Tutorial de uso de CVS/Versión para imprimir

< Tutorial de uso de CVS

Resumen

Breve tutorial introductorio al uso de CVS, con especial énfasis en el uso de la parte cliente de CVS. Se añaden algunas referencias a interfaces adicionales al cliente tradicional, así como enlaces a tutoriales más amplios y detallados. La parte servidora de CVS se trata también con una cierta profundidad.

Autores

La mayoría del texto de este documento fue escrito por Ismael Olea e Ignacio Arenaza, y publicado en 2002 bajo licencia GFDL [1] (http://es.tldp.org/Presentaciones/200211hispalinux/iarenaza/cvs-como.pdf). Las mejoras y correcciones posteriores son obra de la comunidad de Wikilibros y se pueden consultar en el historial de cada página.

Contenido

Introducción

CVS es un sistema de mantenimiento de código fuente (grupos de archivos en general) extraordinariamente útil para grupos de desarrolladores que trabajan cooperativamente usando alguna clase de red.

Para ser más concreto, CVS permite a un grupo de desarrolladores trabajar y modificar concurrentemente ficheros organizados en proyectos. Esto significa que dos o más personas pueden modificar un mismo fichero sin que se pierdan los trabajos de ninguna. Además, las operaciones más habituales son muy sencillas de usar. Es igual de útil para trabajar con código fuente de programas o con toda clase de documentos (como ficheros sgml/html/xml) y también puede almacenar ficheros binarios.

CVS utiliza una arquitectura cliente-servidor: un servidor guarda la(s) versión(es) actual(es) del proyecto y su historia, y los clientes conectan al servidor para sacar una copia completa del proyecto, trabajar en esa copia y entonces ingresar sus cambios. Típicamente, cliente y servidor conectan utilizando Internet, pero cliente y servidor pueden estar en la misma máquina si CVS tiene la tarea de mantener el registro de la historia de las versiones del programa de un proyecto solamente con desarrolladores locales. El servidor normalmente utiliza un sistema operativo similar a Unix, mientras que los clientes CVS pueden funcionar en cualquier de los sistemas operativos más difundidos.

Los clientes pueden también comparar diferentes versiones de ficheros, solicitar una historia completa de los cambios, o sacar una "foto" histórica del proyecto tal como se encontraba en una fecha determinada o en un número de revisión determinado. Muchos proyectos de código abierto permiten el "acceso de lectura anónimo", significando que los clientes pueden sacar y comparar versiones sin necesidad de teclear una contraseña; solamente el ingreso de cambios requiere una contraseña en estos escenarios.

Los clientes también pueden utilizar el comando de actualización con el fin de tener sus copias al día con la última versión que se encuentra en el servidor. Esto elimina la necesidad de repetir las descargas del proyecto completo.

Terminología CVS

Para que no se pierda, un brevísimo vocabulario:

- Servidor CVS: Máquina que ejecuta CVS y actúa como almacén de ficheros.
- Repositorio: Jerarquía de directorios alojada en el servidor CVS que contiene diferentes módulos a disposición de los usuarios.
- Módulo: Árbol de directorios que forma parte del repositorio. Cuenta con un nombre identificador gracias al cual podremos trabajar con él de forma selectiva.

Invocar a CVS

CVS es un programa que se invoca desde intérpretes de órdenes. Según cual sea su configuración (desde el punto de vista de las diferentes formas de autenticación, que veremos en un momento) lo podrá usar en procesos por lotes sin ningún problema.

La manera de invocar cvs es mediante la siguiente instrucción:

```
cvs orden [opciones]
```

Donde *orden* es una palabra reservada que le indica a cvs qué acción queremos realizar. Por ejemplo, podemos enviar los cambios que hayamos hecho en un directorio mediante **cvs commit**, o ver las diferencias entre nuestra copia del proyecto y la versión del repositorio empleando **cvs diff**.

Un aspecto que debe tener en cuenta es que CVS tiene parámetros para cada una de sus órdenes. Para conocerlas tiene dos métodos: invocar CVS como **cvs help** o consultar la ayuda de su página del manual.

