Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Desarrollando en grupo con CVS **(23646 lectures)**
Per **Josep Gayà**, ***Zebub*** ()
Creado el 08/06/2001 23:10 modificado el 08/06/2001 23:10

En este artículo vamos a explicar cómo usar CVS[1]. Un sistema que nos permite llevar a cabo proyectos con múltiples desarrolladores trabajando a la vez sobre el mismo código fuente.

Desarrollando en grupo con CVS

Josep Gayà Miralles[2]

v0.1, 25 de Mayo 2001

En este artículo vamos a explicar cómo usar CVS. Un sistema que nos permite llevar a cabo proyectos con múltiples desarrolladores trabajando a la vez sobre el mismo código fuente. Distribuído bajo licencia FDL[3].

---

1. Introducción

Las siglas CVS[4] corresponden a Concurrent Versions System o Sistema de Versiones Concurrente. Como su propio nombre indica, CVS es principalmente un sistema de control de versiones. Esto quiere decir que es capaz de guardar y mantener accesibles las diferentes versiones que se van produciendo de un mismo fichero (típicamente de código fuente).

Pero lo que hace a CVS especialmente útil no es el control de versiones, sino el control de concurrencia que este permite. En los sistemas tradicionales de control de versiones, un desarrollador consigue una copia del fichero, lo modifica y después devuelve la nueva versión al sistema. Durante todo este proceso, ningún otro desarrollador tiene permiso para acceder a ese fichero.

En cambio CVS permite que varios desarrolladores estén trabajando a la vez sobre el mismo fichero. Y en la mayoría de los casos es capaz también de resolver automáticamente los conflictos que se producen al entregar las modificaciones.

Es por esto que CVS es especialmente útil en los proyectos al estilo open-source desarrollados a través de internet, ya que CVS permite a cada desarrollador trabajar con independencia y sin tener que preocuparse demasiado de las modificaciones que estén haciendo los demás. CVS se ocupa de hacer todo ese trabajo.

Si alguno de los que leéis este artículo estáis pensando en empezar un proyecto de código abierto y queréis que desarrolladores de todo el mundo puedan ayudaros y participar en él no lo dudéis, CVS es vuestro sistema. Si por el contrario creéis que esto del CVS está muy bien pero de momento no lo necesitáis seguid leyendo. Seguro que al terminar el artículo encontráis que os puede ser útil en algo.

2. Instalación del sistema CVS

Describiremos a continuación el proceso que se debe seguir para instalar CVS en nuestra máquina a partir de los fuentes en formato tgz. La versión que usaremos para describir el proceso de instalación es la 1.11, que es la última disponible en el momento de escribir este artículo. Aunque seguramente todo lo que se diga servirá perfectamente para versiones anteriores y posteriores.

Si vuestra distribución posee sistema de paquetes entonces la instalación será tan sencilla como bajaros el paquete y ejecutar algo parecido a $ rpm -i cvs-1.XX.X-X.i386.rpm si usáis RPMs. O ejecutar $ apt-get install cvs si usáis Debian. Pero para describir en proceso de manera general y para ayudar a entender mejor los pasos a seguir mejor vemos cómo instalar CVS a partir del tgz con las fuentes.

Para empezar, nos bajaremos las fuentes de ftp.cvshome.org/pub/cvs-1.11/cvs-1.11.tar.gz y proseguimos con la secuencia de comandos normal en la instalación de un tgz con fuentes. Todo ello como superusuario:

```
# tar xzf cvs-1.11.tar.gz
# cd cvs-1.11
# ./configure
# make
# make install
```

Algunas opciones interesantes del configure son el --prefix=directorio, para cambiar el directorio de instalación. Y también --disable-client --disable-server para el caso en que sólo nos interese usar CVS como servidor o como cliente respectivamente. Si omitimos alguno de estos parámetros se compilará el ejecutable para funcionar de las dos formas. Estas opciones se incluyen porque el ejecutable que se usa para acceder al repositorio (cliente) y el que se usa para servir peticiones y administrar el repositorio (servidor) es el mismo.

