修改过的最新内容合并到自己相应没有修改的地方即可。 如：

```
$svn merge -r 6:7 file:///home/quietheart/svn_repos/test_repos/mystudy/test/trunk
/home/quietheart/test/svn_study/test/branches/quietheart_branch/
```

这样，会把主干上版本 6:7 上面的修改合并到分支上面。然后可以用 svn commit 将合并的结果提交。注意，这里目标是/home/quietheart/test/svn_study/test/branches/quietheart_branch/，它是本地工作拷贝的目录。

实际最后一个参数可以省略，这样会以当前路径作为合并目标，运行如下：

```
$cd /home/quietheart/test/svn_study/test/branches/quietheart_branch
$svn merge -r 6:7 file:///home/quietheart/svn_repos/test_repos/mystudy/test/trunk
```

另外注意，

(a)当某个分支或者主干更新的时候，svn 库的版本号会随着增加，这样，当在其它主干或者分支中进行 update 之后会发现，虽然其内容没有变化，但是最新的版本号却发生了变化。 可见，各个分支和主干之间，共享同一个全局的库版本号。

(b)每当我们想要使用一个新的分支的时候，都要使用”svn checkout ...”来 checkout 整个分支目录。虽然创建分支几乎不占用时间以及 svn 库的空间，但是 checkout 整个分支到本地还是占用许多时间和空间的，这时候应该使用”svn switch ...”来切换。这样可以直接在本地将工作拷贝切换成任何版本的任何分支或者主干，而且不会浪费时间和空间。具体可以参考后面的 svn switch 子命令。

三、svn 命令

这里，简单介绍常用的 svn 子命令以及使用方法举例。具体每个子命令的使用细节，可

以用”$svn help <command>”来获得（这里<command>就是你要查询的那个帮助命令）。

1、查看 svn 可用的所有子命令：

$svn help

输入之后，输出如下：

Available subcommands:

add
blame (praise, annotate, ann)
cat
checkout (co)
cleanup
commit (ci)
copy (cp)
delete (del, remove, rm)
diff (di)
export
help (?, h)
import
info
list (ls)
lock
log
merge
mkdir
move (mv, rename, ren)
propdel (pdel, pd)
propedit (pedit, pe)
propget (pget, pg)
proplist (plist, pl)
propset (pset, ps)
resolved
revert
status (stat, st)
switch (sw)
unlock
update (up)

Subversion 是个版本控制系统的工具。

后面，对其中重要的命令依次进行介绍。

2、svn 子命令

（1）add:

将一个未纳入版本库的目录或者文件添加到 svn 版本管理库中，add 之后需要用 commit 才能体现在 svn 版本库中。

例如：

$svn add testfile

这样，svn 会将 testfile 纳入版本控制。

（2）blame (praise, annotate, ann):

这个命令显示某个文件每一行的版本号以及修改者。

例如：

$svn blame testfile

输出大致如下：

```
158     lv-k #!/bin/bash
901     lv-k ########Notation for development#########
901     lv-k #This script is made by Neusoft.
901     lv-k #1, Comment with "#XXX" needs to be improved in the future.
901     lv-k #2, If you "cd" to some path in sub_function, you must "cd OLDPWD" before return.
901     lv-k ##########################################
158     lv-k
158     lv-k ########Global Variables#########
901     lv-k #This value can be "develop" or "release" now.
```

.....剩下的省略.....

（3）cat:

这个命令的功能类似 shell 的 cat,用于查看文件的内容。

例如：

```
$svn cat testfile
```

之后会输出当前版本文件的内容。

（4）checkout (co)

这个命令用于从版本库中取出指定版本的代码，默认为指定目录中当前版本库中最新代码。

例如：

```
$svn checkout -r 900 testfile
```

之后会从版本库中取出一个工作拷贝 testfile，存放在本地目录中。取出的版本号是 900，取出的内容是目录的话，会包含版本控制信息，可以在相应目录中运行特定 svn 命令。

（5）cleanup

这个命令在本地文件被锁定的使用，用于清楚锁。

SVN 本地更新时，由于一些操作中断更新，如磁盘空间不够，用户取消。可能会造成本地文件被锁定的情况。一般出现这种情况可以使用 cleanup 来清除锁定。如果在根目录下都无法 clean 的话，一般采取的方法是另外找一个目录重新 CHECKOUT,但有时有时 SVN 目录下可能有一些自己本地修改的文件，还未提交到 SVN 服务器，这时重新 CHECKOUT 需要注意本地文件的备份.