Configuración

Puede usar varios ficheros de configuración que CVS reconocerá y usará. Entre otros, tenemos los siguientes:

- ~/.cvsignore : contiene los sufijos de los ficheros que no nos interesa que CVS controle. Por ejemplo podemos tener:

```
*.aux *.dvi *.ps *.log
```

si los ficheros de este módulo son principalmente documentos escritos en (La)TeX, ya que las extensiones mostradas arriba corresponden a los diferentes ficheros temporales creados durante el procesado del fichero fuente original para obtener el fichero listo para impresión.

Así evitamos contaminar el repositorio con ficheros que realmente no es necesario que estén controlados por CVS.

- ~/.cvsrc : contiene aquellos parámetros que CVS usará cada vez que se invoque una determinada orden, de forma automática. Por ejemplo:

```
update -Pd
diff -uw
cvs -z 3
```

Significa que queremos que cuando usemos la orden update, automáticamente se añadan los parámetros -Pd. Algo similar ocurre con la orden diff, a la que se le añadirán los parámetros -uw.

La última línea, cuya orden es cvs, es especial. Se trata de la forma en que se indican parámetros globales, que se aplican a todas las órdenes.

La autenticación

Al trabajar en remoto con CVS pueden elegirse varias alternativas de autenticación (es decir, de demostrar al servidor que somos quienes decimos ser).

Las más utilizadas son vía pserver y vía ssh (aunque existen otras dos formas al

menos: vía rsh, totalmente desaconsejada por motivos de seguridad, y vía Kerberos, que no es habitual por necesitar toda la infraestructura de seguridad de Kerberos).

Deberá elegir alguna de estas técnicas en función de sus necesidades, en caso de que no vengan impuestas de forma externa (por ejemplo, porque el repositorio ya existe y el tipo de acceso ya ha sido elegido previamente).

ssh (OpenSSH)

Para que CVS use este modo de autenticación se deben usar estas variables de entorno:

```
export CVSROOT=:ext:USUARIO@cvs.dominio.org:/var/lib/cvs
export CVS_RSH=/usr/bin/ssh
```

Donde:

- USUARIO es el nombre de usuario que tiene acceso al repositorio.
- cvs.dominio.org es el nombre del servidor donde se aloja el repositorio.
- /var/lib/cvs es el directorio del servidor en el que está el repositorio
- /usr/bin/ssh es la ruta completa al ejecutable de ssh.

Tenga en cuenta que, usando esta técnica, tendrá que autenticarse (es decir, suministrar su contraseña) cada vez que ejecute alguna orden de CVS (a menos que use autenticación con clave pública RSA/DSA para ssh).

La ventaja de usar ssh como método de autenticación es que las comunicaciones con el servidor CVS van completamente cifradas, tanto la autenticación como los datos que intercambiemos con el servidor (cosa que no ocurre con el siguiente método).

El inconveniente de este método de autenticación es que deberá crear cuentas de usuarios locales en el servidor CVS con posibilidad de inicio de sesión (shell válido) para todos aquellos usuarios remotos que necesiten acceso al servidor, lo cual implica un acceso más amplio al equipo donde se ejecuta el servidor CVS que el mero acceso al servicio CVS.

pserver

Si no necesitamos que los datos que se intercambian con el servidor vayan cifrados y nos basta con una autenticación relativamente segura, podemos utilizar el método pserver. Las ventajas de este método sobre el anterior son:

- No requiere cuentas de usuario en el servidor CVS para cada uno de los clientes, ni permite ningún tipo de acceso adicional en el servidor.
- Es un esquema fácil de configurar y administrar de forma moderadamente segura (aunque es mucho menos segura que el método via ssh, desde el punto de vista

de secuestro de contraseñas de usuario CVS, dado que no circulan cifradas).

