Linux 下 SVN 命令使用大全

版本号	V0.1	修订内容	
详细修订日期		文件性质	
备注	正在继续添加和完善...		

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

2、 往版本库中添加新的文件、目录或符号链

svn add PATH...

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)

- 文件、目录或符号链到你的工作拷贝并且预定添加到版本库。它们会在下次提交上传并添加到版本库，如果你在提交之前改变了主意，你可以使用 svn revert 取消预定。
- 是否访问版本库 否
- 选项

  --targets FILENAME

  --non-recursive (-N)

  --quiet (-q)

  --config-dir DIR

  --no-ignore

  --auto-props

  --no-auto-props

  --force

- 例子

  添加一个文件到工作拷贝：

```
$ svn add foo.c
A         foo.c
```

当添加一个目录，svn add 缺省的行为方式是递归的：

```
$ svn add testdir
A         testdir
```

```
A         testdir/a
A         testdir/b
A         testdir/c
A         testdir/d
```

你可以只添加一个目录而不包括其内容：

```
$ svn add --non-recursive otherdir
A         otherdir
```

通常情况下，命令 svn add *会忽略所有已经在版本控制之下的目录，有时候，你会希望添加所有工作拷贝的未版本化文件，包括那些隐藏在深处的文件，可以使用 svn add 的--force 递归到版本化的目录下：

```
$ svn add * --force
A         foo.c
A         somedir/bar.c
A         otherdir/docs/baz.doc
...
```

3、 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

4、 加锁/解锁

svn lock -m “LockMessage“ [--force] PATH

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 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】

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

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 [SUBCOMMAND...]

当手边没有这本书时，这是你使用 Subversion 最好的朋友！

- 别名

?, h

- 使用-?、-h 和--help 选项与使用 help 子命令效果相同。
- 是否访问版本库 否
- 选项 --config-dir DIR

名称

svn merge — 应用两组源文件的差别到工作拷贝路径。

概要

svn merge [-c M | -r N:M] SOURCE[@REV] [WCPATH]

svn merge sourceURL1[@N] sourceURL2[@M] [WCPATH]

svn merge sourceWCPATH1@N sourceWCPATH2@M [WCPATH]

描述

第一种和第二种形式里，源路径（第一种是 URL，第二种是工作拷贝路径）用修订版本号 N 和 M 指定，这是要比较的两组源文件，如果省略修订版本号，缺省是 HEAD。

-c M 选项与-r N:M 等价，其中 N = M-1，使用-c -M 则相反：-r M:N，其中 N = M-1。

第三种形式，SOURCE 可以是 URL 或者工作拷贝项目，与之对应的 URL 会被使用。在修订版本号 N 和 M 的 URL 定义了要比较的两组源。

WCPATH 是接收变化的工作拷贝路径，如果省略 WCPATH，会假定缺省值“.”，除非源有相同基本名称与“.”中的某一文件名字匹配：在这种情况下，区别会应用到那个文件。

不像 svn diff，合并操作在执行时会考虑文件的祖先，当你从一个分支合并到另一个分支，而这两个分支有各自重命名的文件时，这一点会非常重要。

别名

无

改变

工作拷贝 2

是否访问版本库

只有在对 URL 操作时会

选项

```
--revision (-r) REV
--change (-c) REV
--non-recursive (-N)
--quiet (-q)
--force
--dry-run
--diff3-cmd CMD
--extensions (-x) ARG
--ignore-ancestry
--username USER
--password PASS
--no-auth-cache
--non-interactive
--config-dir DIR
```

例子

将一个分支合并回主干（假定你有一份主干的工作拷贝，分支在修订版本 250 创建）：

```
$ svn merge -r 250:HEAD http://svn.red-bean.com/repos/branches/my-branch
U  myproj/tiny.txt
U  myproj/thhgttg.txt
U  myproj/win.txt
```

```
U  myproj/flo.txt
```

如果你的分支在修订版本 23，你希望将主干的修改合并到分支，你可以在你的工作拷贝的分支上这样做：

```
$ svn merge -r 23:30 file:///tmp/repos/trunk/vendors
U  myproj/thhgttg.txt
…
```

合并一个单独文件的修改：

```
$ cd myproj
$ svn merge -r 30:31 thhgttg.txt
U  thhgttg.txt
```

在同一个版本库进行回滚

韩捷对 SystemUI 的修改分别在 46 和 47 版本，现在要将其回滚到未修改之前的状态，则可以这样来做。

进入到 SystemUI 目录，然后运行：

```
svn merge -r 46:45 svn://localhost/MG701/alps/frameworks/base/packages/SystemUI
svn merge -r 47:45 svn://localhost/MG701/alps/frameworks/base/packages/SystemUI
```

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。

svn resolved

上一页　svn 子命令　下一页

名称

svn resolved — 删除工作拷贝文件或目录的“冲突”状态。

概要

svn resolved PATH...

