PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR SEDE AMBATO

UNIDAD DE INGENIERIA DE SISTEMAS

DISERTACIÓN DE GRADO PREVIA LA OBTENCIÓN DEL TITULO DE INGENIERO DE SISTEMAS

"Estudio e Implantación de un Sistema de Control de Proyectos utilizando CVS en la Escuela de Ingeniería de Sistemas de la P.UC.E.S.A."

Marco Vinicio Altamirano Ruíz
Mónica Cristina Ortiz Núñez

DIRECTOR DE TESIS: Ing. Wigberto Sánchez, MS. C.

AMBATO, 2001

PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR
SEDE AMBATO

UNIDAD DE INGENIERÍA DE SISTEMAS

DISERTACIÓN DE GRADO PREVIA
LA OBTENCIÓN DEL TITULO DE
INGENIERO DE SISTEMAS

"Estudio e Implantación de un Sistema de Control de Proyectos utilizando CVS en la Escuela de Ingeniería de Sistema de la P.U.C.E.S.A."

DIRECTOR:

Ing. Wigberto Sánchez, MS. C.

REVISORES:

Ing. Natasha Bayas

Dra. Ana Larrea

Mónica Cristina Ortiz Núñez
Marco Vinicio Altamirano Ruiz

AMBATO, 2001

DEDICATORIA

Esta disertación final de mi carrera quiero dedicarla de manera total a mis padres y hermano quienes supieron brindarme su comprensión y cariño a lo largo de toda mi vida estudiantil especialmente a mi padre, que ha sido mi apoyo absoluto en todo instante con su esfuerzo y sacrificio para culminar con éxito mi carrera.

También dedico esta tesis a mi amado esposo quien con amor supo estar a mi lado alentándome y colaborando para que este sueño se convierta en realidad.

A mi adorada hijita, Joselyn Cristina que es la razón de mi existir, de mi esfuerzo y superación, quien ha sido mi fortaleza e impulso para no dejarme vencer y seguir siempre adelante con su ternura y amor.

Y a todas las personas que con cariño me demostraron su preocupación e interés en este sueño de obtener mi título.

Mónica Cristina Ortiz Núñez

AGRADECIMIENTO

Quiero agradecer de manera especial a mi Dios Padre, quien nunca me falla, siempre me ilumina y me acompaña a cada instante de mi vida.

A mis padres, Luis y Yolanda, quienes me brindaron su apoyo moral, económico y siempre estuvieron a mi lado en especial mi padre que es como mi ángel guardián, dedicándome su vida, su amor, guiándome, aconsejándome y tolerando día a día mi mal carácter, creo que no me alcanzará la vida para retribuir su esfuerzo.

No puedo dejar de agradecer a mi hermano Luis Antonio que con su preocupación y ayuda incondicional formó parte de este trabajo para así alcanzar su culminación.

A mi querido esposo Marco Antonio, quien ha sabido acompañarme y apoyarme en esta etapa final de mi carrera brindándome su colaboración y conocimientos. Además a mi tesoro Joselyn Cristina que con sus travesuras y ternura también fue parte de este trabajo.

A mi compañero y amigo de toda mi carrera universitaria quién con su paciencia ha sabido enseñarme y ayudarme en los momentos dificiles de este etapa.

A la Pontificia Universidad Católica del Ecuador Sede Ambato y todas las personas que pertenecen y pertenecieron en su momento. A mis maestros que supieron compartir sus conocimientos para poder llegar al término de mi carrera y además me brindaron su amistad y confianza. Un agradecimiento sincero a mi director Ing. Wigberto Sánchez y mis asesores Ing. Natasha Bayas y Dra. Anita Larrea quienes no solo en este lapso de tiempo me enseñaron y colaboraron, sino a lo largo del tiempo que fueron mis maestros.

Mónica Cristina Ortiz Núñez

DEDICATORIA

Esta Disertación de grado que para mi representa la realización de un gran sueño la dedico a mis padres Marco y Marujita quienes siempre me han brindado todo su amor esfuerzo, sacrificio y comprensión, a la memoria de mi inolvidable abuelita Mercedes, a mi amada esposa Mercy Elizabeth, que me ha entregado todo su amor paciencia y aliento, a mi adorada hija Andreita Michelle que con su sonrisa, ternura y travesuras ha sido mi inspiración y motivo de constante perseverancia para alcanzar los grandes anhelos , a mis queridos hermanos por su preocupación de toda la vida.

A todas las personas que han formado parte del desarrollo y culminación de este proyecto.

Marco Altamirano R.

AGRADECIMIENTO

Mi eterno agradecimiento a mis queridos padres que me han entregado todo su apoyo a lo largo de mi carrera, a mi esposa e hija que con su amor y comprensión me han impulsado día a día .

Mi eterna gratitud a la Pontificia Universidad Católica del Ecuador sede Ambato, a todos mis maestros que han impartido sus amplios conocimientos, a mi Director de Tesis Ing. Wigberto Sánchez, a mis asesores: Ing. Natasha Bayas y Dra. Anita Larrea quienes han sabido guiar el buen desarrollo de esta Disertación .

A mi compañera y amiga de siempre Cristina que ha estado junto a mi para brindarme su apoyo, y a todas las personas que de una u otra forma contribuyeron en la realización de este trabajo.

Marco Altamirano R.

INDICE

CAPITULO I. INTRODUCCION AL SISTEMA DE CONTROL DE VERSIONES

CAPITULO II. El DEPOSITO

CAPITULO III. COMENZAR UN PROYECTO CON CVS

CAPITULO IV. FICHEROS ESPECIALES

CAPITULO V. INSTALACIÓN E IMPLANTACIÓN

CAPITULO VI.

CAPITULO I

INTRODUCCION AL SISTEMA DE CONTROL DE VERSIONES

1.1. PLATAFORMAS PARA EL SISTEMA DE CONTROL DE VERSIONES (CVS)

El Sistema de Control de Versiones ha tenido un importante desarrollo para trabajar bajo diferentes plataformas como: Linux, Unix, Windows 9x, Windows NT, Mac, Sun.; debido a su buen desempeño multiplataforma se ha convertido en una herramienta indispensable para los verdaderos desarrolladores de software, y de proyectos, en los cuales intervienen múltiples desarrolladores.

Para el estudio de nuestro proyecto, la implantación se realizará en la plataforma Windows NT, como servidor; y las estaciones de trabajo con el sistema operativo Windows 9x, debido a que estas plataformas son las más utilizadas, tanto para el desarrollo de proyectos y de software, y por lo tanto prestando mayor factibilidad de uso e implantarlo en la PUCESA.

1.2. DESCRIPCION DEL SISTEMA DE CONTROL DE VERSIONES

El Sistema de Control de Versiones (CVS) empezó como un grupo de manuscritos, elaborado por Dick Grune en diciembre, 1986. Mientras ningún código actual de estos manuscritos están presentes en la versión actual de CVS muchos de los algoritmos de la resolución del conflicto de CVS viene de ellos. En abril, 1989, Brian Berliner diseñó y

codificó el CVS. Jeff Polk más tarde ayudó a Brian con el diseño del módulo de CVS y el soporte de apoyo del vendedor de CVS.

Es común en proyectos amplios tanto en dimensión como en distribución el uso de herramientas que permitan un control de versiones del mismo, como parte de las herramientas para controlar su ciclo de vida. Un sistema de control de versiones es aquél que permite controlar y conocer las modificaciones realizadas sobre un fichero de texto, un código fuente, etc.. de forma que facilita y gestiona el trabajo simultaneo en los grupos de desarrollo.

CVS ayuda también si es parte de un grupo de gente que trabaja en el mismo proyecto.
Es demasiado fácil sobre grabar cada cambio a menos que se tenga extremadamente cuidado. Algunos editores, intentan cerciorarse de que el mismo fichero nunca sea modificado por dos personas al mismo tiempo. Desafortunadamente, si alguien está utilizando otro editor, esa salvaguardia no trabajará. CVS soluciona este problema aislando los diversos reveladores de uno a uno. Cada revelador trabaja en su propio directorio, y CVS combina el trabajo cuando se hace cada revelador.

Por ejemplo, los fallos de funcionamiento entran silenciosamente a veces cuando se modifica el software lógico, y puede ser que no detecte el fallo de funcionamiento hasta un largo rato después de que se haga la modificación. Con CVS, se puede extraer fácilmente las viejas versiones para ver exactamente que cambios causó el fallo de funcionamiento. Esto puede a veces ser una gran ayuda.

CVS salva todas las versiones de un fichero en uno solo, de una manera lista que salva solamente las diferencias entre las versiones.

1.2.1. QUE NO HACE EL CVS

CVS no es un sistema de estructura.

Aunque la estructura del depósito y los módulos interactúen con su sistema de estructura, son esencialmente independientes.

CVS no dicta cómo construir cualquier cosa. Salva simplemente los ficheros para la extracción en una estructura arborescente que se ha diseñado.

CVS no dicta cómo utilizar el espacio de disco. Si se crea archivos o escrituras en cada directorio, entonces ellos tienen que saber las posiciones relativas de todo.
Al modularizar el trabajo, y construir un sistema de la estructura que comparta ficheros (vía conexiones, montajes, VPATH dentro `Makefile ' s, etc.), se puede arreglar el uso del disco si se lo requiere. Además se tiene que recordar que cualquier sistema en muchos trabajos de construir y de mantener, CVS no trata las ediciones implicadas.

Por supuesto, se debe colocar las herramientas creadas para utilizar tal sistema de la estructura (escrituras, `Makefile ' s, etc) bajo CVS.

El calcular fuera de los ficheros que necesitan ser reconstruidos cuando algo cambia, es otra vez, algo que está fuera del alcance de CVS. Un acercamiento tradicional es utilizar un poco de herramientas automatizadas para generar las dependencias que hacen aplicaciones.

CVS no es un substituto para la gerencia.

Se espera que los encargados de proyectos estén en contacto frecuentemente con la gerencia para asegurarse de que están enterados de horario, de nombres de ramificación y de fechas de desbloquear. Si no , CVS no puede ayudar.

CVS no es un substituto para la comunicación del revelador.

Cuando están hechos frente con conflictos dentro de un solo fichero, la mayoría de los reveladores manejan resolverlos sin demasiado esfuerzo. Pero una definición más general del " conflicto " incluye los problemas demasiado difíciles de solucionar sin la comunicación entre los reveladores.

CVS no puede determinarse cuando los cambios simultáneos dentro de un solo fichero, o a través de una colección entera de ficheros, estarán en conflicto lógicamente uno con otro. CVS no demanda ayudar a todos en calcular fuera de conflictos no-textuales o distribuidos en "program logic".

Por ejemplo: Si se cambia los argumentos a la función X definidos en fichero A . Al mismo tiempo, alguien corrige el fichero B, agregando nuevas llamadas a la función X usando los viejos argumentos. Estaríamos fuera de la capacidad de CVS's.

CVS no tiene control de cambio

El control del cambio se refiere a un número de cosas. Primero puede significar fallo de funcionamiento-bug-tracking, que permite guardar una base de datos de fallos de funcionamiento y el estado de cada uno.
Otro aspecto del control de cambio es que no pierde de vista de los datos que se cambiaron en algunos archivos fueron de hecho cambiados juntos en un cambio lógico.

Si se revisa en algunos ficheros y confía en la operación encargada al cvs; CVS entonces se olvida de que esos ficheros fueron revisados en conjunto, y el hecho de que tienen el mismo mensaje de registro es la única cosa que los tiene juntos.

Otro aspecto del control del cambio, en algunos sistemas, es la capacidad de no perder de vista el estatus de cada cambio. Algunos cambios han sido escritos por un revelador, otros han sido repasados por un segundo revelador, etc. Generalmente, la manera de hacer esto con CVS es generar un diff (usando cvs diff o diff) y email esto con alguien que pueda aplicar de la herramienta utilitaria. Esto es muy flexible, pero depende de los mecanismos exteriores del CVS para cerciorarse de que nada baja a través de las grietas.

CVS no tiene un modelo en el proceso de construcción

Algunos sistemas proporcionan a manera de asegurarse que los cambios o el desbloquear pasen con varios pasos de progresión, con varias aprobaciones según lo necesitado. Generalmente, uno puede lograr esto con CVS pero puede ser que sea un poco más trabajoso. También considere como si las características de las ramificaciones y de las etiquetas se pueden utilizar para realizar tareas tales como hacer el trabajo en un árbol de desarrollo y entonces combinando ciertos cambios a un árbol estable solamente se han probado una vez.

En conclusión podemos decir que: **UN SISTEMA DE CONTROL DE VERSIONES NOS PERMITE**:

- Mantener una base de datos con todos los documentos.
- Gestiona las peticiones de E/S de la base de datos.
- Mantiene una "historia" de la base de datos:
    - Registro de accesos.
    - Registro de modificaciones.
- Permite recuperar documentos previos a cambios.
- Permite diseñar distintas "ramas" de versiones.

Por lo tanto: **POR QUE USAR UN SISTEMA DE CONTROL DE VERSIONES?**

Existen varias razones para usar un sistema de control de versiones así:

- Permite un método para controlar las modificaciones hechas por un grupo grande de gente.

- Los cambios son ordenados y se recuperar fácilmente de desastres.
- Se pueden arreglar conflictos de diseño más rápidamente.
- Una lista exacta de los cambios se puede hacer inmediatamente, así como un control del progreso.

1.2.2. UNA SESIÓN DE PRUEBA

Como una manera de introducir CVS, iremos por una sesión de trabajo típico para empezar a utilizar CVS. La primera cosa que debemos entender es que CVS guarda todos los archivos en un depósito centralizado; el mismo que debe estar ya instalado. Supongamos que se esta trabajando con un compilador simple. La fuente consta de un grupo de archivos C y un Makefile. El compilador es llamado tc (compilador Trivial), y.el depósito es instalado así para que haya un módulo llamado tc.

1.2.2.1 ELABORANDO EL CODIGO FUENTE

La primera cosa que se debe hacer es adquirir su propia copia trabajada de la fuente tc. Para esto se utiliza el comando **checkout**:

```
$ cvs checkout tc
```

Esto creará un directorio nuevo llamado tc y se lo llena con los archivos de la fuente.

```
$ cd tc
$ ls
CVS Makefile backend.c driver.c frontend.c parser.c
```

El directorio del CVS es utilizado internamente por el CVS. Normalmente, no se debe modificar o quitar cualquiera de los archivos en él.

Si comienza con su editor favorito, comprimir el **backend.c**, y en un par de horas más tarde se tendrá agregado un paso de optimización al compilador.
Nota para los usuarios de RCS (Sistema de Control de Revisiones) y SCCS: hay una necesidad de asegurar los archivos que se quiere editar.

1.2.2.2 REALIZANDO SUS CAMBIOS

Cuando se ha verificado que el compilador es aún compilable se decide hacer una nueva versión de **backend.c**. Esto guardará su nuevo **backend.c** en el depósito y lo hará disponible para cualquiera que use el mismo depósito.

```
$ cvs commit backend.c
```

El CVS comienza un editor, que permite entrar a algunos mensajes. Tecleando **"Agregando un paso de optimización"**, guardar el archivo temporario, y salir al editor.

El ambiente inconstante **$CVSEDITOR** determina que editor se inició. Si el **$CVSEDITOR** no está colocado, entonces el ambiente inconstante **$EDITOR** está colocado, y este será utilizado. Si ambos, **$CVSEDITOR** y **$EDITOR** no están

colocados entonces hay una omisión que variará con su sistema operativo, por ejemplo **vi** para Unix o **notepad** para Windows NT/ 95.

Cuando el CVS inicia el editor, este incluye una lista de archivos que se modifican. Para el cliente de CVS, esta lista se basa en comparar las veces de modificación del archivo contra las veces de modificación que el archivo tubo cuando este fue abierto por última vez o actualizado. Además, el tiempo de modificación de un archivo ha cambiado pero su contenido no, este mostrará como se modificó. La manera más simple para manejar esto es absolutamente no preocuparse sobre esto – si se sigue con el **commit,** CVS detectará que los contenidos no son modificados y los tratará como un archivo inmodificable. El próximo **update** indicará al CVS el momento en que el archivo es inmodificable, y este reseteará su tiempo almacenado, así el archivo no se mostrará en futuras sesiones del editor.

Si se quiere evitar iniciar un editor se puede especificar los mensajes en la línea de comando utilizando en su lugar el parámetro **-m** , como éste:

```
$ cvs commit - m "Agregando un paso de optimización" backend.c
```

1.2.2.3 Limpiando

Antes de que se cambie a otras tareas se decide quitar su copia de trabajo del tc. Una manera aceptable para hacer eso es por supuesto :

```
$ cd . .

$ rm -r tc
```

pero una mejor manera para esto es usar el comando **release**:

```
$ cd . .

$ cvs  release  -d  tc

M driver.c

?  tc


Se tiene [ 1] archivos alterados  en este depósito.

Está  seguro que quiere eliminarlos (y borrarlos)  directorio  tc: n **` release'
```

El comando **release** chequea que todas las modificaciones han sido realizadas. Si la historia de cortes no es posible esto hace una nota en la historia del archivo.

Cuando se utiliza el parámetro **-d** con **release**, este además remueve su copia trabajada. En el ejemplo de arriba, el comando **release** escribió un par de líneas de rendimiento.

? tc significa que el archivo tc es desconocido para el CVS. Esto no es nada para preocuparse: tc es el compilador ejecutable, y esto no debería ser almacenado en el depósito. El **M driver.c** es más serio. Significa que el archivo **driver.c** ha sido modificado desde que fue chequeado.

El comando **release** siempre termina diciendo cuantos archivos modificados se tiene en su copia trabajada de las fuentes, y entonces le pide confirmar antes de anular cualquiera de los archivos o hacer cualquier nota en la historia del archivo.

Se debe decidir jugar a lo seguro y responder **n** (Enter), cuando **release** pide confirmación.

1.2.2.4 Diferentes vistas

Si no se recuerdan las modificaciones del **driver.c**, y se quiere ver qué le ha pasado a ese archivo entonces digitaremos:

```
$ cd tc
$ cvs diff driver.c
```

Este comando corre al diff para que compare la versión del **driver.c** que se revisó con su copia trabajada. Cuando se ve el rendimiento se recuerda que se agregó una línea de trabajo que posibilita la optimización del paso. Verifíquelo, y descargue el módulo.

```
$ cvs commit -m "Agregó un paso de optimización" driver.c
    Chequeando en driver.c;
    / usr/ local/ cvsroot/ tc/ driver.c, v
    - driver.c
    revisión nueva: 1.2; revisión previa: 1.1
    done
    $ cd . .
    $ cvs release - d tc
    ? tc
Se tiene los archivos alterados [ 0] en este depósito.
Está seguro que quiere eliminar (y anular) el directorio tc: y
```

CAPITULO II

EL DEPOSITO

El depósito de CVS salva una copia completa de todos los ficheros y los directorios que están bajo el control de la versión.

Normalmente, nunca se tiene acceso a cualquiera de los ficheros en el depósito directamente. En lugar de esto, se utiliza comandos de CVS para conseguir una propia copia de los ficheros en un directorio de funcionamiento, y después trabajar en esa copia. Cuando se ha acabado un conjunto de cambios, se los controla (o confia) nuevamente dentro del depósito. El depósito entonces contiene los cambios que se ha realizado, así como el registro exacto de lo que se cambió, cuando se lo cambió.

Obsérvese que el depósito no es un subdirectorio del directorio de funcionamiento, o viceversa; deben estar en localizaciones separadas.

CVS puede tener acceso a un depósito por una variedad de medios. Puede ser que esté en el ordenador local, o puede ser que esté en un ordenador a través de una red local LAN o a través del mundo. A distinguir varias maneras de tener acceso a un depósito, el nombre del depósito puede comenzar con un método de acceso . Por ejemplo, el método de acceso **:local:** significa tener acceso a un directorio del depósito, así que al

depósito: :local:/usr/local/cvsroot

significa que el depósito está dentro de:

/usr/local/cvsroot'

En el ordenador que ejecuta CVS.

Si se omite el método de acceso, si el depósito no contiene : (dos puntos), entonces **:local:** se asume. Si entonces : (dos puntos) contiene cualquier **:ext:** o **:server:** se asume. Por ejemplo, si usted tiene un depósito local dentro:

/usr/local/cvsroot

se puede utilizar / usr/local/cvsroot en vez de :local:/usr/local/cvsroot.

Pero si (por ejemplo bajo Windows NT) es su depósito local:

c:\src\cvsroot'

entonces se debe especificar el método de acceso, como en :local:c:\src\cvsroot .

El depósito está partido en dos porciones. **$CVSROOT/CVSROOT'** contiene los ficheros administrativos para CVS. Los otros directorios contienen los módulos reales definidos por el usuario.

2.1 INDICAR AL CVS DONDE ESTÁ SU DEPÓSITO

Hay varias maneras de indicar al CVS dónde encontrar el depósito. Se puede nombrar el depósito en la línea de comando explícitamente, con la opción - **d** (para el " directorio"):

cvs - d / usr/local/cvsroot checkout midirectorio/tc

O se puede fijar la variable de entorno **$$CVSROOT** a un camino absoluto a la raíz del depósito, /usr/local/cvsroot en este ejemplo. Para fijar **$$CVSROOT** , el **csh** y los utilizadores del **tcsh** deben tener esta línea en sus ficheros **.cshrc** o **.tcshrc**:

setenv CVSROOT / usr/local/cvsroot

los utilizadores sh deben tener estas líneas en su **.profile** o **.bashrc** :

CVSROOT =/usr/local/cvsroot export CVSROOT

Un depósito especificado con la variable de entorno - **d** reemplazará **$$CVSROOT**.

Una vez que se haya controlado una copia de trabajo hacia fuera del depósito, recordará donde está su depósito (la información se registra en el fichero CVS/Root en la copia de trabajo).

La opción – **d** y **CVS/Root,** ambos reemplazan la variable de entorno **$$CVSROOT**. Si la opción - **d** se diferencia de **CVS/Root**, se utiliza el anterior (y especificando - d causará que CVS/Root sea puesto al día).

2.2 CÓMO SE SALVAN LOS DATOS EN EL DEPÓSITO

Para la mayoría de los propósitos no es importante cómo CVS salva la información en el depósito. De hecho, el formato ha cambiado en el pasado, y es probable cambiar en

el futuro. Puesto que en casi todos los casos uno tiene acceso al depósito vía comandos de CVS, tales cambios no necesitan ser quebrantados.

Sin embargo, en algunos casos puede ser necesario entender cómo CVS salva datos en el depósito, por ejemplo puede ser que se necesite seguir bajo los bloqueos de CVS o puede ser que necesite ocuparse de los permisos del fichero, apropiados para el depósito.