例如：

```
$svn cleanup
```

这样会清除当前锁定。

（6）commit (ci)

这个命令用于将当前目录中本地所做修改提交到版本库中，同时生成一个新的版本。

例如:

```
$svn commit -m "<some information>"
```

这里，使用-m 来指定提交的日志。

（7）copy (cp)

这个命令用于将版本库中某个子目录进行拷贝，经常用于创建分支。这里的拷贝是廉价的拷贝，仅在版本库中保存每个分支拷贝中相应的变化而非全部，但检出的时候却可以将拷贝当做独立的工作拷贝来对待。copy 动作需要 commit 才能提交到版本库中。

例如：

```
$svn copy trunk branches/tmp
```

（8）delete (del, remove, rm)

这个命令类似 linux 的 rm 命令，用于将版本库中的某个文件或者目录删除。
例如：
$svn delete branches/tmp

（9）diff (di)
这个命令查看某两个工作拷贝之间的修改。
例如：
$svn diff -r 901:902
输出内容类似如下：
Index: init.initrd
===
--- init.initrd （修订版 901）
+++ init.initrd （修订版 902）
@@ -5,7 +5,7 @@
 mount -o rw -t proc proc /proc

 #lvkai+{ XXX print time
-cat /proc/uptime
+#cat /proc/uptime
 #lvkai+}

 mount -t sysfs none /sys
@@ -23,7 +23,7 @@
 mount -t vfat -o shortname=winnt /dev/sxs_blk0 /SlotA\:

 #XXX print time
-cat /proc/uptime
+#cat /proc/uptime

 #XXX startup appliation
 #/diablo/diabloMainCpu/diablo.bin&
这里，会比较 901 和 902 两个版本之间的不同，并把相应的修改输出到标准输出。

（10）export
这个命令用于从版本库中取出指定版本的代码，默认为指定目录中当前版本库中最新代码,取出的信息不包含 svn 信息。
例如:
$svn export -r 900 testfile
这个命令之后会从版本库中取出 testfile，存放在本地目录中。取出的版本号是 900，取出的内容是目录的话，就如同一个没有纳入版本控制的目录一样不包含版本控制信息，不能在相应目录中运行特定 svn 命令。

（11）help (?, h)
这个命令用于获取 svn 命令的帮助信息,默认列出所有命令的列表。
例如：
$svn help commit
这样会输出 commit 命令的帮助信息。

（12）import
这个命令用于将指定的目录或者文件纳入版本控制，通常在最初建立 svn 库的时候使用。例如：
$svn import svn_study file:///home/quietheart/svn_repos/test_repos/mystudy/test -m "Initial

import"
这样，会将 svn_study 这个目录下的内容纳入到 file:///home/quietheart/svn_repos/test_repos/mystudy/test 中。这里，svn 库的名称是 test_repos, 同时提交的日志用-m 指定。

（13）info
这个命令用于显示指定路径的版本控制信息。
例如：
$svn info test
会输出如下：
路径：test
地址(URL)：file:///home/quietheart/svn_repos/test_repos/mystudy/test
Repository Root: file:///home/quietheart/svn_repos/test_repos
档案库 UUID：4ee27b8d-b954-4796-92c7-e0d337941423
修订版：9
节点种类：目录
调度：正常
最后修改的作者：quietheart
最后修改的修订版：9
最后修改的时间: 2011-01-20 17:46:35 +0800 (四, 20 1月 2011)

（14）list (ls)
这个命令用于查看指定目录中，已经纳入版本控制的所有文件，类似 linux 中的 ls 命令。

例如：
$svn ls
这个命令输出的内容，不一定是当前目录所有文件，而是当前目录中纳入版本控制的文件。

（15）lock
这个命令用于某个用户将某个文件进行锁定.
例如：
$svn lock test
这里，锁定动作会从当前工作拷贝中自动传播出去，不用通知别人了.锁定之前，这个文件必须是版本库中最新版本的，否则锁定失败。锁定之后，其他用户工作拷贝中无法对此文件加锁，无法提交修改，也无法解锁（即使是同一用户名）.除非加锁的工作拷贝处解锁(可用 unlock 命令),或者别人使用--force 选项的 unlock 强制解锁.实践发现加锁处提交修改之后自动解锁,锁定动作不能针对目录只能单个文件。
注意，可以使用"svn lock test --force"强制把锁抢过来。

（16）log
这个命令查看某个工作拷贝的日志信息。
例如:
$svn log test
输入之后输出类似如下：
---
r17 | quietheart | 2011-01-21 16:49:53 +0800 (五, 21 1月 2011) | 1 line

---
r7 | quietheart | 2011-01-20 17:36:10 +0800 (四, 20 1月 2011) | 1 line

add test

这里，每个日志中有一个 log 说明，就是 commit 命令中的-m 选项指定的。

（17）merge
这个命令用于将某个分支在某个阶段的修改应用到指定的目标上。命令翻译为合并，实际是应用修改的意思，类似打补丁。
例如：
$svn merge -r 6:7 file:///home/quietheart/svn_repos/test_repos/mystudy/test/trunk /home/quietheart/test/svn_study/test/branches/quietheart_branch/
这样，会把主干上版本 6:7 上面的修改应用到分支上面。然后可以用 svn commit 将合并的结果提交。注意，这里目标是/home/quietheart/test/svn_study/test/branches/quietheart_branch/，它是本地工作拷贝的目录。