描述

删除工作拷贝文件或目录的“conflicted”状态。这个程序不是语义上的改变冲突标志，它只是删除冲突相关的人造文件，从而重新允许 PATH 提交；也就是说，它告诉 Subversion 冲突已经“解决了”。关于解决冲突更深入的考虑可以查看“解决冲突（合并别人的修改）”一节。

别名

无

改变

工作拷贝 2

是否访问版本库

否

选项

--targets FILENAME

```
--recursive (-R)
--quiet (-q)
--config-dir DIR
```

例子

如果你在更新时得到冲突，你的工作拷贝会产生三个新的文件：

```
$ svn update
C  foo.c
Updated to revision 31.
$ ls
foo.c
foo.c.mine
foo.c.r30
foo.c.r31
```

当你解决了 foo.c 的冲突，并且准备提交，运行 svn resolved 让你的工作拷贝知道你已经完成了所有事情。

警告

你可以仅仅删除冲突的文件并且提交，但是 svn resolved 除了删除冲突文件，还修正了一些记录在工作拷贝管理区域的记录数据，所以我们推荐你使用这个命令。

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)

19、创建版本库

svnadmin create /mnt/MG701_NEW_DRIVE1/MG701

注意：各个子版本库的上层目录需要自己手动创建。比如：

建立/mnt/MG701_NEW_DRIVE1/的一个子版本库 MG701 可以这样操作：

先建立/mnt/MG701_NEW_DRIVE1/MG701 目录（注意修改权限，让其读写可执行）

svnadmin create /mnt/MG701_NEW_DRIVE1/MG701

要建立另外一个子版本库得先在/mnt/MG701_NEW_DRIVE1/目录下建立一个目录，比如/MG702（注意修改权限，让其读写可执行）。然后 svnadmin create /mnt/MG701_NEW_DRIVE1/MG702

20、开启版本库

svnserve -d -r /mnt/MG701_NEW_DRIVE1

注意：开启的时候有个技巧，如果要多个子版本库都开启，那么必须这样做：

svnserve -d -r /mnt/MG701_NEW_DRIVE1

这样一来，MG701 和 MG702 都开启了。

21、svn import – 递归提交一个路径的拷贝到版本库。

Import 整个目录 svn import files PATH -m “LogMessage“

例如： svn import alps svn://192.168.4.210/MG701/alps -m “add test file for my

test“ test.php

主意：这里红色标记的 alps 目录最好写出来（因为写出来之后，alps 将会成为上传所有文件的顶层目录），否则上传到 svn 服务器上的将会是散的文件，而没有顶层目录。

- 递归提交一个路径 PATH 的拷贝到 URL。如果省略 PATH，默认是“.”。版本库中对应的父目录必须已经创建。
- 是否访问版本库：是
- 选项

--message (-m) TEXT

--file (-F) FILE

--quiet (-q)

--non-recursive (-N)

--username USER

--password PASS

--no-auth-cache

--non-interactive

--force-log

--editor-cmd EDITOR

--encoding ENC

--config-dir DIR

--auto-props

--no-auto-props

--ignore-externals

- 例子

这将本地目录 myproj 导入到版本库的 trunk/misc。trunk/misc 目录在导入之前不需要存在–svn import 会递归的为你创建目录。

```
$ svn import -m "New import" myproj http://svn.red-bean.com/repos/trunk/misc
Adding         myproj/sample.txt
…
Transmitting file data .........
Committed revision 16.
```

需要知道这样不会在版本库创建目录 myproj，如果你希望这样，请在 URL 后添加 myproj：

```
$ svn import -m "New import" myproj http://svn.red-bean.com/repos/trunk/misc/myproj
Adding         myproj/sample.txt
…
Transmitting file data .........
Committed revision 16.
```