3. Configuración del repositorio

Una vez ya tenemos instalado CVS, y si nuestra intención es usar el sistema para montar un repositorio, tenemos que llevar a cabo una serie de acciones. Lo primero es habilitar para CVS un directorio en nuestro sistema de ficheros. Si usamos, por ejemplo, /var/cvs haremos.

```
# export CVSROOT=/var/cvs
# cvs init
```

Con la variable de entorno CVSROOT, CVS sabe sobre que repositorio tiene que realizar las acciones. Es posible por lo tanto tener varios repositorios diferentes sobre la misma máquina simplemente cambiando esta variable de entorno. También es posible especificar el repositorio en la línea de comandos con la opción -d, pero es más cómodo definir la variable de entorno.

Ahora podemos ir al directorio /var/cvs/ y comprobaremos que se ha creado en su interior un directorio llamado CVSROOT. Este directorio es dónde se guardan los archivos necesarios para gestionar el repositorio.

En este punto tendremos que tomar una decisión con respecto a las acciones que queremos permitir que lleven a cabo los usuarios sobre el repositorio. Si las tareas de administración del repositorio como la modificación de ficheros del directorio CVSROOT, la gestión de módulos (que veremos más adelante) o la importación de fuentes la va a llevar a cabo el superusuario entonces los permisos del repositorio deberían ser del estilo de:

```
# ls -al $CVSROOT
drwxrwxr-x    4 root     root         4096 mar 10 16:42 .
drwxr-xr-x   16 root     root         4096 mar 10 16:16 ..
drwxrwxr-x    3 root     root         4096 mar 10 16:16 CVSROOT
```

Si por el contrario queremos tener la opción de permitir que otros usuarios administren el repositorio tendremos que hacer algunos cambios en los permisos. La opción que yo prefiero es crear un grupo de usuarios llamado cvsadmin y añadir al grupo los usuarios a los que decida dar permisos de administración sobre el repositorio. Como ejemplo crearemos el grupo y añadiremos al usuario pepito al grupo de administradores de CVS.

```
# addgroup cvsadmin
# adduser pepito cvsadmin
# chgrp -R cvsadmin $CVSROOT
```

Una vez hecho esto ya estamos en disposición de poner un proyecto bajo CVS. Supongamos que somos el usuario pepito y que en el directorio prueba tenemos todos los archivos de nuestro proyecto.

```
$ export CVSROOT=/var/cvs
$ cd prueba
$ cvs import pruebacvs gnu inicio
```

Con esto crearemos dentro el directorio $CVSROOT/pruebacvs que contendrá todos los ficheros de nuestro proyecto, ahora ya bajo el control de CVS. Los parámetros gnu e inicio son simples etiquetas que no tienen mucha importancia en la mayoría de casos, pero como CVS los requiere hay que ponerlos. Ahora podremos comprobar que todo ha funcionado correctamente.

```
$ cd ..
$ cvs co pruebacvs
```

Ahora se debería haber creado un directorio llamado pruebacvs en el que tendremos el mismo contenido que el directorio prueba. Bueno, no exactamente el mismo contenido. En cada directorio podremos ver un directorio adicional llamado CVS que no conviene borrar y que contiene información necesaria para que CVS haga su trabajo sin problemas.

4. Cómo definir módulos

A medida que vamos incluyendo archivos en nuestro repositorio y el árbol de directorios se va complicando más, puede resultar interesante usar la capacidad de CVS para definir módulos. Esto permite etiquetar determinados directorios para que sea más fácil acceder a ellos. Así bastará con poner el nombre de la etiqueta en vez de tener que poner todo el path necesario para acceder al directorio.

Esto se hace incluyendo estas etiquetas dentro del fichero modules que está situado en $CVSROOT/CVSROOT. Supongamos por ejemplo que queremos etiquetar el directorio situado en $CVSROOT/pruebacvs/documentos/articulos/ y llamarlo atajo. Entonces añadiremos al fichero modules la línea:

```
atajo pruebacvs/documentos/articulos/
```

De esta manera, cada vez que deseemos obtener una copia de $CVSROOT/pruebacvs/documentos/articulos/ bastará con hacer:

```
$ cvs co atajo
```

5. El caso especial de los archivos binarios

Hay que tener especial cuidado si en el proyecto hay ficheros binarios tales como imágenes o ejecutables. No es una buena idea ponerlos bajo el control de versiones de CVS, pero es necesario en muchos casos para mantener juntos todos los archivos necesarios para el proyecto. Este cuidado es necesario porque CVS realiza de forma automática algunas tareas que pueden tener consecuencias catastróficas si se realizan sobre ficheros binarios. Una de estas tareas es la conversión automática del salto de línea de estilo MS-DOS al salto de línea de estilo UNIX.

