|  |
| --- |
| **1、**将文件checkout到本地目录  **svn checkout** path（path是服务器上的目录） 例如：svn checkout svn://**192.168**.**1.1**/pro/domain 简写：svn co  实例：  svn co <https://192.168.57.208/fund/dept1/TmpCode/套利组/接入平台/deploy>  注意，用户名前要加上域名。  **2、**往版本库中添加新的文件  **svn add** file 例如：svn add test.php(添加test.php) svn add \*.php(添加当前目录下所有的php文件)  **3、**将改动的文件提交到版本库  **svn commit** -m "LogMessage" **[**-N**]** **[**--no-unlock**]** PATHhttp://www.blogjava.net/Images/dot.gif(如果选择了保持锁，就使用--no-unlock开关) 例如：svn commit -m "add test file for my test" test.php 简写：svn ci  **4、**加锁/解锁  **svn lock** -m "LockMessage" **[**--force**]** PATHhttp://www.blogjava.net/Images/dot.gif 例如：svn lock -m "lock test file" test.php **svn unlock** PATH  **5、**更新到某个版本  **svn update -r**m path 例如： svn update如果后面没有目录，默认将当前目录以及子目录下的所有文件都更新到最新版本。 svn update -r 200 test.php(将版本库中的文件test.php还原到版本200) svn update test.php(更新，于版本库同步。如果在提交的时候提示过期的话，是因为**冲突，需要先update，修改文件**，然后清除**svn resolved**，最后再提交commit) 简写：svn up  **6、**查看文件或者目录状态  1）**svn status** pathhttp://www.blogjava.net/Images/dot.gif（目录下的文件和子目录的状态，正常状态不显示） 【?：不在svn的控制中；M：内容被修改；C：发生冲突；A：预定加入到版本库；K：被锁定】 2）**svn status -v** pathhttp://www.blogjava.net/Images/dot.gif(显示文件和子目录状态) 第一列保持相同，第二列显示工作版本号，第三和第四列显示最后一次修改的版本号和修改人。 注：svn status、svn diff和 svn revert这三条命令在没有网络的情况下也可以执行的，原因是svn在本地的.svn中保留了本地版本的原始拷贝。 简写：svn st  **7、**删除文件  **svn delete** path -m "delete test fle" 例如：svn delete svn://192.168.1.1/pro/domain/test.php -m "delete test file" 或者直接svn delete test.php 然后再svn ci -m 'delete test file‘，推荐使用这种 简写：svn (del, remove, rm)  **8、**查看日志  **svn log** path 例如：svn log test.php 显示这个文件的所有修改记录，及其版本号的变化  **9、**查看文件详细信息  **svn info** path 例如：svn info test.php  **10、**比较差异  **svn diff** path(将修改的文件与基础版本比较) 例如：svn diff test.php **svn diff** -r m:n path(对版本m和版本n比较差异) 例如：svn diff -r 200:201 test.php 简写：svn di  **11、**将两个版本之间的差异合并到当前文件  **svn merge -r** m:n path 例如：svn merge -r 200:205 test.php（将版本200与205之间的差异合并到当前文件，但是一般都会产生冲突，需要处理一下）  **12、**SVN 帮助  **svn help svn help ci**  ------------------------------------------------------------------------------  以上是常用命令，下面写几个不经常用的  ------------------------------------------------------------------------------  **13、**版本库下的文件和目录列表  **svn list**path 显示path目录下的所有属于版本库的文件和目录 简写：svn ls  **14、**创建纳入版本控制下的新目录  **svn mkdir**: 创建纳入版本控制下的新目录。 用法: 1、mkdir PATH... 2、mkdir URL... 创建版本控制的目录。 1、每一个以工作副本 PATH 指定的目录，都会创建在本地端，并且加入新增 调度，以待下一次的提交。 2、每个以URL指定的目录，都会透过立即提交于仓库中创建。 在这两个情况下，所有的中间目录都必须事先存在。  **15、**恢复本地修改  **svn revert**: 恢复原始未改变的工作副本文件 (恢复大部份的本地修改)。revert: 用法: revert PATH... 注意: 本子命令不会存取网络，并且会解除冲突的状况。但是它不会恢复 被删除的目录  **16、**代码库URL变更  **svn switch (sw):** 更新工作副本至不同的URL。 用法: 1、switch URL [PATH] 2、switch --relocate FROM TO [PATH...]  1、更新你的工作副本，映射到一个新的URL，其行为跟“svn update”很像，也会将 服务器上文件与本地文件合并。这是将工作副本对应到同一仓库中某个分支或者标记的 方法。 2、改写工作副本的URL元数据，以反映单纯的URL上的改变。当仓库的根URL变动  (比如方案名或是主机名称变动)，但是工作副本仍旧对映到同一仓库的同一目录时使用 这个命令更新工作副本与仓库的对应关系。  **17、**解决冲突  **svn resolved:** 移除工作副本的目录或文件的“冲突”状态。 用法: resolved PATH... 注意: 本子命令不会依语法来解决冲突或是移除冲突标记；它只是移除冲突的 相关文件，然后让 PATH 可以再次提交。  **18、**输出指定文件或URL的内容。  **svn cat** 目标[@版本]...如果指定了版本，将从指定的版本开始查找。 svn cat -r PREV filename > filename (PREV 是上一版本,也可以写具体版本号,这样输出结果是可以提交的） |