Para poder usar este tipo de autenticación, deberá usar una variable de entorno similar a esta:

```
export CVSROOT=:pserver:USUARIO@cvs.dominio.org:/var/lib/cvs
```

Donde:

- USUARIO es el nombre de usuario que tiene acceso al repositorio.
- cvs.dominio.org es el nombre del servidor donde se aloja el repositorio.
- /var/lib/cvs es el directorio del servidor en el que está el repositorio

Si usa esta técnica, antes de poder trabajar con CVS deberá autenticarse ante el servidor. Eso se hace con la orden cvs login:

```
cvs login
```

CVS le pedirá la contraseña del usuario que haya configurado en la variable de entorno mencionada arriba. Si la contraseña es correcta, CVS guardará la información que necesita en el fichero ~/.cvspass y no tendrá que volver a autenticarse. En algunos casos es necesario crear el archivo ~/.cvspass antes de hacer el cvs login. En esos casos, cvs se niega a realizar la operación. En ese caso se debe emitir la orden touch ~/.cvspass .

Se debe tener presente que cvs login escribe tu contraseña en el archivo ~/.cvspass muy debilmente cifradas, es decir quien tenga acceso al archivo podrá en cosa de segundos descifrar tu contraseña. La tabla de conversión utilizada para cifrar, es entregada con la documentación de CVS.

Si por algún motivo quisiera «cerrar la sesión» CVS (esto es, dejar de trabajar con ese servidor CVS o con ese repositorio) bastará con hacer:

```
cvs logout
```

Métodos combinados

También es posible usar ambos métodos para el acceso al servidor (vía ssh y vía pserver) de forma conjunta, de forma que ciertos usuarios usen uno de los métodos y otros usuarios el otro. Esto es especialmente interesante para el acceso adminisitrativo al servidor CVS, dado que el método pserver no es lo suficientemente seguro en este caso.

Uso de CVS

Modo de uso

A continuación se propone una sencilla metodología de trabajo con CVS para evitar acciones redundantes. Piense por ejemplo en la eliminación de erratas o errores en documentos o en código fuente.

Antes de cada sesión de trabajo es conveniente hacer **cvs update -Pd** para asegurarnos de que disponemos de las últimas modificaciones registradas en el repositorio.

Justo al acabar cada sesión de trabajo es conveniente hacer **cvs commit** (se puede abreviar en cvs ci) para que todas nuestras modificaciones se registren en el repositorio.

Añadir nuevos módulos al repositorio

Para añadir nuevos modulos al repositorio se usa la orden **cvs import**, que le indica a CVS que debe crear una copia en el repositorio del conjunto de ficheros del directorio actual.

Por ello, para importar un nuevo módulo al repositorio debemos situarnos primero en el directorio raíz del módulo donde están los ficheros del mismo y ejecutar la orden:

```
<<configuración de las variables de entorno de CVS>>
cd directorio-donde-se-encuentran-los-ficheros
cvs import nombre-módulo etiqueta-vendedor etiqueta-versión
```

Donde:

- nombre-módulo: es el nombre que le queremos dar al nuevo módulo.
- etiqueta-vendedor: es el nombre que usa CVS para etiquetar la rama que crea con la importación. Puede ser una cadena cualquiera de letras, números y subrayados.
- etiqueta-versión: es el nombre que usa CVS para etiquetar la versión concreta que se crea con esta importación. Puede ser una cadena cualquiera de letras, números y subrayados.

Descargar un módulo por primera vez

Para crear una copia de trabajo local del módulo CVS deseado debemos usar la orden **cvs checkout** (abreviable como cvs co):

```
<<configuración de las variables de entorno de CVS>>
cd padre-del-directorio-donde-se-alojará-el-módulo
cvs checkout nombre-del-módulo
```

Esto creará una jerarquía de directorios donde se almacenará la copia local de trabajo el módulo. Este paso sólo hay que hacerlo una vez por cada módulo.

A partir de este momento no es necesario configurar las variables de entorno porque CVS sabe a qué repositorio pertenece el módulo con sólo examinar los

subdirectorios CVS. No se debe modificar nunca esos subdirectorios a mano. De lo contrario CVS perderá la pista de a que módulo pertenecen los ficheros, cuáles son las versiones de la copia local, etc.

Actualizar nuestra copia local desde el repositorio

Cuando queramos actualizar la copia local de trabajo del módulo con los cambios que hayan podido hacer otros usuarios y que están recogidos en el repositorio deberemos emplear la orden update:

```
cd directorio-del-módulo
cvs update -Pd
```

Publicar nuestras modificaciones en el repositorio

Se usa la orden commit (o su equivalente ci):

```
cd directorio-del-módulo
cvs commit
```

Tras lo cual el sistema mostrará la pantalla de un editor de textos (el que tengamos configurado como nuestro favorito en la variable de entorno EDITOR) para que introduzcamos el mensaje de log, una o dos líneas describiendo los cambios realizados en el módulo desde el último commit.

