# II Curso de introducción a CVS

**Álvaro Sánchez-Mariscal** (`mariscal@di.uc3m.es`)

***Grupo de usuarios de linux***

***Univ. Carlos III de Madrid***

***2 de abril de 2003***

# Contenido

- ¿Qué es CVS?.
- ¿Qué no es CVS?.
- Conceptos básicos.
- Método pserver vs. método ssh.
- Una sesión de trabajo típica.
- Comandos más importantes.
- Conflictos.
- Comentarios.
- Direcciones de interés.
- Dudas y preguntas.

# ¿Qué es CVS? (I)

- Concurrent Versions System.
- Sistema cliente/servidor de control de versiones.
- Mantiene un repositorio de software centralizado, permitiendo que grupos de personas trabajen simultáneamente sobre él de forma distribuída.
- CVS se encarga de unificar las modificaciones que puedan hacer cada de los usuarios.
- Muy útil para proyectos en los que haya involucradas una o más personas.

# ¿Qué es CVS? (y II)

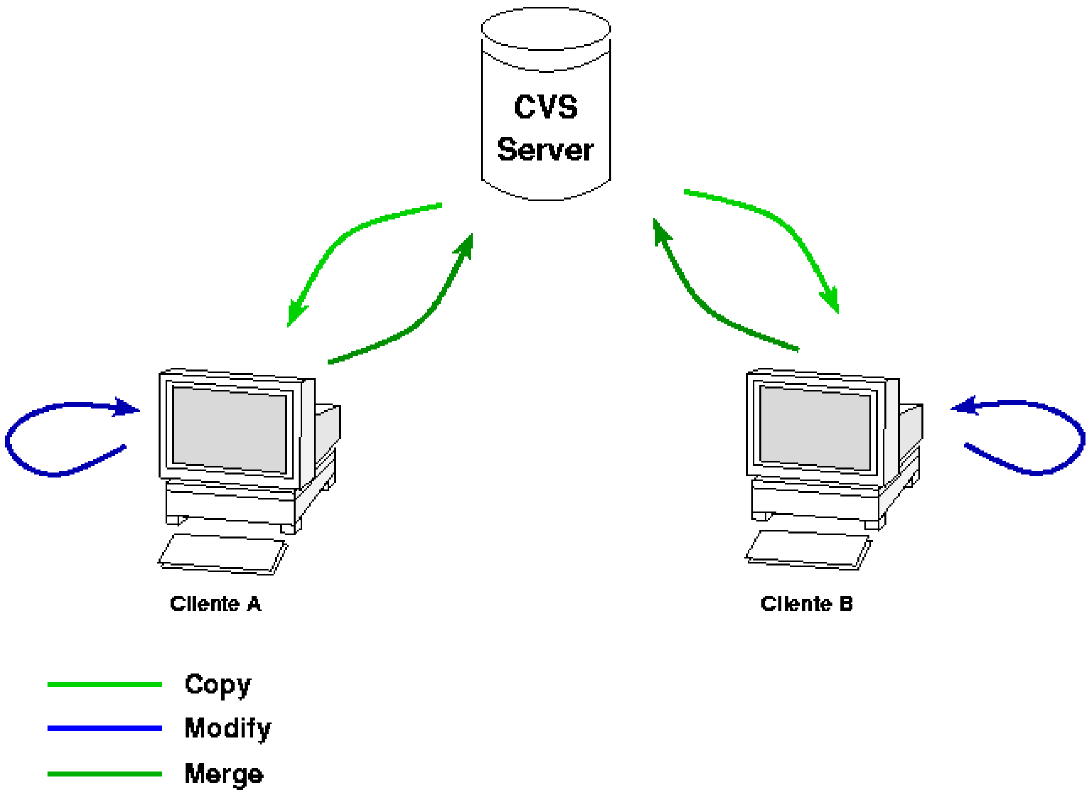


Figura 1: Esquema del funcionamiento de CVS

# ¿Qué no es CVS

- No construye aplicaciones (*make*), sólo almacena archivos.
- No sustituye la figura de coordinador o jefe de proyecto.
- No anula la comunicación entre desarrolladores, sólo trata conflictos de forma textual.
- No tiene un control de cambios sobre bugs.
- No depura ni prueba aplicaciones.

# Conceptos básicos (I)

**Repositorio** El directorio en el servidor a partir del cual se almacenan todos los ficheros (ej: `/home/cvs`).

**Módulo** Un directorio específico del repositorio. Cada proyecto se almacena en un módulo (ej: `practica-iissoo`).

**Check-out** Operación que sirve para **descargar** el contenido de un módulo. Se hace una sóla vez.

**Commit** Operación que sirve para **subir** los cambios que un desarrollador ha hecho sobre su copia local. Se suele hacer tras una sesión de trabajo, y sobre código que funciona (o que al menos compila).