**---------------------------------------------------------**

对于日常操作来说，最常用的就是那几个命令， svn co/update/add/ci   
  
1>修改了某个文件，想传上去： svn ci bob.c   -m "bob modify"  
  
2>如果 想想cvs 那样， cvsi ci bob.c 的时候自动弹出 vi窗口，要设置SVN\_EDITOR变量： export SVN\_EDITOR=/usr/bin/vim  ， 可以写死在 /root/.bash\_profile 里面去。   
  
  
3>新添加了某个文件， 想传上去，   echo 1 > bob.c    ;    svn add bob.c ; svn ci bob.c   
  
  
4>自己修改了， 发现错了， 想恢复到svn上面的版本， 可以用 svn update bob.c -r25   // 25 是版本号 。  
  
  
5>想要删除某个文件bob.c  :  svn del bob.c  ;  svn ci bob.c  
  
  
6> 想彻底更新某个目录（可能改了某个目录的一些文件，都不想要了） ， svn revert -R xxx  （xxx是那个目录）  
  
7>比如 ，我checkout下来后，新建了一个目录，然后要上传整个目录， 这点比cvs强，   
svn add xxx   (xxx就是那个目录)  ，他会递规的增加整个目录的。  
然后 svn ci 即可

再次强调一点使用svn/cvs 的流程：  
  
比如bob ，假如我修改了比较大的地方的代码，   
我的流程是：  
  
1>checkout 下来kernel的最新版本 （如果以前已经有了就update 一下 ， 具体的命令不能用update ，应该用svn revert  
2> 修改，本地调试， 测试ok  
3> check in 自己修改过的文件和增加的文件（svn add xxx) 。   注意人无完人， 可能会忘记传了某个文件，  你可以在根目录下 ，执行svn ci ，它会列出来你改了哪些文件  
4> 上传完毕后， 把全部代码 checkout 下来 ，编译， 测试， 如果ok ， 发mail 给相关的人， 内容，比如“I have updated "drivers/fs/ext3.c , pls update it " ,总之越详细越好，   
最好精确到文件， 如果不能精确到文件 ，也要精确到目录， 不要说 ，"我更新了kernel的代码， 你update一下”这样的话 ， 会给别人增加工作量，kernel 那么多的代码， 别人总不能每次都update ， 那样时间会很长。   
  
5> 等待着别人的问题报告。。。。。。  
  
注意事项： 当你都没有稳定某个c文件的时候 ，不要check in ， 不要三天两头checkin svn上的同一个文件（如果是这样，只能说明，你的改的还不稳定， 先自己测试稳定了再传）  
  