2.2.1 DÓNDE SE SALVAN LOS FICHEROS DENTRO DEL DEPÓSITO

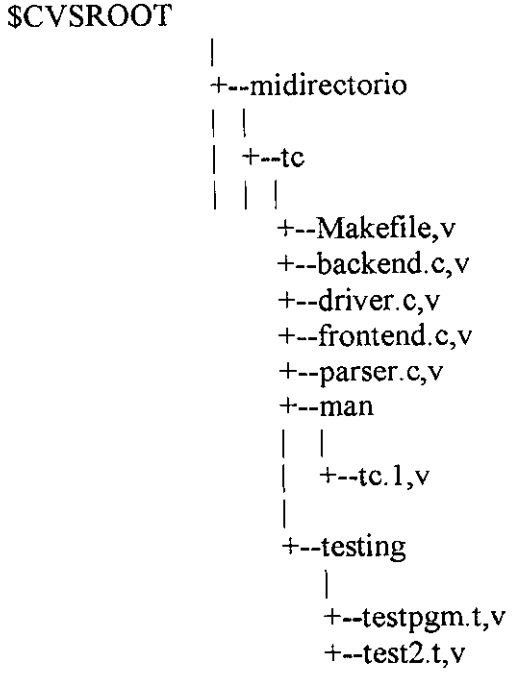
La estructura total del depósito es un árbol del directorio que corresponde a los directorios en el directorio de funcionamiento. Por ejemplo, supongamos que el depósito está dentro de:

/ usr/local/cvsroot

Aquí está un árbol posible del directorio (que muestra solamente los directorios):

```
/usr
|
+--local
| |
| +--cvsroot
| | |
| | +--CVSROOT
|   |    (administrative files)
|   |
|   +--gnu
|   | |
|   | +--diff
|   | | (source code to GNU diff)
|   | |
|   | +--rcs
|   | | (source code to RCS)
|   | |
|   | +--cvs
|   |    (source code to CVS)
|   |
|   +--midirectorio
|     |
|     +--tc
|     | |
|     | +--man
|     | |
|     | +--testing
|     |
|     +--(other Midirectorio software)
```

Con los directorios están los ficheros de historia para cada fichero bajo control de la versión. El nombre del fichero de historia es el nombre del fichero correspondiente con **,v** añadido al final del fichero. Aquí está lo que el depósito presenta **midirectorio/tc'** y el directorio es así:



Los ficheros de historia contienen, entre otras cosas, bastante información para reconstruir cualquier revisión del fichero, un registro de todo confía mensajes y el nombre de usuario de la persona que confió la revisión. Los ficheros de historia se conocen como ficheros de RCS , porque el primer programa para salvar ficheros en ese formato era un sistema de control de la versión conocido como RCS.

Los ficheros de RCS usados en CVS se diferencian de algunas maneras del formato estándar. La diferencia más grande es ramificaciones mágicas. También en CVS los nombres válidos de la etiqueta son un subconjunto de lo que valida RCS.

2.2.2 PERMISOS DEL FICHERO

Todos los ficheros ,v son creados inalterables, y no se debe cambiar el permiso de esos ficheros. Los directorios dentro del depósito deben ser escribibles por las personas que tienen permiso de modificar los ficheros en cada directorio. Esto significa normalmente que se debe crear un grupo de UNIX que consiste en las personas que deben corregir los ficheros en un proyecto, e instalar el depósito de modo que sea ese grupo el que posea el directorio. Esto significa que se puede controlar solamente el acceso a los ficheros sobre una base del pre-directorio.

Observe que los utilizadores deben también poder escribir el acceso del chequeo fuera de ficheros, porque CVS necesita crear los ficheros de bloqueo. También observe que los utilizadores deben poder escribir el acceso al fichero **CVSROOT/val-tags**. CVS lo utiliza para no perder de vista qué etiquetas son nombres válidos de la etiqueta (es a veces actualizado cuando se utilizan las etiquetas, así como cuando se crean).

Cada fichero de RCS será poseído por el utilizador quien pasará controlándole. Esto tiene poco significado; lo qué realmente importa es quién posee los directorios. CVS intenta instalar los permisos razonables del fichero para los nuevos directorios que se agregan dentro del árbol, pero se debe fijar los permisos manualmente cuando un nuevo directorio debe tener diversos permisos como el directorio principal. Si se fija la variable de entorno **CVSUMASK** que controlará los permisos del fichero que CVS utiliza en crear directorios y/o archiva en el depósito. **CVSUMASK** no afecta los permisos de archivo en el directorio de trabajo; tales archivos tienen los permisos que

son típicos para nuevamente crear archivos, excepto que a veces CVS los crea solo de lectura.

Para usar **pserver** (password server), se necesitará generalmente permisos más estrictos sobre el directorio **CVSROOT** y directorios arriba de este en el árbol. Algunos sistemas activos tienen rasgos distintivos que permiten correr un programa particular con capacidad para desempeñar las operaciones las cuales el llamador del programa no podría. Por ejemplo, el instalador del usuario ID (setupid) o el instalador del grupo ID (setgid) resaltan de Unix.

El CVS no estaba escrito para usar tales rasgos y por lo tanto atentan al instalar CVS en este modo que proveerá la protección solamente contra lapsos accidentales; alguien quien trata de circunvenir la medida será capaz de hacerlo, y dependiendo que si tiene instalado podría ganar más acceso que simplemente al CVS.

Se podría considerar instalar el **pserver**. Esto comparte algunos de los mismos atributos, en términos de posibilidad proveyendo un sentido falso de seguridad o abriendo más amplios hoyos de seguridades al que se está tratando de arreglar.

2.2.3 EMITIR UN PERMISO ESPECÍFICO DE UN ARCHIVO A WINDOWS

Algunas emisiones de permiso de archivo son específicos para operar sistemas Windows (Windows 95, Windows NT), y presumiblemente futuros sistemas operacionales en esta familia.

Si se esta usando CVS local y el depósito está sobre un sistema de archivo de red que es servido por el servidor Samba SMB, se han encontrado problemas con permisos. Permitiendo **WRITE=YES** en la configuración samba esto dice arreglar/área de trabajo.

2.2.4 EL ÁTICO

Se notificará que algunas veces el CVS almacena un archivo RCS en el **Attic** . Por ejemplo, si el **CVSROOT** es /usr/local/cvsroot y nosotros estamos hablando sobre el archivo **backend.c** en el directorio midirectorio/tc, entonces el archivo normalmente estaría en :

/usr/local/cvsroot/midirectorio/tc/backend.c,v

pero si va en el **attic**, estaría en:

/usr/local/cvsroot/midirectorio/tc / Attic /backend.c,v

Esto no complicaría desde el punto de vista del usuario si un archivo está en el atico; el CVS guarda la vía de esto y mira en el atico si es necesario. Pero en el caso que se quiera conocer, las reglas del archivo RCS se almacena en el atico y si solamente la revisión principal en el tronco a estado muerta.

Un estado muerto significa que el archivo ha sido removido, o nunca se agregó, para esa revisión. Por ejemplo, si se agrega un archivo en una rama, esta tendrá una revisión de tronco en estado muerto, y una revisión de rama en un estado no muerto.

2.2.5 EL DIRECTORIO DEL CVS EN EL DEPÓSITO

El directorio CVS en cada directorio de depósito contiene información tales como atributos de archivo (en un archivo llamado CVS/fileattr). En los futuros archivos adicionales puede agregarse a este directorio, entonces las implementaciones deberían ignorar silenciosamente archivos adicionales.

Este comportamiento es implementado por CVS 1.7 y posteriores

2.2.6 CVS ASEGURADO EN EL DEPÓSITO

Cualquier archivo en el depósito con un nombre que comienza con **#cvs.rfl** es un archivo de lectura asegurado. Cualquier archivo en el depósito con un nombre que comienza con **#cvs.wfl** es un archivo escrito asegurado. Las versiones viejas de CVS (antes de CVS 1.5) también crearon archivos con nombres que comienzan con **#cvs.tfl**.

El directorio **#cvs.lock** sirve como un seguro maestro. Es decir, se debe obtener primero este seguro antes de crear cualquier otros seguros.

Para obtener un seguro leído (readlock), primero se debe crear el directorio **#cvs.lock**.

Esta operación debe ser atómica (la cual debería ser seguro para crear un directorio bajo la mayoría de los sistemas operativos). Si fracasa porque el directorio ya existió, espere un momento e intente nuevamente. Después de obtener el seguro **#cvs.lock**, se debe crear un archivo cuyo nombre es llamado **#cvs.rfl** seguida por la información de

su elección (por ejemplo, nombre del huésped y el número de identificación de proceso).

Entonces remueva el directorio **#cvs.lock** para liberar el seguro maestro. Luego proceda con la lectura del depósito. Cuando ya se haya realizado esto, remueva el archivo **#cvs.rfl** para liberar el seguro de lectura.

Para obtener un seguro de escritura (writelock), primero se debe crear el directorio **#cvs.lock**, como con un seguro de lectura. Entonces chequee que no haya archivos cuyos nombres comiencen con **#cvs.rfl**. Si existe, remueva el **#cvs.lock**, espere un momento y trate nuevamente. Si no hay lectores, entonces cree un archivo cuyo nombre es **#cvs.wfl'** seguido por la información de su elección (por ejemplo, el nombre del huésped y el número de identificación de proceso). Cuélguelo en el seguro **#cvs.lock**. Procesado con escritura al depósito. Cuando este listo, primero remueva el archivo `**#cvs.wfl** y entonces el directorio **#cvs.lock**. Nótese que es poco común el archivo **#cvs.rfl**, el archivo **#cvs.wfl** es simplemente informativo; este no tiene efecto sobre la operación de seguridad más allá qué ser proveído para autocolgarlo en el seguro **#cvs.lock**.

Nótese que cada seguro (writelock o readlock) solamente asegura un directorio único en el depósito, incluyendo **Attic** y **CVS** pero no incluyen subdirectorios los cuales presentan otras direcciones bajo el control de versión. Para asegurar un árbol entero, se necesita asegurar cada directorio (nótese que si se fracasa para obtener cualquier seguro que se necesita, se debe liberar el árbol entero antes de esperar y tratar nuevamente, para evitar pérdida de los seguros).

Nótese también que el CVS espera seguros escritos para controlar accesos a archivos **foo,v'** individuales. El RCS tiene un plan donde el archivo **,foo,** sirve como un seguro, pero el CVS no lo implementa, entonces eliminar un **Writelock** CVS es recomendado.

2.2.7 COMO LOS ARCHIVOS SON ALMACENADOS EN EL DIRECTORIO CVSROOT

El directorio **$CVSROOT/CVSROOT** contiene los diversos archivos administrativos. De alguna manera este directorio es simplemente como cualquier otro directorio en el depósito; este contiene archivos RCS cuyos nombres terminan en **,v,** y muchos comandos del CVS operan en éste de la misma manera. Sin embargo, hay pocas diferencias.

Para cada archivo administrativo, además del archivo RCS, hay también una copia chequeada del archivo. Por ejemplo, hay un archivo RCS **loginfo,v** y un archivo **loginfo** que contiene la última revisión contenida en **loginfo,v**. Cuando se chequea en el archivo administrativo, CVS debería imprimir:

cvs commit: Rebuilding administrative file database (Reconstruyendo la base de datos del archivo administrativo)

y al actualizar la copia chequeada en **$CVSROOT/CVSROOT**; si no se hace hay algunos errores. Para agregar sus archivos propios a los archivos, se actualiza en esta manera, se puede agregar al archivo administrativo **checkoutlist**.

Por omisión, el archivo **modules** se comporta como se describe arriba. Si los módulos de archivo están muy grandes, almacénelos como un archivo de texto comprimido que haría mirar los módulos lentos. Por lo tanto, al hacer una apropiada edición del código de fuente del CVS uno puede almacenar los módulos de archivo en una base de datos. Si esta opción está en uso, entonces los módulos de la base de datos se almacenará en los archivos **modules.db**, **modules.pag**, y/o **modules.dir**.

2.3 CÓMO SE ALMACENAN LOS DATOS EN EL DIRECTORIO DE TRABAJO

Como con el depósito, CVS maneja esta información y uno puede acceder comúnmente por medio de comandos de CVS. Pero en algunos casos puede ser útil para mirarlos, y los otros programas, tal como para el usuario de interfaz gráfico **jCVS** o el paquete **VC** del paquete para emacs, podría necesitar observarlos. El directorio de **CVS** contiene varios archivos. Los programas que están leyendo este directorio debería ignorar silenciosamente archivos que están en el directorio pero que no están documentados aquí, para permitir futura expansión.

Root

Este archivo contiene la raíz del CVS actual.

Depósito

Este archivo contiene el directorio dentro del depósito el cual corresponde con el directorio actual. Esto puede ser por una ruta de nombre absoluto o una ruta de nombre relativo; el CVS ha tenido la capacidad para leer cualquier formato desde la última versión 1.3 en adelante. La ruta de nombre relativo a la raíz, es el acercamiento más sensible, pero la ruta de nombre relativo es bastante común y las implementaciones deberían aceptarlo. Por ejemplo, después del comando:

cvs - d :local:/usr/local/cvsroot checkout midirectorio/tc

Root, contendrá:

:local:/usr/local/cvsroot

y el **depósito,** contendrá:

/usr/local/cvsroot/midirectorio/tc

o

midirectorio/tc

Entries

Este archivo enumera los archivos y directorios en el directorio de trabajo. Es un archivo de texto según las convenciones apropiadas para el sistema de operación en cuestión. El primer carácter de cada línea indica qué clase de línea es esta. Si el carácter es irreconocido, los programas leídos del archivo deberían saltar silenciosamente esa línea, para permitir futuras expansiones. Si el primer carácter es /, entonces el formato es:

/name/revision/timestamp[+conflict]/options/tagdate/

donde **[and],** no son parte de la entrada, pero por otro lado indican que el + y el marcador de conflicto son optativos. **Name,** es el nombre del archivo dentro del directorio. **Revision,** es la revisión del archivo desde donde deriva el trabajo, o **0** para un archivo agregado, o - seguido por una revisión para un archivo quitado. **Timestamp,** es el tiempo de pega del archivo en el tiempo que el CVS lo creó; si el tiempo de pega (timestamp) difiere con el tiempo real de modificación del archivo esto significa que el archivo se ha modificado. Este está en el Tiempo Universal (UT), almacenado en el formato usado por el ISO C asctime() function (por ejemplo, `Sun Abr 7 01:29:26 1996'). Uno puede escribir un manuscrito que no está en ese formato, por ejemplo, **Result of merge** (Resultado de combinación), para indicar que el archivo debe siempre ser considerado para ser modificado. Esto no es un caso especial; para ver si un archivo es modificado un programa debería tomar el **timestamp** del archivo y simplemente hacer un manuscrito comparado con **Timestamp**. **Conflict**, indica que hubo un conflicto; si este es el mismo en el tiempo real de modificación del archivo significa que el usuario obviamente no ha resuelto el conflicto. **Opcion**, contiene opciones viscosas (por ejemplo `-kb' para un archivo binario). **tagdate** contiene **T** seguidas por un nombre de etiqueta, o **D** para una fecha, seguidas por una fecha o etiqueta viscosa. Nótese que si el **timestamp** contiene un par de fechas de estampe separados por un espacio, en lugar de un **timestamp** único, se reparte con una versión de CVS anterior de CVS 1.5. Si el primer carácter de una línea en **Entries** es **D**, entonces indica un subdirectorio. **D** sobre una línea indica que el programa que escribió el archivo **Entries** no grabó subdirectorios (por lo tanto, si hay tal línea y ningunas otras líneas que comienzan con D, uno sabe que no hay subdirectorios). De otra manera, la línea se mira así:

D/name/filler1/filler2/filler3/filler4

donde **name,** es el nombre de los subdirectorios, y todos los campos **filler** deberían ser ignorados silenciosamente, para futura expansión. Los programas que modifican los archivos de entradas deberían conservar estos campos.

Entries.Log

Este archivo no registra ninguna información más allá que la de **Entries**, pero esto no provee una manera para actualizar la información sin tener que reescribir el archivo **Entries** completamente, incluyendo la capacidad para conservar la información aún cuando el programa escrito **Entries** y **Entries.Log** abruptamente abortados. Los programas que están leyendo el archivo **Entries,** deberían además chequear **Entries.Log**. Si más tarde existe, ellos deberían leer **Entries** y entonces aplicar los cambios mencionados en **Entries.Log**. Después de aplicar los cambios, la práctica recomendada está reescribir **Entries** y entonces borrar **Entries.Log**. El formato de una línea en **Entries.Log** es un comando de carácter único seguido por un espacio seguido por una línea en el formato especificado para una línea en **Entries**. El carácter de comando único es **A** para indicar que la entrada está siendo agregada, **R** para indicar que la entrada está siendo quitada, o cualquier otro carácter para indicar que la línea entera en **Entries.Log** debería silenciosamente ignorarse (para futura expansión). Si el segundo carácter de la línea en **Entries.Log** no es un espacio, entonces esto fue escrito por una versión más vieja de CVS.

Entries.Backup

Este es un archivo temporal. El uso recomendado es para escribir un nuevo archivo de entrada a **Entries.Backup**, y entonces renombrarlo (donde sea posible) a **Entries.**

Entries.Static

La única cosa reelevante sobre este archivo es si existe o no. Si existe, entonces significa que la parte de un directorio se consiguió y CVS no creará archivos adicionales en ese directorio. Para aclararlo, use el comando **update** con la opción **-d,** la cual conseguirá los archivos adicionales y removerá **Entries.Static**.

Tag

Este archivo contiene etiquetas pegajosas pre - directorias o fechas. El primer carácter es **T** para etiqueta de rama, **N** para ninguna etiqueta de rama, o **D** para una fecha, u otro carácter que significa que el archivo debería ser silenciosamente ignorado, para futura expansión. Este carácter es seguido por la etiqueta o la fecha. Nótese que las etiquetas viscosas pre - directorias o fechas son utilizadas para cosas como aplicar archivos los cuales son nuevamente agregados; ellas no podrían ser las mismas como etiquetas pegajosas o fechas en archivos individuales.

Checkin.prog ; Update.prog

Estos archivos almacenan los programas especificados por las opciones - **i** y - **u** en los módulos de archivo, respectivamente.

Notify

Este archivo almacena las notificaciones (por ejemplo, para editar o no editar) que no han sido aún enviadas al servidor.

Notify.tmp

Este archivo es para **Notify** como **Entries.Backup** es para **Entries**. Es decir, para escribir **Notify**, primero escribe los nuevos contenidos a **Notify.tmp** y entonces (atomaticamente donde sea posible), renombrarlo a **Notify**.

Base

Si el observar esta en uso, entonces un comando **edit** almacena la copia original del archivo en el directorio **Base**. Esto permite que el comando **unedit** opere aún cuando es incapáz de comunicarse con el servidor.

Baserev

El archivo enumera las revisiones para cada uno de los archivos en el directorio **Base**. El formato es:

Bname/rev/expansion

Dónde expansion debería ignorarse, para permitir futura expansión.

Baserev.tmp

Este archivo es para **Baserev** como **Entries.Backup** es para **Entries**. Es decir, para escribir **Baserev**, primero escribe los nuevos contenidos a **Baserev.tmp** y entonces (atomicaticamente donde posible), renombrarlo a **Baserev**.

Template

Este archivo contiene la plantilla especificada por el archivo **rcsinfo**. Esto es solamente usado por el cliente; el no - cliente/servidor CVS consulta directamente **rcsinfo**.

2.4 ARCHIVOS ADMINISTRATIVOS

El directorio **$CVSROOT/CVSROOT** contiene algunos archivos administrativos. Se puede usar CVS sin cualquiera de estos archivos, pero algunos comandos trabajan mejor cuando el último archivo **modules** se establezca adecuadamente.

Lo más importante de estos archivos es el archivo **modules**. Esto define todos los módulos en el depósito. Este es una muestra del archivo **modules**.

```
CVSROOT        CVSROOT
modules        CVSROOT modules
        cvs    gnu/cvs
        rcs    gnu/rcs
        diff   gnu/diff
```

```
tc        midirectorio/tc
```

El archivo **modules** es una línea orientada. En su forma más simple cada línea contiene el nombre del módulo, un espacio en blanco, y el directorio donde el módulo radica. El directorio es una ruta relativa a **$CVSROOT**. Las últimas cuatro líneas en el ejemplo de arriba son ejemplos de tales líneas.

2.4.1 EDICIÓN DE ARCHIVOS ADMINISTRATIVOS

Editar los archivos administrativos del mismo modo que se editaría cualquier otro módulo. Utilice **cvs checkout CVSROOT** para conseguir una copia de trabajo, edítela y guarde los cambios en la manera normal. Es posible guardar un archivo administrativo erróneo. Se puede a menudo arreglar el error y chequear una nueva revisión, pero a veces un mal error particularmente en el archivo administrativo hace imposible guardar nuevas revisiones.

2.5 MÚLTIPLES DEPÓSITOS

En algunas situaciones es una buena idea tener más de un depósito, por ejemplo si se tiene dos grupos de desarrollo que trabajan sobre proyectos separados sin compartir cualquier código. Todos tienen que tener varios depósitos para especificar el depósito apropiado, usando la variable de ambiente **CVSROOT**, la opción **- d** a CVS, o (una vez que se ha chequeado un directorio de trabajo) simplemente permitiendo al CVS utilizar el depósito que se usó para chequear el directorio de trabajo

La gran ventaja de tener depósitos múltiples es que ellos pueden radicar sobre diferentes servidores. La gran desventaja es que no se puede tener un comando CVS único dentro de los directorios que vengan desde depósitos diferentes. Generalmente hablando, si se piensa instalar varios depósitos sobre la misma máquina, se podría querer considerar usar varios directorios dentro del mismo depósito.

2.6 CREAR UN DEPÓSITO

Para Instalar un depósito a un CVS, primero se debe escoger la máquina y el disco en que se quiere guardar la revisión de la historia de la fuente de archivos. Para estimar los requerimientos del espacio del disco, si se esta importando archivos RCS de otro sistema, el tamaño de esos archivos es el tamaño aproximado inicial de su depósito, o si comienza sin ninguna versión de la historia, es mejor dejar para el servidor aproximadamente tres veces el tamaño de la codificación de CVS por el depósito. En las máquinas en que los diseñadores trabajarán, el espacio del disco deberá ser para aproximadamente un directorio trabajado por cada desarrollador (sea el árbol entero o una porción de él, depende con lo que cada desarrolador trabaje).

El depósito debe ser accesible (directorio o un sistema de vía en un archivo de red) desde todas las máquinas las cuales quieran que utilice CVS ya sea directamente en el servidor o de forma local; las máquinas del cliente no necesitan tener un acceso a este por otra vía del protocolo del CVS. No es posible usar el CVS para leer desde un depósito el cual solamente tiene acceso de lectura. CVS necesita poder crear archivos seguros.

Para crear un depósito, hay que correr el comando **cvs init**. Se instalará un depósito vacío en una raíz específica del CVS en la forma usual.

Por ejemplo,

cvs - d/ usr/ local/ cvsroot init

El **cvs init** es cuidadoso de nunca sobreescribir cualquier archivo existente en el depósito, así no se daña si se corre **cvs init** en un depósito ya instalado.

El **cvs init** no podrá eliminar la historia; si no se quiere remover el archivo de la historia después de que se corre el **cvs init**.

