

## 第二章

---

# CVS 快速入门指南

为了让你快点学会使用 CVS，这一章会说明最常见的 CVS 用法。本章所列举的命令和范例都是以常见的情况为主，只谈最常用的选项。后面各章节会对本章所谈到的每个主题进行深入的探讨。

本章的范例和指令都是以 Unix/Linux 命令行上所运行的 CVS 客户端程序为基础。各种图形界面客户端程序的菜单选项和按钮多半会以 CVS 的命令为名，所以，如果你用的是图形界面客户端程序，应该可以很容易理解本章所说的内容。附录一列出了 Unix/Linux 以外其他操作系统适用的图形界面客户端程序和命令行客户端程序。

本章的内容不一定要全部阅读，要领如下：

- 如果你现有的项目已存入 CVS，可以跳过前面几节，直接阅读本章的“调出文件”一节。
- 如果 CVS 已安装并且已运行，而且备有仓库可供项目使用，那么请直接阅读本章的“导入项目”一节。

如果你不确定有无安装和运行 CVS，请先阅读“安装 CVS”，它会告诉你如何查证。如果你不确定有无仓库，请试着搜索 *CVSROOT* 这个目录。仓库的根目录就是 *CVSROOT* 所在的目录，而仓库之下其他位于顶层的目录就是 CVS 项目的目录。

## 安装 CVS

CVS 是一种客户机/服务器式的软件，可以在 Unix 和 Linux 平台上运行。当你在 Unix/Linux 服务器上安装 CVS 时，就等于自动取得 CVS 的服务器和客户端程序。要从任何

Unix/Linux 机器通过网络来访问 CVS 时，只要把 CVS 安装到你要用的那台机器上就行了。CVS 的服务器和客户端程序都是同一个软件。

CVS 可以从 <http://cvsgui.sourceforge.net/> 下载。许多 GNU/Linux 发行包（包括 Debian、Red Hat 及 SuSE）也会随附 CVS 安装包。

如果你喜欢 GUI 客户端程序，建议你到 <http://www.wincvs.org> 看一下。在那里你可以找到 gCVS、WinCVS 以及 MacCVS，它们分别是 Unix 和 GNU/Linux、Windows 以及 Macintosh 的 GUI 客户端程序。

---

注意：如果你是个运行 OS X 的 Macintosh 用户，可以安装标准的 Unix CVS 服务器和客户端程序。

---

你可以从 <http://www.cvsnt.org> 取得与 Windows 兼容的 CVS 服务器。这个服务器与 Unix 所用的服务器并不相同，它们之间的差异被明确地列在该网站的 FAQ 中。你还可以从该网站取得安装指南。

如果你打算使用 SSH (Secure Shell) 协议来建立 CVS 客户端程序和 CVS 服务器之间的安全连接，还必须安装与 SSH 兼容的版本到客户端和服务端所在的机器上。对客户端来讲，可能还需要找到一个能够从命令行使用的 SSH 版本。有关这个主题的其他信息可以参考“访问远程仓库”。

要查找关于 SSH 和 SSH 客户端的信息，OpenSSH 的网站 (<http://www.openssh.com>) 是很好的起点。你也可能会想要阅读 Daniel J. Barrett 博士和 Richard Silverman 合著的《SSH, The Secure Shell: The Definitive Guide》(O'Reilly 出版)。如果还有不足之处，可以参考《SSH FAQ》(<http://www.employees.org/~satch/ssh/faq/>)。你还可以在 [http://directory.google.com/Top/Computers/Security/Products\\_and\\_Tools/Cryptography/SSH/Documentation/](http://directory.google.com/Top/Computers/Security/Products_and_Tools/Cryptography/SSH/Documentation/) 找到 Google 为 SSH 说明文件所列的列表。

多数 Unix 和 Linux 系统都会将 SSH 客户端程序视为标准程序集的一部分而予以安装，Mac OS X 也会事先安装一个 SSH 客户端程序。如果没有自动安装，通常表示你的发行包将 SSH 客户端程序视为选用的程序。

在 MacCVS 的网站 (<http://www.heilancoo.net/MacCVSClient/MacCVSClientDoc/>) 上可以找到讨论如何为 Macintosh 使用 SSH 的文章。无论你使用哪一种 Macintosh CVS 客户端程序，文章中所提到的指令都很有用。

查找 Windows 版的 SSH 时，我通常会从 <http://www.openssh.com/windows.html> 开始找起。

如果你用的是 Unix 或 Linux , CVS 大概已经安装好了。如果已经安装而且安装在搜索路径的某个位置上 , 那么你只要在命令行输入 *cv*s , 就能见到例 2-1 所示的结果。

#### 例 2-1 : 显示 CVS 的辅助说明

```
$ cvs
Usage: cvs [cvs-options] command [command-options-and-arguments]
  where cvs-options are -q, -n, etc.
    (specify --help-options for a list of options)
  where command is add, admin, etc.
    (specify --help-commands for a list of commands
     or --help-synonyms for a list of command synonyms)
  where command-options-and-arguments depend on the specific command
    (specify -H followed by a command name for command-specific help)
Specify --help to receive this message

The Concurrent Versions System (CVS) is a tool for version control.
For CVS updates and additional information, see
  the CVS home page at http://www.cvshome.org/ or
  Pascal Molli's CVS site at http://www.loria.fr/~molli/cvs-index.html
```

如果 CVS 已经安装 , 你可以跳过这一节 , 直接看本章的 “ 创建第一个仓库 ” 一节。

要在 Unix 系统中安装 CVS , 通常是下载源代码再从头编译。这种做法也适用于 Linux , 不过许多 Linux 发行包还会提供选项 , 让你安装已编译好的 CVS 可执行文件。下一小节你将可以看到各种 Linux 发行包中用来安装 CVS 的指令。