另外， svn 不是个人的备份的仓库，是大家的财产，要一字千金的改。

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

很多人用VPS做subversion服务器来管理代码。在subversion里面怎样撤销一次错误的check in呢？一般用svn merge来达成目的。

比如我们刚才有了一次错误的checkin，我们想回滚到正确的版本去。

首先保证我们拿到的是最新代码，运行svn update，假设是28版本。

然后找出要撤销的确切版本：

|  |
| --- |
| svn log contacts.java |

根据log怀疑是27版本改坏的，比较一下：

|  |
| --- |
| svn diff -r 26:27 contacts.java |

发现果真是27版本坏事。

撤销27版本的改动：

|  |
| --- |
| svn merge -r 27:26 contacts.java |

为了保险起见，再次确认合并的结果：

|  |
| --- |
| svn diff contacts.java |

发现已正确撤销了改动，提交。

提交改动

|  |
| --- |
| svn commit -m "Revert wrong change from r27" |

提交后版本变成了29。

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

|  |
| --- |
| **svn switch解决svn服务器地址改变工作拷贝问题** |
|  |
|  |
| |  | | --- | | **我的指令： svn sw --relocate svn://192.168.1.144/device/branches svn://192.168.1.56/device/branches .**  **Name**  svn switch — Update working copy to a different URL.  **Synopsis**  svn switch URL [PATH]  switch --relocate FROM TO [PATH...]  **Description**  This subcommand updates your working copy to mirror a new URL—usually a URL which shares a common ancestor with your working copy, although not necessarily. This is the Subversion way to move a working copy to a new branch. See [the section called “Switching a Working Copy”](http://svnbook.red-bean.com/en/1.1/ch04s05.html) for an in-depth look at switching.  **Alternate Names**  sw  **Changes**  Working copy  **Accesses Repository**  Yes  **Switches**  --revision (-r) REV --non-recursive (-N) --quiet (-q) --diff3-cmd CMD --relocate --username USER --password PASS --no-auth-cache --non-interactive --config-dir DIR  **Examples**  If you're currently inside the directory vendors which was branched to vendors-with-fix and you'd like to switch your working copy to that branch:  $ svn switch http://svn.red-bean.com/repos/branches/vendors-with-fix . U myproj/foo.txt U myproj/bar.txt U myproj/baz.c U myproj/qux.c Updated to revision 31.  And to switch back, just provide the URL to the location in the repository from which you originally checked out your working copy:  $ svn switch http://svn.red-bean.com/repos/trunk/vendors . U myproj/foo.txt U myproj/bar.txt U myproj/baz.c U myproj/qux.c Updated to revision 31.  **Tip**  You can just switch part of your working copy to a branch if you don't want to switch your entire working copy.  Sometimes an administrator might change the “base location” of your repository—in other words, the contents of the repository doesn't change, but the main URL used to reach the root of the repository does. For example, the hostname may change, the URL scheme, or any part of the URL which leads to the repository itself. Rather than checkout a new working copy, you can have the**svn switch** command “rewrite” the beginnings of all the URLs in your working copy. Use the --relocate option to do the substitution. No file contents are changed, nor is the repository contacted. It's similar to running a Perl script over your working copy .svn/ directories which runs **s/OldRoot/NewRoot/**.  $ svn checkout file:///tmp/repos test A test/a A test/b …  $ mv repos newlocation $ cd test/  $ svn update svn: Unable to open an ra\_local session to URL svn: Unable to open repository 'file:///tmp/repos'  $ svn switch --relocate file:///tmp/repos file:///tmp/newlocation . $ svn update At revision 3.  **Warning**  Be careful when using the --relocate option. If you mistype the argument, you might end up creating nonsensical URLs within your working copy that render the whole workspace unusable and tricky to fix. It's also important to understand exactly when one should or shouldn't use --relocate. Here's the rule of thumb:   * If the working copy needs to reflect a new directory within the repository, then use just **svn switch**. * If the working copy still reflects the same repository directory, but the location of the repository itself has changed, then use **svn switch --relocate**. | |