

SVN 命令使用手册

修改版本记录

序号	版本	操作	操作章节	修正者	时间
1.	V0.1	A		牛杰	2008-11-26
2.					
3.					
4.					
5.					
6.					
7.					
8.					

M: 修改 A: 添加 D: 删除

SVN 命令使用

一、 常用命令

1. Svnadmin create 创建库

svnadmin create path

在 windows 版本上:

D:\>svnadmin create test2

D:\>

通过 dir 来列出目录中文件, 已经包含 test2, 如图 1 所示:



```
D:\>dir
驱动器 D 中的卷是 Soft
卷的序列号是 1CF1-97D4

D:\ 的目录

2008-11-22  19:56    <DIR>          001-常用软件
2008-10-13  19:14    17,637 Favorites.rar
2008-11-22  00:03    <DIR>          myproject
2008-11-23  20:02    <DIR>          Program Files
2008-11-21  22:33    <DIR>          SUN-A
2008-11-21  22:18    <DIR>          test
2008-11-26  14:56    <DIR>          test2
2008-11-07  19:10    36 WomccServices.Desc

                2 个文件          17,673 字节
                6 个目录 17,122,557,952 可用字节
```

图 1

说明此时已经成功建立

2. Svn import 导入项目

svn import project_path svn_lib_path -m "comment"

举例:

假设有一个工程名称 unismg, 代码的文件目录是 unicom;

A、我们在 D 盘新建目录 unismg, 在此目录下新建三个文件目录, 如图 2 所示:



```
2008-11-26  15:02    <DIR>          branches
2008-11-26  15:02    <DIR>          tags
2008-11-26  15:02    <DIR>          trunk
```

图 2

trunk 中存放的是项目主线；branches 中存放源码分支；tags 存放在开发过程中做的标签。

B、我们将代码 unisimg 放到 d:\unismg\trunk\中

C、执行命令 D:\>svn import d:\unismg file:///d:/test2/unismg -m "initial import unismg"
结果如图 3 所示：

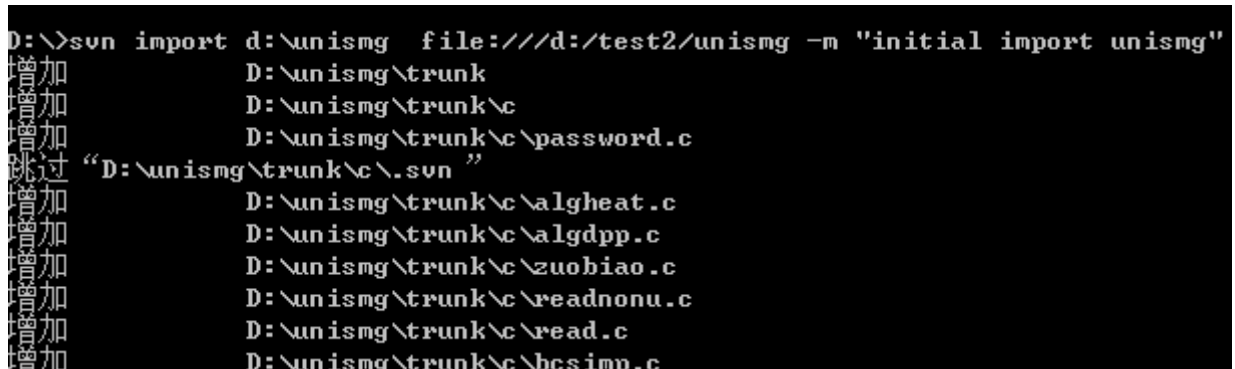


图 3

这样我们就将工程代码导入 svn 库中管理。此时删除 D:\>unismg 目录也没有关系，因为你的源代码已经在 SVN 库中管理了。

有人会有疑问，为什么我到 test2 目录中去找*.c 文件怎么一个没有找到啊，是的 SVN 管理代码，并不是简单的保存文件，而是利用 bdb 管理的，所以你看不到源码存在。

之后你可以使用后续的命令来工作了。

多说一句，关于 svn_lib_path 的几种形式：

[file:///直接版本访问](#)（本地磁盘）

[http://通过配置 subversion 的 Apache 服务器的 WebDAV 协议](#)

[https://与 http://相似](#)，只不过增加了 ssh 协议

Svn://通过 svnserver 服务自定义的协议

Svn+ssh://与 svn://相似，但是通过 SSH 协议封装

比如，联通在信网关在 30.251linux 服务器上，使用的是 svnserver 服务自定义的协议，那么，导入工程代码时应采用的命令是：

```
svn import $path/proj/unismg svn://192.168.30.251:3482 -m "initial import unismg"
```

