Concurrent Version System

GarZa, garzalin@worldonline.es
3 de noviembre de 2002
http://www.terra.es/personal/garzones/index.html

Introducción

CVS (Concurrent Version System) es un sistema cliente/servidor para el control de versiones que permite a los desarrolladores guardar sus archivos, organizados en proyectos, en un depósito central llamado repositorio. Usando las herramientas de cliente de los cvs, los desarrolladores pueden realizar cambios al contenido del depósito. CVS sigue cada cambio realizado a cada archivo, creando una historia completa de la evolución del proyecto en desarrollo; los desarrolladores pueden recuperar versiones más antiguas de un archivo, ver un registro de cambios, y realizar otras tareas útiles según lo necesitado. CVS está dirigido al trabajo en grupo, aunque también es muy útil para un programador individual por la posiblidad de recuperar viejas copias y ver los cambios hechos. Supongo que en muchas ocasiones habrás modificado un programa en el cual han aparecido inconsistencias y has tenido que volver atrás, teniendo que recordar como estaba antes, con un sistema de control de versiones restaurar a un estado anterior es tan fácil que cuando lo uses por primera vez será algo indispensable en tus proyectos.

Muchos de los proyectos abiertos de software libre tienen sus propios servidores de CVS, que son utilizados por los programadores como depósito central para todo su trabajo. CVS proporciona mecanismos para que desarrolladores de todo el mundo puedan unirse a proyectos centralizados, sin que sus trabajos interfieran entre sí, un mismo archivo puede ser modificado por varios programadores sin que se pierdan los cambios que cada uno ha realizado.

CVS no bloquea ficheros, dos o más programadores pueden simultáneamente modificar el mismo archivo, esto es posible ya que los desarrolladores primero deben obtener una copia de de todos los directorios y ficheros que conforman el proyecto, y el trabajo se efectúa sobre esta copia local. CVS es el encargado de fusionar y guardar un registro de los cambios así como informar de los conflictos que puedan surgir.

CVS trabaja con documentos de texto, por ello puede utilizarse tanto para el código fuente de lenguajes de programación como para documentación en HTML, XML, o cualquier otro, siempre que sea ASCII.

Cliente CVS

Existen numerosos clientes gráficos para distintas plataformas, mucho más explícitos que la línea de comandos, no obstante conociendo los comandos cvs podrás manejar cualquier cliente cvs en linux, windows, macintosh o cualquier otro sistema.

Antes de nada hay instalar CVS, probablemente lo tengas en tu distribución favorita, si no es así, las fuentes están en http://www.cvshome.org/, una vez descargado el paquete, procede a compilar e instalar:

```
$ tar xvfz cvs-1.11.2.tar.gz
$ cd cvs-1.11.1
$ ./configure
$ make
$ make install
```

CVS soporta distintos métodos de acceso que dependen de la configuración establecida en la máquina que sirve el repositorio, la elección de un método u otro depende de la seguridad requerida. Los protocolos más usados son pserver y ssh. El más simple es *pserver* que no necesita ningún añadido en la parte servidora o cliente, por otra parte, es el menos seguro ya que la comunicación se realiza en texto claro. *ssh* cifra todo el tráfico entre servidor y cliente, pero necesita que ambos soporten el protocolo SSH.

Para conectar con un repositorio se utiliza la variable CVSROOT, a la cual le asignamos una secuencia, como una URL, que dice a cvs donde está el depósito remoto y el método de conexión.

```
$ export CVSROOT=":pserver:garza@hostcvs.com:/usr/local/cvs"
```

Como puede verse en este ejemplo el método de acceso elegido ha sido *pserver*; deseo conectar con el usuario *garza* en el host *hostcvs.com* y el repositorio remoto está en /usr/local/cvs.

```
$ export CVSROOT=:ext:garza@hostcvs.com:/usr/local/cvs
$ export CVS_RSH=/usr/bin/ssh
```

En este otro ejemplo el método usado es ssh, para ello tendremos que tener instalado openssh-client y en el servidor del repositorio openssh-server.

Para entrar en el servidor cvs hay que hacer *login*:

$ cvs login
Logging in to :pserver:garza@myhost.com:2401/usr/local/cvs
CVS password: < entrar la clave aqui >

Una vez identificado, en futuras sesiones ya no es necesario volver a introducir la clave ya que se guarda en un fichero llamado ~/.cvspass.