在导入数据之后，你会发现原先的目录树并没有纳入版本控制，为了开始工作，你还是要运行 svn checkout 得到一个干净的目录树工作拷贝（为了避免这么麻烦，可以选择自己搜集和总结出来的方法）。

附加说明：

- svn import [PATH] URL——将未版本化文件导入版本库的最快方法，会根据需要创建中介目录

eg: svn import -m 'note' mytree file:///var/svn/newrepos/some/project/mytree

- svn 客户端是可以配置忽略哪些些文件，比如说我们不想提交*.o 之类的文件。那么可以通过设置 svn:ignore 或者 global-ignore 来进行过滤相关的文件。

有时基于某种需要，或者是导入一个别人之前维护的工程，里面有些*.so，*.a 文件必须保留，那么如果在第一次 import 的时候全部导入呢？

单个的可以使用 svn add 后再提交这个文件，对于这样的文件较多且分散，可以使用--no-ignore 选项。

eg: svn import [source] [repository] --no-ignore 或 svn add [source] --no-ignore 会自动将 source 目录下所有的内容全部 import 或者 add。

22、svn export – 导出一个干净的目录树

- svn export [-r REV] URL[@PEGREV] [PATH]

从版本库导出干净工作目录树：指定 URL，如果指定了修订版本 REV，会导出相应的版本，如果没有指定修订版本，则会导出 HEAD，导出到 PATH。如果省略 PATH，URL 的最后一部分会作为本地目录的名字。

- svn export [-r REV] PATH1[@PEGREV] [PATH2]

从工作拷贝导出干净目录树：是指定 PATH1 到 PATH2，所有的本地修改将会保留，但是不再版本控制下的文件不会拷贝。

- 是否访问版本库：只有当从 URL 导出时会访问
- 选项

```
--revision (-r) REV
--quiet (-q)
```

--force

--username USER

--password PASS

--no-auth-cache

--non-interactive

--non-recursive (-N)

--config-dir DIR

--native-eol EOL

--ignore-externals

- 例子

从你的工作拷贝导出（不会打印每一个文件和目录）：

```
$ svn export a-wc my-export
Export complete.
```

- 从版本库导出目录（打印所有的文件和目录）：

```
$ svn export file:///tmp/repos my-export
A  my-export/test
A  my-export/quiz
...
Exported revision 15.
```

当使用操作系统特定的分发版本，使用特定的 EOL 字符作为行结束符号导出一棵树会非常有用。--native-eol 选项会这样做，但是如果影响的文件拥有 svn:eol-style = native 属性，举个例子，导出一棵使用 CRLF 作为行结束的树（可能是为了做一个 Windows 的.zip 文件分发版本）：

```
$ svn export file:///tmp/repos my-export --native-eol CRLF
A  my-export/test
A  my-export/quiz
...
Exported revision 15.
```

你可以为--native-eol 选项指定 LR、CR 或 CRLF 作为行结束符。

23、svn move — 移动一个文件或目录。

svn move SRC DST

描述

这个命令移动文件或目录到你的工作拷贝或者是版本库。

提示

这个命令同 svn copy 加一个 svn delete 等同。

注意

Subversion 不支持在工作拷贝和 URL 之间拷贝，此外，你只可以一个版本库内移动文件—Subversion 不支持跨版本库的移动。

WC -> WC

移动和预订一个文件或目录将要添加（包含历史）。

URL -> URL

完全服务器端的重命名。

别名

mv, rename, ren

改变

如果是对 URL 操作则会影响版本库，否则是工作拷贝

是否访问版本库

只有在对 URl 操作时会

选项

--message (-m) TEXT

--file (-F) FILE

--revision (-r) REV (废弃的)

--quiet (-q)

--force

--username USER

```
--password PASS
--no-auth-cache
--non-interactive
--editor-cmd EDITOR
--encoding ENC
--force-log
--config-dir DIR
```

例子

移动工作拷 bede 一个文件：

```
$ svn move foo.c bar.c
A         bar.c
D         foo.c
```

移动版本库中的一个文件（一个立即提交，所以需要提交信息）：

```
$ svn move -m "Move a file" http://svn.red-bean.com/repos/foo.c \
                          http://svn.red-bean.com/repos/bar.c

Committed revision 27.
```