3. Svn co: 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

举例：

```
svn co svn://192.168.30.251:3482/trunk/unicom
```

下面信息就是从库中下载的代码信息。

4. Svn add: 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 *.c(添加当前目录下所有的 c 文件)

举例:

svn add unismg-misc.conf

5. Svn ci: 将改动的文件提交到版本库

svn commit -m "LogMessage" [-N] [--no-unlock] PATH(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

举例:

svn ci -m "comment" file/path

注意: 此处必须添加 comment

Comment 是: 详细的说明修改代码的原因或者功能, 也即: 能够通过 svn log 获知你提交代码的原因就行。

6. Svn up: 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 up -r 4 filename/path: 是将代码更新到版本 4。用此命令可以更新/还原代码至指定版本。

svn update filename.c

(更新, 与版本库同步。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, 然后使用 svn resolved 命令清除目录下多余文件, 最后再提交 svn commit)

简写: svn up

举例:

```
/export/home/mcpp20/niuj/proj/mcpp2.0>svn up
```

```
U    mcpp/include/mp_cfg.h
```

```
U    mcpp/include/mp_csp_oper.h
```

```
G    mcpp/include/mp.h
```

```
U    mcpp/include/mp_kernel.h
```

```
U    mcpp/include/mp_glo.h
```

```
U    mcpp/include/mp_msisdh.h
```

```
U    mcpp/include/mp_common_mc.h
```

```
U    mcpp/include/mp_biz.h
U    mcpp/include/mp_kernel_smg.h
U    mcpp/include/mp_core_update.h
.....
```

7. Svn st: 查看文件或者目录状态

1) `svn status path` (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

G: 表示代码合并成功

举例:

```
/export/home/mcpp20/niuj/proj/unicom>svn st Make.rules
/export/home/mcpp20/niuj/proj/unicom>vi Make.rules
修改 Make.rules 后, 再执行之:
/export/home/mcpp20/niuj/proj/unicom>svn st Make.rules
M      Make.rules
```

注意:

一旦遇到 C 这种情况, 应当及时手动整合代码。

2) `svn status -v path`(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`

举例:

```
/export/home/mcpp20/niuj/proj/unicom>svn st -v Makefile
                                4          1 niu-jie      Makefile
/export/home/mcpp20/niuj/proj/unicom>vi Makefile
修改一下 Makefile 文件, 再次执行
/export/home/mcpp20/niuj/proj/unicom>svn st -v Makefile
M                                4          1 niu-jie      Makefile
```

8. Svn del: 删除文件

`svn delete file/path -m "delete test file"`

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)`

`Svn del filename/path -m "comment"`

注意：`comment` 必须填写，内容同 `svn ci`

9. Svn log: 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
举例：

r1 | niu-jie | 2008-11-25 11:28:03 +0800 (二, 25 11 月 2008) | 1 line

initial import unismg

此处黑色粗体就是你在签入代码或者删除文件时添加的 `comment` 信息。

10.Svn info: 查看文件详细信息

`svn info file/path`

列出一些版本，修改者，时间等详细信息

例如：

`/export/home/mcpp20/niuuj/proj/unicom>svn info conf`

Path: conf

URL: `svn://192.168.30.251:3482/trunk/unicom/conf`

Repository Root: `svn://192.168.30.251:3482`

Repository UUID: `be25efa0-5121-4a33-b4ce-07ae4bf004f4`

Revision: 4

Node Kind: directory

Schedule: normal

Last Changed Author: niu-jie

Last Changed Rev: 1

Last Changed Date: 2008-11-25 11:28:03 +0800 (二, 25 11 月 2008)

11.Svn diff: 比较差异

`svn diff file/path`

简写：`svn di`

1) `svn diff file/path`(将修改的文件与基础版本比较)

例如：

`/export/home/mcpp20/niuuj/proj/unicom>svn di Make.rules`

Index: Make.rules

=====

--- Make.rules (revision 4)

+++ Make.rules (working copy)

```
@ @ -1,4 +1,3 @ @
```

```
-
```

```
#
```

```
# Project:          Short Message Internet Access Solution
```

```
# Version:          1.0
```

注意:

红色粗体部分是说明下面输出的信息中:

前面带有“-”表示版本 4(svn 库中代码)中存在而你目录下当前版本删除掉的;

前面带有“+”表示版本 4(svn 库中代码)中没有, 你当前版本新添的代码。

2) svn diff -r m:n file/path(对版本 m 和版本 n 比较差异)