2.7 RESPALDAR UN DEPÓSITO

No hay nada particularmente mágico sobre los archivos en el depósito; para la mayoría de los archivos es posible respaldarlos como cualquier otros archivos. De cualquier modo hay pocos problemas que considerar.

El primero es estar paranoico, uno no debería usar el CVS durante el **backup**, o tener un programa seguro de **backup** para el CVS mientras se hace el **backup**. Para no utilizar el CVS, se puede prohibir el registro (logins) a máquinas que pueden acceder al depósito, apagar el servidor del CVS o mecanismos similares. Los detalles dependerían del sistema operativo y cómo se tiene instalado el CVS.

Para asegurar el CVS, se crearía **#cvs.rfl** asegurándose cada directorio del depósito. Habiendo hecho todo esto, si se crea un **backup** sin cualquiera de estas precausiones,

los resultados son improbables, son particularmente fatales. Restaurando desde el **backup**, el depósito podría estar en un estado incoherente, pero esto no sería particularmente difícil para arreglarlo manualmente.

Cuando se restaura un depósito desde el **backup**, se asume que los cambios en el depósito fueron hechos después de que se creó el **backup**, directorios trabajados los cuales no fueron afectados por la falla podrían referirse a revisiones las cuales no son más extensas de las que existen en el depósito. Tratar de correr el CVS en tales directorios típicamente producirá un mensaje de error. De cualquier manera para arreglar estos cambios en el depósito es de la siguiente forma:

- Crear un directorio nuevo de trabajo.
- Copiar los archivos del directorio trabajado desde antes de la falla al nuevo directorio trabajado (por supuesto no copiar los contenidos de los directorios del CVS).
- Para trabajar en el nuevo directorio, utilizar comandos tales como **cvs update** y **cvs diff** para resolver que ha cambiado y entonces cuando está listo, **commit** cambia en el depósito.

2.8 CAMBIANDO UN DEPÓSITO

Respaldar los archivos en el depósito es tan fácil como respaldar cualquier otros archivos, si se necesita mover un depósito de un lugar a otro es tan fácil como mover cualquier otra colección de archivos.

La principal cosa a considerar es señalar los directorios trabajados al depósito. La manera más simple para tratar con un depósito cambiado es conseguir un directorio trabajado reciente después del cambio. Por supuesto, se debe asegurar que los antiguos directorios trabajados hayan sido revisados antes del cambio, o imaginarse cualquier otra forma para asegurarse que no se pierda cualquier cambio. Si verdaderamente se quiere reusar la existencia de los directorios trabajados sería posible con una operación manual sobre los archivos del CVS/ Depósito.

2.9 REPOSITORIOS REMOTOS

La copia hecha de las fuentes puede estar en una máquina diferente a la del depósito. Utilizando el CVS en esta manera se conoce como operación cliente/ server. Se corre el CVS en una máquina en la cual puede armar su directorio trabajado, conociéndole como el cliente, y decirle que se comunique a la máquina la cual puede armar el depósito, conociéndole como el servidor. Generalmente, usando un depósito remoto es como usar uno local, excepto que el formato del nombre del depósito es:

:method:[[user][:password]@]hostname[:[port]]/path/to/repository

Los detalles de exactitud de qué se necesita para ser instalado depende de como se está conectando al servidor.

Si el método no es especificado, y el nombre del depósito contiene : (dos puntos), entonces la falla es **ext** o **server**, dependiendo de su plataforma.

2.9.1 REQUISITOS DEL SERVER

La respuesta rápida para que responda la máquina satisfactoriamente como para un servidor es que los requisitos sean modestos, un servidor con 32M de memoria para que pueda manejar un árbol de fuente medianamente grande con una regular cantidad de actividad.

La respuesta real, por supuesto, es más complicado. Estimar las áreas conocidas del gran consumo de la memoria debería ser suficiente para estimar los requerimientos de la memoria. La primera área de importancia del consumo de la memoria es el gran espacio, cuando se utiliza el servidor del CVS. El servidor consta de dos procesos para cada cliente que lo esta utilizando. El consumo de la memoria en el proceso hijo debería permanecer medianamente pequeño. El consumo de la memoria en el proceso padre, particularmente si la conexión en red del cliente es lenta, se puede esperar que crezca ligeramente más que el tamaño de las fuentes en un directorio simple, o dos megabytes, de la que es más grande.

Multiplicando el tamaño de cada servidor del CVS por el número de servidores de los cuales se espera tener actividad se debería dar de una vez una idea de los requerimientos de la memoria para el servidor. Para la mayoría de las partes, la memoria consumida por el proceso padre probablemente puede ser un espacio cambiado en lugar de memoria física. La segunda área del gran consumo de memoria es **diff**, cuando se registra grandes archivos. Esto es requerido aún para archivos binarios.

La regla principal es dejar aproximadamente diez veces el tamaño del archivo más grande que se quiere verificar, aunque cinco veces podría ser adecuado. Por ejemplo, si se quiere verificar un archivo que está en 10 megabytes, se debería tener 100 megabytes de memoria en la máquina para hacer el chequeo (el servidor de la máquina para cliente/ servidor, o la máquina que corre el CVS para non-cliente/ servidor). Esto puede ser un espacio cambiado más que una memoria física. A pesar de que la memoria es requerida brevemente, no hay un particular necesario que deje a la memoria para más de un tal chequeo a la vez.

El consumo del recurso para el cliente es aún más modesto, cualquier máquina con capacidad suficiente para correr el sistema operativo en cuestión tendría pequeños problemas.

2.9.2 CONECTANDOSE CON RSH

El CVS utiliza el protocolo del rsh para desarrollar estas operaciones, así el huésped, del usuario del remoto necesita tener un archivo **.rhosts** el cual concede acceso al usuario local.

Por ejemplo supongamos que el usuario **mozart** en una máquina local **toe.grunge.com**, y la máquina del servidor es **chainsaw.yard.com**. Sobre **chainsaw** se pone la siguiente línea dentro del archivo **.rhosts** en **bach** del directorio de origen:

toe.grunge.com mozart

Entonces se prueba que rsh está trabajando con

Rsh - l bach chainsaw.yard.com 'echo $PATH'

Asegúrese que el camino que el rsh imprimió en el ejemplo de arriba incluya el directorio conteniendo un programa llamado cvs que es el servidor. Se necesita colocar la ruta en **.bashrc**, **.cshrc**, etc, no **.login** o **.profile**. Alternativamente, se puede colocar el ambiente variable **CVS_SERVER** sobre la máquina del cliente al nombre del archivo del servidor que se quiere utilizar, por ejemplo: / usr/ local/ bin/ cvs-1.6.

No es necesario editar **inetd.conf** o empezar un servidor CVS.

Hay dos métodos de acceso en **CVSROOT** para **rsh : server:** especifica un interno rsh el cual está apoyado por algunos puertos del CVS. **: ext:** especifica un programa rsh externo. Por defecto este es rsh pero no podría colocarse el ambiente variable **CVS_RSH** para invocar otro programa el cual puede acceder al servidor remoto (por ejemplo, remsh sobre HP-UX 9 porque el rsh es algunas veces diferente).

Este debe ser un programa que puede transmitir datos a/y desde el servidor sin modificarlo; por ejemplo el Windows NT, rsh no es conveniente puesto que por falla traduce entre el **RLF** y **LF**. El puerto OS/ 2, CVS tiene un atajo para pasar **- b** a **rsh** logrando rodear este, pero esto podría potencialmente causar problemas para otros programas que el estándar rsh, esto podría cambiar en el futuro.

Continuando el ejemplo, suponiendo que se quiere acceder al módulo **foo** en el **repository/ usr/ local/ cvsroot/**, sobre la máquina **chainsaw.yard.com**, está listo para continuar:

Cvs - d: ext: bach@ chainsaw.yard.com:/ usr/ local/ cvsroot checkout foo

(El bach@ se puede omitir si el nombre del usuario es el mismo en el huésped local y remoto).

2.9.3 CONEXIÓN DIRECTA CON LA CLAVE DE AUTENTICIDAD

El cliente de CVS puede conectarse al servidor usando un protocolo de contraseña. Esto es particularmente útil si utilizando el rsh no es factible (por ejemplo, el servidor esta detrás de un obstáculo), y Kerberos también no está disponible. Utilizar este método, es necesario para hacer algunos ajustes en el servidor y el cliente.

2.9.3.1 INSTALAR EL SERVIDOR PARA CONTRASEÑA DE AUTENTICIDAD

Lo primero de todo, probablemente se quiera enlazar los permisos en los directorios **$CVSROOT** y **$CVSROOT/ CVSROOT**.

En el lado del servidor, el archivo **/etc/ inetd.conf** necesita ser editado así **inetd**, sabe como correr el comando **cvs pserver** cuando recibe una conexión en el puerto correcto. Si por falla, el número del puerto es 2401; aunque esto sería diferente si el cliente estuvo recopilando con el puerto **CVS_AUTH_PORT** que define algo más.

Si su **inetd** permite nuevo puerto numerado en **/etc/inetd.conf**, entonces lo siguiente (todo en una sola línea inetd.conf') sería suficiente:

2401 stream tcp nowait root / usr/ local/ bin/ cvs

cvs -f -- allow – root = /usr / cvsroot pserver

Se podría usar también la opción -T para especificar un directorio temporal.
La opción **-- allow-root** especifica lo aceptable del directorio **CVSROOT**. Los clientes los cuales intentan usar un directorio diferente del **CVSROOT** no podrán conectarse.

Si hay más de un directorio **CVSROOT** que se quiera dejar, repita la opción.

Si su **inetd** quiere el servicio de un nombre simbólico en lugar de un nuevo número de puerto, entonces se pone lo siguiente:

/ etc/ services:

cvspserver 2401/ tcp y añada **cvspserver** en lugar de **2401** en **inetd.conf**.

Una vez que lo de arriba es guardado, reinicie su **inetd**, o haga cualquier cosa si es necesario para fortalecerlo o reeler sus archivos de inicialización.

A pesar de que el cliente guarda y transmite las contraseñas en claros textos, un archivo de contraseña CVS separado podría ser utilizado, así las personas no comprometen sus contraseñas regulares cuando ellos acceden al depósito. Este archivo es **$CVSROOT/ CVSROOT/ passwd**. Su formato es similar a **/etc/ passwd**, excepto que sólo este tiene dos o tres campos, nombre de usuario, contraseña, y nombre de usuario opcional para el servidor que utilice. Por ejemplo:

bach: ULtgRLXo7NRxs

spwang: 1sOp854gDF3DY

melissa:tGX1fS8sun6rY:pubcvs

qproj:XR4EZcEs0szik:pubcvs

(La contraseña es inscrita de acuerdo al estándar de la función del Unix crypt (), así es posible pegar directamente la contraseña desde los archivos regulares del Unix passwd).

Cuando una contraseña es autentificada, el servidor primero chequea por el usuario en el archivo **CVS passwd**. Si encuentra el usuario, se compara con la contraseña. Si no encuentra el usuario, o si el archivo **CVS passwd** no existe, entonces el servidor trata de coincidir la contraseña utilizando el sistema de rutina **user-lookup** (al utilizar el sistema de rutina **user-lookup**, se lo puede desactivar instalando **SystemAuth= no** en el archivo del **config**). Cuando se utiliza el archivo **CVS passwd**, el servidor corre

con el nombre del usuario especificado tercer argumento en la entrada, o como el primer argumento si no hay tercer argumento (de esta manera el CVS permite nombres de usuarios imaginarios con tal que el archivo **CVS passwd** indique el correspondiente sistema válido del nombre del usuario).

Es posible **mapear** los nombres del los usuarios del **cvs-specific** dentro del sistema nombres de usuario (por ejemplo, dentro del sistema login names) en el archivo **$CVSROOT/ CVSROOT/ passwd** añadiendo **:** (dos puntos) y el nombre del usuario del sistema después de la contraseña. Por ejemplo:

cvs: ULtgRLXo7NRxs: kfogel

generic: 1sOp854gDF3DY: spwang

anyone: 1sOp854gDF3DY: spwang

Así, alguien remotamente accederá al depósito en **chainsaw.yard.com** con el siguiente comando:

Cvs –d :pserver: cvs@chainsaw.yard.com: /usr/ local/ cvsroot checkout foo

Terminaría corriendo el servidor bajo el sistema identificado **kfogel** asumiendo una exitosa autentificación. De cualquier modo, el usuario remoto no necesariamente necesitaría saber la contraseña del sistema **kfogel's**, así como el archivo **$CVSROOT/ CVSROOT/ passwd** podría contener una contraseña diferente, utilizado solamente para el CVS. Y como el ejemplo anterior indica, es posible mapear múltiples nombres de usuario del cvs dentro de un simple sistema de nombres de usuario.

Esta distinción es diseñada para permitir que la gente del depósito acceda sin un completo sistema de acceso. Cualquier clase de acceso de depósito muy probablemente implica también un grado de acceso al sistema general. Inmediatamente, la manera de poner solamente una contraseña en el archivo CVS es para ubicarlo allí desde cualquier lugar.

2.9.3.2 USO DEL CLIENTE CON LA AUTENTIFICACIÓN DE LA CONTRASEÑA

Antes de conectarse al servidor, el cliente debe registrarse con el comando **cvs login**.

Registrándose se verifica la contraseña con el servidor, y además se graba la contraseña para transacciones posteriores con el servidor. El comando **cvs login** necesita saber el nombre del usuario, **hostname** del servidor, y el **path** del depósito completo, y se adquiere esta información desde el resumen del depósito o desde un ambiente variable del **CVSROOT**.

Cvs login is interactive == it prompts for a password:

Cvs - d: pserver: bach@ chainsaw.yard.com:/ usr/ local/ cvsroot login cvs password:

La contraseña es chequeada por el servidor; si es correcta, el **login** tiene éxito, por otro lado si falla, entiéndase que la contraseña fue incorrecta.

Una vez que se ha registrado se puede obligar al CVS a conectarse directamente al servidor y autentificarse con la contraseña guardada:

Cvs –d :pserver: bach@chainsaw.yard.com: /usr /local/ cvsroot checkout foo

El **:pserver:** es necesario porque sin este, el CVS asumirá que debería utilizar rsh para conectarse con el servidor. (Una vez que se tiene una copia chequeada y está corriendo el comando CVS desde adentro no hay ninguna necesidad para especificar el depósito explícitamente, ya que el CVS lo graba en el sub-directorio de la copia del CVS trabajada).

Las contraseñas son guardadas por presunción en el archivo $HOME/ .cvspass. Su formato es human-readable, pero no edita a menos que se sepa lo que se está haciendo. Las contraseñas no son almacenadas en **cleartext**, pero son trivialmente codificados para protegerlos de un compromiso inocente (por ejemplo, inadvertidamente es visto por un administrador de sistemas quien pasa mirando todo el archivo). La contraseña para una selección corriente del depósito remoto puede ser removido desde el **CVS_PASSFILE** utilizando el comando **cvs logout**.

El ambiente variable del **CVS_PASSFILE** atropella esta presunción. Si se utiliza esta variable, se debe asegurar de instalar antes el **cvs login** para que se corra. Si se instaló este después de correr el **cvs login**, entonces más tarde el comando CVS no podrá buscar la contraseña para transmitir al servidor.

2.9.3.3 CONSIDERACIONES DE SEGURIDAD CON LA AUTENTIFICACIÓN DE LA CONTRASEÑA.

Las contraseñas son almacenadas en el lado del cliente en un código trivial del **cleartext**, y transmitido con el mismo código. El código es hecho solamente para prevenir contraseñas inadvertidas que comprometan (por ejemplo, un administrador del sistema accidentalmente mira un archivo), y no prevendrá igual que un atacador ingenuo obtenga la contraseña.

El separar el archivo **cvs password** permite a las personas utilizar una contraseña diferente para que acceda al depósito, y no por el acceso **login**. En cambio, una vez que el usuario no ha leído solamente el acceso al depósito, ella puede procesar programas en el sistema del servidor a través de una variedad de recursos. Así, el acceso al depósito implica un amplio acceso al sistema regular. Podría ser posible modificar el CVS para prevenir que nadie haya hecho alguno de estos escritos. Además, podría haber otras maneras en las cuales teniendo acceso al CVS se permita a las personas ganar más acceso general al sistema; nadie ha hecho una auditoría cuidadosa.

Nótese que a pesar de que el directorio **$CVSROOT/ CVSROOT** contiene **passwd** y otros archivos los cuales son utilizados para chequear la seguridad, se debe controlar los permisos en este directorio tan herméticamente como los permisos en **/etc**. Las mismas aplicaciones al directorio **$CVSROOT** él mismo y cualquier directorio sobre él en el árbol. Cualquiera que tiene un acceso escrito como un directorio tendrá la

habilidad para llegar a ser cualquier usuario en el sistema. Nótese que estos permisos son típicamente más herméticos que se utilizarían si no se está utilizando **pserver**.
En resumen, cualquiera que consiga la contraseña logra un acceso al depósito, y en alguna medida un acceso al sistema general. La contraseña esta disponible para cualquiera que pueda indagar paquetes de redes o leer un archivo protegido (por ejemplo, utilizador de lectura solamente). Si se quiere real seguridad se debería adquirir **Kerberos**.

2.9.4 CONEXIÓN DIRECTA CON GSSAPI

El GSSAPI es una interfase genérica para los sistemas de seguridad de red como el **Kerberos 5**. Si se tiene una biblioteca de GSSAPI trabajando, se puede tener un CVS conectado por la vía de una conexión directa de un TCP, autentificada con GSSAPI.

Para hacer esto, el CVS necesita estar compilado con un soporte GSSAPI; cuando se configura el CVS se trata de descubrir si las bibliotecas GSSAPI que utilizan kerberos versión 5 están presentes. Se puede además utilizar la insignia **--with-gssapi** para configurar.
La conexión es autentificada utilizando GSSAPI, pero el mensaje corriente no es autentificado por omisión. Se debe utilizar la opción **- a global** para la autentificación de la solicitud corriente. El dato transmitido no está inscrito por falla. El soporte de inscripción debe ser compilado en ambos, el cliente y el servidor; utilizar la opción de configuración - **enable -encrypt** para comenzar. Se debe entonces utilizar la opción **–x global** para la solicitud de inscripción.

Las conexiones GSSAPI son manejados en el lado del servidor por el mismo servidor el cual maneja la contraseña de autentificación del servidor. Si se está utilizando un mecanismo GSSAPI como el Kerberos el cual provee una autentificación enérgica, probablemente se querrá desactivar la capacidad de autentificar por la vía de las contraseñas del **cleartext**. Hacerlo así, crea un vacío en el archivo de contraseña **CVSROOT/ passwd**, y se instala **SystemAuth = no** en el archivo configurado.

El servidor GSSAPI usa un nombre principal de **cvs/ hostname**, donde el nombre del **hostname** es el nombre canónico del huésped servidor. Se tendrá que instalar este como requerimiento para el mecanismo GSSAPI. Para conectarse con GSSAPI, utilice

: gserver:. Por ejemplo:

Cvs – d : gserver: chainsaw.yard.com:/ usr/ local/ cvsroot checkout foo

2.9.5 CONEXIÓN DIRECTA CON KERBEROS

La manera más fácil para usar kerberos es usar el **Kerberos rsh**. La principal desventaja de usar rsh es que todos los datos necesitan pasar por programas adicionales, así esto sería más lento. Así, si tiene kerberos instalado se puede conectar por la vía de conexión directa TCP , autentificándose con kerberos. Esta sección concierne el sistema de seguridad de red kerberos, versión 4. Kerberos versión 5 soporta por la vía de la interface de seguridad de red genérica GSSAPI.

Para hacer esto, el CVS necesita estar compilado con el apoyo de kerberos; cuando se configura el CVS se trata de descubrir si el kerberos esta presente o se puede utilizar la insignia **-with-krb4** para configurar.

El dato transmitido no es encriptado por presunción. El apoyo de inscripción debe estar compilado en ambos, el cliente y el servidor; utilice la opción - **a global** para configurar – **enabler -encryption** para correrlo. Se debe usar entonces la opción **-x global** para la solicitud de inscripción.

Se necesita editar el **inetd.conf** en la máquina del servidor para correr el **cvs kserver**. El cliente usa el puerto 1999 por omisión; si se quiere usar otro puerto lo especifica en el ambiente variable **CVS_CLIENT_PORT** en el cliente.

Cuando se quiere utilizar el CVS, consiga una entrada de la manera usual (generalmente kinit); esta debe ser una entrada la cual le permita cortar dentro de la máquina del servidor. Entonces está listo para ir:

Cvs - d: kserver: chainsaw.yard.com :/ usr/ local/ cvsroot checkout foo

2.10 ACCESO DE LECTURA AL DEPOSITO

Es posible conceder o transferir lectura solo de acceso del depósito para personas que utilizan la contraseña autentificada por el servidor. (Los otros métodos del acceso no tienen apoyo explícito para solamente leer usuarios porque todos esos métodos asumen

accesos login para cualquier forma de la máquina del depósito , sin embargo, el usuario puede hacer que cualquier permiso del archivo local sea hecha por ella).

El usuario quien tiene solo acceso de lectura puede hacer solamente esas operaciones con el CVS el cual no modifica al depósito, excepto por ciertos archivos administrativos (tales como archivos asegurados y archivo de historia).

Sería deseable usar este rasgo en relación con **user-aliasing**. Poco común con versiones previas del CVS, los usuarios de lectura solamente podrían meramente leer el depósito, y no excluir programas en el servidor o por otra parte ganar niveles inesperados de acceso. O para ser más exacto, conocer los agujeros que han sido tapados. A pesar de que este rasgo es nuevo y no ha recibido una intervención de seguridad comprensiva, se debería utilizarlo si los niveles de precaución parecen garantizados dado a su actitud de seguridad.

Hay dos maneras para especificar el acceso a la lectura solamente por el usuario: por inclusión y por exclusión.

Inclusión: Significa tener a punto ese usuario específicamente en el archivo **$CVSROOT/ CVSROOT/ readers**, el cual es simplemente una nueva lista separada de usuarios.

Aquí una muestra del archivo **readers**:

melissa

splotnik

jrandom

(no olvide la nueva línea después del último usuario.)

Exclusión: Significa una lista explícitamente de cada uno, quien tiene acceso de escritura si el archivo **$CVSROOT/ CVSROOT/ writers** existe, entonces solamente esos usuarios enlistados en este tienen acceso de escritura, cada uno además tiene acceso de lectura solamente (por supuesto, igual que los usuarios de lectura solamente aún todavía necesitan ser enlistados en el archivo CVS passwd). El archivo **writers** tiene el mismo formato que el archivo **readers**.

Nota: Si el archivo **CVS passwd** ubica a los usuarios del cvs dentro del sistema de usuarios, asegúrese de negar u otorgar acceso de lectura solamente usando los nombres de usuarios del cvs, no el nombre de usuario del sistema. Esto es, los archivos **writers** y **readers** contienen nombres de usuarios del cvs, que puede o no ser el mismo de los nombres de usuarios del sistema.