因为你可以使用版本库的 URL 作为唯一参数取出一个工作拷贝，你也可以在版本库 URL 之后指定一个目录，这样会将你的工作目录放到你的新目录，举个例子：

```
$  svn checkout http://svn.collab.net/repos/svn/trunk subv
A    subv/Makefile.in
```

```
A    subv/ac-helpers
A    subv/ac-helpers/install.sh
A    subv/ac-helpers/install-sh
A    subv/build.conf
…
Checked out revision 8810.
```

这样将把你的工作拷贝放到 subv 而不是和前面那样放到 trunk，如果 subv 不存在，将会自动创建。

附录 一 svn 选项

Subversion 命令行客户端：svn

为了使用命令行客户端，只需要输入 svn 和它的子命令以及相关的选项或操作的对象——输入的子命令和选项没有特定的顺序，下面使用 svn status 的方式都是合法的：

```
$ svn -v status
$ svn status -v
$ svn status -v myfile
```

虽然 Subversion 的子命令有一些不同的选项，但有的选项是全局的—也就是说，每个选项保证是表示同样的事情，而不管是哪个子命令使用的。举个例子，--verbose（-v）一直意味着“冗长输出”，而不管使用它的命令是什么。

--auto-props

开启 auto-props，覆盖 config 文件中的 enable-auto-props 指示。

--change (-c) ARG

作为引用特定“修改”（也叫做修订版本）的方法，这个选项是“-r ARG-1:ARG”语法上的甜头。

--config-dir DIR

指导 Subversion 从指定目录而不是默认位置（用户主目录的.subversion）读取配置信息。

--diff-cmd CMD

指定用来表示文件区别的外部程序，当 svn diff 调用时，会使用 Subversion 的内置区别引擎，默认会提供统一区别输出，如果你希望使用一个外置区别程序，使用--diff-cmd。你可以通过--extensions（本小节后面有更多介绍）把选项传递到区别程序。

--diff3-cmd CMD

指定一个外置程序用来合并文件。

--dry-run

检验运行一个命令的效果，但没有实际的修改—可以用在磁盘和版本库。

--editor-cmd CMD

指定一个外部程序来编辑日志信息或是属性值。如何设定缺省编辑器见“配置”一节的 editor-cmd 小节。

--encoding ENC

告诉 Subversion 你的提交日志信息是通过提供的字符集编码的，缺省时是你的操作系统的本地编码，如果你的提交信息使用其它编码，你一定要指定这个值。

--extensions (-x) ARGS

指定一个或多个 Subversion 传递给提供文件区别的外部区别程序的参数，如果你要传递多个参数，你一定能够要用引号（例如，svn diff --diff-cmd /usr/bin/diff -x "-b -E"）括起所有的参数。这个选项只有在使用--diff-cmd 选项时使用。

--file (-F) FILENAME

为特定子命令使用命名文件的的内容，尽管不同的子命令对这些内容做不同的事情。例如，svn commit 使用内容作为提交日志，而 svn propset 使用它作为属性值。

--force

强制一个特定的命令或操作运行。Subversion 有一些操作防止你做普通的使用，但是你可以传递 force 选项告诉 Subversion “我知道我做的事情，也知道这样的结果，所以让我做吧”。这个选项在程序上等同于在打开电源的情况下做你自己的电子工作—如果你不知道你在做什么，你很有可能会得到一个威胁的警告。

--force-log

将传递给--message（-m）或者--file（-F）的可疑参数指定为有效可接受。缺省情况下，如果选项的参数看起来会成为子命令的目标，Subversion 会提出一个错误，例如，你传递一个版本化的文件路径给--file（-F）选项，Subversion 会认为出了点错误，认为你将目标对象当成了参数，而你并没有提供其它的—未版本化的文件作为日志信息的文件。为了确认你的意图并且不考虑这类错误，传递--force-log 选项给命令来接受它作为日志信息。

--help (-h 或-?)