例如:

```
/export/home/mcpp20/niuuj/proj/unicom>svn di -r 4:1 CHANGELOG
```

```
Index: CHANGELOG
```

```
=====
```

```
--- CHANGELOG (revision 4)
```

```
+++ CHANGELOG (revision 1)
```

```
@ @ -1,10 +1,21 @ @
```

```
-#凡是修改代码者,需要先到此处申请修改编号
```

```
-#然后确定没有重复后,提交此文件,再行修改代码.
```

```
-#
```

```
* version:
```

```
+* description: 配置文件初始化
```

```
+* code:
```

```
+* changelog: <- 0003 zhangchen
```

```
+* recorder: zhangchen
```

```
+* date:
```

```
+
```

```
+* version:
```

```
+* description: mo 短息自动提示功能:用户发起的 MO 消息流程中没有找到对应的目的 EC/SI 或 EC/SI 无应答;
```

```
+* code:
```

```
+* changelog: <- 0002 zhaoxy
```

```
+* recorder: zhaoxy
```

```
+* date:
```

```
+
```

```
+* version:
```

```
* description:
```

```
* code:
```

```
-* changelog: <- 0001 zhaoxy
```

```
+* changelog: <- 0001 maxw
```

```
* recorder: zhaoxy
```

```
* date:
```

12.svn resolved: 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: **svn resolved file/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。

举例:

更新代码后, 如果发现有 **C**, 也即有冲突。此时会产生多个文件:

filename.c, **filename.c.mine**, **filename.c.rxxxx**(**x** 是版本号)等文件
文件 **filename.c** 代码中会含有:

```
>>>>>
```

```
Codes
```

```
=====
```

```
Codes
```

```
<<<<<
```

这个时候需要解决冲突, 然后保存文件。

运行命令: **svn resolved filename.c**

当遇到有多个文件冲突时, 建议单个文件进行解决, 并运行此命令解决之, 不要多个命令一起操作。

13.svn help: 帮助

svn help

svn help ci

举例:

```
/export/home/mcpp20/niuuj/proj/unicom>svn help ci
```

commit (ci): Send changes from your working copy to the repository.

usage: commit [PATH...]

A log message must be provided, but it can be empty. If it is not given by a **--message** or **--file** option, an editor will be started.

If any targets are (or contain) locked items, those will be unlocked after a successful commit.

Valid options:

-q [--quiet]	: print as little as possible
-N [--non-recursive]	: operate on single directory only
--targets arg	: pass contents of file ARG as additional args
--no-unlock	: don't unlock the targets
-m [--message] arg	: specify log message ARG
-F [--file] arg	: read log message from file ARG

<code>--force-log</code>	: force validity of log message source
<code>--editor-cmd arg</code>	: use ARG as external editor
<code>--encoding arg</code>	: treat value as being in charset encoding ARG
<code>--username arg</code>	: specify a username ARG
<code>--password arg</code>	: specify a password ARG
<code>--no-auth-cache</code>	: do not cache authentication tokens
<code>--non-interactive</code>	: do no interactive prompting
<code>--config-dir arg</code>	: read user configuration files from directory ARG

二、 非常用命令:

1. Svn merge: 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.c`

(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)
建议不要使用此命令

2. Svn lock: 加锁/解锁

`svn lock -m "LockMessage" [--force] PATH`

例如: `svn lock -m "lock test file" test.php`

`svn unlock PATH`

3. Svn list: 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`

4. Svn mkdir: 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。

5. Svn revert: 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)。

用法: **svn 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
举例:

```
/export/home/mcpp20/niuj/proj/unicom>svn diff Makefile
Index: Makefile
=====
--- Makefile      (revision 4)
+++ Makefile      (working copy)
@@ -1,4 +1,4 @@
-#
+#
# Project:          Push Server
# Version:          1.0
# Author:           Darwin
/export/home/mcpp20/niuj/proj/unicom>svn revert Makefile
Reverted 'Makefile'
/export/home/mcpp20/niuj/proj/unicom>svn diff Makefile
/export/home/mcpp20/niuj/proj/unicom>
```

6. Svn switch: 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。

7. Svn cat: 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)

举例:

```
/export/home/mcpp20/niuuj/proj/unicom>svn cat -r 1 CHANGELOG
```

```
* version:  
* description: 配置文件初始化  
* code:  
* changelog: <- 0003 zhangchen  
* recorder: zhangchen  
* date:
```

```
* version:  
* description: mo 短息自动提示功能:用户发起的 MO 消息流程中没有找到对应的  
目的 EC/SI 或 EC/SI 无应答;  
* code:  
* changelog: <- 0002 zhaoxy  
* recorder: zhaoxy  
* date:
```

```
* version:  
* description:  
* code:  
* changelog: <- 0001 maxw  
* recorder: zhaoxy  
* date:
```

```
/export/home/mcpp20/niuuj/proj/unicom>svn cat -r 4 CHANGELOG
```

```
#凡是修改代码者,需要先到此处申请修改编号  
#然后确定没有重复后,提交此文件,再行修改代码.
```

```
#  
* version:  
* description:  
* code:  
* changelog: <- 0001 zhaoxy  
* recorder: zhaoxy  
* date:
```

```
/export/home/mcpp20/niuuj/proj/unicom>
```