Crear un proyecto

Supongamos que tengo un programa denominado Neko y quiero ponerlo bajo el control de CVS, tengo que importar el código desde mi ordenador hasta el repositorio de cvs, primero me sitúo en el directorio donde se encuentra el programa Neko:

$ cd /home/garza/Neko

Y efectuó la importación:

$ cvs import -m "importación inicial a CVS" Neko garza start

N Neko/Neko.class
N Neko/Neko.html
N Neko/Neko.java
N Neko/Readme
cvs server: Importing /usr/local/cvs/Neko/imagenes
N Neko/imagenes/fondo.jpg
N Neko/imagenes/scratch1.gif
N Neko/imagenes/yawn.gif
N Neko/imagenes/ladybug.gif
N Neko/imagenes/right1.gif
N Neko/imagenes/right2.gif
N Neko/imagenes/scratch2.gif
N Neko/imagenes/sleep1.gif
N Neko/imagenes/sleep2.gif
N Neko/imagenes/stop.gif
N Neko/imagenes/awake.gif
No conflicts created by this import

La importación se ha realizado satisfactoriamente, he subido mi proyecto Neko al ordenador *hostcvs*, la letra *N* indica que se trata de un fichero nuevo.

Como dije anteriormente, para trabajar bajo el control de CVS es necesario obtener una copia del proyecto, para no liarme le cambio el nombre al directorio donde se aloja el proyecto Neko original:

$ cd /home/garza
$ mv Neko Neko.old

Y obtengo la copia del servidor CVS:

$ cvs checkout Neko

cvs server: Updating Neko
U Neko/Neko.class
U Neko/Neko.html
U Neko/Neko.java
U Neko/Readme
cvs server: Updating Neko/imagenes
U Neko/imagenes/awake.gif
U Neko/imagenes/fondo.jpg
U Neko/imagenes/ladybug.gif
U Neko/imagenes/right1.gif
U Neko/imagenes/right2.gif
U Neko/imagenes/scratch1.gif
U Neko/imagenes/scratch2.gif
U Neko/imagenes/sleep1.gif
U Neko/imagenes/sleep2.gif
U Neko/imagenes/stop.gif

U Neko/imagenes/yawn.gif

En */home/garza* se ha creado una copia de *Neko* con todos los ficheros y directorios que componen el proyecto, además existe un subdirectorio llamado *CVS* donde se guarda información sobre las revisiones.

Resumiendo: tengo una copia remota de Neko alojada en el repositorio que sirve el host *hostcvs*, aquí están todos los ficheros maestros que componen el proyecto, y una copia local de Neko alojada en mi ordenador, obtenida del repositorio, y que será la que utilize para trabajar.

Modificar un proyecto

Los programadores trabajaran en sus copias locales y podrán modificar o añadir ficheros, crear directorios, renombrarlos, etc., para enviar las modificaciones locales al repositorio se usa *commit:*

$ cvs commit -m "Comentario"

Para obtener una copia actualizada del repositorio se usa *update*:

$ cvs update

update también nos informa sobre los cambios que hemos hecho en nuestra copia local y no hemos enviado aún al depósito central. Siguiendo con nuestro ejemplo, modifico el fichero Neko.html y ejecuto cvs update, y obtengo lo siguiente:

$ cvs update
cvs server: Updating .
M Neko.html
cvs server: Updating imagenes

La letra M indica que el fichero Neko.html ha sido modificado y no ha sido enviado al repositorio.

Para ver las diferencias existentes entre el depósito central y nuestra copia local usaremos:

$ cvs diff

Si necesitas añadir un fichero créalo con tu editor favorito y añádelo a CVS con:

$ cvs add nuevo.java
$ cvs commit -m "Añado fichero nuevo.java"

Si lo que necesitas es un directorio nuevo, por ejemplo ./lib, crealo con *mkdir* y añádelo a CVS:

$ mkdir lib
$ cvs add lib
$ cvs commit -m "Añado directorio lib"

Para eliminar un fichero, por ejemplo nuevo.java, primero bórralo de la copia local y ejecuta:

$ rm nuevo.java
$ cvs remove nuevo.java
$ cvs commit -m "Borro fichero nuevo.java"

Para eliminar un directorio, primero deberás borrar todos los ficheros contenidos en él, despues haz un:

$ cvs update -P

La opción -P indica a update que elimine de la copia de trabajo los directorios vacíos.

Renombrar ficheros y directorios es equivalente a crearlos con el nuevo nombre y eliminar el antiguo. Veamos un ejemplo, si deseo cambiar el nombre al fichero Readme por Leeme.txt:

$ mv Readme Leeme.txt
$ cvs remove Readme.txt
$ cvs add Leame.txt
$ cvs commit -m "Renombrado Readme como Leame.txt"

cvs log muestra un resumen de los cambios realizados en un archivo con información sobre el número de revisión asignado por CVS, fecha y hora en la que se hizo la modificación, autor, comentario, número de líneas modificada etc.