Es una cosa muy útil en los ficheros de texto cuando los usuarios de CVS usan diferentes plataformas para desarrollar, pero cuando se realiza la conversión sobre ficheros binarios el resultado puede resultar cuando menos curioso.

Para evitar problemas hay que informar a CVS de cuándo un fichero es binario para que lo trate como tal. Durante el proceso de importación del proyecto esto se hace con la opción -W. Por ejemplo, con la opción -W "*.exe -k 'b'" le decimos que todos los ficheros con la extensión .exe los trate como binarios. Podemos añadir tantas opciones de este estilo como deseemos para cada tipo de archivo binario que contenga nuestro proyecto. Con el ejemplo anterior, si tenemos varias imágenes en formato PNG haremos:

```
$ cvs import -W "*.png -k 'b'" pruebacvs gnu inicio
```

Podemos comprobar después que el CVS ha reconocido los ficheros como binarios con el comando cvs status, mirando que el campo Sticky Options contiene el valor -kb.

```
$ cvs status imagen.png
===================================================================
File: imagen.png                Status: Up-to-date
   Working revision:    1.1.1.1 Sat Mar 10 15:42:00 2001
   Repository revision: 1.1.1.1 /var/cvs/pruebacvs/php.png,v
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      -kb
```

También se deberá tener en cuenta el tipo de los ficheros al añadirlos con posterioridad al repositorio mediante el comando add. La forma correcta de añadirlos se comenta más abajo.

6. Trabajando con CVS

Como ya supongo que habréis imaginado, el uso de CVS va a cambiar un poco la manera habitual de trabajar. Pero todos estos cambios nos permitirán olvidarnos casi por completo de si alguien está trabajando sobre el mismo fichero que nosotros o de preocuparnos por poder volver a una versión anterior de nuestro fichero si más adelante lo necesitamos.

Lo más importante para coger rápidamente la filosofía de CVS es tener claro que no trabajamos sobre el proyecto es sí, sino que lo hacemos sobre una copia del proyecto. De esta forma, es fácil deducir que las principales tareas que tendremos que realizar con CVS serán las de conseguir una copia del proyecto para empezar el trabajo y entregar los cambios al servidor.

Esto se hace con los comandos de CVS checkout(o co para abreviar) y commit. Para conseguir la copia del módulo sobre el que deseamos trabajar hacemos:

```
$ cvs co nombre_del_modulo
```

Después ya podemos efectuar las modificaciones que deseemos y, una vez finalizadas ejecutar:

```
$ cvs commit nombre_del_modulo
```

Sencillo, ¿no?. Efectivamente, no todo es tan sencillo. Para empezar, hemos dicho antes que una de las cosas que nos soluciona CVS es todo lo relacionado con las modificaciones concurrentes. Entonces, ¿que ocurre si entre mi checkout y mi commit alguien ha hecho también un commit de un fichero que he modificado?.

Pues bien, en este caso al hacer nosotros el commit CVS nos informará de que alguien ha efectuado cambios en el mismo fichero y no nos dejará continuar. Necesitaremos ejecutar un comando adicional llamado update, que se encargará de sincronizar nuestra copia del repositorio con la copia del servidor.

```
$ cvs update
```

Al ejecutar este comando su salida nos informará del estado de cada uno de los ficheros con una letra que precede al nombre del fichero. Los ficheros actualizados los marcará con una 'U', los ficheros con modificaciones pendientes de entregar con una 'M' y los ficheros que han generado conflictos durante la actualización con una 'C'.

Hay que tener cuidado y tomar el tiempo necesario para revisar la salida de este comando, que es una de las mayores fuentes de desaguisados en el repositorio. Sobretodo hay que buscar los posibles conflictos que se han generado y resolverlos adecuadamente.