## 使用源代码来安装和创建 CVS

从 <http://www.cvshome.org> 下载经过压缩的 .tar 文件 ( 即 CVS 包 ) 。解压缩此文件。如果编译之后想留下源代码 , 你可以把压缩文件的内容取出并放到 */usr/src/cvs* 中 ; 否则 , 你可以把它放到 */tmp* 中。接着 , 切换 ( *cd* ) 到 *cvs* 目录 , 阅读 *INSTALL* 和 *README* 这两个文件 , 执行这些文件所提及的命令。我建议新手不要使用 *automake* 和 *autoconf* 。

---

注意 : 你应该以一般用户的身份来编译 CVS , 不要使用超级用户 ( *superuser* ) 的特权来编译 CVS 。但是当你要安装 CVS ( 执行 *make install* ) 的时候 , 就必须使用超级用户的特权。

---

你可以在例 2-2 看到以源代码来安装 CVS 的方法。此例中 , 我会把 .tar 压缩文件的内容 ( 即源码树 , *source tree* ) 取出并放到 */tmp* 中 , 并且切换 ( *cd* ) 到源码树的顶层目录。然后 , 我会以 CVS 包所提供的 *noautomake* 脚本来关掉 *automake* 和 *autoconf* 的功能 , 就像 *INSTALL* 文件所说的那样 ( 译注 1 ) 。接下来 , *INSTALL* 文件针对 CVS 1.11.5 所提示的

---

译注 1 : 编译本书当前的最新版本 CVS 1.11.17 已不再需要执行此步骤。

安装步骤包括：运行 *configure* 脚本，成功之后执行 *make*，接着将用户的身份切换（*su*）成 *root*，再执行 *make install*。

#### 例 2-2：使用源代码进行安装

```
/tmp$ ls
cvs-1.11.5.tar.gz
/tmp$ gunzip cvs-1.11.5.tar.gz
/tmp$ tar -xpf cvs-1.11.5.tar
/tmp$ cd cvs-1.11.5
/tmp/cvs-1.11.5$ ./noautomake.sh --noautoconf
/tmp/cvs-1.11.5$ ./configure
creating cache ./config.cache
checking for a BSD compatible install... /usr/bin/install -c
.
.
.
creating config.h
config.status: executing depfiles commands
/tmp/cvs-1.11.5$ make
make all-recursive
make[1]: Entering directory `/tmp/cvs-1.11.5'
.
.
.
make[2]: Leaving directory `/tmp/cvs-1.11.5'
make[1]: Leaving directory `/tmp/cvs-1.11.5'
/tmp/cvs-1.11.5$ su - root
Password:
/tmp/cvs-1.11.5$ make install
Making install in lib
make[1]: Entering directory `/tmp/cvs-1.11.5/lib'
.
.
.
make[2]: Leaving directory `/tmp/cvs-1.11.5'
make[1]: Leaving directory `/tmp/cvs-1.11.5'
/tmp/cvs-1.11.5$
```

## 使用 apt 来安装 CVS

*apt* 是 Debian Linux 的包管理程序。先确定 *apt* 源代码列表（source list）已设置好，并在安装 CVS 之前执行 *apt-get update* 命令。这个 CVS 包的名称是 *cvs*，因此安装命令是：

```
apt-get install cvs
```

例 2-3 所示为 *apt* 的安装方法，其中省略了 *apt-get update* 命令的大部分输出结果。

### 例 2-3：使用 apt 安装 CVS

```
$ apt-get update
.
.
.
Reading Package Lists ...Done
Building Dependency Tree ...Done
$ apt-get install cvs
Reading Package Lists ...Done
Building Dependency Tree ...Done
The following NEW packages will be installed:
  cvs
0 packages upgraded, 1 newly installed, 0 to remove and 10 not upgraded.
Need to get 0B/1085kB of archives. After unpacking 2626kB will be used.
Reading changelogs... Done
Preconfiguring packages ...
Selecting previously deselected package cvs.
(Reading database ... 39545 files and directories currently installed.)
Unpacking cvs (from .../cvs_1.11.1pldebian-8_i386.deb) ...
Setting up cvs (1.11.1pldebian-8) ...
```

## 使用 rpm 安装 CVS

*rpm* 是 Red Hat Linux 使用的包管理程序。CVS 随附在 Red Hat 7.3 发行包的第二张光盘上。下列指令会假设 CD 已经放入光驱而且已经被挂载到文件系统上，或者 *rpm* 包管理程序能以其他方式来取得此包。

如果你是从命令行使用 *rpm*，请检查 CD 中 CVS 包的名称，名称的格式应该是 *cvs-version.rpm*。要安装 CVS，请使用如下的命令：

```
rpm -Uh cvs-version.rpm
```

如果运行成功，则不会有任何输出结果。如果想看安装进度，请另外加入 *-v*（冗长模式）选项。例 2-4 所示为在命令行以冗长模式安装 CVS 的方法。

### 例 2-4：使用 rpm 安装 CVS

```
$ rpm -Uvh cvs-1.11.1pl-7.i386.rpm
Preparing...          ##### [100%]
1:cvs                 ##### [100%]
```