Algunas opciones que admite la orden *commit* son:

- -l : Solamente se aplica a los ficheros del directorio actual, no a sus subdirectorios.
- -R : Se aplica a los subdirectorios de forma recursiva (por defecto).
- -F fichero: Lee el mensaje de log de un fichero, en lugar de invocar al editor.
- -m "mensaje" : Permite indicar el mensaje de log (debe ir entre comillas), en lugar de invocar al editor.
- -f: Obliga a realizar un commit sobre un fichero incluso si no hemos hecho modificaciones al mismo.

Resolución de conflictos

Habrá ocasiones en las que tengamos que resolver los conflictos que surjan entre diferentes versiones de un módulo o fichero del mismo (recuerde que puede haber múltiples personas trabajando de forma concurrente sobre el mismo módulo) para que CVS continúe trabajando. Estos conflictos son normales cuando dos o más personas modifican a la vez exactamente la mismas partes de un fichero.

El procedimiento es simple:

1. CVS se quejará de un fichero al hacer un update o un commit.

2. Editamos ese fichero y encontraremos marcas del tipo:

```
[...]
>>>>>>>>>>>>>>
texto-opción-1
===========
texto-opción-2
<<<<<<<<<<<<<<
[...]
```

3. El texto entre marcas es el que produce el conflicto. Hay que elegir qué modificación nos gusta y borramos todo lo demás.

4. Si no quedan más conflictos volvemos a hacer el commit o update.

Añadir ficheros al módulo

No olvide que CVS controlará sólo los ficheros que se hayan descargado inicialmente desde el repositorio. Cualquier otro fichero o directorio de la jerarquía del módulo CVS será ignorado.

Si quiere añadir un nuevo fichero o directorio al módulo CVS hay que emplear la orden **cvs add**:

```
cd directorio-del-módulo
cvs add fichero
```

Si el fichero es binario (por ejemplo, una imagen .jpg) hay que tener la precaución de añadir la opción -kb:

```
cd directorio-del-módulo
cvs add -kb fichero
```

¿Por qué?, se preguntará el lector más intrépido. Resulta que CVS usa varias variables (en realidad son de RCS, que funciona por debajo de CVS). Si el fichero es binario es posible que se dé una combinación de bytes que coincidan con alguna de estas variables. Si así fuera, RCS/CVS modificaría el contenido y lo corrompería. También se debe a que el sistema de cálculo de diferencias que usan estos sistemas no está diseñado para trabajar con información binaria. Si se obra equivocadamente es probable que corrompamos los datos.

También quiero señalar que si bien se pueden gestionar ficheros binarios, no se hará control de versiones de los mismos. Sólo se guardará la última versión (al no disponer CVS de la funcionalidad necesaria para calcular diferencias de

ficheros binarios).

Tras la orden cvs add hay que hacer ejecutar de nuevo la orden **cvs commit** para incluir los nuevos ficheros en el repositorio CVS.

Algunas opciones que admite la orden *add* son:

- -ka : Trata los ficheros añadidos como texto ASCII (por defecto)
- -kb : Trata los ficheros añadidos como binarios.
- -m "mensaje" : Permite indicar el comentario inicial con que se añeden los ficheros (el mensaje debe ir entre comillas)

Eliminar ficheros del módulo CVS

Para eliminar un fichero del módulo CVS hay que emplear la orden **cvs remove** , despues de haber borrado el fichero de la copia local de trabajo (se puede usar cvs rm como abreviatura):

```
cd directorio-del-módulo
cvs remove fichero
```

Si queremos borrar físicamente los ficheros a la vez que los eliminamos del módulo deberemos usar:

```
cd directorio-del-módulo
cvs remove -f fichero
```

Ver diferencias respecto al repositorio

Podemos pedir a CVS que compare todos los ficheros de nuestra copia local del proyecto con los ficheros del repositorio mediante el comando **cvs diff**:

```
cvs diff [opciones]
```

La orden anterior produce un listado de todos ficheros modificados y sus diferencias, en el mismo formato que emplean las herramientas *diff* y *patch* de unix.