7. Reslución de conflictos

En la mayoría de casos, CVS es capaz de encajar dos versiones de un mismo archivo en una nueva versión el la que se reflejen todos los cambios de las dos versiones. Pero, en ocasiones, CVS no es capaz de realizar esta tarea automáticamente y tendremos que echarle una mano. Esto sucede cuando las modificaciones que se han hecho en las dos versiones tienen líneas en común. En este caso tendremos que editar el código fuente y decidir nosotros mismos que cambios son los que deben permanecer. Todo esto se ve más claro en el siguiente ejemplo:

Supongamos que la versión 1.4 del fichero driver.c contiene esto.

```
#include <stdio.h>

 void main()
 {
     parse();
     if (nerr == 0)
         gencode();
     else
         fprintf(stderr, "No code generated.\n");
     exit(nerr == 0 ? 0 : 1);
 }
```

La versión 1.6 esto otro.

```
 #include <stdio.h>

 int main(int argc,
          char **argv)
 {
     parse();
     if (argc != 1)
     {
         fprintf(stderr, "tc: No args expected.\n");
         exit(1);
     }
     if (nerr == 0)
         gencode();
     else
         fprintf(stderr, "No code generated.\n");
     exit(!!nerr);
 }
```

Y nuestra copia local, basada en la versión 1.4, contiene:

```
 #include <stdlib.h>
 #include <stdio.h>

 void main()
 {
     init_scanner();
```

```
        parse();
        if (nerr == 0)
            gencode();
        else
            fprintf(stderr, "No code generated.\n");
        exit(nerr == 0 ? EXIT_SUCCESS : EXIT_FAILURE);
    }
```

Como no tenemos la última versión del fichero, al hacer un commit no nos permitirá continuar y tendremos que hacer antes un update.

```
$ cvs update driver.c
  RCS file: /usr/local/cvsroot/yoyodyne/tc/driver.c,v
  retrieving revision 1.4
  retrieving revision 1.6
  Merging differences between 1.4 and 1.6 into driver.c
  rcsmerge warning: overlaps during merge
  cvs update: conflicts found in driver.c
  C driver.c
```

Entonces, CVS nos informa de que ha habido conflictos durante el reensablaje de las dos versiones y de que tendremos que resolver los conflictos manualmente. Editamos el fichero drivers.c y vemos lo siguiente:

```
  #include <stdlib.h>
  #include <stdio.h>

  int main(int argc,
           char **argv)
  {
      init_scanner();
      parse();
      if (argc != 1)
      {
          fprintf(stderr, "tc: No args expected.\n");
          exit(1);
      }
      if (nerr == 0)
          gencode();
      else
          fprintf(stderr, "No code generated.\n");
  <<<<<<< driver.c
      exit(nerr == 0 ? EXIT_SUCCESS : EXIT_FAILURE);
  =======
      exit(!!nerr);
  >>>>>>> 1.6
  }
```

Vemos que la zona de conflicto está claramente delimitada con `<<<<<<<', `=======' y `>>>>>>>'. La primera parte corresponde con nuestra versión del fichero, mientras que la segunda corresponde con la versión 1.6. Entonces decidimos los cambios a efectuar (obviamente deberemos como mínimo borrar las líneas `<<<<<<<', `=======' y `>>>>>>>') y dejamos el fichero de la siguiente forma:

```
  #include <stdlib.h>
  #include <stdio.h>
  int main(int argc,
           char **argv)
  {
      init_scanner();
      parse();
      if (argc != 1)
      {
          fprintf(stderr, "tc: No args expected.\n");
          exit(1);
      }
      if (nerr == 0)
          gencode();
      else
          fprintf(stderr, "No code generated.\n");
```

```
        exit(nerr == 0 ? EXIT_SUCCESS : EXIT_FAILURE);
    }
```

Entonces ya podemos hacer el commit con tranquilidad.

```
$ cvs commit driver.c
  Checking in driver.c;
  /usr/local/cvsroot/yoyodyne/tc/driver.c,v  <--  driver.c
  new revision: 1.7; previous revision: 1.6
  done
```

8. Cómo añadir y quitar ficheros del repositorio

Otros comandos útiles en CVS son los utilizados para añadir o quitar ficheros en el repositorio. Para añadir ficheros el comando utilizado es el add. Supongamos que queremos añadir el fichero holamundo.c al repositorio, entonces haremos:

```
$ cvs add holamundo.c
$ cvs commit holamundo.c
```

Es importante hacer un commit después de añadir un fichero, ya que sólo entonces se sube el fichero al servidor.