如果你运行的是 Gnome，可能会用到 *GnoRPM*：

1. 选择“Install”，打开新窗口，从光盘加载各种包。
2. 打开“Packages”文件夹，再依次打开“Development”和“Tools”文件夹。

- 3. 选择 *cvs-version* 包 ( 目前是 *cvs-1.11.1p1-7* )。
- 4. 单击 “ Install ” 按钮。

图 2-1 所示为 GnoRPM 的范例。

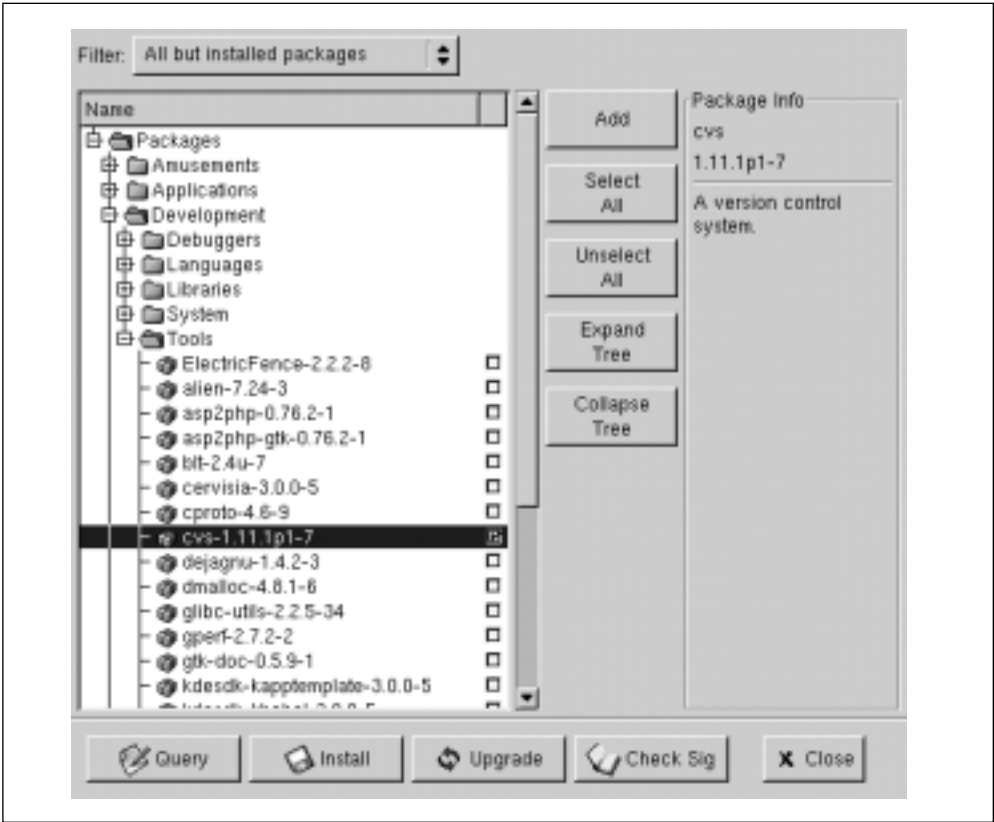


图 2-1 : GnoRPM 的范例

如果你运行的是 KDE , 可能会用到 *KPackage* :

- 1. 选择 “ New ” 标签页 ( tab ) , 则各种包会从光盘载入。
- 2. 找出并选取 cvs 包。
- 3. 单击 “ Install ” 按钮 , 这样会看到打开的新窗口 , 再在新窗口单击 “ Install ” 按钮以确认你的安装意图。

图 2-2 所示为 KPackage 的范例。

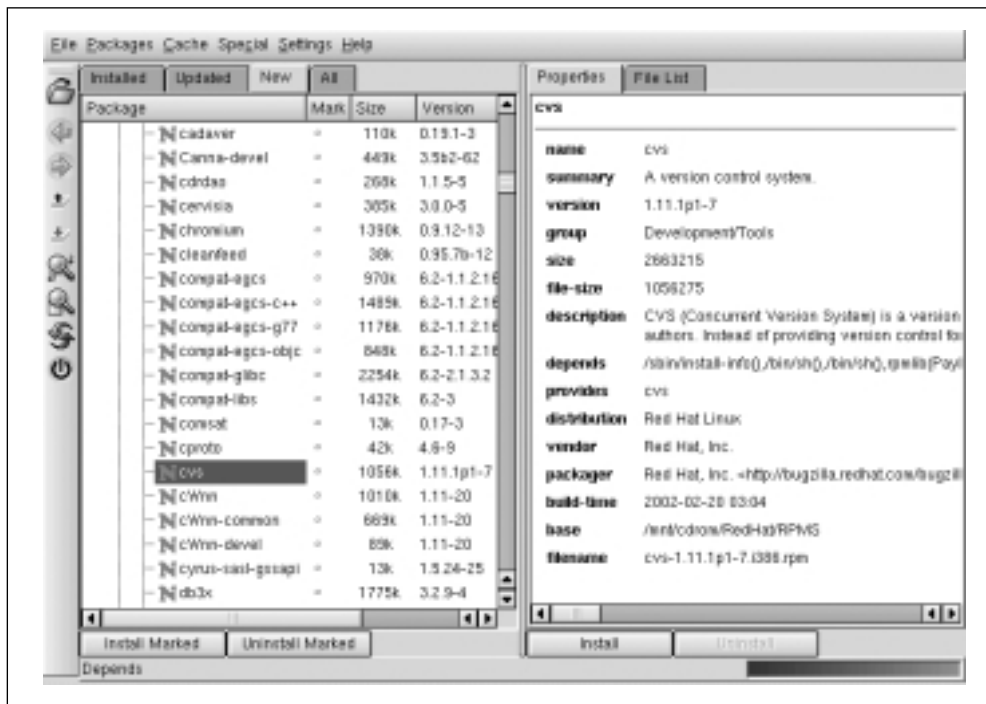


图 2-2 : KPackage 的范例

## 使用 yast 安装 CVS

*yast* 是 SuSE Linux 的包管理程序。CVS 包就随附在 SuSE 7.3 的第二张光盘上，不过新版的 SuSE 可能会将它放到其他的光盘上，或者将它放在此例所示的列表中的其他组里。将光盘放入光驱，然后运行 YaST2 控制中心 ( control center )：