En caso de que queramos ver únicamente las diferencias de un fichero, podemos indicarlo como argumento a la orden diff

```
cvs diff [opciones] nombre_fichero
```

Algunas opciones que admite la orden *diff* son:

- -B : Ignora diferencias consistentes en añadir o eliminar únicamente líneas

en blanco.
- -b : Ignora diferencias en la cantidad de espacio en blanco. Esta opción trata todos los espacios en blanco como iguales.
- -c : Añade 3 líneas de contexto por encima y debajo de las líneas de diferencia. Util para emplear con el programa *patch*.
- -C NUM : Como la anterior, pero emplea NUM líneas de contexto.
- -i : Ignora diferencias entre mayúsculas y minúsculas.
- -u : Muestra la salida en formato de diff unificado.
- -w : Ignora todas las diferencias de espacio en blanco. Básicamente, una versión más fuerte de -b.

Cómo configurar servidor y cómo incormporar nuevos módulos al repositorio

Configuración del servidor

Para configurar un servidor CVS con acceso remoto hay básicamente tres formas:

- rsh/ssh: Esta primera opción implica tener una cuenta equivalente en el servidor CVS y no hace falta tocar ni inetd.conf ni ejecutar el servicio pserver. La seguridad de esta solución es nula a menos que se use ssh como sustituto de rsh.
- pserver: La segunda opción usa una cuenta genérica (la misma para todo el mundo o varias diferentes si se desean, pero no son cuentas con acceso local al servidor), necesita activar pserver desde inetd.conf (o xinetd.conf, si se usa este último). La seguridad es superior al método precedente en caso de usar rsh pero inferior al uso de ssh. La gran ventaja de este método es que los usuarios de CVS no necesitan ningún tipo de acceso local al servidor.
- Kerberos: Kerberos queda fuera de juego a mi modesto entender, ya que la infraestructura necesaria para usar este método de autenticación simplemente no tiene sentido para uso de CVS fuera de un mismo dominio administrativo (sin contar la complejidad de instalación y operación de un dominio Kerberos).

En la práctica, buena parte de los servidores CVS públicos usan la opción de pserver, al no ser necesario dar de alta cuentas de sistema a los usuarios en el servidor. Si el acceso va a ser mucho mas restringido (un pequeño grupo estable y de confianza), la opción de usar ssh es claramente preferible.

Si optamos por el método pserver, deberemos añadir una línea como la siguiente en el fichero inetd.conf:

```
cvspserver stream tcp nowait root /usr/sbin/tcpd /usr/bin/cvs -b /usr/bin -f --allow-root=/var/lib/
```

El valor /var/lib/cvs corresponde al directorio donde ubicaremos el repositorio en el servidor CVS y que denotaremos de ahora en adelante por la variable de entorno $CVSROOT.

Hay que asegurse también, en caso de estar usando TCP Wrappers (como es el caso del ejemplo) de que permitimos el acceso al servicio CVS (en el fichero /etc/hosts.allow con una línea como la siguiente. De lo contrario se van a rechazar las conexiones.

```
cvs: ALL
```

Clases de usuarios y tipo de acceso permitido.

El segundo punto a tener en cuenta es quién tiene acceso a los repositorios CVS y qué tipo de operaciones se pueden realizar. Basicamente, una vez creado el repositorio, se suelen realizar dos grandes grupos de operaciones:

- Adición/actualización de ficheros del repositorio (implica acceso en modo escritura). Aqui van los commit, add, remove, ...
- Actualizaciones en el cliente/creación de diferencias. Esto sólo implica acceso en modo lectura. Aqui van los checkout, update, diff, ...

El acceso en modo escritura al repositorio sólo se debe otorgar a las personas estrictamente necesarias, ya que en teoría el acceso al sistema se abre potencialmente bastante.

Creacion inicial del repositorio

A continuación se muestran los pasos a realizar para la creación inicial del repositorio CVS.