**Update** Operación que sirve para actualizar la copia local con los cambios que hayan podido hacer otros desarrolladores. Se hace siempre antes de una sesión de trabajo.

# Conceptos básicos (y II)

**Revisión** Indicador numérico o alfanumérico que existe sobre cada fichero del proyecto (ej: `1.3`).

**Rama** Versión especial que se puede crear sobre una versión estable de un repositorio para optimizar código, probar nuevas funcionalidades, etc.

**Tag** Agrupación lógica que se puede hacer sobre un estado concreto de un proyecto (ej: `VERSION_1_0`). Un tag concreto no es más que las revisiones que tenían los fuentes en el momento de crear el tag.

`$CVSROOT` Variable de entorno definida en los clientes que indica el método de conexión con el servidor, el usuario, la ruta al repositorio en el servidor, etc (ej: `:pserver:anonymous@cvs.gnome.org:2401:/home/cvs`).

# Método pserver vs. SSH

- La autenticación por **pserver** carece totalmente de seguridad.
- Tanto las contraseñas como los ficheros viajan por la red sin cifrar.
- Además, un servidor pserver mal configurado puede comprometer la seguridad de toda la máquina.
- La ventaja es que los usuarios del servidor pserver no tienen porqué ser usuarios de la máquina.
- Con el método **ssh**, la comunicación con el servidor se realiza de forma segura.

- Pero con el método ssh, los usuarios del CVS tienen que ser usuarios del sistema (es decir, tener cuenta en `/etc/passwd` con una shell válida).
- En muchas situaciones no vamos a poder (o no vamos a querer) crear cuentas de usuario con acceso a una shell para que usen el CVS.
- Con el método pserver, el único usuario del sistema es el usuario no privilegiado `cvs`.
- Su uso actualmente sólo está recomendado para cuentas de usuario anónimas.

- Existen otras alternativas para aumentar la seguridad del servidor pserver.
- Muchas de ellas se basan en aplicar parches SSL al servidor CVS.
- El problema es que a veces también hace falta parchear los clientes.
- Una buena aproximación intermedia es la creación de un entorno *chroot* (enjaulado) en el que ejecutar el servidor SSH y el servidor CVS.
- De esta manera, podemos tener cuentas ficticias que se comunicarían con el servidor vía SSH.
- Si intentaran acceder por SSH a una shell, estarían dentro de un entorno restringido (la jaula) del que no podrían salir.
- Los usuarios normales de la máquina podrían acceder normalmente con otro servidor SSH en otro puerto.

 

# Configuración: método SSH

- Variables de entorno en el cliente:
  - Variable `CVS_RSH=/usr/bin/ssh`
  - Variable `CVSROOT=":ext:mariscal@cvs.hispalinux.es:/home/cvs"`
- El servidor sólo necesita tener instalado el paquete CVS (es a la vez cliente y servidor).
- Cada vez que se ejecute un comando CVS, se deberá introducir la contraseña (son accesos SSH normales y corrientes) a menos que esté configurado el acceso sin contraseña mediante claves públicas.

# Una sesión de trabajo típica (I)

- Nos autenticamos en el servidor (no necesario con método SSH):

```
$ cvs login
Logging in to :pserver:mariscal@cvs.hispalinux.es:2401/home/cvs
CVS password:
```

- Nos bajamos un módulo:

```
$ cvs co curso-cvs
cvs server: Updating curso-cvs
U curso-cvs/Makefile
U curso-cvs/curso-cvs.tex
U curso-cvs/gul.epsi
U curso-cvs/slnuevo.sty
```

# Una sesión de trabajo típica (II)

- Editamos y realizamos los cambios que sean necesarios.
- Subimos los cambios al repositorio:

```
$ cvs ci -m "Añadida sección 3" curso-cvs.tex
Checking in curso-cvs.tex;
/home/cvs/curso-cvs/curso-cvs.tex,v  <--  curso-cvs.tex
new revision: 1.5; previous revision: 1.4
done
```

# Una sesión de trabajo típica (y III)

- . . . días después volvemos a trabajar sobre el proyecto. Actualizamos nuestra copia local con los cambios que hayan podido hacer otros usuarios:

```
$ cvs up -d -P
? curso-cvs.dvi
? curso-cvs.ps
cvs server: Updating .
P Makefile
P curso-cvs.tex
```

