CVS 教學

<bfennema@galaxy.csc.calpoly.edu>

中譯：柯志裕，許明彥

<myhsu@cyberdude.com>

March 20, 2000

前言

本文件譯自 **Ben Fennema** 先生寫的 *CVS Tutorial*，原文刊載於 http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/cvs/。經取得原作者授權及同意，譯成中文，並加入譯者部份的增修。

1 目的

閱讀完此文件後，希望能夠讓你學會以下幾個主題：

1. 如何安裝 CVS
2. 如何將專案放入 CVS 中進行版本控管
3. 基本的 CVS 用法
4. 了解分支 (Branching) 的功能及其用途

觀念

CVS[1] 是一種程式原始碼的版本控制系統，用來記錄程式原始碼的變化歷程。

Bug (程式臭蟲) 常在程式修改時產生，但可能在修改之後很長一段時間問題才被發現，利用 CVS 你可以取出舊版程式碼，以找出是在那一次改版時產生這個 bug。

在多人處理同一專案時，很容易發生覆蓋掉別人程式碼的情形。遇到這種困擾， CVS 也是有極大幫助的。為了解決這個問題，CVS 讓每個人在自己的目錄下進行程式的修改，當工作完成後，再經由 CVS 進行合併 (merge) 的工作。

CVS 檔案是採用 RCS 的格式儲存。如果想進一步了解 RCS 檔案格式請參閱 **RCS tutorial**[2]。

說明

接下來的內容中，所有的 CVS 命令以粗體顯示：**cvs commit bob.c**
輸出結果以斜體顯示：*U bob.c*。

2 準備好使用 CVS

首先設定 CVSROOT 這個環境變數到你想要使用的 CVS 存放目錄，以 sh/ksh/bash 為例

```
CVSROOT=~/cvsroot;export CVSROOT (於本機建立
CVS Server)
```

若為 csh/tcsh 的環境，則為

```
setenv CVSROOT ~/cvsroot;export CVSROOT
```

[1] CVS: Concurrent Version System
[2] http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/rcs/

或是透過網路遠端存取已經設定好的 CVS 伺服器，

```
CVSROOT=:pserver:john@cvs.foo.bar:/var/cvsroot
export CVSROOT (使用網路上的 CVS Server)
```

pserver 是 CVS 伺服器採用的使用者認證方法之一 (尚有其他種類)。john 是在機器上的帳號名稱。cvs.foo.bar 是表示已經安裝及設定好的 CVS 伺服器。 /var/cvsroot 是 CVS 伺服器上供存取 CVS 檔案的目錄。

(欲使用網路上的 CVS 伺服器者可以跳至3)。

要確定你用 CVSROOT 指定的目錄已經建立好了，接著執行 **cvs init**，這個命令會在環境變數 CVSROOT 所指定的目錄 (~/cvsroot 或是 cvs.foo.bar 上的 /var/cvsroot) 下建立一個 CVSROOT 子目錄，並在其中放入一些 CVS 系統必要的檔案。

3 如何將專案放入 CVS 中進行版本控管

假設在 /class/bfennema/project 下有一個由許多檔案所組成的程式，要把這個程式放入 CVS 開始進行版本控管，首先執行

cd /class/bfennema/project

接著

cvs import -m "Sample Program" project vendor start

-m 之後是表示此次動作的訊息記錄，project 為專案名稱，vendor 為廠商註記，start 為版本註記。 CVS 會把目前目錄下的東西 (含子目錄) 通通都放到 CVS 中。