Aquí una completa descripción del comportamiento del servidor para decidir si otorga acceso de lectura solamente o de lectura escritura:

Si **readers** existe, y el usuario está enlistado en él, entonces ella logra acceso de lectura solamente. O si **writers** existe, y el usuario no está enlistado en él , entonces

también logra acceso de lectura solamente (esto es verdad aún si readers existe pero el usuario no está enlistado allí). Por otra parte, el usuario logra un acceso de lectura escritura completo.

Por supuesto hay un conflicto si el usuario está enlistado en los dos archivos. Esto es resuelto de la forma más conservativa, siendo mejor proteger al depósito mucho más: así el usuario logra acceso de lectura solamente.

2.11 DIRECTORIOS TEMPORARIOS CREADOS POR EL SERVIDOR

Mientras se corre, el servidor del CVS crea directorios temporarios. Llamados

cvs-serv

pid

En **pid** está el proceso de identificación del número del servidor. Ellos están localizados en el directorio especificado por el ambiente variable **TMPDIR**, la opción global **T global**, o faltando **/ tmp**.

En la mayoría de los casos el servidor quitará el directorio temporario cuando este ya este hecho, ya sea que esté terminado normalmente o anormalmente. De cualquier modo, hay pocos casos en que no remueve o no puede remover directorios temporarios, por ejemplo:

- Si el servidor aborta debido a un error del servidor interno, este podría preservar el directorio para ayudarse en la puesta a punto del programa.
- Si el servidor es eliminado de cualquier forma este no tiene manera de depurarse (la mayoría notablemente, en kill-KILL unix).
- Si el sistema cierra sin un cierre ordenado, que le dice al servidor depurarse.

En casos como este, se necesitará remover manualmente los directorios del **cvs-srv pid.**

Tan largo como sea, ningún servidor corre sin proceso de identificación de números **pid**, es seguro hacer así.

CAPITULO III

COMENZAR UN PROYECTO CON CVS

Para comenzar un proyecto en CVS, es necesario tomar en consideración que la retitulación de ficheros y la mudanza de ellos entre los directorios es algo incómodo, para lo cual primero se debe crear un nuevo proyecto pensando en su debida organización del fichero.

No es imposible retitular o mover ficheros, sino que aumenta el potencial para la confusión y CVS tiene algunos caprichos determinado en el área de retitular directorios.

3.1 CONFIGURANDO LOS FICHEROS

El primer paso de progresión es crear los ficheros dentro del depósito. Esto se puede hacer de diversas maneras.

- **De ficheros:** Este método es útil con los viejos proyectos donde ya existen los ficheros.
- **De otros sistemas de control de la versión:** Viejos proyectos donde se desea preservar historia de otro sistema.
- **De rasguño:** Crear un árbol del directorio del rasguño.

3.1.1 CREAR UN ÁRBOL DEL DIRECTORIO DE UN NÚMERO DE FICHEROS

Cuando se comienza a usar CVS, se tendrá probablemente ya varios proyectos que se puedan poner bajo control de CVS. En estos casos la manera más fácil es utilizar el comando de la importación. Un ejemplo es probablemente la manera más fácil de explicar cómo utilizarla. Si los ficheros que se desea instalar en CVS reside dentro de **wdir** y quisiera que aparecieran en el depósito como **$CVSROOT/midirectorio / rdir**, se deberá hacer esto:

```
$ cd wdir
$ cvs import -m "Imported sources" midirectorio/rdir yoyo start
```

A menos que se provea un mensaje del registro del indicador - **m**, CVS comienza un editor e incita para un mensaje. La cadena yoyo es una etiqueta del vendedor , y **start** es una etiqueta del desbloquear . No pueden llenar ningún propósito en este contexto, sino que CVS los requiere, entonces deben estar presentes.

Se puede ahora verificar qué trabajara, y ahora puede quitar su directorio de fuente original.

```
$ cd ..
$ cvs checkout midirectorio/rdir
$ diff -r wdir midirectorio/rdir
$ rm -r wdir
```

Borrar las fuentes originales es una buena idea, de cerciorarse de que no se corrige accidentalmente en el **dir** , desviando CVS. Por supuesto, sería sabio cerciorarse de que se tiene una salvaguardia de las fuentes antes de que se las quite.

El comando de comprobación puede tomar un nombre de módulo como argumento (como ha hecho en todos los ejemplos anteriores) o un nombre del camino en relación con **$$CVSROOT**, como hizo en el ejemplo arriba.

Es una buena idea controlar que los conjuntos de los permisos CVS en el interior de los directorios **$CVSROOT** son razonables, y que pertenecen a los grupos apropiados. Si algunos de los ficheros que se desea importar son binarios, se puede desear utilizar las características de las envolturas para especificar qué ficheros son binarios y cuáles no son.

3.1.2 CREAR FICHEROS DE OTROS SISTEMAS DE CONTROL DE LA VERSIÓN

Si se tiene un proyecto que se está manteniendo con otro sistema de control de la versión, tal como RCS, y se desea poner los ficheros de ese proyecto en CVS, y conservar la historia de la revisión de los ficheros.

De Rcs

Si se ha estado utilizando RCS (, encuentre los ficheros de RCS generalmente un fichero nombrado **foo.c** tendrá su fichero de RCS dentro **RCS/foo.c, v**. Entonces se

debe crear los directorios apropiados en CVS si estos aún no existen. Luego se copian los ficheros en los directorios apropiados en el depósito de CVS (el nombre en el depósito debe ser el nombre del fichero de fuente con ,v agregado; los ficheros entran directamente en el directorio del depósito apropiado, no en el subdirectorio RCS).

Este es uno de los pocos momentos en que es una buena idea tener acceso al depósito de CVS directamente, más bien se ordena CVS. Entonces se está listo para controlar fuera de un nuevo directorio de funcionamiento. El fichero de RCS no debe ser bloqueado cuando se lo mueve en CVS.

De otro sistema de control de la versión

Muchos sistemas de control de la versión tienen la capacidad de exportar ficheros de RCS en el formato estándar. Si este es el caso, exportar el RCS clasifica y después sigue las instrucciones antedichas. Fallando eso, la mejor propuesta es probablemente buscar un método que controle fuera de los ficheros una revisión al mismo tiempo que usa el interfaz de línea de comando al otro sistema, y después controla las revisiones en CVS.

De Sccs

Hay una escritura en **contrib** el directorio de la distribución de la fuente de CVS llama a **sccs2rcs** el mismo que convierte ficheros de SCCS a los ficheros de RCS.

Nota: Esto se debe ejecutar en una máquina que tenga SCCS y RCS instalados.

De Pvcs

Hay una escritura en **contrib** el directorio de la distribución de la fuente de CVS **pvcs_to_rcs** el mismo que convierte archivos de PVCS a los ficheros de RCS. Esto se debe ejecutar en una máquina que tenga PVCS y RCS instalados.

Crear un árbol del directorio del rasguño

Para un nuevo proyecto, lo más fácil a hacer es probablemente crear una estructura vacía del directorio, como esto:

```
$ mkdir tc
$ mkdir tc/man
$ mkdir tc/testing
```

Después de esto, si se utiliza el comando de importación para crear la estructura (vacía) correspondiente del directorio dentro del depósito:

```
$ cd tc
$ cvs import -m "Created directory structure" midirectorio/dir yoyo start
```

Entonces, utilizar **add** para agregar ficheros (y nuevos directorios) según como aparezcan. Controle que los conjuntos de los permisos de CVS en el interior de los directorios **$CVSROOT** son razonables.

3.2 DEFINIR EL MÓDULO

El paso de progresión siguiente es definir el módulo en el fichero **modules**. Esto no es terminantemente necesario, pero los módulos pueden ser convenientes en agrupar juntos ficheros y directorios relacionados.

En casos simples estos pasos de progresión son suficientes para definir un módulo.

1. Consiga una copia de trabajo del fichero de los módulos.

```
$ cvs checkout CVSROOT/modules
$ cd CVSROOT
```

2. Corrija el fichero e inserte una línea que defina el módulo. Se puede utilizar la siguiente línea para definir el módulo tc :

```
tc midirectorio/tc
```

3. Confie sus cambios al fichero de los módulos.

```
$ cvs commit -m "Added the tc module." Modules
```

4. Borrar el módulo modules.

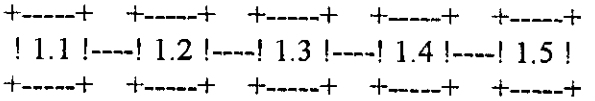
```
$ cd ..
$ cvs release -d CVSROOT
```

3.3 REVISIONES

Para muchas aplicaciones del CVS, uno no necesita preocuparse demasiado de números de la revisión; el CVS asigna números tales como 1,1 ; 1,2, etcétera, y eso se necesita saber. Si uno desea no perder de vista un conjunto de revisiones que implican más de un fichero, tal como las revisiones entraron a un desbloqueador determinado, uno utiliza una etiqueta, que es una revisión simbólica que se puede asignar a una revisión numérica en cada fichero.

3.3.1 NÚMEROS DE LA REVISIÓN

Cada versión de un fichero tiene un número único de la revisión. Los números de la revisión aparecen 1.1, 1.2, 1.3.2.2 o aún 1.3.2.2.4.5. Un número de la revisión tiene siempre un número par de números decimales enteros período-separados. Por valor por defecto la revisión 1,1 es la primera revisión de un fichero. Cada revisión sucesiva es dado un nuevo número aumentando al número de la derecha en uno. La figura siguiente visualiza algunas revisiones, con más revisiones nuevas a la derecha.



Es también posible continuar hacia arriba con los números que contienen más de un período, por ejemplo 1.3.2.2. Tales revisiones representan revisiones en ramificaciones.

3.3.2 VERSIONES, REVISIONES Y DESBLOQUEAR

Un fichero puede tener varias versiones, según lo descrito arriba. Asimismo, un producto de software puede tener varias versiones. Un producto de software se da a menudo un número de versión por ejemplo 4.1.1. Las versiones en el primer sentido se llaman revisiones en esta oportunidad y las versiones en el segundo sentido se llaman desbloquear.

3.3.3 ASIGNAR REVISIONES

Por defecto, CVS asignará revisiones numéricas dejando al primer número igual e incrementando el segundo número. Por ejemplo, 1,1; 1,2; 1,3; etc.

Al agregar un fichero nuevo, el segundo número será siempre uno y el primer número igualará el primer número más alto de cualquier fichero en ese directorio. Por ejemplo, el directorio actual contiene los ficheros que revisiones numeradas más altas son 1,7; 3,1; y 4,12, después un fichero agregado será dado la revisión numérica 4,1.

Normalmente no hay razón de cuidar sobre los números de la revisión es más fácil tratarlos como números internos que CVS mantenga, y las etiquetas proporcionan a una manera mejor de distinguir entre las cosas como el desbloquear 1 contra el desbloquear 2 de su producto. Sin embargo, si se desea fijar las revisiones numéricas, la opción **-r** de cvs commit puede hacer eso. La opción **-r** implica la opción **-f**, en el sentido que hacen los ficheros ser confiados incluso si no se modifican.

Por ejemplo, para traer todos sus ficheros hasta la revisión 3,0 (incluyendo los que no han cambiado), puede ser que se invoque:

$ cvs commit -r 3.0

Observe que el número que se especifica con **-r** debe ser más grande que cualquier número existente de la revisión. Es decir, si existe la revisión 3,0, no se puede **cvs commit -r 1.3** . Si se desea mantener varios desbloquear en paralelo, se necesita utilizar una ramificación.

3.3.4 ETIQUETAS REVISIONES SIMBÓLICAS

Los números de la revisión viven una vida propia. No necesitan tener cualquier cosa para hacer con todos los números el desbloquear de su producto de software. Dependiendo cómo se utiliza CVS de la revisión los números pudieran cambiar varias veces entre dos desbloquear. Como ejemplo, algunos de los ficheros de fuente arriba de RCS 5,6 tiene la siguiente revisión:

ci.c	5.21
co.c	5.9
ident.c	5.3
rcs.c	5.12
rcsbase.h	5.11
rcsdiff.c	5.10
rcsedit.c	5.11
rcsfcmp.c	5.9
rcsgen.c	5.10
rcslex.c	5.11
rcsmap.c	5.2
rcsutil.c	5.10

Se puede utilizar el comando de la etiqueta para dar un nombre simbólico a cierta revisión de un fichero. Se puede utilizar el parámetro **-v** para indicar el estatus para ver todas las etiquetas que un fichero tenga, y que representan los números de la revisión de ellos. Los nombres de la etiqueta deben comenzar con una letra mayúscula o minúscula y pueden contener las letras mayúsculas y minúsculas, dígitos, - y _ . La BASE y la PISTA de dos nombres de la etiqueta son reservadas para el uso del CVS. Se espera que los nombres futuros que son especiales a CVS serán nombrados principalmente, por ejemplo comenzando con . (un punto), más bien siendo nombrado análogo para BASE y HEAD, evitar conflictos con el nombre real de la etiqueta.

Se deberá elegir a cierta convención para nombrar etiquetas, basada en la información tal como el nombre del programa y del número de versión del desbloquear. Por ejemplo, uno puede tomar el nombre del programa, seguido inmediatamente por el número de versión con . (un punto) cambiado a - , de modo que CVS 1,9 fuera marcado con la etiqueta con el nombre **cvs1-9**. Si se elige a una convención constante, después no conjeturará constantemente si una etiqueta es **cvs-1-9** o **cvs1_9** o lo que sea.

El ejemplo siguiente muestra cómo se puede agregar una etiqueta a un fichero. Los comandos se deben publicar dentro de su directorio de funcionamiento. Es decir, se debe publicar el comando en el directorio donde **backend.c** reside.

```
$ cvs tag rel-0-4 backend.c
T backend.c
$ cvs status -v backend.c
===================================================================
File: backend.c          Status: Up-to-date

Version:          1.4    Tue Dec  1 14:39:01 1992
RCS Version:      1.4    /u/cvsroot/midirectorio/tc/backend.c,v
Sticky Tag:       (none)
Sticky Date:      (none)
Sticky Options:   (none)

Existing Tags:
rel-0-4           (revision: 1.4)
```

Hay raramente razón de marcar un fichero con etiqueta de aislamiento. Un uso más común es marcar todos los ficheros con etiqueta que constituyen un módulo con la misma etiqueta en las puntas estratégicas en el ciclo vital del desarrollo, por ejemplo cuando se hace un desbloquear.

```
$ cvs tag rel-1-0 .
cvs tag: Tagging .
T Makefile
T backend.c
T driver.c0
T frontend.c
T parser.c
```

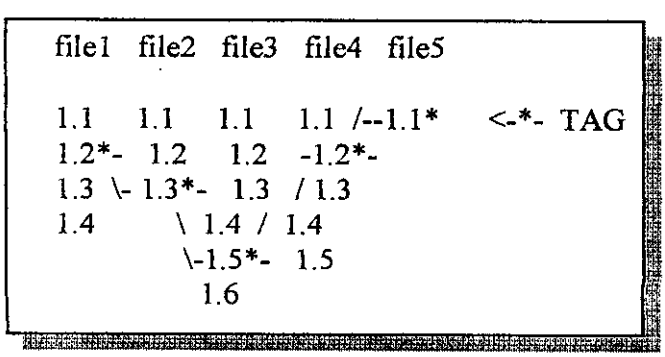
Cuando se da a CVS un directorio como argumento, aplica generalmente la operación a todos los ficheros en ese directorio, y (recurrentemente), a cualquiera de los subdirectorios que pueda contener.

El comando de la comprobación tiene un indicador, **-r**, que le deja controlar fuera de cierta revisión de un módulo. Este indicador hace fácil extraer las fuentes son hacia arriba del desbloquear 1,0 del módulo **tc** en cualquier momento después:

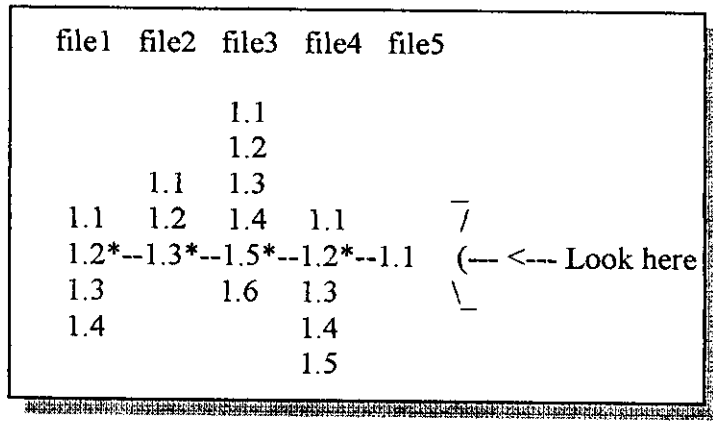
$ cvs checkout -r rel-1-0 tc

Esto es útil, por ejemplo, si alguien demanda que hay un fallo de funcionamiento al desbloquear, pero no se puede encontrar el fallo de funcionamiento en la copia del trabajo actual. Se puede también controlar fuera de un módulo mientras cualquier fecha es dada.

Al especificar **- r** a cualquiera de estos comandos, se necesitará cuidarse de etiquetas pegajosas. Cuando se marca con etiqueta más de un fichero con la misma etiqueta, se puede pensar de la etiqueta como " curva trazada a través de una matriz del nombre de fichero contra número de la revisión". Así tenemos: 5 ficheros con las revisiones siguientes:



En cierto momento en el pasado, * las versiones fueron marcadas con etiqueta. Se puede pensar en la etiqueta como manija asociada a la curva trazada con las revisiones marcadas con etiqueta. Cuando se continúa el manejo, se logra etiquetar todas las revisiones. Otra manera para mirar esto es que se observe a través de un instalador de revisiones que están "apartamentadas" a lo largo de las revisiones etiquetadas, así:



3.3.4.1 ESPECIFICAR QUÉ ETIQUETAR DESDE EL DIRECTORIO TRABAJADO

El ejemplo en la sección previa demostró una de las maneras más comunes para escoger que revisiones etiquetar. Esto es, corriendo el comando **cvs.tag** sin argumentos causa que el CVS seleccione las revisiones que son chequeadas en el directorio recién trabajado. Por ejemplo, si la copia del **backend.c** en el directorio trabajado fue chequeado desde la revisión 1.4, entonces el CVS etiquetará la revisión 1.4. Nótese que la etiqueta es aplicada inmediatamente a la revisión 1.4 en el depósito; etiquetar no es como modificar un archivo, u otras operaciones en las cuales primero se

modifica el directorio trabajado y entonces se corre el **cvs commit** para transferir esa modificación al depósito.

Un aspecto potencialmente sorprendente del hecho es que el **cvs tag** opera en el depósito que se está etiquetando las revisiones chequeadas, las cuales podrían diferir de los archivos localmente modificados en su directorio trabajado. Si se quiere evitar hacer esto por equivocación, especifique la opción **- c** al **cvs tag**. Si hay algunos archivos modificados localmente, el CVS abortará con un error antes de etiquetar cualquier archivo:

```
$ cvs tag -c rel-0-4
cvs tag : backend.c es modificado localmente
cvs [tag aborted]: corrige los primeros errores de arriba
```

3.3.4.2 ESPECIFICAR QUÉ ETIQUETAR POR FECHA O REVISIÓN

El comando **cvs rtag** etiqueta al depósito según una fecha reciente o tiempo (o puede ser usado para etiquetar la última revisión). El **rtag** trabaja directamente sobre los contendos del depósito (no se requiere chequeo previo ni buscar un directorio trabajado).

Las siguientes opciones especifican que fecha o revisión etiquetar.

- D date

Etiqueta las revisiones más recientes y no más anteriores de la fecha.

- f

Solamente es útil con los parámetros **- D date** o **– r tag**. Si la revisión emparejada, no se encuentra use la revisión más reciente (en lugar de ignorar el archivo).

- r tag

Solamente se etiquetan esos archivos que contienen la existencia tag tag.

El comando **cvs tag** además permite a alguien revisar o fechar archivos específicos, utilizando las mismas opciones **- r, - D** y **- f**. De cualquier modo, este rasgo no es probablemente lo que se quiere. La razón es que el **cvs tag** escoge que archivos etiquetar basado en los archivos que existen en el directorio trabajado, en lugar de los archivos que existieron según se dio el **tag/date**. Además, es generalmente mejor a distancia utilizar el **cvs rtag**. Las excepciones podrían ser causas como:

Cvs tag - r 1.4 backend.c

3.3.4.3 BORRANDO , MOVIENDO, Y RENOMBRANDO ETIQUETAS

Normalmente uno no modifica etiquetas. Ellas existen en orden para grabar la historia del depósito y así borrarlos o cambiar sus principales.

De cualquier modo, podría haber casos en los que alguien utilice una etiqueta temporalmente o coloque accidentalmente una etiqueta en el lugar equivocado. Por eso, alguien podría borrar, mover o renombrar una etiqueta.

Advertencia: los comandos en esta sección son peligrosos; ellos permanentemente desechan información histórica y podría dificultar o imposibilitar recobrar desde los errores. Si se es un administrador del CVS, podría considerarse restringir estos comandos con **taginfo**.

Para borrar una etiqueta, especifique la opción – **d** a cualquiera de ellos **cvs tag** o **cvs rtag**. Por ejemplo:

Cvs rtag - d rel-0-4 tc

Borrar la etiqueta rel-0-4 del módulo tc.

Cuando se dice mover una etiqueta, significa señalar el mismo nombre a diferentes revisiones. Por ejemplo, la etiqueta estable podría señalar de vez en cuando a la revisión 1.4 del **backend.c** y quizás queramos señalar la revisión 1.6. Para mover una etiqueta, especificar la opción - **F** al **cvs tag** o **cvs rtag**. Por ejemplo, la tarea recién mencionada podría ser completada así:

Cvs tag -r 1.6- F stable backend.c

Cuando se dice renombrar una etiqueta, significa señalar nombres diferentes a las mismas revisiones según la etiqueta vieja. Por ejemplo uno podría haber escrito mal el nombre de la etiqueta y quiere corregirlo. Para renombrar una etiqueta, primero cree una etiqueta nueva usando la opción - **r** al **cvs rtag**, y entonces elimine el nombre antiguo. Esto deja a la nueva etiqueta exactamente sobre el mismo archivo de la etiqueta vieja. Por ejemplo:

Cvs rtag - r old-name-0-4 rel-0-4 tc

Cvs rtag - d old-name-0-4 tc

3.3.4.4 ETIQUETANDO, AGREGANDO Y ELIMINANDO ARCHIVOS

El asunto de exactamente cómo etiquetar interactuando con agregar y quitar archivos es algo obscuro; pero la mayor parte del CVS mantendrán una pista de si los archivos existen o no sin demasiada precaución. Por omisión las etiquetas son aplicadas a solamente archivos que tiene una revisión correspondiente de qué está siendo etiquetado. Los archivos que no existen todavía, o que se han sido quitados, simplemente se omite la etiqueta, y el CVS sabe tratar la ausencia de una etiqueta como manifestando que el archivo no existió con esa etiqueta.