1. 选择 “ Software ”，打开 “ Install/Remove software ”。
2. 选择 “ Development/Version Control ” 组。
3. 选择 *cvs*，单击 “ OK ” 按钮。
4. 如果光盘不在光驱中或是没被挂载到文件系统上，则 YaST2 会要求你把光盘放入光驱。

图 2-3 所示为 YaST2 的范例。

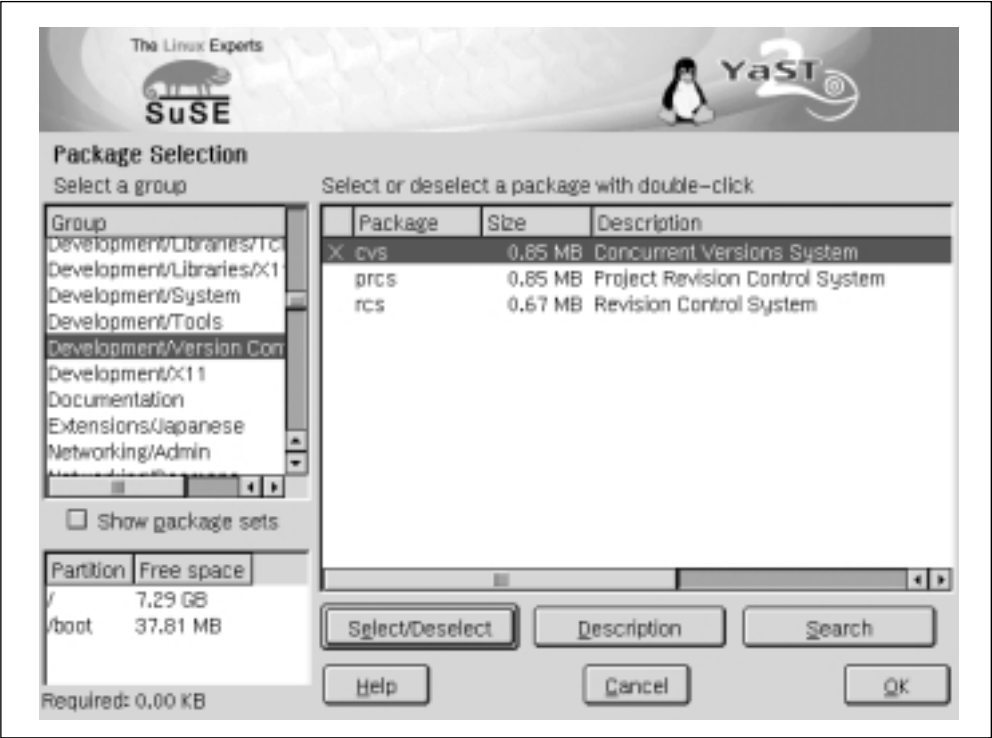


图 2-3 : YaST2 的范例

## 创建第一个仓库

CVS 需要一个数据库，我们称之为 CVS 仓库 (repository)，里面放的都是文件。仓库需要放在一台机器上，它必须有足够的磁盘空间来存储文件以及容纳各种项目的所有数据变更。所有的用户都应该可以从他们的工作站来访问这个仓库。

如果仓库位于远程计算机上，建议让用户通过 SSH 来访问仓库。确认一下远程服务器是否运行 SSH 服务器，而各工作站是否有兼容的 SSH 客户端程序可用。有关远程仓库和 SSH 的信息，可参考“访问远程仓库”。第八章将会全面讨论远程仓库。

对任一项目而言，务必让磁盘空间具备三倍于项目最终可能的大小的容量。如果你打算存储编译后的二进制文件，则至少要五倍。当你做过第一个项目之后，就会对所需的磁盘空间的大小比较有感觉。

仓库可以存储很多项目。如果你所建立的仓库可能要存储好几个大小难料的项目，那么你就需要评估自己能做什么事，确定日后还能再添加新的存储空间。



CVS 仓库所存放的都是用户的数据,应该放在具有备份的分区中,所以即使存储空间被填满,机器也不会因此瘫痪。仓库通常被会放在 `/cvsroot`、`/var/lib/cvsroot`、`/home/cvsroot` 或 `/usr/local/cvsroot` 下。依照《Filesystem Heirarchy Standard》(<http://www.pathname.com/fhs/>) 的规定,存放仓库的较佳位置应该在 `/var/lib/cvsroot` 下。

---

注意: 另一个可能的位置是把仓库放在 `/cvsroot` 下,这是用户最容易记住的位置。

---

例 2-5 说明了创建仓库需要哪些步骤。首先,在 CVS 服务器上创建仓库根目录。在例 2-5 中,创建了 `/var/lib/cvsroot` 目录。

#### 例 2-5: 创建仓库

```
# mkdir /var/lib/cvsroot
# chgrp anthill /var/lib/cvsroot
# ls -la /var/lib
total 2
drwxr-xr-x  2 root    anthill    4096 Jun 28 16:31 cvsroot
# chmod g+srwx /var/lib/cvsroot
# ls -la /var/lib
total 2
drwxrwsr-x  2 root    anthill    4096 Jun 28 16:33 cvsroot
# cvs -d /var/lib/cvsroot init
# ls -la /var/lib
total 3
drwxrwsr-x  3 root    anthill    4096 Jun 28 16:56 cvsroot
# ls -la /var/lib/cvsroot
total 12
drwxrwsr-x  3 root    anthill    4096 Jun 28 16:56 .
drwxrwsr-x 10 root    staff      4096 Jun 28 16:35 ..
drwxrwsr-x  3 root    anthill    4096 Jun 28 16:56 CVSROOT
# chown -R cvs /var/lib/cvsroot
```