En el caso de que el fichero que vamos a añadir sea binario tendremos que incluir el modificador -kb a la línea de comandos. Un ejemplo:

```
$ cvs add -kb imagen.jpg
$ cvs commit imagen.jpg
```

Si lo que queremos hacer es quitar ficheros del repositorio el comando a utilizar es remove. De la misma forma tenemos que hacer un commit para reflejar los cambios en el repositorio.

```
$ cvs remove -f holamundo.c
$ cvs commit
```

9. Algunos comandos adicionales

Además de los que ya hemos comentado, existen otros comandos que pueden resultar útiles en determinadas ocasiones. A continuación comentaremos algunos de ellos:

- login - Para autentificarnos en un servidor CVS remoto.
- logout - Para dar por terminada la sesión en el servidor remoto.
- status - Para observar el estado de un fichero en el repositorio.
- history - Para ver el historial de un fichero.
- release - Para revisar el estado de la copia local del repositorio y para borrar la copia local.

10. Acceso remoto al repositorio

Todo lo visto hasta ahora os habrá podido parecer útil o interesante, pero seguro que más de uno habrá pensado ya que si sólo se puede acceder al repositorio de manera local el sistema CVS está un poco limitado. Pues no os preocupéis, que CVS permite la posibilidad de acceso remoto. Vamos a explicar aquí una de las varias opciones que tenemos que es el acceso mediante el servidor de passwords o pserver.

Para poner en funcionamiento este método de acceso remoto usaremos los servicios del demonio inetd. Añadiremos en el fichero de configuración inetd.conf que normalmente está en el directorio /etc/ la siguiente línea:

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

Y en el fichero /etc/services añadiremos lo siguiente:

```
cvspserver      2401/tcp
cvspserver      2401/udp
```

A continuación tendremos que crear a mano un fichero llamado passwd en el directorio $CVSROOT/CVSROOT/. Este directorio deberá contener líneas del estilo del fichero /etc/passwd:

```
usuario:clave_encriptada
```

Dado que CVS no proporciona ninguna herramienta para la creación de este fichero tendremos que echar mano de otras herramientas. La que suelo usar yo es htpasswd que viene con el apache. Para crear el fichero y añadir el usuario pepito haremos:

```
# cd $CVSROOT/CVSROOT/
# htpasswd -c passwd pepito
```

Con lo que nos pedirá una contraseña y su confirmación y nos creará el fichero. Después rearrancamos el inetd y ya tenemos funcionando el acceso remoto.

```
# killall -HUP inetd
```

Ahora ya podremos acceder remotamente a nuestro repositorio. Para ello tendremos que cambiar un poco la variable de entorno CVSROOT. Si queremos acceder a la máquina remota noesaqui.com con el usuario pepito usando el servidor pserver al repositorio situado en /var/cvs haremos:

```
$ export CVSROOT=:pserver:pepito@noesaqui.com:/var/cvs
$ cvs login
```

Entonces se nos pedirá la contraseña y se creará el fichero local .cvspass en el que se guardará la contraseña. De ahora en adelante podremos acceder al repositorio de la misma manera que si estuviéramos en la misma máquina.

Una cosa importante a tener en cuenta es que todos los usuarios que se autentifican mediante el pserver necesitan tener su usuario equivalente en la máquina. Esto es debido a que, cuando el usuario se haya autentificado, accederá a los ficheros del repositorio con los permisos que tenga ese usuario en el sistema.

Para evitar que tenga que definirse un usuario en la máquina por cada usuario de CVS se puede añadir un tercer campo al fichero de $CVSROOT/CVSROOT/passwd que indica el usuario con el que se accederá al repositorio. Así la estructura de las entradas en el fichero quedaría de la siguiente manera:

```
usuario_cvs:clave_encriptada:usuario_real
```

Así podremos crear, por ejemplo, un usuario llamado cvsuser y hacer que todos los usuarios que accedan a CVS lo hagan a través de este usuario genérico. Independizando así totalmente los usuarios de CVS de los usuarios de la máquina.

Lista de enlaces de este artículo:

1. http://www.cvshome.org
2. mailto:jgayaARROBAwanadooPUNTOes
3. http://www.gnu.org/copyleft/fdl.html
4. http://www.cvshome.org/

E-mail del autor: zebub _ARROBA_ ono.com
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=664