CVS 會回應

N project/Makefile

N project/main.c

N project/bar.c

N project/foo.c

No conflicts created by this import

如果這些程式是你自己的，現在可以把原本的原始碼刪除 (即完全以 CVS 系統中的檔案為主)，在刪除之前先把舊的原始碼備份起來是個好習慣。

4 基本的 CVS 用法

現在你的原始碼已經安全的存放在 CVS 的倉庫 (repository) 中了，想要取出你的原始碼，先切換至你的家目錄，

cd

輸入

cvs checkout project

CVS會回應：

cvs checkout: Updating project

U project/Makefile

U project/bar.c

U project/foo.c

U project/main.c

這個動作會在你現在所在的目錄下，以專案的名稱建立一個目錄，並將所有檔案放在其中，另外在專案的目錄中會有一個 CVS 目錄，裡面存放有關這些檔案的資訊。

現在你可以開始對這些原始檔案作修改，比如說，你在 main.c 中呼叫 bar() 之後加了 **printf("DONE\n");**，現在你要將新的程式碼存回 (check in)

cvs commit -m "Added a DONE message." main.c

CVS會回應

Checking in main.c;

/class/'username'/cvsroot/project/main.c,v <– main.c

new revision: 1.2; previous revision: 1.1

done

注意，-m 選項讓你在命令列直接輸入修改訊息，如果你忽略這個選項，你會被帶到編輯器中，在那裡輸入修改訊息。

5 多人使用 CVS

要模擬多人使用的狀況，首先以第二位開發者的身份再建立一個目錄 (devel2)，取出 project。[3]

接著進入 devel2/project 的目錄中，在 bar.c 中的 **printf("BAR\n");** 後加上 **printf("YOU\n");** 接著以第二位開發人員的角色作 check in bar.c 的動作[4]，

現在以第一位開發者的身份回到第一個目錄 (假設是 devel1)，觀察 bar.c，你可以看到在 devel2 下所作的修改並未影響到第一位開發者的版本，這時候第一位開發者必需執行

cvs update bar.c

CVS會回應：

U bar.c

現在再看看 bar.c，devel1 看起來應該和 devel2 的版本一致了。

接著以第一位開發者身份在 devel1/project 下編輯 foo.c，在 **printf("FOO\n");** 之後加上 **printf("YOU\n");**

對第一位開發者的 foo.c 作 check in[5]。

再回到第二位開發者的 devel2/project 下，在 **printf("FOO\n");** 後加上 **printf("TOO\n");**

接著輸入

cvs status foo.c

[3]提示：**cvs checkout project**

[4]提示：**cvs commit -m "Added a YOU" bar.c**

[5]提示：**cvs commit -m "Added YOU" foo.c**

CVS會回應：

==========

File: foo.c Status: Needs Merge

Working revision: 1.1.1.1 'Some Date'
Repository revision: 1.2 /class/'username'/cvsroot/project/foo.c,v
Sticky Tag: (none)
Sticky Date: (none)
Sticky Options: (none)

一個檔案的 status (狀態) 可能是：
Up-to-date：表示檔案和倉庫中的最新版相同。
Locally Modified：表示你已經對檔案作修改，但尚未作 check in。
Needing Patch：表示有人已經放入新版本到倉庫中。
Needs Merge：表示有人已經放入新版本到倉庫中，而你也對自己目錄下的檔案作過修改。

目前的 status 是 Need Merge 表示我們需要合併 (merge) 第一位開發者所作的改變，輸入：

cvs update foo.c

CVS會回應：

RCS file: /class/'username'/cvsroot/project/foo.c,v
retrieving revision 1.1.1.1
retrieving revision 1.2
Merging differences between 1.1.1.1 and 1.2 into foo.c
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in foo.c
C foo.c

因為我們對 (devel1, devel2) 所作的改變具有關連性 (位置相同)，所以我們必須手動調整 foo.c，把他改成我們希望的樣子，先看一下 foo.c：

```
void foo()
{
   printf("FOO\n");
<<<<<<< foo.c
   printf("TOO\n");
=======
   printf("YOU\n");
>>>>>>> 1.2
}
```