要让其他人可以使用这个仓库,可替所有 CVS 用户创建一个 Unix 组,并以 `chgrp` 命令将仓库的根目录的所属组变更成该组。要将用户加入该组,请使用系统所提供的适当命令(通常是 `gpasswd`)。接着设定目录的 SGID 位,使得该目录下所创建的文件组标识符与目录的组标识符一样(这样就能避免以后出问题)。目录所属组的使用权限应该设定为可写、可读以及可执行。

要把安全风险降到最低,可创建一个名为 `cvs` 的用户,使其拥有这个仓库及其管理文件。使用 `chown` 命令把仓库的根目录和管理文件的拥有者变更为 `cvs` 这个用户。第六章将会探讨安全方面的内容。

执行如下的命令,将你所选的目录设置成 CVS 仓库:

```
cvs -d repository_root_directory init
```

CVS 命令的格式如下所示：

```
cvs [cvs-options] command [command-options]
```

其中，*cvs-options* 会改变 CVS 整体的行为，而 *command-options* 则会修改某个 CVS 命令的行为。第三章会更详细地加以说明。

例 2-5 说明了创建一个目录后以该目录创建 CVS 仓库的过程。在此例中，*cvs-options* 是 “-d repository\_root\_directory”，而 *command* 是 “init”，没有 *command-options*。*anthill* 是访问 CVS 的组的名称，而 *cvs* 是 CVS 拥有者的用户名称。

---

注意：Debian Linux 有一个 *cvs-makerepos* 脚本，可根据现有的 Debian 配置脚本来创建仓库。你可以用 *man cvs-makerepos* 来了解更多的信息，以及用 *man cvsconfig* 来了解如何将 Debian CVS 的仓库设定成自动化系统。

---

设置仓库等同于提供了一个用来存储项目的地方，也就是新增了一个 *CVSROOT* 目录。这个 *CVSROOT* 目录包含了配置文件 (configuration file) 和元数据文件 (metadata file)。第六章将会详细说明这个目录。项目就被存放在仓库的根目录下的子目录里，而此例中仓库的根目录就是 */var/lib/cvsroot*。

## 导入项目

创建新的仓库之后，你可能会想要导入你的第一个项目 (project) —— 在单一目录下所存放的一群相关的文件。CVS 也允许你存储单一文件，不过这也算是一个项目，所以你需要准备一个项目目录来存放这个单一文件。CVS 还需要创建一个子目录来存放项目的元数据 (metadata)。

---

注意：你的仓库可能只保存了一个项目，也可能保存了许多不同的项目。CVS 的规模大小皆宜：仓库可以协助个人完成小操作，也可以让大公司用来为上百个独立的团队提供版本控制的服务。

---

将项目加载进 CVS 之前，得先考虑项目的目录结构。如果你在 CVS 已经创建和存储某个文件之后再移入相同的文件，CVS 会视之为两个文件：原来位置上的文件以及新位置上的文件。该文件的历史数据 (history) 就会被分成两个部分。因此，在你把项目导入 CVS 之前，得先决定项目的源代码文件的结构。

如果最后你想把项目中的文件分散到几个不相关的目录下,最好先在单一根目录下开发该项目,然后再把分散文件的工作写在安装脚本里。第七章会详细探讨项目结构的内容。

---

注意：如果有二进制文件或其他非纯文本文件,在你把它们加入仓库之前,请先参考第三章提供的关于二进制文件的信息。

---

先为项目创建最初的目录结构,你可以把它放在 */tmp* 中。一旦项目存入 CVS 之后,最初的版本就可以移除了,这样日后你才不会误把它当成沙箱(项目文件的个人副本)来用。而且项目在 CVS 中已经有了副本,所以没有理由留着它。

一旦项目有了最初的结构,便可以加入任何你想要的初始文件。先切换(*cd*)到项目结构的根目录,然后以如下的命令导入该项目:

```
cvs -d repository_path import name_of_project vendor_tag release_tag
```

如果仓库位于本地机器上,则可以为 *repository\_path* (仓库路径)使用仓库目录的完整路径;如果仓库位于远程服务器上,则可参考本章的“访问远程仓库”中有关指定仓库路径的说明。

多数情况下,你并不需要知道 *vendor\_tag* 和 *release\_tag*。不过 CVS 需要它们的信息。目前,你可以把 *vendor\_tag* 设定成项目的名称,并把 *release\_tag* 设定成当前的修订版的名称。这些名称必须以字母开头,而且只能包含字母数字(alphanumeric)、下划线(\_)以及连字符(-)。例 2-6 说明了创建项目结构、项目文件以及将项目导入 CVS 的过程。

*vendor\_tag* 和 *release\_tag* 会在第八章中加以说明。

#### 例 2-6：导入项目

```
/tmp$ mkdir example
/tmp$ touch example/file1
/tmp$ touch example/file2
/tmp$ cd example
/tmp/example$ cvs -d /var/lib/cvsroot import example example_project ver_0-1
```

执行例 2-6 的命令之后, CVS 会开启一个编辑器窗口,如图 2-4 所示。你可以输入一些信息,以便在日后提醒自己这个项目在做什么。

编辑器窗口中那些以“CVS:”开头的文本行不会被包含在项目的历史数据中。编辑器中所显示的文本也可以通过第七章所提到的 *rcsinfo* 文件予以设定。

默认的编辑器是 *vi*。插入文本前,必须先输入“i”;按“ESC”键会回到可以移动光标的模式;移动键是“h”、“j”、“k”及“l”;要保存和离开,先按“ESC”键,再输入“:wq”并按“RETURN”键。第三章将会说明如何把默认编辑器更换成 *vi* 以外的编辑器。