De cualquier modo, éste puede perder una cantidad pequeña de información. Por ejemplo, se supone que un archivo fue agregado y removido. Entonces, si la etiqueta falta para ese archivo, no hay ninguna manera de saber si la etiqueta se refiere al tiempo antes de que el archivo fuera agregado, o el tiempo después de que se quitó. Si se especifica la opción **- r** al **cvs rtag,** entonces el CVS etiquetará los archivos que se han quitado, y de este modo se evita este problema. Por ejemplo uno puede especificar **- r head** a la etiqueta de **head**.

Sobre el tema de agregar y remover archivos, el comando **cvs rtag** tiene una opción **- a** la cual significa limpiar la etiqueta de los archivos removidos que por otro lado no serían etiquetados. Por ejemplo se podría especificar esta opción en conjunción con

- **F** cuando se mueve una etiqueta. Si se movió una etiqueta sin - **a**, entonces la etiqueta en los archivos removidos podría aún referirse a la vieja revisión, en lugar de reflejar el hecho de que el archivo ha sido removido. No se debe pensar que esto es necesario si - **r** es especificado, como se anotó arriba.

3.3.5 ETIQUETAS PEGAJOSAS

Algunas veces la revisión de una copia trabajada tiene datos extras asociadas con ésta, por ejemplo ésto podría ser sobre una rama, o restringir versiones previas o un dato cierto por medio del **checkout - D** o **update – D**. A pesar de que este dato persiste – esto es, aplicar comandos subsecuentes a la copia trabajada; nos referimos a esto como pegajoso.

La mayoría de las veces, la pegajosidad es un aspecto oscuro del CVS que no se lo necesita pensar. Sin embrago, aun cuando no quiere usar este rasgo, se necesitaría saber algo acerca de etiquetas pegajosas (por ejemplo, cómo evitarlos!). Puede usar el comando **status** para ver si algunas etiquetas pegajosas o fechas están instaladas:

```
$ cvs status driver.c
=================================================================

File: driver.c          status: Up-to-date

Versión:                1.7.2.1     Sat Dec 5 19: 35: 03 1992

RCS Versión:            1.7.2.1  / u/cvsroot/ midirectorio/ tc/ driver.c, v

Sticky Tag:             rel-1-0-patches (branch: 1.7.2)

Sticky Date:            (none)

Sticky  Options:        (none)
```

Las etiquetas pegajosas permanecerán en sus archivos trabajados hasta que los borre con **cvs update - A**. La opción **- A** recupera la versión del archivo desde la cabeza del tronco, y olvida cualquiera de las etiquetas pegajosas, fechas, u opciones.

El uso más común de etiquetas pegajosas es identificar sobre que rama se está trabajando. De cualquier modo, las etiquetas pegajosas sin rama tienen uso también. Por ejemplo, se supone que se quiere evitar actualizar su directorio trabajado, para aislar la posibilidad que le desestabilicen los cambios que otras personas están haciendo. Se puede, por supuesto, sólo abstenerse de correr **cvs update**. Pero si quiere la posibilidad solamente de actualizar una porción de un árbol más grande, entonces las etiquetas pegajosas pueden ayudar. Si se quiere chequear una revisión segura esto llegará a ser pegajoso. Subsecuente los comando **cvs update** no recuperarán la última revisión hasta que se resetee la etiqueta con **cvs update - A**.

Igualmente, se utiliza la opción **- D** para **update** o **checkout** al colocar una fecha pegajosa, la cual, similarmente, causa que esa fecha sea utilizada para futuras recuperaciones.

Las personas a menudo quieren recuperar una versión vieja de un archivo sin instalar una etiqueta pegajosa. Esto puede ser hecho con la opción **- p** para **checkout** o **update**, la cual envía los contenidos del archivo a un rendimiento standard. Por ejemplo:

```
$ cvs update – p - r 1.1 file1>file1

Checking out file1
RCS: / tmp/ cvs-sanity/ cvsroot/ first-dir/ Attic/ file1, v
VERS: 1.1
$
```

De cualquier modo, ésta no es la manera más fácil, si se pide cómo deshacer una revisión previa (en este ejemplo, se puso file1, regresar a la manera que estaba como revisión 1.1). En este e caso es mejor utilizar la opción **-j** para **update**.

3.6 RAMIFICACIÓN Y COMBINACIÓN

CVS permite que se aisle cambios sobre una línea separada del desarrollo, conocida como ramificación. Cuando se cambia ficheros en una ramificación, esos cambios no aparecen en el tronco principal u otras ramificaciones. Se puede mover más adelante cambios a partir de una ramificación a otra ramificación (o al tronco principal) combinándose. La combinación implica la primera actualización corriente de los **cvs -j**, para combinar los cambios en el directorio de funcionamiento.

Se puede entonces confiar esa revisión, y copia con eficacia los cambios sobre otra ramificación.

3.6.1 PARA QUÉ ES BUENO USAR RAMIFICACIONES

Supongamos que el desbloquear 1,0 del **tc** se ha hecho. Después de un rato sus clientes comienzan a quejarse por un fallo de funcionamiento fatal. se controla fuera del desbloquear 1,0 y encuentra el fallo de funcionamiento (que resulta tener un arreglo trivial). Sin embargo, no se espera que la revisión actual de las fuentes esté en un estado de flux y sea estable para por lo menos un tiempo más. No hay manera de hacer un desbloquear del **bugfix** basado en las más nuevas fuentes.

Lo que se debe hacer en una situación como esta es crear una ramificación en los árboles de la revisión para todos los ficheros que hacen el desbloquear 1,0 del **tc**. Se puede entonces hacer modificaciones a la ramificación sin dañar el tronco principal.

Cuando se acaban las modificaciones se puede elegir si se las incorpora en el tronco principal, o las deja en la ramificación.

3.6.2 CREAR UNA RAMIFICACIÓN

Se puede crear una ramificación con la etiqueta **- b**; por ejemplo, si se asume que está en una copia de trabajo:

```
$ cvs tag -b rel-1-0-patches
```

Esto parte de una ramificación basada en las revisiones actuales en la copia de trabajo, asignando a esa ramificación el nombre `rel-1-0-patches' .
Es importante entender que las ramificaciones consiguen ser creadas en el depósito, no en la copia de trabajo. Crear una ramificación basada en revisiones actuales, como el ejemplo antedicho, no cambiará automáticamente la copia de trabajo para estar en la nueva ramificación.

Se puede también crear una ramificación sin referencia a cualquier copia de trabajo, usando el **rtag**:

```
$ cvs rtag -b -r rel-1-0 rel-1-0-patches tc
```

Esta ramificación -r rel-1-0 se debe arraigar en la revisión que corresponde a la etiqueta rel-1-0. No necesita ser la revisión más reciente es a menudo útil partir de una ramificación de una revisión vieja. Como con **tag**, el parámetro **-b** le dice a **rtag** que cree una ramificación (más bien apenas un nombre simbólico de la revisión). Observe que el item numérico de la revisión que corresponde con rel-1-0 será probablemente diferente de fichero a fichero.

Así pues, el efecto completo del comando es crear una nueva ramificación nombrada rel-1-0-patches en el módulo **tc**, arraigado en el árbol de la revisión en la punta marcada con etiqueta rel-1-0.

3.6.3 RAMIFICACIONES QUE TIENEN ACCESO

Se puede extraer una ramificación de dos maneras: controlándola fuera del depósito, o cambiando una copia de trabajo existente arriba de la ramificación.

Para controlar fuera de una ramificación del depósito, invoque **checkout** con el indicador **- r**, seguido por el nombre de la etiqueta de la ramificación:

```
$ cvs checkout -r rel-1-0-patches tc
```

O, si se tiene una copia de trabajo, se puede cambiarla a una ramificación dada con **update – r**:

```
$ cvs update -r rel-1-0-patches tc
```

o equivalente:

```
$ cd tc
$ cvs update -r rel-1-0-patches
```

No importa si la copia de trabajo estaba originalmente en el tronco principal o en otra ramificación el comando antedicho le cambiará a la ramificación nombrada. Y semejantemente a un comando **update** regular, **update** – **r** combina cualquier cambio que se haya realizado, notificándole de los conflictos donde ocurren.

Una vez que se tenga una copia de trabajo atada a una ramificación determinada, se sigue manteniendo allí hasta que se le ordene de otra manera. Esto significa que los cambios llegados de la copia de trabajo agregarán nuevas revisiones en esa ramificación, mientras que deja el tronco principal y las ramificaciones inafectadas.

Para descubrir qué ramificación es una copia de trabajo, se puede utilizar el comando **status**. En su salida, busque el campo nombrado **Sticky tag** que es una manera de decirle a la ramificación CVS's, si cualquiera, del trabajo actual clasifica:

```
$ cvs status -v driver.c backend.c
===================================================================
File: driver.c          Status: Up-to-date

   Version:             1.7    Sat Dec  5 18:25:54 1992
   RCS Version:         1.7    /u/cvsroot/midirectorio/tc/driver.c,v
   Sticky Tag:          rel-1-0-patches (branch: 1.7.2)
   Sticky Date:         (none)
   Sticky Options:      (none)

   Existing Tags:
   rel-1-0-patches      (branch: 1.7.2)
   rel-1-0              (revision: 1.7)

===================================================================
File: backend.c         Status: Up-to-date

   Version:             1.4    Tue Dec  1 14:39:01 1992
   RCS Version:         1.4    /u/cvsroot/midirectorio/tc/backend.c,v
   Sticky Tag:          rel-1-0-patches (branch: 1.4.2)
   Sticky Date:         (none)
   Sticky Options:      (none)

   Existing Tags:
   rel-1-0-patches      (branch: 1.4.2)
   rel-1-0              (revision: 1.4)
   rel-0-4              (revision: 1.4)
```

Versión: 1,4 DEC 1 14:39:01 de Tue 1992 versiones de RCS:

1,4 /u/cvsroot/midirectorio/tc/backend.c, etiqueta pegajosa de v:

rel-1-0-patches (ramificación: 1,4,2) Fecha Pegajosa: (ningunos)

opciones pegajosas: (ningunos) etiquetas existentes: rel-1-0-patches

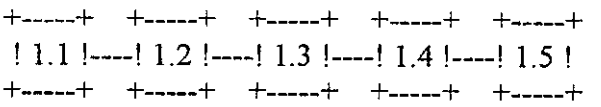
(ramificación: 1,4,2) rel-1-0 (revisión: 1,4) rel-0-4 (revisión: 1,4)

No hay que confundirse por el hecho de que los números de la ramificación para cada fichero sean diferentes (1.7.2 y 1.4.2 respectivamente). La etiqueta de la ramificación

es igual, **rel-1-0-patches**, y los ficheros están de hecho en la misma ramificación. Los números reflejan simplemente la punta en la historia de la revisión de cada fichero en la cual la ramificación fue hecha. En el ejemplo antedicho, uno puede deducir que **driver.c** ha pasado a través de más cambios que **backend.c** antes de que esta ramificación fuera creada.

3.6.4 RAMIFICACIONES Y REVISIONES

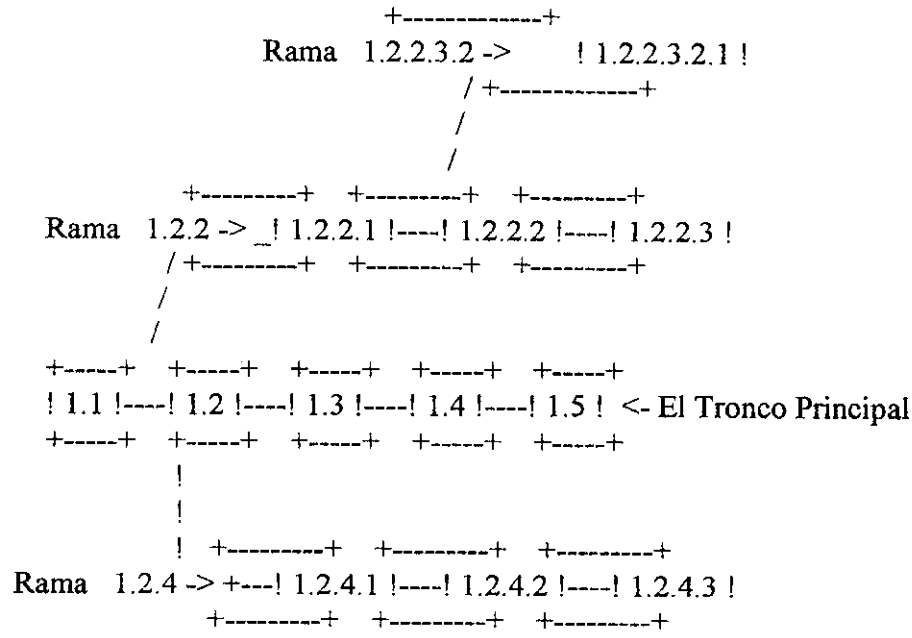
Ordinariamente, la historia de la revisión de un fichero es una serie linear de incrementos:



Sin embargo, CVS no se limita al desarrollo linear. El árbol de la revisión se puede partir en ramificaciones, donde está una línea para cada ramificación mantenida del desarrollo. Los cambios realizados en una ramificación se pueden mover fácilmente de nuevo al tronco principal.

Cada ramificación tiene un numero de ramificación, consistiendo en un número impar de números enteros decimales período - separados. El número de la ramificación es creado añadiendo un número entero al final del fichero al número de la revisión donde separa la correspondiente ramificación. Tener números de la ramificación permite que más de una ramificación sea separada de cierta revisión. Todas las revisiones en una ramificación tienen números formados de la revisión añadiendo un número al final del

fichero ordinal, al número de la ramificación. La figura siguiente ilustra la ramificación con un ejemplo.



Los detalles exactos de cómo se construye el número de la ramificación no es algo que normalmente se necesita conocer, pero aquí esta como esto trabaja: Cuando CVS crea un número de la ramificación escoge el primer número entero, comenzando con 2. Si nosotros creamos una ramificación de la revisión 6.4; este será numerado 6.4.2. Todos los números de la ramificación que terminan en un cero (tales como 6.4.0) se usan internamente por CVS.

Se puede usar el comando admin para reasignar un nombre simbólico para una ramificación the way RCS expects it to be. Si **R4patches** es asignado para la ramificación 1.4.2 (número de la rama mágica 1.4.0.2) en el archivo **numbers.c**, se puede hacer esto:

$ cvs admin -NR4patches:1.4.2 numbers.c

3.6.5 NÚMEROS DE LA RAMA MÁGICA

Esta sección describe una falla del CVS llamada ramas mágicas. Para la mayoría de propósitos, no se necesita preocuparse sobre las ramas mágicas; el CVS las maneja. De cualquier modo, éstas son visibles en ciertas circunstancias, entonces podría ser útil para tener alguna idea de cómo trabaja.

Externamente, los números de las ramas constan de un número impar separados por un punto de decimales enteros. Sin embargo esto no es completamente cierto. Por razones de eficiencia el CVS a veces inserta un 0 extra en segunda posición contando desde el último decimal (1.2.4 llega a ser 1.2.0.4, 8.9.10.11.12 llega a ser 8.9.10.11.0.12 y así sucesivamente).

El CVS hace un trabajo bastante bueno al esconder estos llamados ramas mágicas, pero en algunos lugares la ocultación es incompleta:

- El número de rama mágica aparece en el rendimiento del **cvs log**.
- No se puede especificar un nombre simbólico de rama al **cvs admin**.

Se puede usar el comando **admin** para reasignar un nombre simbólico a la rama de la manera de que el RCS espera para hacerlo. Si el **R4patches** es asignado a la rama 1.4.2 (el número de la rama mágica 1.4.0.2) en el archivo **numbers.c** se puede hacer esto:

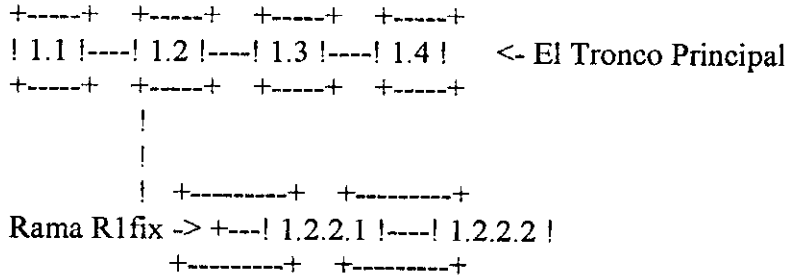
$ cvs admin - NR4patches: 1.4.2 numbers.c

Esto solamente trabaja si en la última revisión está ya cometido en la rama. Hay que ser muy cuidadoso si no, se asigna una etiqueta al número equivocado. (no hay forma de ver cómo se asignó la etiqueta ayer).

3.6.6 COMBINACIÓN DE UNA RAMA ENTERA

Se puede combinar cambios hechos en un rama dentro de su copia trabajada dando el parámetro **- j branch** para el comando **update**. Con la opción **- j branch** esta combina los cambios hechos entre el punto donde la rama se divide y la revisión más nueva en esa rama (dentro de la copia trabajada). La **-j** representa "junta."

Considere esta revisión obligada a refugiarse en un árbol:



La rama 1.2.2 ha sido dada al parámetro **R1fix** (nombre simbólico). El siguiente ejemplo asume que el módulo **mod** contiene solamente un archivo, **m.c**.

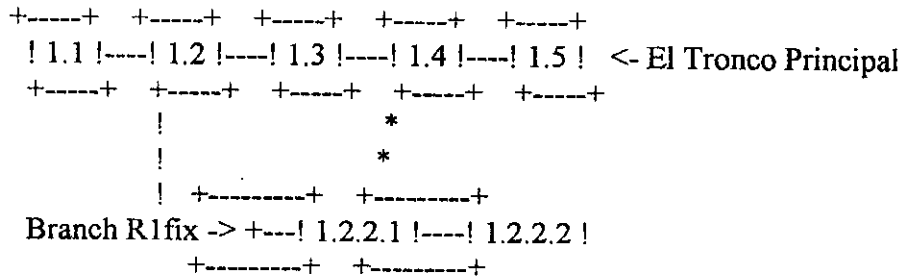
```
$ cvs  checkout mod          # Recupera la revisión última, 1.4
$ cvs update - j R1fix  m.c  # Combina todos los cambios hechos en la rama,
                             # por ejemplo los cambios entre revisión 1.2
                             # y 1.2.2.2, dentro de la copia trabajada
                             # del archivo.
          $ cvs commit - m "Incluye R1fix"  # Crea la revisión 1.5.
```

De la combinación de operaciones puede resultar un conflicto. Si esto pasa, se debería resolver antes de cometer la revisión nueva. El comando **checkout** también apoya al parámetro **-j branch**. El mismo efecto de arriba se puede alcanzar con ésto:

```
$ cvs  checkout - j R1fix  mod
$ [cvs] commit - m "Incluye  R1fix"
```

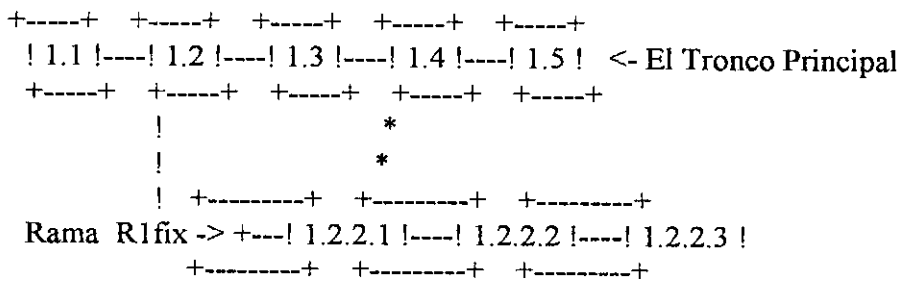
3.6.7 COMBINACIÓN DE ALGUNAS RAMAS A LA VEZ

Continuando con nuestro ejemplo, la revisión del árbol se mira como esto:



Donde la línea estrellada representa la combinación de la rama **R1fix** al tronco principal.

Ahora se supone que el desarrollo continúa con la rama R1fix :



Y entonces se quiere combinar estos nuevos cambios sobre el tronco principal. Si recién se utilizó otra vez el comando **cvs update - j R1fix m.c**, el CVS intentará combinar los cambios que ya se han combinado lo cual puede tener efectos indeseables.

Por otro lado se necesita especificar que solamente se quiere combinar los cambios en la rama que no se ha unido todavía en el tronco. Al hacer esto se especifica dos opciones – **j** , y el CVS combina los cambios desde la primera revisión a la segunda revisión.

Por ejemplo, en este caso la manera más simple podría ser:

```
Cvs update - j 1.2.2.2 - j R1fix  m.c        # Combina los cambios desde 1.2.2.2  a la
                                             # cabeza de la rama R1fix
```

El problema con esto es que se necesita especificar la revisión 1.2.2.2 manualmente. Un acercamiento mejor podría ser utilizar la fecha en que la última combinación fue hecha:

Cvs update - j R1fix: yesterday - j R1fix m.c

Mejor todavía, etiquetar la rama **R1fix** después de cada combinación dentro del tronco, y así utilizar esa etiqueta para combinaciones subsecuentes:

Cvs update - j merged_from_R1fix_to_trunk- j R1fix m.c

3.6.8 COMBINACIONES DE DIFERENCIAS ENTRE DOS REVISIONES CUALQUIERA

Con dos parámetros **- j revision**, el comando **update** y **checkout** pueden combinar las diferencias entre dos revisiones cualquiera dentro del archivo trabajado.

$ cvs update - j 1.5 - j 1.3 backend.c

Se deshacerán todos los cambios hechos entre la revisión 1.3 y 1.5. Nótese el orden de las revisiones. Si se trata de usar esta opción cuando se opera en archivos múltiples, recuerde que las revisiones numéricas estarán probablemente muy diferentes entre los varios archivos. Casi siempre se utiliza etiquetas simbólicas en lugar de revisión numeradas cuando se opera en archivos múltiples.

Especificando dos opciones – **j** se pude además deshacer archivos trasladados o adicionados. Por ejemplo, se supone que se tiene un archivo llamado file1 el cual existió como revisión 1.1 y entonces se lo removió (así se adicionó un archivo completamente 1.2). Ahora se supone que se quiere sumarlo otra vez, con el mismo contenido que tenía previamente. Aquí está como hacerlo:

```
$ cvs update -j 1.2 -j 1.1 file1
U file1
$ cvs commit -m test
Checking in file1;
/tmp/cvs-sanity/cvsroot/first-dir/file1,v <-- file1
new revision: 1.3; previous revision: 1.2
done
$
```

3.6.9 COMBINACIONES QUE PUEDE ADICIONAR O REMOVER ARCHIVOS

Si los cambios que se están combinando envuelve remover o agregar algunos archivos, **update - j** reflejará tales adiciones o eliminaciones. Por ejemplo:

```
Cvs update -A
touch a b c
cvs add a b c; cvs ci - m "added" a b c
```

```
cvs tag -b branchtag

cvs update -r branchtag

touch d; cvs add d

rm a; cvs rm a

cvs ci -m "added d, remove a"

cvs update – A

cvs update - jbranchtag
```

Después que estos comandos son ejecutados y un **cvs commit** es hecho, el archivo **a** será removido y el archivo **d** agregado en la rama principal.