- Configurar /etc/inetd.conf para que lance el servidor pserver como se ha comentado arriba, en caso de usar el método de acceso pserver.
- Definir una cuenta administrativa local para CVS (en el ejemplo se llamará cvsadm) que será quien administre CVS. Esta cuenta será la propietaria de los ficheros de control del repositorio y quien pueda crear nuevos módulos en el mismo. Aprovecharemos para crear un grupo llamado cvs para gestionar más cómodamente los permisos del repositorio y sus módulos. El usuario cvsadm deberá ser parte del grupo cvs. Sin embargo esta cuenta no tienen porque tener acceso local al sistema en caso de usar el método de acceso pserver.
- Crear el repositorio en local en el servidor (en $CVSROOT), ejecutando las ordenes (si queremos colocar el repositorio en /var/lib/cvs):

```
export CVSROOT=/var/lib/cvs
mkdir -p $CVSROOT
cvs init
```

El repositorio conviene crearlo como root y tratarlo, desde el punto de vista de los permisos, como si fuera el directorio /etc. Todos sus ancestros sólo deberian ser escribibles por root. El propio $CVSROOT debería ser escribible por

el administrador del servidor CVS (cvsadm, como hemos indicado más arriba). NADIE MAS debería tener permisos de escritura en dicho directorio.

Esto se hace asignando su propiedad al usuario cvsadm y haciendo al grupo cvs el grupo de $CVSROOT. Además, sería conveniente que $CVSROOT tuviera activado el bit SGID, para que los subdirectorios y ficheros que se creen debajo hereden el grupo principal del fichero o directorio (e indirectamente, por el ajuste del valor de umask que hace el propio CVS en modo pserver, el permiso de escritura para el grupo cvs y que tenga así control sobre todo lo que se introduzca allí). En resumen, $CVSROOT debería tener los permisos:

```
drwxr-s---  10 cvsadm   cvs     1024 Oct  2 12:31  $CVSROOT
```

- El usuario cvsadm será el propietario de los ficheros que haya en $CVSROOT/CVSROOT. El único fichero que es necesario que tenga permisos de escritura para los usuarios de CVS es $CVSROOT/CVSROOT/history. El resto, deben tener sólo permiso de lectura. Esto es:

```
$CVSROOT/CVSROOT:           drwxr-x---  2 cvsadm   cvs   1024 Oct  2 12:52 CVSROOT
$CVSROOT/CVSROOT/history:   -rw-rw----  1 cvsadm   cvs   9609 Oct  2 13:01 history
$CVSROOT/CVSROOT/commitlog: -rw-rw----  1 cvsadm   cvs   9609 Oct  2 13:01 commitlog  (*)
$CVSROOT/CVSROOT/log.pl:    -r-xr-x---  1 cvsadm   cvs   9609 Oct  2 13:01 log.pl     (*)
$CVSROOT/CVSROOT/*:         -r--r-----  1 cvsadm   cvs   9609 Oct  2 13:01 (el resto de ficheros)

(*) Esto sólo es necesario si se va a usar la caracteristica loginfo con el script log.pl o similar
```

- Incorporar al grupo cvs los diferentes usuarios locales que van a representar a los usuarios remotos de pserver. En concreto, en mi sistema, existen cvsadm, cvsuser y anoncvs, que representan respectivamente al administrador del servidor CVS, a los usuarios CVS con permiso de escritura en el repositorio (previa autenticación) y a los usuarios con acceso sólo lectura en modo anónimo. La idea de este grupo (como se ha comentado arriba) es poder asignar los permisos locales a los ficheros de forma que tanto el administrador como los usuarios tengan acceso a ellos.
- Si se va a usar pserver, es muy importante que los usuarios que tengan que ver con CVS (cvsadm, anoncvs y cvsuser) no puedan tener acceso local al servidor (esto es, poner una contraseña no válida, un shell no válido, etc.)
- Los módulos iniciales para cada proyecto (manual, como, programa, etc.) los debe crear el administrador del servidor CVS. Luego los puede usar cualquiera. Así garantizamos que los permisos son los adecuados, y que todo esta «controlado y en orden».
- Para funcionalidades adicionales (envio de mensaje de corre cada vez que se hace un commit, actualización de un log de commit en el repositorio, publicacion en el web del estado del repositorio, etc.) el propio administrador del servidor CVS se puede encargar de todo. Todo lo necesario esta en la documentación oficial de CVS.