图 2-4：输入并导入信息

退出编辑器之后，就算 CVS 完成了导入的工作，如例 2-7 所示。

例 2-7：完成导入的工作

```
"/tmp/cvs2hTSiX" 6L, 365C written
N example/file2
N example/file1

No conflicts created by this import
```

在仓库中，所导入的项目会被存储成一群 RCS 文件。例 2-8 列出了例 2-7 所导入的项目的文件。

例 2-8：仓库的内容

```
$ ls -la /var/lib/cvsroot
total 16
drwxrwsr-x  4 root    anthill  4096 Jun 28 17:06 .
drwxrwsr-x 10 root    staff     4096 Jun 28 16:35 ..
drwxrwsr-x  3 cvs     anthill  4096 Jun 28 16:56 CVSROOT
drwxrwsr-x  2 jenn    anthill  4096 Jun 28 17:09 example
$ ls -la /var/lib/cvsroot/example
total 16
drwxrwsr-x  2 jenn    anthill  4096 Jun 28 17:09 .
drwxrwsr-x  4 cvs     anthill  4096 Jun 28 17:06 ..
-r--r--r--  1 jenn    anthill   387 Jun 28 17:06 file1,v
-r--r--r--  1 jenn    anthill   387 Jun 28 17:06 file2,v
```

创建好项目之后，请备份你的 CVS 仓库。项目开发期间，应该定期备份仓库。一旦备份

了仓库，项目也经过确认之后，就可以删除原本的文件。备份的时候，仓库不能开放给别人使用。

---

注意：第六章将会说明如何备份仓库。

---

在你为项目做任何工作之前，得先调出文件放入沙箱，借此确认项目是否已导入 CVS（参阅本章的“调出文件”）。千万不要把原本的文件当成沙箱来使用。将文件调出并放到沙箱之后，才可以开始进行任何新的工作。把原本的文件移除，方能避免自己不小心修改到原本的文件。

## 访问远程仓库

访问远程仓库的方法不只一种。此处所要使用的是 *ext* 访问法和 SSH 协议，但如果系统管理员所提供的是不同的指令，你就应该照做。第八章将会说明远程仓库的使用细节。此处所介绍的做法使用的是仓库的 *ext* 访问法，并通过 SSH 客户端程序来完成连接的工作。相关内容第八章还会予以详述。

你的第一步（至少是我建议的第一步）是在客户端机器上安装 SSH。确定客户端的 SSH 协议能够和服务端端的 SSH 协议相吻合。SSH 密钥或密码设好之后测试连接。使用 SSH 的好处是可以和远程仓库建立安全的连接。

接着，如果你用的是 Unix 或 Linux 系统，可以在你的客户端机器上把 CVS\_RSH 环境变量设为 SSH 程序的名称（通常是 *ssh* 或 *ssh2*）。支持“*ext* 和 SSH”方式的图形界面客户端程序，在其要求你输入仓库路径的对话框中，可能会有一个以 *ssh* 作为认证类型的选项。详情可查阅客户端程序的说明文件。

---

注意：在 WinCVS 和 gCVS 中，你可以在 Admin 菜单中开启 Preferences 对话框，并且在 General 标签页（tab）中将“*ssh*”选为认证方法。在 MacCVSX（供 Mac OS X 使用）中，*ssh* 就放在 MacCVSX 菜单中。供 OS 9 及较早版本使用的 MacCVS，在其标准的选项中并不包含 SSH。但是，你还是可以通过 port tunnelling 让 MacCVS 使用 SSH（附录一会说明）。

---

如果仓库和客户端程序位于相同的机器上，则仓库路径就是指向仓库根目录的完整路径。如果仓库位于远程机器上，则仓库路径的形式如下：

```
[ :method: ] [ [user] [:password]@ ] hostname [ :port ] ] /path
```

*method* 就是用来连接仓库的协议。要使用 SSH 的话，必须将 *method* 指定为 *ext*。如果服

服务器的用户名称和客户端机器的用户名称不同，还必须包括 *user* 和 *@*。如果主机上没有你的 SSH 密钥，当你试着登录（log in）主机时，系统会要求你输入密码。

---

注意：*ext* 访问法并不会用到仓库路径中的密码（*password*）或端口号（*port*）。

---

在操作系统的提示符下使用如下的命令，以便针对远程仓库执行 CVS 命令：

```
cvs -d repository_path command
```

例如，例 2-9 示范了如何把项目导入远程的 CVS 仓库中。

#### 例 2-9：导入项目到远程仓库

```
bash-2.05a$ cvs -d cvs:/home/cvs import example no-vendor release-0
"/tmp/cvs002008" 6L, 365C written
N example/file1h
No conflicts created by this import
bash-2.05a$
```

## 调出文件

CVS 把项目和文件集中存放在一个仓库里，而你所编辑的副本（称为沙箱）则存放在你本地的目录下。用来创建沙箱的命令是 *cvs checkout*。

CVS 会把沙箱创建成当前工作目录的子目录。我喜欢把用来存放所有 CVS 沙箱的目录创建在 *~/cvs* 下。从你想创建沙箱的目录中执行如下的命令：

```
cvs -d repository_path checkout project_name
```

如上的命令会从我所指名的项目中调出所有文件。如果你的仓库位于本地的机器上，则 *repository\_path* 就是 CVS 仓库的完整路径；如果仓库位于远程服务器上，则参考本章的“访问远程仓库”一节。例 2-10 展示了从本地仓库调出项目文件的方法。

#### 例 2-10：从本地仓库调出项目的文件

```
$ mkdir ~/cvs
$ cd ~/cvs
$ cvs -d /var/lib/cvsroot checkout example
cvs checkout: Updating example
U example/file1
U example/file2
```