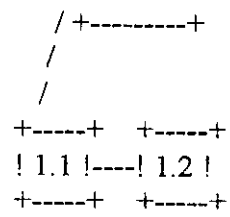
3.6.10 COMBINACIONES Y PALABRAS CLAVES

Si se combina archivos que contengan palabras claves, se tendrá normalmente numerosos conflictos durante la combinación, porque las palabras claves son expandidas diferentemente en las revisiones que se están combinando.

Además se querrá a menudo especificar el cambio – **kk** a la línea de comando de unión.

Para sustituir solo el nombre de la palabra clave, no valorar la expanción de esa palabra clave, esta opción asegura que las revisiones que se están combinando serán iguales a las otras. Y evita conflictos falsos. Por ejemplo se supone que se tiene un archivo así:

```
+---------+
_! 1.1.2.1 ! <- br1
```



Y el directorio trabajado está sobre un tronco corriente (revisión 1.2). Entonces se podría lograr el siguiente resultado desde una combinación:

```
$ cat file1
key $Revision: 1.2 $
. . .
$ cvs update -j br1
U file1
RCS file: /cvsroot/first-dir/file1,v
retrieving revision 1.1
retrieving revision 1.1.2.1
Merging differences between 1.1 and 1.1.2.1 into file1
rcsmerge: warning: conflicts during merge
$ cat file1
<<<<<<< file1
key $Revision: 1.2 $
=======
key $Revision: 1.1.2.1 $
>>>>
```

Lo que pasó fue que la combinación trató de combinar las diferencias entre 1.1 y 1.1.2.1 dentro del directorio trabajado. Así, desde que la palabra clave cambió desde revisión 1.1 a revisión 1.1.2.1, el CVS trató de combinar los cambios dentro del directorio trabajado, lo cual chocó con los datos que el directorio trabajado porque había contenido

Revisión: 1.2. Aquí está lo que pasa si se hubiera utlilizado **–kk**:

```
$ cat file1
key $Revision: 1.2 $
. . .
$ cvs update -kk -j br1
U file1
RCS file: /cvsroot/first-dir/file1,v
retrieving revision 1.1
retrieving revision 1.1.2.1
Merging differences between 1.1 and 1.1.2.1 into file1
$ cat file1
key $Revision$
. . .
```

Lo que esta pasando aquí es que la revisión 1.1 y 1.1.2.1 se expandieron como revisión sencilla, y además se combinaron los cambios entre ellos dentro del directorio trabajado que no necesita cambiar cualquier cosa. Además no hay conflicto.

Hay, sin embargo, una mejor forma con el uso del **-kk** en combinaciones. A saber, esto anularía cualquier expansión de la palabra clave en el modo CVS que normalmente se habría utilizado. En particular, esto es un problema si el modo ha sido **- kb** para un archivo binario. Además, si el depósito contiene archivos binarios, se necesitará tratar con los conflictos más bien que al utilizar **- kk**.

3.7 COMPORTAMIENTO RECURRENTE

Casi todos los submandatos del trabajo de CVS, recurrentemente cuando se especifica un directorio como argumento. Por ejemplo, considere esta estructura de directorio:

```
$HOME
   |
   +--tc
   | |
     +--CVS
     |    (internal CVS files)
     +--Makefile
     +--backend.c
     +--driver.c
     +--frontend.c
     +--parser.c
     +--man
     | |
     | +--CVS
     | | (internal CVS files)
     | +--tc.1
     |
     +--testing
        |
        +--CVS
        | (internal CVS files)
        +--testpgm.t
        +--test2.t
```

Si **tc** es el directorio de trabajo actual, el siguiente es:

- cvs update testing es equivalente a:

cvs update testing/testpgm.t testing/test2.t

- cvs update testing man: pone al día todos los ficheros en los subdirectorios.
- cvs update reune todos los ficheros en el módulo tc.

Si no existe ningún problema, pondrá al día todos los ficheros en el directorio de trabajo actual y todos sus subdirectorios. Es decir **.** (un punto) es un argumento de valor por defecto a ponerse al día. Esto es también verdad para la mayoría de los submandatos de CVS, no solamente el comando de actualización.

El comportamiento recurrente de los submandatos de CVS se puede dar con la opción **- l**. Inversamente, la opción - **R**, se puede utilizar para forzar la repetición si - **l** se especifica dentro del archivo **~/.cvsrc**.

$ cvs update -l # no pone al día ficheros en subdirectorios

3.8 AGREGANDO, QUITANDO Y RETITULANDO FICHEROS Y DIRECTORIOS

En el curso de un proyecto, uno agregará a menudo ficheros nuevos. Así mismo pasa con quitar o retitulación de directorios. El concepto general a tener presente en todos estos casos es este en lugar de realizar un cambio irreversible que se quisiera que CVS registrara el hecho de que ha ocurrido un cambio, apenas como con la modificación de

un fichero existente. Los mecanismos exactos para hacer esto en CVS varían dependiendo de la situación.

3.8.1 ADICIÓN DE FICHEROS A UN DIRECTORIO

Para agregar un fichero nuevo a un directorio, se debe seguir estos pasos de progresión:

- Tener una copia de trabajo del directorio.
- Crear el nuevo interior del fichero con su copia de trabajo del directorio.
- Utilice **cvs add filename** esto indica al CVS que se desea que se controle la versión del fichero. Si el fichero contiene datos binarios, especifique – **kb.**
- Utilice **cvs commit filename** para que llegue realmente el fichero al depósito. Otros reveladores no pueden ver el fichero hasta que se realiza este paso de progresión.

Se puede también utilizar el comando add para agregar un nuevo directorio. Diferente a la mayoría de los otros comandos, el comando de la adición no es recurrente. Incluso no se puede pulsar **cvs add foo/bar.** En lugar de esto, se tiene:

```
$ cd foo
$ cvs add bar
```

```
cvs add [-k kflag] [-m message] files ...
```

Los ficheros o los directorios especificados con add deben existir ya en el directorio actual. Para agregar una nueva jerarquía entera del directorio al depósito de la fuente

(por ejemplo, ficheros recibidos de un vendedor de tercera persona), utilice el comando de la importación en lugar de otro.

Los ficheros agregados no se colocan en el depósito de la fuente hasta que se utiliza **commit** para hacerla permanente del cambio. Hacer una adición en un fichero que fue quitado con el comando **remove** deshará este efecto, a menos que interviniera un comando **commit**, por ejemplo:.

La opción **- k** especifica la manera del valor por defecto que este fichero será controlado hacia fuera.

La opción - **m** especifica una descripción para el fichero. Esta descripción aparece en el registro de la historia. También será salvada en la historia de la versión dentro del depósito cuando el fichero está confiado. El comando del registro visualiza esta descripción. La descripción puede ser el usar **admin - t**. Si se omite el indicador - **m description**, una cadena vacía será utilizada. Por ejemplo, los comandos siguientes agregan el fichero **backend.c** al depósito:

```
$ cvs add backend.c
$ cvs commit -m "versión temprana. No compilable todavía" backend.c
```

Cuando se agrega un fichero se agrega solamente en la ramificación en la cual se está trabajando. Se puede combinar más adelante las adiciones a otra ramificación si se desea.

3.8.2 QUITAR FICHEROS

Aquí se describe lo que se puede hacer para quitar un fichero, pero sigue siendo capaz de extraer viejas revisiones:

- Cerciórese de que no haya hecho ninguna modificación sin compromiso al fichero, una forma para hacer eso, se puede también utilizar el comando **estatus** o **update.**
- Quite el fichero de su copia de trabajo del directorio. Se puede por ejemplo utilizar el parámetro **rm** .
- Usando **cvs remove filename** le indica al CVS que realmente desea suprimir el fichero.
- Usando **cvs commit filename** realiza realmente el retiro del fichero del depósito.

Es posible que un fichero exista solamente en algunas ramificaciones y no en otras, o re-agregue otro fichero con el mismo nombre más adelante. CVS creará o no correctamente el fichero, basado en las opciones - **r** y - **D** especificados a la comprobación o a la actualización . Comando:

cvs remove [options] files ...

Este comando no quita realmente el fichero del depósito hasta que se confía el retiro. Aquí está un ejemplo para quitar varios ficheros:

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

```
cvs remove: Removing .
cvs remove: scheduling a.c for removal
cvs remove: scheduling b.c for removal
cvs remove: use 'cvs commit' to remove these files permanently
$ cvs ci -m "Removed unneeded files"
cvs commit: Examining .
cvs commit: Committing .
```

Mientras que para conveniencia se puede quitar el fichero y cvs lo quita en un paso de progresión, especificando la opción **-f**. Por ejemplo, el ejemplo antedicho se podía también hacer como esto:

```
$ cd test
$ cvs remove -f *.c
cvs remove: scheduling a.c for removal
cvs remove: scheduling b.c for removal
cvs remove: use 'cvs commit' to remove these files permanently
$ cvs ci -m "Removed unneeded files"
cvs commit: Examining .
cvs commit: Committing .
```

Si se ejecuta **remove** para un fichero, y después cambia de idea antes de que se confíe, se puede deshacer el **remove** con un comando de adición.

```
$ ls
CVS  ja.h  oj.c
$ rm oj.c
$ cvs remove oj.c
cvs remove: scheduling oj.c for removal
cvs remove: use 'cvs commit' to remove this file permanently
$ cvs add oj.c
U oj.c
cvs add: oj.c, version 1.1.1.1, resurrected
```

Si se detecta el error antes de que se ejecute el comando **remove** se puede utilizar la actualización para resucitar el fichero:

```
$ rm oj.c
$ cvs update oj.c
cvs update: warning: oj.c was lost
U oj.c
```

Cuando se quita un fichero se quita solamente en la ramificación en la cual se está trabajando. Se puede combinar más adelante los retiros a otra ramificación si se desea.

3.8.3 QUITAR DIRECTORIOS

Quitar directorios es algo similar a quitar un fichero si desea que el directorio no exista en su directorio de trabajo actual, pero si también se desea ser capaz de extraer un viejo desbloqueador en el directorio existente.

La manera para quitar un directorio es quitar todos los ficheros de el. No se puede quitar el directorio lleno; no hay manera de hacer eso. En lugar de esto se especifica la opción **- P** a la actualización, comprobación o exportación del cvs, este quitará los directorios vacíos de directorios de funcionamiento. Probablemente la mejor manera de hacer esto es especificar siempre **- P**; si se desea mantener un directorio vacío entonces se puede utilizar por ejemplo **.Keepme**, este previene – **P** de quitarlo. Observe que - **P** es implicado por las opciones - **r** o - **D** de la comprobación y exportación. De esta manera CVS podrá crear correctamente el directorio o no dependiendo de si la versión determinada que se está controlando hacia fuera contiene cualquier fichero en ese directorio.

3.8.4 MOVIÉNDOSE Y RETITUTALANDO FICHEROS

Los ficheros móviles a un diverso directorio o a retitularlos no son difíciles, pero algunas de las maneras de las cuales ésta trabaja pueden ser no obvias. (la mudanza o la retitulación de un directorio es incluso más dura). Los ejemplos abajo asumen que el viejo fiechero está retitulado a nuevo .

3.8.4.1 LA MANERA NORMAL DE RETITULAR

La manera normal de mover un fichero es copiar el viejo al nuevo, y después publicar los comandos normales de CVS de quitar el fichero viejo del depósito, y agregar el nuevo a el.

$ mv old new

```
$ cvs remove old

$ cvs add new

$ cvs commit -m "Renamed old to new" old new
```

Esta es la manera más simple de mover un fichero, y preserva la historia de qué fue hecha. Observe que para tener acceso a la historia del fichero se debe especificar el viejo o nuevo nombre, dependiendo de qué porción de la historia se está teniendo acceso.

3.8.4.2 MUDANZA DE FICHEROS DE HISTORIA

Este método es más peligroso porque involucra mover archivos dentro del depósito.

```
$ cd $CVSROOT/dir

$ mv old,v new,v
```

Ventajas:

- Los registros de cambio se mantienen intactos.
- Las revisiones numeradas no son afectadas.

Desventajas:

- Las versiones antiguas eliminadas del módulo no pueden ser fácilmente buscadas desde el depósito. (El archivo se mostrará como nuevo aún en revisiones desde la fecha antes que fue renombrado).
- No hay registro de información de cuando los archivos fueron renombrados.

- Las cosas peores que podrían pasar si alguien accede a la historia del archivo mientras se lo está moviendo. Asegúrese que nadie más corra cualquiera de los comandos del cvs, mientras lo está moviendo.

3.8.4.3 COPIANDO LA HISTORIA DEL ARCHIVO

Esta forma además involucra modificaciones directas al depósito. Está a salvo pero no sin desventajas.

```
# Copy the RCS file inside the repository
$ cd $CVSROOT/dir
$ cp old,v new,v
# Remove the old file
$ cd ~/dir
$ rm old
$ cvs remove old
$ cvs commit old
# Remove all tags from new
$ cvs update new
$ cvs log new
$ cvs tag -d tag1 new
$ cvs tag -d tag2 new
...
```

Al remover las etiquetas se podrá chequear revisiones viejas del módulo.

Ventajas:

- Revisar si versiones viejas trabajan satisfactoriamente, en tanto que se utiliza **–r tag** y no **–D date** para recuperar las revisiones.
- El registro de los cambios se mantienen intactos.
- Las revisiones numeradas no son afectadas.

Desventajas:

- No se puede ver fácilmente la historia de los archivos a través del renombre.

3.8.5 MOVIENDO Y RENOMBRANDO DIRECTORIOS

La manera normal para renombrar o mover un directorio es renombrar o mover cada cosa como se describió anteriormente. Entonces chequee con la opción **-P**.

Si realmente se quiere cortar el depósito para renombrar o borrar un directorio en el depósito, se puede hacer así:

1. Informar a cada uno que tiene una copia del módulo que el directorio será renombrado. Ellos deberán comitir todos sus cambios, y remover sus copias trabajadas del módulo, antes que se tome el paso de abajo.

2. Renombrar los directorios dentro del depósito.

```
$ cd $CVSROOT/parent-dir
$ mv old-dir new-dir
```

3. Arreglar los archivos administrativos del CVS, si es necesario (por ejemplo si renombró un módulo entero).

4. Comunicar a todo el mundo que pueden chequear el módulo y continuar trabajando.

Si alguien tiene una copia trabajada del módulo, los comandos del CVS cesarán de trabajar para él, hasta que él remueva los directorios que desaparecieron dentro del depósito.

Es casi siempre mejor remover los archivos en lugar de mover el directorio. Si se mueve el directorio se está improbablemente capacitado para recuperar viejos escapes correctamente, ya que ellos probablemente dependen del nombre de los directorios.

3.9 MIRAR LA HISTORIA

Una vez que se haya utilizado CVS para salvar una historia del control de la versión qué ficheros han cambiado cuando, cómo, y por quién, hay una variedad de mecanismos para mirar con la historia.

3.9.1 MENSAJES DE REGISTRO

Siempre que se confie un fichero se especifica un mensaje del registro. Para mirar a través de los mensajes del registro que se han especificado para cada revisión que ha estado confiada, utilice el comando del registro del cvs.

3.9.2 LA BASE DE DATOS DE LA HISTORIA

Se puede utilizar el fichero de historia para registrar varias acciones de CVS. Para extraer la información del fichero de historia, se utiliza el comando de la historia del cvs.

3.9.3 REGISTRACIÓN DEFINIDA POR EL USUARIO

Se puede modificar CVS para requisitos particulares a varias clases del registro de acciones, de cualquier manera se puede elegir el mejor. Estos mecanismos funcionan ejecutando una escritura en varias horas. La escritura pudo añadir un mensaje al final del fichero a un fichero que enumeraba a la información y al programador que lo creó, o envía el correo a un grupo de reveladores, o, quizás, fija un mensaje a un **newsgroup** determinado. Para registrar, se utiliza el fichero **loginfo**.

Para registrar, las comprobaciones, exportaciones, y las etiquetas, respectivamente, se puede también utilizar la opciones en los módulos **- i , -o , -e** , y **-t**. Para una manera más flexible de dar las notificaciones a los varios utilizadores, que requiere menos la manera de mantener las escrituras actualizadas, utiliza el comando **cvs watch add**. Este comando es útil incluso si no está utilizando el cvs watch on. El fichero **taginfo** define programas para ejecutarse cuando alguien ejecuta un comando de etiqueta o del **rtag**.

El fichero **taginfo** tiene la forma de estándar para los ficheros administrativos, donde está una expresión cada línea regular seguida por un comando de ejecutarse. Los argumentos pasados al comando son, in order, tagname, in operation (add para tag, move para la tag - F , y del para tag - d), depósito, y cualquier restante es igual de la revisión del nombre de fichero . Una salida diferente a cero del programa del filtro hará la etiqueta ser abortada.

Aquí está un ejemplo utilizando **taginfo** para registrar comandos de la etiqueta y del **rtag**. En el fichero del **taginfo** puesto:

```
ALL /usr/local/cvsroot/CVSROOT/loggit
Where `/usr/local/cvsroot/CVSROOT/loggit' contains the following script:
#!/bin/sh
echo "$@" >>/home/kingdon/cvsroot/CVSROOT/taglog
```

3.10 MANIPULACIÓN DE FICHEROS BINARIOS

El uso más común para CVS es salvar ficheros de texto. Con los ficheros de texto, CVS puede combinar revisiones, visualiza las diferencias entre las revisiones en una manera humano visible, y otras tales operaciones. Sin embargo, si se está dispuesto a dar para arriba algunas de estas capacidades, CVS puede salvar ficheros binarios. Por ejemplo, uno pudo salvar un **Web site** en CVS incluyendo ficheros del texto e imágenes binarias.

3.10.1 EDICIONES CON LOS FICHEROS BINARIOS

Mientras que la necesidad de manejar ficheros binarios puede parecerse obvia si los ficheros con los cuales se trabaja acostumbradamente son binarios, ponerlos en control de la versión presentan algunas ediciones adicionales.

Una función básica del control de la versión es mostrar las diferencias entre dos revisiones. Por ejemplo, si alguna otra persona llevó una nueva versión de un fichero, se puede desear observar en lo que él cambió y se determina si sus cambios son buenos.

Para los ficheros de texto, CVS proporciona a estas funciones vía el comando del **diff** de los cvs. Para los ficheros binarios, puede ser posible extraer las dos revisiones y después compararlas con una herramienta externa a CVS (por ejemplo, el software lógica del tratamiento de textos tiene a menudo tal característica). Si no hay tal herramienta, se debe optar por otros mecanismos, tales como impulsión de la gente a escribir buenos mensajes del registro, y esperar que los cambios que realizaron realmente eran los cambios que se propusieron realizar.

Otra capacidad de un sistema de control de la versión es la capacidad de combinar dos revisiones. Para CVS esto sucede en dos contextos. El primero es cuando los utilizadores realizan cambios en directorios de funcionamiento separados. El segundo es cuando uno se combina explícitamente con el comando **update -j**.

En la caja de ficheros del texto, CVS puede combinar los cambios realizados independientemente, y señala el conflicto o si están en conflicto los cambios. Con los ficheros binarios, lo mejor que CVS puede hacer es realizar dos diversas copias del fichero, y dejarlo al utilizador para resolver el conflicto. El utilizador puede elegir una copia o la otra, o puede ejecutar una herramienta externa de la fusión que sepa sobre ese formato determinado del fichero, si existe uno. Observe que tiene el utilizador que la fusión confía sobre todo en el utilizador para omitir no accidentalmente algunos cambios, y es así potencialmente error propenso.

Si este proceso se piensa para ser indeseable, la mejor opción puede ser evitar la combinación. Para evitar las fusiones que resultan de directorios de funcionamiento separados. Para evitar las fusiones resultando de ramificaciones, restrinja el uso de ramificaciones.

3.10.2 CÓMO SALVAR FICHEROS BINARIOS

Hay dos puntos usando CVS para salvar ficheros binarios. Primero es que CVS por valor por defecto convierte la línea de conclusión en forma canónica en que ellos se almacenan en el depósito, y la forma apropiada al sistema operativo en uso sobre el cliente. El segundo es que un fichero binario pudo contener los datos que parece una palabra clave, así que la extensión de la palabra clave se debe dar por concluido.

La opción - **kb** disponible con comandos de algunos CVS aseguran que ni la conversión de la conclusión de la línea ni la extensión de la palabra clave serán

finalizadas. Aquí está un ejemplo de cómo se puede crear un fichero nuevo usando el comando **-kb flag**:

```
$ echo '$Id$' > kotest
$ cvs add -kb -m"Un fichero de prueba" kotest
$ cvs ci -m"Primer Chequeo, Contiene el Password" kotest
```

Si un fichero consigue accidentalmente agregar - **kb**, se puede utilizar el comando **admin** para recuperarse. Por ejemplo:

```
$ echo '$Id$' > kotest
$ cvs add -m"Un fichero de prueba" kotest
$ cvs ci -m" Primer Chequeo, Contiene el Password " kotest
$ cvs admin -kb kotest
$ cvs update -A kotest
# For non-unix systems:
# Copy in a good copy of the file from outside CVS
$ cvs commit -m "Hecho binario" kotest
```

Cuando se chequea en el fichero **kotest** no se preserva como fichero binario, porque no llegó como fichero binario. El comando **cvs admin – kb** el método de la substitución de la palabra clave del valor por defecto para este fichero, sino él no altera la copia de trabajo del fichero que se tiene. Si se necesita hacer frente a las conclusiones de la línea (es decir, se está utilizando CVS en un sistema de que no sea

unix), entonces se necesita llevar una nueva copia del fichero. En unix, la actualización del comando **cvs - a** es suficiente.

Sin embargo, al usar **cvs admin - k** para cambiar la extensión de la palabra clave, se debe saber que el modo de la extensión de la palabra clave no es versión controlada. Esto significa, por ejemplo, que si se tiene un fichero del texto en viejos desbloquear, y un fichero binario con el mismo nombre en nuevos desbloquear, CVS no proporciona de ninguna manera el control fuera del fichero en texto o modo binario dependiendo de qué versión se está controlando hacia fuera. No hay buen workaround para este problema.

Se puede también fijar un valor por defecto si los cvs agregan y el convite de la importación de los cvs un fichero como binario basado en su nombre; por ejemplo usted podría decir que los ficheros que nombra el extremo adentro `.exe' son binarios. Vea la sección el fichero de los cvswrappers . No hay actualmente manera de hacer que CVS detecte si un fichero es binario basado en su contenido. La dificultad principal con diseñar tal característica es que no está clara cómo distinguir entre los ficheros binarios y no-binarios, y las reglas a aplicarse variarían considerablemente con el sistema operativo.

3.11 REVELADORES MÚLTIPLES

Cuando más de una persona trabaja en cosas de un proyecto del software lógica a menudo es complicado. Por lo general, dos personas intentan corregir el mismo fichero

simultáneamente. Una solución, conocida como bloqueo de fichero o comprobaciones reservadas, es permitir que solamente una persona corrija cada fichero al mismo tiempo.