（18）mkdir
这个命令用于为版本控制系统中增加一个目录,类似 linux 中的 mkdir 命令。
例如：
$svn mkdir test1
这里，添加的目录必须用 commit 提交，通知到版本库。

（19）move (mv, rename, ren)
这个命令用于在版本库中移动文件，或者给某个文件命名，类似 linux 中的 mv 命令。
例如：
$ svn move good test2
输入之后，输出如下：
A test2
D good
这里，类似用 add 添加一个目录 test2，然后把原来的目录用 delete 删除。必须用 commit 提交到版本库别人才能够看到你的动作。

（20）propdel (pdel, pd)
这条命令涉及 svn 库属性，省略。

（21）propedit (pedit, pe)
这条命令涉及 svn 库属性，省略。

（22）propget (pget, pg)
这条命令涉及 svn 库属性，省略。

（23）proplist (plist, pl)
这条命令涉及 svn 库属性，省略。

（24）propset (pset, ps)
这条命令涉及 vn 库属性，省略。

（25）resolved
这个命令用在解决冲突之后。
例如：
$svn resolved test
这里，发生冲突的文件就是 test.当你提交时候，svn 会自动 update 一下将版本库最新版

本合并到你的工作拷贝，如果有人修改了和你同样文件的同样行，那么就产生冲突，解决冲突之后(如果不删除冲突备份文件)，使用这个命令就可以继续提交了，如果删除冲突备份文件，那么不用这个命令也行。

（26）revert

这条命令会撤消当前没有提交的本地工作拷贝的修改，假设修改了一个文件，再用这个命令，就可以把修改撤消。

例如：

$svn revert test

这里，可以结合 status 命令进行合适的撤消操作。

注意：本子命令不会存取网络，并且会解除冲突的状况。但是它不会自动恢复被删除的目录，除非你指定了那个被删除的当前看不见的文件名称。

（27）status (stat, st)

这条命令用于在提交之前查看当前当前工作拷贝中哪些文件被修改，添加或者删除了，哪些没有在版本控制之下。

例如：

$svn status

会输出类似如下：

?　　test4

D　　test1/testtest

M　　test3

A　　test5

这里，?表示 test4 不在版本控制之下，D 表示提交时会再版本控制库中删除 testtest 文件，M 表示 test3 做了修改，A 表示 test5 是将要被添加的文件。

如果在上述命令继续运行,输出如下：

$ svn revert *

已恢复“test3”

跳过“test4”

已恢复“test5”

$svn status

?　　test4

?　　test5

D　　test1/testtest

$svn revert test1/testtest

$ svn revert test1/testtest

已恢复“test1/testtest”

$svn status

?　　test4

?　　test5

可见 revert 不是智能地恢复被删除的文件。

（28）switch (sw)

这条命令用于在主干工作拷贝，各个分支工作拷贝之间自由切换（节省了 checkout 一个独立的分支或者 trunk 的时间和空间）；以及在 svn 服务器改变了之后，将本地工作拷贝的服务器地址更新。

例如：

$ svn info

路径：.

地址(URL):

file:///home/quietheart/svn_repos/test_repos/mystudy/test/branches/quietheart_branch

Repository Root: file:///home/quietheart/svn_repos/test_repos

档案库 UUID：4ee27b8d-b954-4796-92c7-e0d337941423
修订版：19
节点种类：目录
调度：正常
最后修改的作者：quietheart
最后修改的修订版：10
最后修改的时间: 2011-01-20 17:49:19 +0800 (四, 20 1月 2011)
$svn switch file:///home/quietheart/svn_repos/test_repos/mystudy/test/trunk
D test
A test1/testtest
A test2
U test3
$ svn info
路径：.
地址(URL)：file:///home/quietheart/svn_repos/test_repos/mystudy/test/trunk
Repository Root: file:///home/quietheart/svn_repos/test_repos
档案库 UUID：4ee27b8d-b954-4796-92c7-e0d337941423
修订版：19
节点种类：目录
调度：正常
最后修改的作者：quietheart
最后修改的修订版：19
最后修改的时间: 2011-01-21 17:12:25 +0800 (五, 21 1月 2011)

上面，原来我在工作分支中工作，现在想要切换到 trunk 主线上面，那么使用 switch 可以很快地将当前工作拷贝变成 trunk，而不用另外 checkout 整个 trunk 主线了。

（29）unlock
这条命令用于解锁 lock 命令锁住的文件。
例如：
$svn unlock test

（3）update (up)
这条命令用于更新工作拷贝，将当前路径工作拷贝更新为 svn 库中相应路径最新版本。
例如：
$svn update