*checkout* 命令会把项目文件的副本以及子目录的副本放入以项目为名的目录（即 *example*）中，这个项目目录就放在当前的工作目录（即 *~/cvs*）下。此外，它还会在项目目录的 CVS 子目录中放入一些管理性质的文件。细节参阅第三章。

CVS 会把仓库的路径存成沙箱的一部分，所以在沙箱内执行命令时不必使用 *-d repository\_path*。

如果从远程仓库把项目的文件调出并放到沙箱中，则 *repository\_path* 和前例所示的路径会有些不同。例 2-11 展示了从远程仓库调出项目文件的方法。

#### 例 2-11：从远程仓库调出项目的文件

```
$ cvs -d :ext:cvs:/home/cvs checkout cvsbook
cvs server: Updating cvsbook
U cvsbook/1-introduction.html
U cvsbook/2-installation.html
U cvsbook/3-quickstart.html
U cvsbook/acknowledgements
U cvsbook/outline_schedule
U cvsbook/proposal
U cvsbook/sources
U cvsbook/template
```

## 提交变更后的数据

把项目的文件调出并放到沙箱后，你就可以用编辑器编辑这些文件。变更之处并不会和仓库同步，除非你执行了 *cvs commit* 命令。这个命令最好从沙箱的根目录来执行，而且只能从沙箱内执行。

变更之处要经常提交给仓库。原则上，每当程序代码可以正确编译时、每天午餐前以及下班前都应该进行提交。

---

注意：有多人参与开发的程序设计项目中，应该避免提交不能编译的程序代码给仓库。

---

提交时，CVS 会检查当前工作目录下的每个目录和子目录，搜索有变更记录的文件，然后把所有的变更细节提交给仓库。例 2-12 是一个提交文件的例子。记住，仓库路径已存储在沙箱中，因此你不需要在 *cvs commit* 命令中指明仓库路径。

#### 例 2-12：提交文件

```
$ cd ~/cvs/example
$ cvs commit
cvs commit: Examining .
```

如果仓库不在本地的机器，而且仓库所在服务器上没有你的 SSH 公开密钥 (public key)，则 CVS 会要求你输入密码以登录远程机器。如果服务器上有公开密钥，则你的 SSH 客户端程序就可以使用私有密钥 (private key) 让你通过认证。

如果任何文件有变更，则 CVS 会启动编辑器让你记录一条变更信息。默认的编辑器是 *vi*，就是导入项目时所启动的编辑器。第三章将会说明如何替换默认编辑器。

强烈建议写些有意义的变更注释。如果你需要进行恢复 (rollback) 的操作，结果你的信息只有 “fixed a few bugs” (修复若干缺陷)，那么你大概搞不清楚究竟恢复的是哪个修订版。

例 2-13 是变更注释的最佳典范。

#### 例 2-13：输入提交信息

```
CVS:-----
CVS: Enter Log. Lines beginning with 'CVS:' are removed automatically
Corrected bug #35. 'hello' was misspelled as 'helo'.
CVS:
CVS: Committing in .
CVS:
CVS: Modified Files:
CVS:  file1
CVS:-----
```

退出编辑器之后，CVS 就算完成了提交工作，所显示的信息与例 2-14 的结果相同。

#### 例 2-14：完成提交工作

```
"/tmp/cvs7ckpN4" 9L, 343C written
Checking in file1;
/var/lib/cvsroot/example/file1,v <-- file1
new revision: 1.2; previous revision: 1.1
done
```

如果仓库中有一个修订版比沙箱所依据的修订版还新，则 *cvs commit* 会失败。此时，你必须以 *cvs update* 命令来合并有变更的文件，重新执行一次 *cvs commit*。例 2-15 显示了提交失败时的响应信息。

#### 例 2-15：提交失败

```
cvs server: Up-to-date check failed for 'file2'
cvs [server aborted]: correct above errors first!
cvs commit: saving log message in /tmp/cvst7onmJ
```

## 更新沙箱

*cvs update* 命令会比对你的沙箱和仓库，把任何有变更的文件下载 (download) 到你的沙箱中，等同于 *cvs commit* 命令的互补命令，因为 *cvs commit* 命令是从沙箱中把有变更的文件上传 (upload) 到仓库。使用 *-d* 这个命令选项还可以下载新的目录。例 2-16 是 *cvs update* 命令的用法。



## 例 2-16：更新沙箱

```
$ cvs update -d
cvs update: Updating .
U file2
cvs update: Updating directory
$ ls
CVS directory file1 file2
```

如同提交时那样，你不用指明仓库的路径，它应该已经存入沙箱中的 CVS 子目录里。你必须在沙箱中执行 *cvs update*，而且最好是从沙箱的根目录来执行，以确保此命令会检查所有的子目录。

请注意，*-d* 有两种不同的意义，要看它出现在命令中的何处。CVS 命令具有如下的形式：

```
cvs [cvs-options] command [command-options]
```

作为 CVS 选项 (cvs-options) 时，*-d* 用来定义目录的路径；作为 *update* 命令的命令选项 (command-options) 时，*-d* 是指所要下载的目录和文件。第三章会有详细的说明。

当执行 *update* 命令时，会产生已修改文件的列表。在每个文件名的左边，你会看见一个大写的字母。这些字母用来报告列表中每个文件的状态，它们分别具有如下的意义：

*U filename*

更新成功。仓库中较新的版本已经代替沙箱中较旧的版本。

*A filename*

代表新增，但还未加入仓库（需要执行 *cvs commit*）。

*R filename*

代表移除，但尚未从仓库移除（需要执行 *cvs commit*）。

*M filename*

已在工作目录中做了修改。沙箱中的文件比仓库的还要新，或是沙箱和仓库两者都有变更，而系统可将变更之处安全地合并到你的沙箱副本中（需要执行 *cvs commit*）。