Esta es la única solución con algunos sistemas de control de la versión, incluyendo RCS y SCCS. Actualmente la manera generalmente de conseguir comprobaciones reservadas con CVS es con el comando **cvs admin - l**. Esto no se integra tan agradablemente en CVS como las características del reloj, descritas a continuación, pero parece que la mayoría de gente con esta necesidad encuentra adecuada las comprobaciones reservadas. También puede ser posible utilizar las características de los relojes, junto con los procedimientos convenientes (no hechos cumplir por el software lógica), para evitar que dos personas corrijan al mismo tiempo.

El modelo del valor por defecto con CVS se conoce como comprobaciones sin reservas. En este modelo, los reveladores pueden corregir su propia copia de trabajo de un fichero simultáneamente. La primera persona que confía sus cambios no tiene ninguna manera automática de saber que otra ha comenzado a corregirla. Otros conseguirán un mensaje de error cuando intentan confiar el fichero. Deben entonces utilizar comandos de CVS de traer su copia de trabajo actualizada con la revisión del depósito. Este proceso es casi automático.

CVS también utiliza los mecanismos que facilitan varias clases de comunicación, sin realmente hacer cumplir reglas como lo hacen las comprobaciones reservadas.

3.11.1 ESTADO DEL FICHERO

De acuerdo con qué operaciones se ha realizado el control fuera del fichero, y qué operaciones han realizado otros a ese fichero en el depósito, uno puede clasificar un fichero en un número de estados. Los estados, según lo señalado por el comando del estatus, son:

Up-to-date (Actualizado)

El fichero es idéntico con la última revisión del depósito para la ramificación en uso.

Locally Modified (Localmente Modificado)

Se ha corregido el fichero, y todavía no se ha confiado sus cambios.

Locally Added (Localmente Agregado)

Se ha agregado el fichero con add, y todavía no confiado sus cambios.

Locally Removed (Localmente Quitado)

Se ha quitado el fichero con remove, y todavía no confiado sus cambios.

Needs Checkout (Comprobación De las Necesidades)

Algún otro ha confiado una nueva revisión más al depósito. El nombre es levemente engañoso; se utilizará ordinariamente la actualización más bien que la comprobación para conseguir nueva revisión.

Needs Patch (Corrección De las Necesidades)

Como la comprobación de las necesidades, pero el servidor de CVS enviará una corrección mejor que el fichero entero. Enviar una corrección o enviar un fichero entero logra la misma cosa.

Needs Merge (Fusión De las Necesidades)

Alguna otra persona ha confiado una nueva revisión más al depósito, y también ha hecho modificaciones al fichero.

File had conflicts on merge (El fichero tenía conflictos en la fusión)

Esto es igual a localmente modificado, excepto que un comando anterior de la actualización dio un conflicto.

Unknown (Desconocido)

CVS no sabe cualquier cosa sobre este fichero. Por ejemplo, se ha creado un fichero nuevo y no se ha ejecutado add.

Para ayudar a clarificar el estatus del fichero, el estatus también señala la revisión de trabajo que es la revisión de la cual el fichero en el directorio de funcionamiento deriva, y la revisión del depósito que es la última revisión del depósito para la ramificación en uso. Se puede pensar en los comandos del estatus y de la actualización como algo complementario. Se utiliza la actualización para traer sus ficheros actualizados, y se puede utilizar estatus para darle una cierta idea de lo que una actualización haría (por

supuesto, el estado del depósito pudo cambiar antes de que se ejecute realmente la actualización). En hecho, si se desea un comando de visualización al estatus del fichero en un formato más abreviado que es visualizado por el comando estatus, se puede invocar :

$ cvs -n -q update

Los medios de la opción **-n** no hacen realmente la actualización, pero simplemente visualiza el estatus; la opción **-q** evita imprimir el nombre de cada directorio.

3.11.2 TRAYENDO UN FICHERO ACTUALIZADO

Cuando se desea poner al día o combinar un fichero, utilice el comando de la actualización. Para los ficheros que no son actualizados esto es áspero equivalente a un comando de la comprobación: la más nueva revisión del fichero se extrae del depósito y se pone en su copia de trabajo del módulo.

Sus modificaciones a un fichero nunca se pierden cuando se utiliza la actualización. Si no existe ninguna otra nueva revisión, se puede ejecutar la actualización no tiene ningún efecto. Si se ha corregido el fichero, y una nueva revisión más está disponible, CVS combinará todos los cambios en su copia de trabajo.

Por ejemplo, imagínese que se controló fuera de la revisión 1,4 y comenzó a corregirla. En el medio tiempo algún otro confió la revisión 1,5 y poco después esa revisión 1,6. Si ahora ejecuta la actualización en el fichero, CVS incorporará todos los cambios entre la revisión 1,4 y 1,6 en su fichero.

3.11.3 ESTÁ EN CONFLICTO EL EJEMPLO

Supongamos que la revisión 1,4 de **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);
}
```

Revisión 1,6 de **driver.c** contiene esto:

```
#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);
}
```

La copia de trabajo de **driver.c**, basado en la revisión 1,4, contiene esto antes de que se ejecute cvs update:

```
#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);
}
```

Al ejecutar **cvs update**:

```
$ cvs update driver.c
RCS file: /usr/local/cvsroot/midirectorio/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
```

CVS le dice que si existe algunos conflictos. Su fichero de trabajo original se salva sin modificar dentro **.#driver.c.1.4**. La nueva versión de **driver.c**, contiene esto:

```
#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
}
```

***** non-overlapping*****

Se resuelve el conflicto corrigiendo el fichero, quitando las etiquetas de plástico y la línea errónea. Supongamos terminar lo de arriba con este fichero:

```
#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);
}
```

Se puede ir a continuación y guardar esto como revisión 1.7

```
$ cvs commit - m "Initialize scaner. Use valores de la salida simbólicos."
driver.c
Registrar driver.c;

/ usr/ local/ cvsroot/ midirectorio/ tc/ driver.c, v
<
-- driver.c
revisión nueva: 1.7; revisión previa: 1.6
```

3.11.4 INFORMANDO A OTROS ACERCA DE GUARDAR

Es a menudo útil informar a otros cuando se guarda una revisión nueva de un archivo.

La opción - **i** del archivo **modules**, o el archivo **loginfo**, se puede usar para automatizar este proceso. Se puede usar estos rasgos del CVS para, por ejemplo, instruir al CVS para enviar un mensaje a todos los desarrolladores, o fijar un mensaje en un grupo local de noticias.

3.11.5 ALGUNOS DESARROLLADORES INTENTAN CORRER CVS SIMULTÁNEAMENTE

Si varios desarrolladores tratan de correr CVS al mismo tiempo, uno podría tener el siguiente mensaje:

[11: 43: 23] esperando por un seguro de rama en / usr/ local/ cvsroot/ foo

CVS probará de nuevo cada 30 segundos, y si continúa con el funcionamiento o imprime el mensaje de nuevo, si todavía necesita esperar. Si una cerradura parece pegarse alrededor de una cantidad de tiempo, se encuentra la persona detenida por la cerradura y pregunta acerca de los comandos del cvs que ellos están corriendo. Si ellos no están corriendo un comando del cvs, observe en el depósito el directorio mencionado en el mensaje y remueva los archivos propios de ellos cuyos nombres empiezan con **# cvs.rfl , # cvs.wfl, o # cvs.lock.**

Note que estas cerraduras son para proteger estructuras de datos internos del CVS y no tienen relación a la palabra cerradura en el sentido usado por RCS lo cual se refiere para reservar chequeos.

Cualquier número de personas pueden estar leyendo en un depósito dado a un tiempo; sólo cuando alguien escribe se hace las cerraduras para prevenir a otras personas desde la lectura o escritura.

Se puede esperar por la propiedad siguiente

Si alguien guarda algunos cambios en un comando del cvs, entonces una actualización por alguien más se obtendría todos los cambios, o ninguno de ellos.

Pero el CVS no tiene esta propiedad. Por ejemplo se dan los archivos:

a/ one.c

a/ two.c

b/ three.c

b/ four.c

si alguien corre:

cvs ci a/ two.c b/ three.c

y alguien además corre **cvs update** al mismo tiempo, la persona que corre **update** podría obtener solamente los cambios de **b/ three.c** y no los cambios de **a/ two.c**.

3.11.6 MECANISMOS PARA RASTREAR QUIEN EDITA ARCHIVOS

Para muchos grupos, el uso del CVS en su modo estándar es absolutamente satisfactorio. Usuarios podrían algunas veces ir a chequear una modificación solamente para encontrar que otra modificación ha intervenido, pero ellos tratan con esto y proceden con su chequeo. Otros grupos prefieren poder conocer quién esta editando y

que archivos, así que si dos personas tratan de editar el mismo archivo ellos pueden escoger hablar sobre quién esta haciendo qué cuando en lugar de ser sorprendido más tarde al chequear el tiempo. Los rasgos en esta sección permiten tal coordinación, mientras retiene la habilidad de dos desarrolladores para editar el mismo archivo al mismo tiempo.

Para máximo beneficio de los desarrolladores se debería utilizar **cvs edit** (no chmod) para hacer archivos leído-escrito para editarlos, y el **cvs release** (no rm) para descartar un directorio trabajado el cual no es tan grande en uso, pero CVS no puede fortalecer esta conducta.

3.11.6.1 DECIR A CVS QUE OBSERVE CIERTOS ARCHIVOS

Para capacitar, observar rasgos, primero especifique que ciertos archivos están para ser observados.

cvs watch on [- lR] files...

Especificar que diseñadores se deben correr del **cvs edit** antes de editar los archivos. El CVS creará copias trabajadas de archivos leídos solamente, recuerde los desarrolladores que corren el comando **cvs edit** antes de trabajar con ellos. Si los archivos incluyen el nombre de un directorio, el CVS acuerda mirar todos los archivos añadidos al correspondiente directorio del depósito, coloca una omisión para archivos agregados en el futuro; esto permite al usuario colocar políticas de notificación o una base pre-directoria.

Los contenidos del directorio son procesados recursivamente, a menos que la opción –**l** sea dada. La opción **-R** puede ser utilizada para forzar la recursión si la opción – **l** es colocada en ~/ **.cvsrc**. Si se omiten archivos, esto omite al directorio presente.

cvs watch off [- lR] files. . .

No se proveen notificaciones sobre trabajos en archivos, el CVS creará copias trabajadas de los archivos leidos-escritos. Los archivos y las opciones son procesadas por **cvs watch on**.

3.11.6.2 DICIENDO AL CVS QUE NOTIFICAR

Se puede decir al CVS que se quiere recibir notificaciones acerca de varias acciones tomadas en un archivo. Puede hacer esto sin el uso del **cvs watch on** para el archivo, pero generalmente se querrá utilizar **cvs watch on**, así que los desarrolladores utilizan el comando **cvs edit**.

cvs watch add [- a action][- lR] files. . .

Agregue al usuario presente la lista de personas que recibe la notificación del trabajo hecho en archivos. La opción **- a** especifica qué clase de eventos el CVS debería notificar al usuario. **Action** es uno de los siguientes:

edit

Otro usuario tiene aplicado el comando cvs para un archivo.

unedit

Otro usuario tiene aplicado el orden **cvs unedit** o el comando **cvs release** a un archivo, o tiene borrado el archivo y permite que el **cvs update** lo recree.

commit

Otro usuario tiene cambios guardados a un archivo.

all

Todo de lo descrito anteriormente

none

Nada de lo descrito anteriormente. (Esto es útil con cvs edit.)

La opción **-a** podría aparecer más de una vez, o nada. Si es omitida, la acción omitirá **all**. Los archivos y las opciones son procesadas por los comandos **cvs watch**.

cvs watch remove [- a action][- lR] files. . .

Para remover una notificación solicitada establecida utilizando **cvs watch add**; los argumentos son los mismo. Si la opción **- a** es presentada, solamente los **watches** para acciones especificadas son removidas.

Cuando las condiciones existen para notificación, CVS llama al archivo administrativo **notify**. El **notify** edita como se hace con los otros archivos administrativos. Este archivo permite las convenciones usuales para archivos administrativos, donde cada

línea es una expresión regular seguida por un comando **execute**. El comando debería contener una ocurrencia simple de **% s** la cual será reemplazada por el usuario a notificar; el resto de la información observada la notificación será proveída al comando estándar **input**. El estándar pone al archivo **notify** que es una línea simple:

All mail % s - s\ "CVS notification\

Esto causa que los usuarios sean notificados por correo electrónico.

Nótese que si se hace esto en la forma correcta, los usuarios reciben notificaciones en la máquina del servidor. Uno podría por supuesto escribir un manuscrito **notify** el cual directamente notifica a cualquier lugar, así lo hace fácil, el CVS permite asociar una dirección de notificación para cada usuario. Así se crea un archivo **users** en **CVSROOT** con una línea para cada usuario en el formato **user : value**. Entonces en lugar de pasar el nombre del usuario para ser notificado a **notify**, el CVS pasará al **value** (normalmente una dirección del email en alguna otra máquina).

El CVS no nos notifica nuestros propios cambios. Corrientemente este chequeo se hace basado en si la persona realiza la acción con un juego de blancos de notificación, entonces la persona logra la notificación. De hecho, los rasgos observados solamente rastrean una edición por cada usuario. Esto probablemente sería más útil si se observaran sendas para cada directorio trabajado separadamente, así este comportamiento podría valer cambiarlo.

3.11.6.3 CÓMO EDITAR UN ARCHIVO QUE ESTÁ SIENDO OBSERVADO

Si un archivo que está siendo observado, es chequeado con lectura solamente, no se puede editarlo simplemente. Esto se hace leyendo y escribiendo, e informar a otros que se piensa editar, se utiliza el comando **cvs edit**. Algunos sistemas lo llaman a esto un chequeo, pero el CVS utiliza ese termino para obtener una copia de las fuentes, una operación que estos sistemas lo llaman un logro o un traer.

cvs edit [options] files . . .

Prepararse para editar el archivo trabajado **files**. El CVS hace que los archivos leídos-escritos, y notifica al usuario que ha solicitado **edit** para cualquiera de los archivos.
El comando **cvs edit** acepta las mismas opciones como el comando **cvs watch add**, y establece una mirada temporaria al usuario de los archivos; El CVS remueve la observación cuando los archivos son ineditables **ed** o guardados **ted**. Si el usuario no quiere recibir notificaciones, debe especificar **- a none**.

Los archivos y las opciones son procesadas como para los comandos **cvs watch**.

Precaución: Si la opción **Preserve Permissions** está habilitada en el depósito, el CVS no cambiará los permisos de cualquiera de los archivos. La razón para este cambio es asegurarse que el **cvs edit** no interfiera con la habilidad para guardar permisos de archivo en el depósito del CVS.

Normalmente cuando se hace con un set de cambios, se usa el comando **cvs commit**, el cual chequea los cambios y retorna los archivos observados a su estado usual leído solamente. Pero si en cambio se decide abandonar los cambios, o no hacer cualquier cambio, se puede usar el comando **cvs unedit**.

cvs unedit [- lR] files. . .

Abandonar el trabajo de los archivos trabajados **files**, y revertirlos a la versión del depósito en el cual ellos están basados. CVS hace estos archivos leídos solamente para los cuales el usuario ha solicitado notificación utilizando **cvs watch on**. CVS notifica al usuario quien ha solicitado notificación ineditable para cualquiera de los archivos. Los archivos y opciones son procesados como para los comandos **cvs watch**.

Si las miras no están en uso, el comando **unedit** probablemente no trabaje, y la forma para revertir a la versión del depósito es quitar el archivo y entonces usar **cvs update** para lograr una nueva copia. El significado no es precisamente el mismo; quitar y actualizar podría además traer algunos cambios que han sido hechos en el depósito desde la última vez que se actualizó.

Cuando se usa el CVS cliente/ servidor, se puede usar los comandos **cvs edit** y **cvs unedit** aun cuando CVS es incapaz de comunicarse exitosamente con el servidor; las notificaciones serán enviadas al próximo comando CVS.

3.11.6.4 INFORMACIÓN ACERCA DE QUIÉN ESTÁ OBSERVANDO Y EDITANDO

cvs watchers [- lR] files. . .

Lista de los usuarios corrientes que observan cambios de archivos. El informe incluye los archivos que están siendo observados y la dirección de correo de cada observador. Los archivos y las opciones son procesadas como para los comandos **cvs watch**.

cvs editor [- lR] files. . .

Lista de usuarios corrientes que trabajan en archivos. El informe incluye la dirección de correo de cada usuario, el tiempo cuando el usuario empezó a trabajar con el archivo, el huésped y trayectoria del directorio trabajado que contiene el archivo.

Los archivos y las opciones son procesadas como para los comandos **cvs watch**.

3.11.6.5 USANDO OBSERVACIONES CON VERSIONES VIEJAS DEL CVS

Si se utiliza el rasgo observado en un depósito, esto crea directorios de CVS en el depósito y almacena la información acerca de las observaciones en ese directorio. Si se intenta usar CVS 1.6 o más antiguas con el depósito, se logra un mensaje de error tal como el siguiente (todo en una línea):

Cvs update: cannot open cvs/ Entries for reading:

No such file or directory

Y la operación comúnmente será abortada. Al usar rasgos observados, se debe clasificar todas las copias del CVS que usan ese depósito en un modo local o servidor. Si no se puede clasificar utilice los comandos **watch off** y **watch remove** para remover todo lo observado.

3.11.7 SELECCIONANDO ENTRE CHEQUEOS RESERVADOS O NO RESERVADOS

Los chequeos reservados y no reservados tienen pro y contras. Permite que se diga que mucho de esto es un problema de opinión o que los trabajos dan estilos de trabajo diferentes, pero aquí está una breve descripción de algo de estas emisiones. Hay muchas maneras para organizar un equipo de diseñadores. El CVS no trata de dar fuerza a una organización segura. Es una herramienta que se puede usar en varias maneras.

Los chequeos reservados pueden ser de muy contada producción. Si dos personas quieren revisar partes diferentes de un archivo, allí no podría estar una razón para prevenir que cualquiera de ellos lo haga así. También, es común para alguien asegurar un archivo, debido a que se está planeando editarlo, pero entonces olvide el seguro.

Las personas, quienes conocen la revisión reservada, a menudo se preguntan cómo a menudo ocurren conflictos si las revisiones no reservadas son utilizadas, y cuan difícil son de resolverlas. La experiencia con muchos grupos es que ocurren raramente y usualmente son relativamente directas de resolver. La extraño de los conflictos serios podrían ser sorprendentes, hasta uno se da cuenta que ocurren sólo cuando dos diseñadores discrepan en el propio diseño para una sección dada del código; tal discordancia sugiere que el equipo no ha sido comunicado propiamente en primer lugar. En orden para colaborar bajo cualquier régimen de manejo de fuente, los diseñadores deben estar de acuerdo en el diseño general del sistema; dado este acuerdo, los cambios están usualmente directos a unirse.

En algunos casos las revisiones no reservadas son claramente inapropiadas. Si no existen herramientas de unión para la clase de archivo que se esta manejando, y esto no es deseable para cambiar a un programa que usa un formato de datos combinables, entonces resolver los conflictos no esta siendo lo suficientemente placentero, eso que generalmente será mejor apartar o simplemente evitar los conflictos por otro lado, usar revisiones reservadas.

Cuando se va ha editar un archivo, es posible encontrar alguien más que está editándolo. Y en lugar de tener el sistema simplemente se prohibe a las dos personas editar el archivo, esto puede decirle en la situación que está, y esto permite figurar si este problema es un caso en particular o no.

Por su protección CVS se negará a verificar en un archivo si un conflicto ocurrió y no se ha resuelto el conflicto. Comúnmente para resolver un conflicto, debe cambiar el tiempo de sello (timestamp) en el archivo. En versiones previas de CVS, también necesita asegurar que el archivo no contiene conflictos marcados. Porque el archivo podría legítimamente crear conflictos marcados (esto es, ocurrencias de > > > > > > > en el inicio de una línea que no marca un conflicto), la versión presente de CVS imprimirá una advertencia y procede a registrar el archivo.

A menudo está útil informar a otros cuando comete una revisión nueva de un archivo. La opción **i** del archivo **modules**, o el archivo **loginfo**, se puede usar para automatizar este proceso.

Varios diseñadores simultáneamente intentan correr CVS

Si varios diseñadores tratan de correr CVS al mismo tiempo, se haría el mensaje siguiente:

[11:43:23] waiting for bach's lock in /usr/local/cvsroot/foo

CVS tratará nuevamente cada 30 segundos, y continúan con la operación o imprime el mensaje nuevamente, si todavía necesita esperar. Si un seguro parece pegar alrededor de una cantidad desmedida de tiempo, encontrar la persona que está a cargo del seguro y pedir que se corra el comando cvs. Si no se corre un comando de cvs, se observa

en el directorio del depósito mencionando el mensaje y quita archivos que poseen, cuyos nombres comienzan con **#cvs.rfl, #cvs.wfl, o #cvs.lock**.

Nótese que estos seguros están para proteger las estructuras internas de datos del CVS y no tiene ninguna relación a la palabra de seguro en el sentido usado por RCS WHICH que se refiere a **checkouts** reservados.

Cualquier número de personas puede leer desde un depósito determinado a la vez; solo cuando alguien escribe hace los seguros impidiendo al resto de gente leer o escribir.

Uno podría esperar para la propiedad siguiente

Si alguien comete algunos cambios en un comando cvs, entonces se hará una actualización por alguien a voluntad o conseguirá todo el cambio, o ninguno de ellos. Pero CVS no tiene esta propiedad. Por ejemplo, dado los archivos

a/one.c

a/two.c

b/three.c

b/four.c

Si alguien corre:

cvs ci a/two.c b/three.c

Y alguien corre una actualización cvs a la vez, la persona que corre la actualización podría conseguir únicamente el cambio de **b/three.c** y no el cambio de **a/two.c**.

Mecanismos para investigar quien revisa archivos

Para muchos grupos, el uso de CVS en su modo stándar es perfectamente satisfactorio. Los usuarios pueden ir a veces a registrar una modificación única para encontrar que otra modificación ha intervenido, pero lo reparten y proceden con su registro. Otros grupos prefieren poder saber quien revisa qué archivos, para que si dos personas tratan de revisar el mismo archivo, pueden escoger hablar de quien hace alguna cosa, en lugar

de sorprender el tiempo de registro. Los aspectos en esta sección permiten tal coordinación, mientras la retención de la capacidad de dos desarrolladores para editar el mismo archivo a la vez.

Para beneficio de los desarrolladores máximos deberían usar **cvs editor** para hacer archivos lectura - escritura y cvs edit para desechar un directorio de trabajo que no está más en uso, pero CVS no es capaz de imponer este comportamiento.

3.12 GERENCIA DE LA REVISIÓN

Este tema nos facilitará información importante cuando más de una persona está trabajando en un depósito.

3.12.1 CUÁNDO CONFIAR

Su grupo debe decidir qué política utilizar con confianza. Varias políticas son posibles, y como la experiencia con CVS crece se descubrirá probablemente qué trabaja correctamente para un mejor desarrollo.

Si se confía ficheros rápidamente puede ser que confíe los ficheros que incluso no compilan. Si su socio pone al día sus fuentes de trabajo para incluir su fichero, él no podrá compilar el código. Por otra parte, otras personas no podrán beneficiarse de las mejoras que se lleva a cabo al código si se confía muy raramente, y los conflictos serán probablemente más comunes.

Es común confiar solamente ficheros después de cerciorarse que pueden ser compilados. Las políticas como estas se pueden hacer cumplir usando el fichero del **commitinfo**, pero se debe pensar dos veces antes de que se haga cumplir a tal convención. Haciendo un ambiente de desarrollo controlado también puede convertirse regimented y así contraproducente a la meta verdadera, que es conseguir software lógica escrito.

3.13 SUBSTITUCIÓN DE LA PALABRA CLAVE

Se corrige el interior de los ficheros de fuente, su copia de trabajo de un módulo se puede descubrir mientras siempre el estado de sus ficheros esten **cvs status** y **cvs**

log. Pero tan pronto como se exporte los ficheros de su ambiente de desarrollo se convierte más dificilmente para identificar las revisiones aquellas.

CVS puede utilizar un mecanismo conocido como la substitución de la palabra clave (o extención de la palabra clave) para ayudar a identificar los ficheros. Cadenas embutidas de la palabra clave $ de $ de la forma y de la palabra clave de $:... $ en un fichero se substituyen por las cadenas de la palabra clave de $ de la forma : valore $ siempre que se obtenga una nueva revisión del fichero.

3.13.1 LISTA DE LAS PALABRAS CLAVES

Esta es una lista de las palabras claves:

$$Author$

El nombre de la conexión del utilizador que llegó la revisión.

$$Date$

La fecha y la hora (UTC) que la revisión llegó

$$Header$

Una cabecera estándar que contiene el pathname completo del fichero de RCS, del número de la revisión, de la fecha (UTC), del autor, del estado (si es bloqueado). Los ficheros normalmente nunca serán bloqueados cuando se utiliza CVS.

$$Id$

Igual como $$Header$, excepto el nombre de fichero de RCS está sin un camino.

$$Name$

Marque el nombre con etiqueta usado para controlar fuera de este fichero.

$$Locker$

El nombre de la conexión del utilizador que bloqueó la revisión (vacia si no está bloqueado, y así casi siempre inútil cuando se está utilizando CVS).

$$Log$

El mensaje del registro provisto confía, precedido por una cabecera que contiene el nombre de fichero de RCS, el número de la revisión, el autor, y la fecha (UTC). Los mensajes existentes del registro no se substituyen. En este lugar, el nuevo mensaje del registro se inserta después $$Log:... de $. Cada nueva línea se prefija con la misma cadena que precede $$Log keyword. Por ejemplo, si el fichero contiene