如果同一个或多个子命令一起使用，会显示每个子命令内置的帮助文本，如果单独使用，它会显示常规的客户端帮助文本。

--ignore-ancestry

告诉 Subversion 在计算区别（只依赖于路径内容）时忽略祖先。

--ignore-externals

告诉 Subversion 忽略外部定义和外部定义管理的工作拷贝。

--incremental

打印适合串联的输出格式。

--limit NUM

只显示第一个 NUM 日志信息。

--message (-m) MESSAGE

表示你会在命令行中指定日志信息，紧跟这个开关，例如：

```
$ svn commit -m "They don't make Sunday."
```

--new ARG

使用 ARG 作为新的目标（结合 svn diff 使用）。

--no-auth-cache

阻止在 Subversion 管理区缓存认证信息（如用户名密码）。

--no-auto-props

关闭 auto-props，覆盖 config 文件中的 enable-auto-props 指示。

--no-diff-added

防止 Subversion 打印添加文件的区别。缺省的行为方式是，当添加一个文件时，svn diff 打印的信息和比较一个空白文件相同。

--no-diff-deleted

防止 Subversion 打印删除文件的区别信息，缺省的行为方式是当你删除了一个文件后运行 svn diff 打印的区别与删除文件所有的内容得到的结果一样。

--no-ignore

在状态列表中显示 global-ignores 配置选项或者是 svn:ignore 属性忽略的文件。见“配置”一节和“忽略未版本控制的条目”一节查看详情。

--no-unlock

不自动解锁文件（缺省的提交行为是解锁提交列出的所有文件），更多信息见“锁定”一节。

--non-interactive

如果认证失败，或者是不充分的凭证时，防止出现要求凭证的提示（例如用户名和密码）。这在运行自动脚本时非常有用，只是让 Subversion 失败而不是提示更多的信息。

--non-recursive (-N)

防止子命令迭代到子目录，大多数子命令缺省是迭代的，但是一些子命令—通常是那些潜在的删除或者是取消本地修改的命令—不是。

--notice-ancestry

在计算区别时关注祖先。

--old ARG

使用 ARG 作为旧的目标（结合 svn diff 使用）。

--password PASS

指出在命令行中提供你的密码—另外，如果它是需要的，Subversion 会提示你输入。

--quiet (-q)

请求客户端在执行操作时只显示重要信息。

--recursive (-R)

让子命令迭代到子目录，大多数子命令缺省是迭代的。

--relocate 目的路径[PATH...]

svn switch 子命令中使用，用来修改你的工作拷贝所引用的版本库位置。当版本库的位置修改了，而你有一个工作拷贝，希望继续使用时非常有用。见 svn switch 的例子。

--revision (-r) REV

指出你将为特定操作提供一个修订版本（或修订版本的范围），你可以提供修订版本号，修订版本关键字或日期（在华括号中）作为修订版本开关的参数。如果你希望提供一个修订版本范围，你可以提供用冒号隔开的两个修订版本，举个例子：

```
$ svn log -r 1729
$ svn log -r 1729:HEAD
$ svn log -r 1729:1744
$ svn log -r {2001-12-04}:{2002-02-17}
$ svn log -r 1729:{2002-02-17}
```

见“修订版本关键字”一节查看更多信息。

--revprop

操作针对修订版本属性，而不是 Subversion 文件或目录的属性。这个选项需要你传递--revision（-r）参数。

--show-updates (-u)

导致客户端显示本地拷贝哪些文件已经过期，这不会实际更新你的任何文件—只是显示了如果你运行 svn update 时更新的文件。

--stop-on-copy

导致 Subversion 子命令在传递历史时会在版本化资源拷贝时停止收集历史信息—也就是历史中资源从另一个位置拷贝过来时。

--strict

导致 Subversion 使用严格的语法，就是明确使用特定而不是含糊的子命令（也就是，svn propget）。

--targets FILENAME

告诉 Subversion 从你提供的文件中得到希望操作的文件列表，而不是在命令行列出所有的文件。

--username NAME

表示你要在命令行提供认证的用户名—否则如果需要，Subversion 会提示你这一点。

--verbose (-v)

请求客户端在运行子命令打印尽量多的信息，会导致 Subversion 打印额外的字段，每个文件的细节信息或者是关于动作的附加信息。

--version

打印客户端版本信息，这个信息不仅仅包括客户端的版本号，也有所有客户端可以用来访问 Subversion 版本库的版本库访问模块列表。

--xml

使用 XML 格式打印输出。