*C filename*

仓库的副本和你（在沙箱中）的副本间有冲突。这种冲突需要人为的干预才能解决。

*? filename*

此文件位于你的工作目录中，但在仓库中找不到，CVS 不知道该怎么办。

其中，A、R 和 M 是指你的沙箱已经有所变更，但还没反映到仓库，此刻应该是执行 *cvs commit* 的好时机。

如果CVS无法成功地把修改过的文件和仓库中的副本合并在一起，则会在`cvs update`的输出结果中告诉你发生冲突的地方，如例 2-17 所示。

#### 例 2-17：文件发生冲突

```
cvs/example$ cvs update
cvs server: Updating .
RCS file: /var/lib/cvsroot/example/file1,v
retrieving revision 1.3
retrieving revision 1.4
Merging differences between 1.3 and 1.4 into file1
rcsmerge: warning: conflicts during merge
cvs server: conflicts found in file1
C file1
```

如果变更之处不在同一行，则 CVS 会自动合并文件；如果变更之处在同一行，则 CVS 会汇报冲突并创建一个文件，内含该行的两种修订版，并以特殊的符号加以标示，如例 2-18 所示。

#### 例 2-18：冲突标示符号

```
<<<<<<<file2
This line came from the sandbox.
= = = = =
This line came from the repository..
>>>>>>> 1.4
```

原文件的内容会被存入`.#file.revision`文件中，并被放在该原文件的工作目录下，而合并后的文件则会以原文件的文件名来存储。因此你可以在合并后的文件（原文件的文件名）中搜索冲突标示符号，手动编辑文件，然后再把修改过的文件提交给仓库。

## 新增文件

要为仓库里的项目新增文件，首先应该在沙箱中创建文件。确定你考虑到了项目的结构并把文件放入正确的目录，然后从包含该文件的沙箱目录执行如下的命令：

```
cvs add filename
```

如上的命令会把新文件标示成需要纳入仓库。新增目录也是使用相同的命令。新增目录之后，目录中的文件方能跟着新增。执行`cvs add`之后，只会将相应的文件标示为新增，只有在执行`cvs commit`之后，才会真正地将该文件加入仓库。但目录的新增则是立刻加入仓库。例 2-19 所示为新增文件以及将其加入仓库的过程。记住，直到`cvs commit`命令执行之后，才会真正地将该文件加入仓库。

### 例 2-19：新增文件

```
$ touch file3
$ cvs add file3
cvs add: scheduling file `file3' for addition
cvs add: use 'cvs commit' to add this file permanently
$ cvs commit
...
Log message editor opens
...
"/tmp/cvs002176" 9L, 308C written
RCS file: /var/lib/cvsroot/example/file3,v
done
Checking in file3;
/var/lib/cvsroot/example/file3,v <-- file3
initial revision: 1.1
done
```

---

注意：如果你有二进制文件或是其他非纯文本文件，将之新增到仓库之前，请先参考第三章中有关二进制文件的说明。这些文件在新增时要加上 *-kb* 命令选项。

---

如同提交那样，这将会开启一个编辑器窗口，要求你输入登录信息，以描述你所新增的文件。

## 移除文件

要从仓库中移除文件，首先要从沙箱目录中移除该文件，然后再在包含该文件的沙箱目录下执行如下的命令：

```
cvs remove filename
```

执行 *cvs commit* 命令之前，删除操作并不会生效。在删除操作生效之前，相应的文件会一直留在仓库中。

例 2-20 展示了文件删除的过程。当 *cvs commit* 执行之后，CVS 并不会真的将相应的文件删除，而是将该文件放到仓库中一个称为 *Attic* 的目录里。这个目录是用来存储文件的历史数据，日后可以让相应的文件再度恢复到仓库中。

CVS 会启动一个编辑器，让你得以记录删除该文件的原因，如同提交变更文件那样。

### 例 2-20：移除文件

```
$ rm file3
$ cvs remove file3
```

```
cvcs remove: scheduling `file3' for removal
cvcs remove: use 'cvcs commit' to remove this file permanently
$ cvcs commit
...
Log message editor opens
...
"/tmp/cvcs002092" 9L, 308C written
Removing file3;
/var/lib/cvcsroot/example/file3,v <-- file3
new revision: delete; previous revision: 1.1
done
```

CVS 不会从仓库中删除目录，因为这样做会破坏变更记录。使用 *cvcs checkout* 和 *cvcs update* 时，可添加 *-P* 标记（flag）以避免沙箱中出现空目录。

## 技巧整理

CVS 是改善项目开发和系统管理的工具。就像其他工具那样，有好几种方法可以让 CVS 的使用更有效率。以下便是使用 CVS 的一些技巧：

- 让仓库所在计算机的时钟和国际标准时钟同步。CVS 是根据文件的时间戳（timestamp）来判断哪些文件被修改过。
- 让每位开发人员拥有自己的沙箱，并通过 CVS 来传递所有文件的变更。这样可以维持住变更的跟踪和记录，避免开发人员改写别人的成果，造成无法弥补的错误。
- 时常更新，至少每天开始工作之前进行更新，以确保沙箱能保持最新状态。
- 时常提交，让仓库保持最新的状态。程序设计人员可以在程序编译无误时进行提交；其他人可以在每完成一段工作时进行提交。
- 程序设计团队：使用编译管理工具，确保编译过程中所有文件都可被提交给仓库。要确保测试和发行的编译版本都是来自仓库，而不是来自沙箱，但是可以接受程序设计人员以沙箱的编译版本进行程序完成前的测试（pre-alpha test）。