Todo lo anterior (especialmente el tema de permisos, usuarios necesarios, etc.) son conclusiones mías después de leer la documentacion de CVS y su FAQ. No está indicado en ningún sitio que se deba hacer exactamente así, ya que sólo se dan consejos muy generales y bastante dispersos por toda la documentación. Con esto

quiero decir que habría que hacer un análisis un poco más profundo sobre el posible impacto de seguridad.

Alta de usuarios

Para el método de acceso ssh

En este caso, los usuarios de CVS son directamente usuarios del sistema. Por tanto, el administrador del sistema donde resida el servidor CVS debe dar de alta los usuarios normales y habilitar los permisos adecuados en el repositorio (y sus diferentes ficheros) para que puedan llevar a cabo el tipo de acceso deseado.

Para el método de acceso pserver

En este caso, tenemos que usar los ficheros $CVSROOT/CVSROOT/passwd , $CVSROOT/CVSROOT/readers y $CVSROOT/CVSROOT/writers . El primero de ellos sirve para indicar qué usuarios remotos tienen acceso al repositorio y el segundo y tercero indican qué usuarios de entre los anteriores tiene acceso en modo sólo lectura y cuáles en modo lectura/escritura, respectivamente.

El formato del primero de ellos es similar al de los ficheros de contraseñas de Apache y de hecho se puede crear y manipular con la herramienta htpasswd. Decimos similar porque no es exactamente igual, al disponer de un tercer campo opcional que nosotros si usaremos. Veamos un ejemplo:

```
cvsadm:8BQYT0o1J7FI6:cvsadm
anoncvs:
hermes-team:8Ba2dFouJkxI6:cvsuser
lucasiano:gA7sdFouJk9X6:cvsuser
```

En el ejemplo anterior tenemos cuatro usuarios remotos, pero en realidad sólo tres usuarios locales. Los usuarios «remotos» son cvsadm, anoncvs, hermes-team y lucasiano. Estos son los nombres de usuario que se usarán durante el proceso de autenticación.

Sin embargo, una vez superada la autenticación correctamente (el segundo campo del fichero es la contraseña cifrada con el método crypt estándar; si no existe, como en el caso del usuario anoncvs, valdrá cualquier contraseña), se usarán los usuarios que se indican en el tercer campo para el acceso local a los ficheros del repositorio. Esto es, el usuario remoto lucasiano en realidad será el usuario local cvsuser en el servidor y por tanto tendrá acceso a los ficheros y directorios a los que pueda acceder dicho usuario. En caso de no existir este tercer valor, el nombre de usuario remoto y el local serán el mismo.

Esta funcionalidad es muy interesante, ya que nos permite crear únicamente una cuenta de usuario local en el servidor por cada tipo de acceso diferente necesario, y agregar después cuantas cuentas remotas queramos y mapearlas al

usuario local correspondiente. Como el fichero donde se realiza el mapeo es $CVSROOT/ROOT/passwd y este fichero está disponible a través del propio CVS, el administrador del servidor CVS no necesita permisos de administrador global de la máquina donde este ubicado el repositorio para dar de alta nuevos usuarios de CVS. Nótese que en este caso, el fichero con las contraseñas viaja sin cifrar por la red, con lo cual se recomienda usar el método de acceso ssh para este tipo de operaciones).

Para crear nuevos usuarios CVS podemos usar la orden htpasswd de Apache:

```
export CVSROOT=:ext:cvsadm@cvs.servidor.org:/var/lib/cvs
export CVS_RSH=/usr/bin/ssh
cd directorio-temporal-de-trabajo
cvs checkout CVSROOT
cd CVSROOT
htpasswd passwd usuario-nuevo  (nota 1)
  New password: no-se-ve-mientras-se-escribe
  Re-type new password: no-se-ve-mientras-se-escribe
vi passwd     (nota 2)
cvs add passwd (notas 1 y 3)
cvs commit
```