# Comandos más importantes (I). Add

- `cvs add [-k flag] [-m "comentario"] ficheros...`
- Añade ficheros y/o directorios a un repositorio
- Para añadir un fichero que está en un directorio que no existe en el repositorio, primero se añade el directorio y luego el fichero
- Después de añadirlo hay que hacer un commit del fichero añadido
- Ejemplo
  - ```
    $ cvs add LEEME.txt
    cvs add: scheduling file ‘LEEME.txt’ for addition
    cvs add: use ’cvs commit’ to add this file permanently
    ```
  - `$ cvs add -kb logo.gif`
    ```

# Comandos más importantes (II). Checkout

- `cvs checkout|co|get [opciones] modulos...`
- Se baja del servidor CVS una copia del módulo o módulos especificados
- Se realiza sólo una vez
  - En el resto de ocasiones se harán *updates* sobre la copia local
- Ejemplo
  - `$ cvs co curso-cvs/curso-cvs.tex`

# Comandos más importantes (III). Commit

- `cvs commit|ci [-lnR] [-m 'msj' | -f fich] [-r revisión] \\ [ficheros...]`
- Sube al repositorio los cambios realizados en la copia local
- **Importante**: para poder hacer un commit debemos estar trabajando sobre la última versión de los archivos que hayamos modificado
- Ejemplo
  - `$ cvs ci -m "Añadido método toString()" Bot.java`

# Comandos más importantes (IV). Diff

- `cvs diff [-kl] [opciones] [[-r rev1 | -D fecha1] \\`
  `[-r rev2 | -D fecha2]] [ficheros...]`
- Compara archivos locales con las versiones del repositorio, o bien dos revisiones entre sí
- Como la utilidad `diff` de Unix/Linux
- Ejemplo
  - `$ cvs diff -r RELEASE_1_0 -r EXPR1`
  - Compara las diferencias entre los *tags* `RELEASE_1_0` y `EXPR1`

# Comandos más importantes (V). Export

- `cvs export [-flNnR] [-r rev|-D fecha] [-k flag] \\`
  `[-d dir] modulo...`
- Extrae los archivos de un módulo para producir una distribución independiente del CVS
- Es necesario indicar un *tag* (`-r`) o una fecha (`-d`)
- Funciona como *checkout*, pero no baja los directorios administrativos `CVS`

# Comandos más importantes (VI). Import

- `cvs import -m "comentario" nombre-modulo \\`
  `nombre-vendedor nombre-version`
- Instala un conjunto de archivos en el repositorio
- Se mantiene la organización de directorios original
- Añade todos los archivos y directorios que se encuentren en el directorio donde se ejecutó el import.
- Ejemplo:

```
$ cvs import -m "Repositorio Prueba" prueba malvarez v0
N prueba/README.txt
cvs import: Importing /home/i5/CVS/is0/prueba/test
N prueba/test/README.txt

No conflicts created by this import
```

# Comados más importantes (VII). Log

- `cvs log [opciones] [ficheros...]`
- Muestra el historial de los archivos especificados
- El historial es la información facilitada en los *commits*
- Como todos los comandos, si no se especifica ningún fichero, muestra un comportamiento recursivo

# Comandos más importantes (VIII). Remove

- `cvs remove [-flR] [ficheros...]`
- Elimina ficheros/directorios del repositorio
- Para poder eliminar un fichero, debe haber sido eliminado previamente de la copia local
- Esto se puede hacer en un sólo paso con la opción `-f`
- Después de eliminarlo, hay que hacer un *commit*
- Ejemplo
  - Este comando . . .

    ```
    $ rm *.c
    $ cvs remove
    ```

  - . . . equivale a este otro . . .

    ```
    $ cvs remove -f *.c
    ```

# Comandos más importantes (IX). Tag

- `cvs tag [opciones] nombre-tag [ficheros...]`
- Añade una etiqueta simbólica a un fichero o conjunto de ficheros de un módulo.
- Agrupa las revisiones que en ese momento tuviera cada archivo
- Por ejemplo, el tag `VERSION_1_0` podría estar formado por:

| | |
|---|---|
| ci.c | 1.4 |
| ident.c | 1.28 |
| rcs.c | 2.17 |
| rcsbase.h | 1.14 |
| rcsedit.c | 3.26 |
| rcsfcmp.c | 5.0 |
| rcslex.c | 2.7 |

# Comandos más importantes (X). Update

- `cvs update|up [opciones] [ficheros]`
- Actualiza la copia local con las versiones que existen en el repositorio
- Debe hacerse siempre antes de modificar cualquier archivo, para asegurarnos que estamos trabajando sobre las últimas versiones
- Con la opción `-d`, se crearán todos los archivos y directorios que no existan en nuestra copia local pero sí en el repositorio
- Con la opción `-P`, no se bajará aquellos directorios que estén vacíos
- Ejemplo:

  `$ cvs up -d -P`

# Conflictos (I)

- Existe la posibilidad de que dos desarrolladores estén modificando simultáneamente el mismo archivo.
- Es algo que se debe evitar: la **comunicación** es imprescindible.
- Uno de los dos hará primero el commit, aumentando la versión del fichero
- En ese instante, la segunda persona estará trabajando sobre una versión obsoleta. Cuando intente hacer el commit, CVS no le va a dejar
- Esta segunda persona debe entonces actualizar (update) el archivo
- Muy probablemente la copia local modificada no coincidirá con la nueva versión que subió el primer desarrollador
- Se ha producido un **conflicto**

# Conflictos (II)

- Veamos un ejemplo:

```
$ cvs up
cvs update: Updating .
RCS file: /var/cvs/web/menu.php,v
retrieving revision 1.5
retrieving revision 1.6
Merging differences between 1.5 and 1.6 into menu.php
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in menu.php
C menu.php
```

- Si la mezcla se puede hacer de forma limpia, CVS resolverá el conflicto automáticamente
- Pero la gran mayoría de las veces esto no va a ser posible

# Conflictos (III)

- Si el conflicto no ha podido ser solucionado, el segundo desarrollador deberá resolverlo
- CVS inserta unas marcas en el fichero fuente indicando lo que tenemos nosotros, y lo que había en el repositorio:

```
<<<<<<< menu.php
<a href="foro.cgi">Foro</a>
=======
<a href="libro.cgi">Libro de visitas</a>
>>>>>>> 1.6
```

- El segundo desarrollador deberá editar las líneas que dan conflicto, eliminar las marcas, y hacer un commit:

```
$ cvs ci -m "Enlace al libro de visitas arreglado" menu.php
```

# Comentarios (I)

- La opción `-k` de todos los comandos sirve para establecer el modo de expansión de las palabras clave (*keywords*)
  - Por ejemplo, con `-kb` le decimos que el fichero es binario, y que no intente realizar sustitución de palabras clave
- Lista de palabras clave más importantes

  **$Id$** `$Id: file1,v 1.1 1993/12/09 03:21:13 joe Exp $`

  **$Author$** `$Author: joe $`

  **$Date$** `$Date: 1993/12/09 03:21:13 $`

  **$Revision$** `$Revision: 1.1 $`

# Comentarios (II)

- La palabra clave `Log` se comporta de forma especial
  - Colocando esto en el fichero fuente ...

```
/*
 * $Log$
 */
```

  - ... se sustituiría por ...

```
/*
 * $Log: file1,v $
 * Revision 1.1  1993/12/09 03:30:17  joe
 * Revisión inicial subida al repositorio
 */
```

# Comentarios (III)

- Montando un servidor CVS con `xinetd`

```
service cvspserver {
        port        = 2401
        socket_type = stream
        protocol    = tcp
        wait        = no
        user        = root
        passenv     = PATH
        server      = /usr/bin/cvs
        server_args = -f --allow-root=/cvsroot pserver
}
```

- Agregando ficheros binarios

  `cvs add -kb ant.jar`

# Comentarios (IV)

- Alternativa a `htpasswd`: `crypt.pl`

```
#!/usr/bin/perl
# Uso: crypt.pl usuario
srand (time());
my $rl="(int(rand(26))+(int(rand(1)+.5)%2?65:97))";
my $salt = sprintf("%c%c", eval $rl, eval $rl);
my $plaintext = shift || die "Uso: crypt.pl usuario";
my $cryptext = crypt ($plaintext, $salt);
```

# Comentarios (V)

- Significado de las letras al hacer update o commit:

  **U** La copia local del fichero ha sido actualizada (updated).

  **P** Igual que U, pero sólo se realiza una descarga de un diff, en vez del fichero original (patched).

  **A** El fichero ha sido añadido a la copia local (added).

  **R** El fichero ha sido eliminado, y hay que actualizar el repositorio con un commit (removed).

  **M** El fichero ha sido modificado localmente (modified).

  **C** Ha habido un conflicto al intentar mezclar la copia local con la del repositorio (conflict).

  **?** El fichero está en el subdirectorio local, pero no pertenece al módulo.

# Direcciones de interés

- `http://www.cvshome.org` - Página oficial CVS.
- `http://cvsbook-red-bean.com` - Uno de los mejores libros sobre CVS. Disponible como paquete Debian :)
- `http://www.wincvs.org` - Front-end's gráficos para windows, linux y mac.
- `http://www.cvsnt.org` - Servidor CVS para Windows NT/2k.
- `http://www.red-bean.com/cvs2cl` - Crea ficheros Changelog al estilo GNU a partir del CVS
- `http://www.cooptel.qc.ca/ limitln/cvsadmin` - Herramienta para administrar las cuentas de un repositorio.
- `http://cvsauth.sourceforge.net` - Conjunto de parches para CVS que sirven para no ejecutar el servidor como root. Añade además cifrado SSL en las conexiones pserver tradicionales.

# Dudas y preguntas

¿. . . ?