$ cvs log Neko.html

RCS file: /usr/local/cvs/Neko/Neko.html,v
Working file: Neko.html
head: 1.2
branch:
locks: strict
access list:
symbolic names:
start: 1.1.1.1
jgarzon: 1.1.1
keyword substitution: kv
total revisions: 3; selected revisions: 3
description:
----
revision 1.2
date: 2002/11/03 20:13:05; author: jgarzon; state: Exp; lines: +1 -0
Modifico texto
----
revision 1.1
date: 2002/11/01 14:04:12; author: jgarzon; state: Exp;
branches: 1.1.1;
Initial revision
----
revision 1.1.1.1
date: 2002/11/01 14:04:12; author: jgarzon; state: Exp; lines: +0 -0
importación inicial a CVS

==========

CVS asigna un número de revisión cada vez que se hace un cambio en un fichero, éste número es importante puesto que se utiliza para volver a una versión anterior:

$ cvs update -r 1.1.1 Neko.html
U Neko.html

La letra U significa que se ha actualizado, ahora tengo el fichero *Neko.html*, tal y como estaba cuando se creo el repositorio, sin ninguna modificación posterior.

Resolver conflictos

Cuando dos o más personas modifican simultáneamente el mismo fichero puede que después de hacer un *cvs update* obtengamos algo como esto:

RCS file: /usr/local/cvs/Neko/Neko.html,v
retrieving revision 1.21
retrieving revision 1.22
Merging differences between 1.21 and 1.22 into Neko.html
rcsmerge: warning: conflicts during merge
cvs server: conflicts found in Neko.html
C Neko.html

Por el mensaje que nos muestra CVS, es evidente que se ha producido un conflicto, si edito el fichero Neko.html veo una parte del texto con unas marcas que indican las modificaciones que cada programador ha realizado, tendré que optar por una de las dos modificaciones:

<<<<<<< Neko.html
date 03 Nov 2002

System.out.println("Hola mundo");
=======
date 03 Nov 2002

System.out.println("Hello world");

>>>>>>> 1.22

Servidor CVS

Tal y como expliqué al principio de este artículo instalo y compilo el software CVS, que previamente he obtenido de http://www.cvshome.org/.

Creo el directorio raiz del repositorio:

```
# export CVSROOT=/usr/local/cvs
# cvs init
```

En /usr/local se ha creado el directorio cvs y un subdirectorio llamado CVSROOT.

Ahora creo un archivo de password donde se registran los usuarios que van a tener acceso al depósito:

```
# cd /usr/local/cvs/CVSROOT
# htpasswd -c passwd garza
```

htpasswd -c crea el fichero passwd, para crear los siguientes usuarios omitir la opción -c (si no tienes *htpasswd* en tu sistema, éste pertenece al paquete apache).

Si al igual que en los servidores ftp, deseas que accedan a cvs en modo lectura usuarios anónimos, edita el fichero passwd y añade el usuario *anonymous*, despues crea el fichero *readers* en /usr/local/cvs/CVSROOT y añade el usuario *anonymous*.

Por último hay que editar el fichero htpasswd y añadir un tercer campo con el nombre del grupo linux al que pertenecen los usuarios, a los directorios del proyecto le asigno el grupo que corresponda y permisos 2750.

En la parte servidora ya sólo nos queda iniciar el demonio que atienda las peticiones de los clientes, cvs utiliza el super-demonio inetd o xinetd:

a) Si tu distribución linux dispone de inetd, añade al fichero de configuración */etc/inetd.conf* la siguiente línea:

```
cvspserver  stream  tcp  nowait  root  /usr/bin/cvs  cvs  --allow-root=/usr/local/cvs pserver
```

Y reinicia inetd:

```
# /etc/init.d/inetd restart
```

b) Si tu sistema usa xinetd (p.e. RedHat 7 y v. posteriores), crea el archivo *cvspserver* en */etc/xinetd.d*

```
Fichero: /etc/xinetd.d/cvspserver
# description: The cvs allow the creation of remote repositories
service cvspserver
   {
      disable = no
      socket_type     = stream
      protocol        = tcp
      wait            = no
      passenv         =
      user            = root
      server          = /usr/bin/cvs
      server_args     = --allow-root=/usr/local/cvs -f pserver
   }
```

Para que entren en vigor los cambios reinicio xinetd:

```
# /etc/init.d/xinetd restart
```

Ya tenemos a nuestro servidor cvs esperando peticiones por el puerto 2401.

Nótese que en ambos casos he usado *pserver* como mé todo de acceso al servidor cvs. CVS soporta distintos mé todos de acceso: pserver, rsh, ssh, kserver (kerberos) y algunos má s. El má s simple es *pserver* que no necesita ningún añadido en la parte servidora o cliente, por otra parte, es el menos seguro ya que la comunicación se realiza en texto claro.

Mas información en:

- http://www.cvshome.org/
- http://www.gentoo.org/doc/cvs-tutorial.html

- Micro-como empezar a trabajar con cvs
- The CVS Book y su traducción al castellano en http://cvs.hispalinux.es/cgi-bin/cvsweb/doc-cvsbook-es/

GarZa, garzalin@worldonline.es
3 de noviembre de 2002
http://www.terra.es/personal/garzones/index.html