- nota 1: Si fuese el primer usuario del repositorio, el fichero passwd no existirá, y por tanto será necesario usar la opción -c de la orden htpasswd. De igual modo, habrá que indicar a CVS que existe un nuevo fichero llamado passwd para que lo tenga en cuenta y lo añada al repositorio al hacer el commit.
- nota 2: Si se desea la funcionalidad de mapear el usuario remoto a un usuario local concreto, se puede editar el fichero a mano y añadir el tercer campo, separándolo del segundo por ':'.
- nota 3: Para que esto funcione es necesario haber incluido el nombre del fichero passwd en el fichero $CVSROOT/checkoutlist previamente. (no se aconseja por motivos de seguridad a menos que el acceso administrativo a CVS se realice por medio de ssh, como se ha comentado más arriba y se ha hecho en el ejemplo mostrado).

Hemos mencionado más arriba los ficheros $CVSROOT/CVSROOT/writers y $CVSROOT/CVSROOT/readers. El proposito de ambos es poder delimitar aún más el tipo de acceso permitido a los usuarios remotos. Si un nombre de usuario aparece en el fichero $CVSROOT/CVSROOT/writers este usuario tendrá acceso a las operaciones que impliquen modificaciones al repositorio. Este fichero contiene simplemente el nombre de los usuarios «remotos», uno por línea, que tienen este tipo de acceso:

```
cvsadm
hermes-team
lucasiano
```

Si por el contrario aparece en el fichero $CVSROOT/CVSROOT/readers, sólo tendrá acceso en modo lectura y no podrá hacer ningún cambio en el repositorio. El formato es idéntico al fichero anterior.

En caso de que aparezca en ambos ficheros, obtiene acceso de sólo lectura. En caso de que no existan ninguno de los dos ficheros, el usuario obtiene acceso de escritura, así que asegúrese de crearlos durante la configuración inicial del repositorio (no vienen creados de serie).

Por último, hay que tener muy presente que el control de acceso a los módulos se hace en base a los permisos de los directorios y ficheros del repositorio. Por tanto, si se quieren módulos con usuarios completamente independientes, habrá que extender el esquema usuarios/grupos locales aquí presentado y jugar con los permisos. Esto supone una cierta complicacion para el administrador tanto del sistema como del repositorio CVS.

Apéndices

Bibliografía

- La página del propio CVS es http://www.nongnu.org/cvs/
- La documentación oficial de CVS se instala como un fichero info en sistemas GNU/Linux.
- En http://cvsbook.red-bean.com/ está disponible un libro (en inglés) documentando CVS.

Interfaces gráficas de usuario para CVS

Existen varias interfaces gráficas para CVS, en mayor o menor estado de desarrollo. Todas ellas se pueden encontrar en http://freshmeat.net

- Tortoise (http://www.tortoisecvs.org): Interfaz y cliente de CVS para MS Windows, integrado en el explorador de archivos. Muy completo y sencillo de usar.
- cvsgui (http://cvsgui.sourceforge.net): Una interfaz multiplataforma para CVS.
- tkcvs (http://www.twobarleycorns.net/tkcvs.html): Interfaz gráfica para CVS escrita en Tcl/Tk muy establecida y estable.
- cervisia (http://cervisia.sourceforge.net/): Interfaz gráfica para KDE.
- lincvs (http://www.lincvs.org/): Interfaz gráfica multiplataforma que usa la biblioteca grafica Qt.
- eccvs (http://eccvs.sourceforge.net/index.html): Interfaz gráfica fácil de usar diseñada para Gnome.
- PCL-CVS: Es una extensión de (X)Emacs que permite manipular ficheros gestionados con CVS de forma automática y transparente, disponible como parte de XEmacs.

Extensiones y otros recursos

- viewcvs (http://www.viewvc.org/): Es un interfaz web para CVS (sólo lectura).
- CVSweb (http://www.freebsd.org/projects/cvsweb.html): Otro interfaz web para CVS, desarrollado por los autores de FreeBSD.
- sandweb (http://sandweb.sourceforge.net): Otro interfaz web para CVS, con opciones de administración.
- cvs2cl.pl (http://www.red-bean.com/cvs2cl/): que es una herramienta escrita en Perl, para crear ficheros Changelog al estilo GNU. Puede encontrarse en

Licencia