可以看到 devel1 所加的文字是介於 ======= 和 >>>>>>> 1.2. 間，而在 devel2 加入的文字是介於 ======= 和 <<<<<<< foo.c 間，要解決這個問題，把 printf("TOO\n") 移到 printf("YOU\n") 之後，再把 CVS 插入的內容殺掉，接著將 foo.c 作 check in，由於你使用了 cvs update 命令，並且針對 devel1 所作的修改作了整合，所以整合的結果就被放回了倉庫。

6 CVS 的分支

對一個產品維護多個版本時，CVS 有一個非常有用的特性，就是在 revision tree 中能夠產生分支。CVS有一個 tag 命令讓你指定一個名字給特定版本的檔案，cvs status 的 -v 選項讓你可以看到檔案的tag。

回到我們的例子，到 devel1 的目錄，執行

cvs update foo.c

以取得 foo.c 目前的版本接著，我們想要發行這個產品的第一版，我們希望以 release-1 來標記 (tag) 所有檔案

cvs tag release-1 .

CVS會回應

cvs tag: Tagging .

T Makefile

T bar.c

T foo.c

T main.c

如果有其他的發展者想要取出 release-1，他要輸入：

cvs checkout -r release-1 project

他會得到 release-1 版本而不是最新的版本（如果有的話）

現在我們已經有了一個 release 版本，讓我們假裝我們開始第二版的工作，我們知道顧客在抱怨第一版的一個嚴重錯誤，而第二版要在幾個月後才會完成，但是第一版的這個錯誤應該在現在就要解決。

現在呢，我們先把之前課程中的兩個發展樹移除掉，在CVS中有個簡單的方法可以作到，先換到你專案所在的目錄，執行

cvs release -d project

-d 的選項告訴 CVS 把這份拷貝殺掉

CVS會回應：

You have [0] altered files in this repository.

Are you sure you want to release (and delete) odule 'project':

回答：

y

用同樣的命令把 devel2 殺掉

現在我們要產生版本 1 的分支，回到原始的目錄，執行

cvs rtag -b -r release-1 release-1-patches project

CVS會回應：

cvs rtag: Tagging project

其次我們需要取出剛建立的分支：
cvs checkout -r release-1-patches project

CVS會回應：
cvs checkout: Updating project
U project/Makefile
U project/bar.c
U project/foo.c
U project/main.c

現在你可以修改版本 1 的 bug，所有更改後的 check in 動作會把檔案放在分支中，而不是原來的主幹中，假設 bug 的修正是將印出 YOU 和 TOO 的敘述互 換，將 foot.c 作 check in，你也可以把分支再組合回主幹中，要這樣作，首先把修改版刪掉
cd ..
cvs release -d project

接著用 -j 選項把 release-1 和 release-1-patches 組合起來
cvs checkout -j release-1-patches project

CVS會回應：
cvs checkout: Updating project
U project/Makefile
U project/bar.c
U project/foo.c
RCS file: /class/'username'/cvsroot/project/foo.c,v
retrieving revision 1.3
retrieving revision 1.3.2.1

Merging differences between 1.3 and 1.3.2.1 into foo.c
U project/main.c

如果合併過程中有任何衝突 (conflict) 發生，將會需要手動修正，完成之後你可以

cvs commit -m "Merged patch"

把和 release-1-patch 組合後的所有檔案放入目前的 source tree

7 其它 CVS 的指令

7.1 要在模組中加入新的檔案

1. 先 check out 模組
2. 在模組所在的目錄下建立新檔案
3. 用 **cvs add** *filename* 告訴 CVS 這個檔案也要作版本控制
4. 再用 **cvs commit** *filename* 把檔案放入倉庫中。

7.2 從模組中將檔案移除

1. 首先確定這個檔案已經完成 check in 的動作
2. 以 **rm** *filename* 移除該檔案
3. 用 **cvs remove** *filename* 告訴 CVS 你要刪除這個檔案
4. 用 **cvs commit** *filename* 時才真正執行由倉庫中移除檔案的動作。

想了解更多關於 CVS 的用法，可以參考 cvs 的 manual page (**man cvs**)。