```
/* Here is what people have been up to:
 *
 * $Log: frob.c,v $
 * Revision 1.1  1997/01/03 14:23:51  joe
 * Add the superfrobnicate option
 *
 */
```

Entonces las líneas adicionales agregan que al ampliar la palabra clave $Log serán precedidas por *. .La palabra clave **$$Log** es útil para acumular una conexión completa del cambio un fichero de fuente, pero por varias razones puede ser problemática.

$$RCSfile$

El nombre del fichero de RCS sin un camino.

$$Revision$

El número de la revisión asignado a la revisión.

$Source$

El pathname completo del fichero de RCS.

$State$

El estado asignado a la revisión. Los estados se pueden asignar con los cvs que los admin - s.

3.13.2 USAR PALABRAS CLAVES

Para incluir una cadena de la palabra clave se incluye simplemente la cadena de texto relevante, por ejemplo **$$Id$**, dentro del fichero, y confía el fichero. CVS ampliará automáticamente la cadena como parte de la operación de confiar. Es común embutir la cadena de **$ Id$** en los ficheros de fuente de modo que consiga pasar a través de los

ficheros generados. Por ejemplo, si se está manejando código de fuente del programa de computadora, puede ser que incluya una variable que se inicializa para contener esa cadena.

El comando de la identificación se puede utilizar para extraer palabras claves y valores de un fichero. Esto puede ser práctico para los ficheros de texto, pero es aún más útil para extraer palabras claves de ficheros binarios.

```
$ ident samp.c
samp.c:
$Id: samp.c,v 1.5 1993/10/19 14:57:32 ceder Exp $
$ gcc samp.c
$ ident a.out
a.out:
$Id: samp.c,v 1.5 1993/10/19 14:57:32 ceder Exp $
```

SCCS es otro sistema de control popular de la revisión. Tiene un comando que es muy similar a la identificación y se utiliza para el mismo propósito. Muchos sitios sin RCS tienen SCCS. Desde que busca la secuencia del carácter @ (#) es fácil incluir las palabras claves que son detectadas por cualquier comando. Prefije simplemente la palabra clave de RCS con la frase magica SCCS, como esto:

```
static char *id="@(#) $Id: ab.c,v 1.5 1993/10/19 14:57:32 ceder Exp $";
```

3.13.3 EVITAR LA SUBSTITUCIÓN

La substitución de la palabra clave tiene sus desventajas. Puede ser que se quiera a veces que la cadena de texto literal `**$' Author$** apareciera dentro de un fichero que CVS la interpretaba como palabra clave y que la ampliaba en algo como `**$' autor: ceder $**.

No hay desafortunadamente manera de dar vuelta apagado a la substitución de la palabra clave. Se puede utilizar **-ko** dar vuelta apagado a la substitución de la palabra clave enteramente. En muchos casos se puede evitar usar palabras claves en la fuente, aunque aparecen en el producto final.

3.13.4 MODOS DE SUBSTITUCIÓN

Cada fichero tiene un modo de substitución salvado del valor por defecto, y cada copia del directorio de funcionamiento de un fichero también tiene un modo de substitución. El anterior es fijado por la opción **-k** a los cvs add y los **cvs admin**; el último es fijado por las opciones **-k** u **-A** a la comprobación de los cvs o a la actualización de los cvs .

El diff de los cvs también tiene una opción **-k**.

Los modos disponibles son:

-kkv

Genera las cadenas de la palabra clave usando la forma del valor por defecto, por ejemplo $Revision: 5.7 $ para la palabra clave de la revisión.

-kkvl

Como **-kkv**, el nombre de un armario se inserta siempre si la revisión dada está bloqueada actualmente. Esta opción no es normalmente útil cuando se utiliza CVS admin – l.

-kk

Genera solamente los nombres de la palabra clave en cadenas de la palabra clave; omite sus valores. Por ejemplo, para la palabra clave de la revisión, genere la cadena $Revision$ en vez de $Revision: 5.7 $. , estas opciones son útiles para no hacer caso de las diferencias debido a la substitución de la palabra clave al comparar diversas revisiones de un fichero.

-ko

Genera la vieja cadena de la palabra clave, presente en el fichero de trabajo momentos antes que fue llegado. Por ejemplo, para la palabra clave de la revisión, genere $Revision: 1.1 $ en vez de $Revision: 5.7 $, si es así cómo la cadena apareció cuando el fichero fue llegado.

-kb

Como **-ko**, pero también inhiba la conversión de las conclusiones de la línea entre la forma canónica en la cual se salvan en el depósito, y la forma apropiada al sistema

operativo en uso en el cliente. Para los sistemas, como unix, que utilizan linefeed para terminar solamente líneas, éste es igual que **-ko**.

-kv

Genera solamente los valores de la palabra clave para las cadenas de la palabra clave. Por ejemplo, para la palabra clave de la revisión, genere la cadena 5,7 en vez de la revisión $:5,7 $, que esto puede ayudar para generar ficheros en lenguajes de programación donde está duro eliminar delimitadores de la palabra clave como la revisión $: $ de una cadena. Sin embargo, la substitución adicional de la palabra clave no puede ser realizada una vez que se quiten los nombres de la palabra clave, así que esta opción se debe utilizar con cuidado. Uno quisiera a menudo utilizar **-kv** con los cvs de la exportación. Pero se debe saber que no maneja una exportación que contiene ficheros binarios correctamente.

3.13.5 PROBLEMAS CON LA PALABRA CLAVE $$LOG$

La palabra clave de **$ Log$** es algo polémica. Si se está trabajando en un sistema de desarrollo la información es fácilmente accesible incluso si no se utiliza la palabra clave de **$ Log$** apenas haga un registro de los cvs . Una vez que se exporte el fichero la información de la historia pudo ser inútil de todos modos.

Una preocupación más seria es que CVS no es bueno en la manipulación de entradas de **$ Log$** cuando una ramificación se combina sobre el tronco principal. Los conflictos resultan a menudo de la operación de combinación. La gente también tiende " a fijar

las entradas del registro en el fichero " (corrigiendo errores del deletreo y quizá incluso errores efectivos). Si se hace eso la información del registro de los cvs no será constante con la información dentro del fichero.

Se ha sugerido que la palabra clave de **$ Log$** fuera insertada por último en el fichero, y no en la cabecera del fichero, si se va a ser utilizado en todos..

3.14 SEGUIR FUENTES DE TERCERA PERSONA

Si se modifica un programa para mejorar el sitio de ajuste, si se desea probablemente incluir las modificaciones cuando llega el desbloquear siguiente del programa. CVS puede ayudarle con esta tarea. En la terminología usada en CVS, se llama al surtidor del programa un vendedor. La distribución sin modificar del vendedor se llega en su propia ramificación, la ramificación del vendedor. CVS reserva la ramificación 1,1,1 para este uso.

Cuando se modifica la fuente y la confía, su revisión terminará hacia arriba en el tronco principal. Cuando un nuevo desbloquear es hecho por el vendedor, se lo confía en la ramificación del vendedor y copia las modificaciones sobre el tronco principal. Utilice el comando de la importación para crear y poner al día la ramificación del vendedor.

Cuando se importa un fichero nuevo, la ramificación del vendedor se hace revisión del **head**, cualquier persona que controle fuera de una copia del fichero consigue esa

revisión. Cuando una modificación local está confiada se pone en el tronco principal, y entonces se realizó una revisión del **head**.

3.14.1 IMPORTACIÓN DE UN MÓDULO POR PRIMERA VEZ

Cuando se utiliza el comando de la importación de seguir fuentes de tercera persona, las etiquetas de la etiqueta del vendedor y del desbloquear son útiles. La etiqueta del vendedor es un nombre simbólico para la ramificación (que es siempre 1,1,1, a menos que se utilice -b branch las ramificaciones múltiples). Las etiquetas del desbloquear son nombres simbólicos para un desbloquear determinado, por ejemplo FSF_0_04.

Observe que la importación no cambia el directorio en el cual se lo invoca. En detalle, no instala ese directorio como directorio de funcionamiento de CVS; si se desea trabajar con las fuentes impórtelas primero y en seguida contrólelas hacia fuera en un diverso directorio. Supongamos que se tiene las fuentes a un programa llamado **wdiff** en un directorio **wdiff-0.04**, y van a hacer las modificaciones privadas que se quisiera que pudieran utilizar incluso cuando los nuevos desbloquear se hacen en el futuro. Se comienza importando la fuente a su depósito:

```
$ cd wdiff-0.04
$ cvs import -m "Import of FSF v. 0.04" fsf/wdiff FSF_DIST WDIFF_0_04
```

La etiqueta del vendedor se nombra **FSF_DIST** en el ejemplo antedicho, y la única etiqueta del desbloquear asignada es **WDIFF_0_04**.

3.14.2 PUESTA AL DÍA DE UN MÓDULO CON EL COMANDO DE LA IMPORTACIÓN.

Cuando llega un nuevo desbloquear de la fuente, se la importa en el depósito con el mismo comando de la importación que se instalaba el depósito en primer lugar. La única diferencia es que se especifica una diversa etiqueta del desbloquear esta vez.

```
$ tar xfz wdiff-0.05.tar.gz
$ cd wdiff-0.05
$ cvs import -m "Import of FSF v. 0.05" fsf/wdiff FSF_DIST WDIFF_0_05
```

Para los ficheros que no se han modificado localmente, la revisión nuevamente creada se convierte en la revisión principal. Si se ha realizado cambios locales, la importación le advertirá que se debe combinar los cambios en el tronco principal, y le recomienda utilizar **checkout -j**.

```
$ cvs checkout -jFSF_DIST:yesterday -jFSF_DIST wdiff
```

El comando antedicho controlará fuera de la última revisión de **diff**, combinando los cambios realizados en la ramificación del vendedor **FSF_DIST** desde ayer en la copia de trabajo. Si algunos conflictos se presentan durante la fusión deben ser resueltos de la manera normal. Entonces, los ficheros modificados pueden ser confiados. Usando una fecha, según lo sugerido arriba, asume que no se importe más de un desbloquear de un producto por día. Si se puede utilizar siempre algo como esto en lugar de otro:

```
$ cvs checkout -jWDIFF_0_04 -jWDIFF_0_05 wdiff
```

En este caso, los dos comandos antedichos son equivalentes.

3.14.3 INVERSIÓN AL ÚLTIMO DESBLOQUEAR DEL VENDEDOR

Se puede también invertir cambios locales totalmente y volver al último desbloquear del vendedor cambiando la revisión del **head** de nuevo a la ramificación del vendedor en todos los ficheros. Por ejemplo, si se tiene una copia controlada hacia fuera de las fuentes **~/work.d/wdiff**, y se desea invertir a la versión del vendedor para todos los ficheros en el directorio, se pulsaría:

```
$ cd ~/work.d/wdiff
$ cvs admin -bWDIFF .
```

Se debe especificar fuera **-bWDIFF** cualquier espacio después de la opción **-b**.

3.14.4 CÓMO MANEJAR FICHEROS BINARIOS CON LA IMPORTACIÓN DE LOS CVS

Se debe utilizar la opción **-k** de la envoltura para decir a importación qué ficheros son binarios.

3.14.5 CÓMO MANEJAR LA SUBSTITUCIÓN DE LA PALABRA CLAVE CON LA IMPORTACIÓN DE LOS CVS

Las fuentes que se está importando pueden contener palabras claves. Por ejemplo, el vendedor puede utilizar CVS u otro sistema que utilice sintaxis similar de la extensión de la palabra clave. Si las extensiones de la palabra clave hacen la importación justa de los ficheros en manera de valor por defecto, entonces las extensiones de la palabra clave provistas por el vendedor substituye por su propia copia de CVS. Puede ser más conveniente mantener las extensiones provistas por el vendedor, de modo que esta información pueda proveer la exploración sobre las fuentes que se importó del vendedor.

Para mantener las extensiones de la palabra clave provistas por el vendedor, provea la opción **-ko** a la importación de los cvs por primera vez que se importa el fichero. Esto dará vuelta apagado a la extensión de la palabra clave para ese fichero enteramente, así que si se desea ser más selectivo se tendrá que pensar de lo que se desea y utilizar la opción **-k** para **cvs update** o **cvs admin** como apropiado.

3.14.6 RAMIFICACIONES MÚLTIPLES DEL VENDEDOR

Todos los ejemplos asumen hasta ahora que hay solamente un vendedor de quien se está consiguiendo fuentes. En algunas situaciones puede ser que consiga fuentes de una variedad de lugares. Por ejemplo, supongamos que se está ocupando de un proyecto donde diversas personas y equipos están modificando el software lógica.

Para manejar las situaciones en las cuales puede haber más de un vendedor, se puede especificar la opción **-b** para **cvs import** . Toma como argumento la ramificación del vendedor a la importación. El valor por defecto es **-b 1.1.1**. Por ejemplo, supongamos que hay dos equipos, el equipo rojo y el equipo azul, que le están enviando fuentes. Se desea importar los esfuerzos del equipo rojo de ramificar 1,1,1 y de utilizar el ROJO de la etiqueta del vendedor. Se desea importar los esfuerzos del equipo azul de ramificar 1,1,3 y de utilizar el AZUL de la etiqueta del vendedor. Los comandos que se puede utilizar son:

$ cvs import dir RED RED_1-0

$ cvs import -b 1.1.3 dir BLUE BLUE_1-5

Observe que si la etiqueta del vendedor no corresponde con su opción **-b**, CVS no detectará este caso! Por ejemplo,

$ cvs import -b 1.1.3 dir RED RED_1-0

3.15 CÓMO SU SISTEMA DE LA ESTRUCTURA OBRA RECÍPROCAMENTE CON CVS

Según lo mencionado en la introducción, CVS no contiene el software lógica para construir su software lógica del código de fuente. Aquí se describe cómo los varios aspectos del sistema de estructura pudieron obrar recíprocamente con CVS.

Una pregunta común, especialmente de la gente que está acostumbrada a RCS, es cómo hacer que su estructura consigua una copia actualizada de las fuentes. Primero que

todo, puesto que el recurso de CVS así mismo a través de directorios, no hay ninguna necesidad de modificar su **Makefile** (o cualquier fichero de la configuración utiliza su herramienta de la estructura) cerciorarse que cada fichero es actualizado. Por otro lugar, los comandos de uso dos, primero utilizar cvs – q update y entonces el comando invoca su herramienta de la estructura. En segundo lugar, si no desea necesariamente conseguir una copia de un cambio de algún otro hecho hasta que se ha acabado su propio trabajo. Un acercamiento sugerido es que primero pone al día sus fuentes, entonces, construye y prueba el cambio en ejecución que se pensaba, y después confía sus fuentes.

Periódicamente poniendo al día su árbol entero, se asegura de que las fuentes son suficientemente actualizadas. Una necesidad común es registrar qué versiones de las cuales los ficheros de fuente empezaron una estructura determinada. Esta clase de funciones a veces se llama cuenta de los materiales o algo similar. La mejor manera de hacer esto con CVS es utilizar el comando de la etiqueta de registrar qué versiones comenzaron una estructura dada.

Usando CVS en la manera más directa posible, cada revelador tendrá una copia del árbol entero de la fuente que se utiliza en una estructura determinada. Si el árbol de la fuente es pequeño, o si los reveladores geográficamente se dispersan, ésta es la solución preferida. En el hecho uno, el acercamiento para proyectos más grandes es romper un proyecto abajo en subsistemas separado - compilados más pequeños, y arreglar una manera de **release/version** internamente de modo que cada chequeo de necesidad del

revelador fuera solamente de esos subsistemas que ellos estén trabajando sean activamente encendidos.

Otro acercamiento es instalar una estructura que permita que los reveladores tengan sus propias copias de algunos ficheros, y para que otros ficheros tengan acceso a ficheros de fuente de una localización central. Mucha gente ha venido con cierto sistema usando características tales como la característica simbólica de la conexión encontrada en muchos sistemas operativos, o la característica de **VPATH** encontrada en muchas versiones.

CAPITULO IV

FICHEROS ESPECIALES

En circunstancias normales, CVS trabaja solamente con los ficheros regulares. Cada fichero en un proyecto que se asume para ser persistente; debe ser posible abrirse, leerlos y cerrarlos, etc. CVS también no hace caso de los permisos y de las propiedades del fichero, dejando tales ediciones que se resolverán por el revelador en el tiempo de la instalación. Es decir no es posible " llegar " a un dispositivo en un depósito; si el fichero del dispositivo no puede ser abierto, CVS rechazará manejarlo. Los ficheros también pierden sus propiedades y permisos durante transacciones del depósito.

Si la configuración **PreservePermissions** variable se fija en el depósito, CVS salvará las características siguientes del fichero en el depósito:

- Propiedad del utilizador y del grupo
- Permisos
- Números de dispositivo importantes y de menor importancia
- Conexiones simbólicas
- Estructura dura de la conexión

Usar la opción de **PreservePermissions** afecta el comportamiento de CVS de varias maneras. Primero, algunas de las nuevas operaciones utilizadas por CVS no son accesibles a todos los utilizadores. En detalle, la propiedad del fichero y las características del fichero especial se pueden cambiar solamente por el **superuser**.

Cuando se fija la variable de la configuración de **PreservePermissions**, los utilizadores tendrán que ser **root** para realizar operaciones de CVS. Cuando **PreservePermissions** está en uso, las operaciones del CVS (por ejemplo cvs status) no reconocerán la estructura dura de la conexión de un fichero, y así emitirán alertas falsas sobre unir mal conexiones duras. La razón es que la estructura interna de CVS no hace fácil para que estas operaciones recojan todos los datos necesarios sobre conexiones duras, así que controlan si hay conflictos del fichero con datos inexactos.

Una diferencia más sutil es que CVS considera que un fichero ha cambiado solamente si su contenido ha cambiado (específicamente, si la época de la modificación del fichero de trabajo no corresponde con el del fichero del depósito). Por lo tanto, si solamente los permisos, la propiedad o el acoplamiento duro han cambiado, o si los números importantes o de menor importancia de un dispositivo han cambiado, CVS no lo notará.

Para confiar tal cambio al depósito, se debe forzar el confiar con **cvs commit -f**. Esto también significa que si los permisos de un fichero han cambiado y el fichero del depósito es más nuevo que la copia de trabajo, la ejecución **cvs update** cambiará silenciosamente los permisos en la copia de trabajo. Cambiar conexiones duras en un depósito de CVS es delicado. Supongamos que el fichero **foo** se conecta para clasificar **old**, pero es enlazado más adelante para clasificar **new**. Se puede mirar para arriba en la situación inusual donde, aunque **foo, old** y **new** tenga todos sus modelos subyacentes de la conexión cambiantes, solamente **foo** y **new** se han modificado, **old** no se considera un candidato a llegar. Puede ser muy fácil producir

resultados contrarios de esta manera. Por lo tanto, recomendamos que cuando es importante salvar conexiones duras en un depósito, la línea de conducta prudente es tocar cualquier fichero que el acoplamiento o estatus haya cambiado desde el enregistramiento pasado. De hecho, puede ser sabio tocar * antes que cada uno confie en un directorio con las estructuras duras complejas de la conexión.

Vale observar de que solamente los ficheros regulares pueden ser combinados, por las razones que esperanzadamente son obvias. Si **cvs update** o **cvs checkout –j,** las tentativas de combinar una conexión simbólica con un fichero regular, o dos ficheros del dispositivo para diversas clases de los dispositivos, CVS señala un conflicto y rechaza realizar la fusión. En el mismo tiempo, **cvs diff** no señalará ninguna diferencia entre estos ficheros, puesto que ninguna comparación textual significativa se puede hacer en los ficheros que no contienen ningún texto.

Las características de **PreservePermissions** no trabajan con CVS **client/server**. Otra limitación es que las conexiones duras deben estar a otros ficheros dentro del mismo directorio; las conexiones duras a través de directorios no se utilizan.

4.1 GUÍA A LOS COMANDOS DE CVS

Este apéndice describe la estructura total de los comandos de CVS, y describe algunos comandos detalladamente .

4.1.1 ESTRUCTURA TOTAL DE LOS COMANDOS DE CVS

El formato total de todos los comandos de CVS es:

cvs_command [cvs_options] [command_options] [command_args]

cvs

El nombre del programa de CVS.

cvs_options

Algunas opciones que afectan todos los submandatos de CVS.

cvs_command

Uno de varios y diversos submandatos. Algunos de los comandos tienen pseudónimos que se puedan utilizar uno en lugar de otro. Hay solamente dos situaciones donde se puede omitir **cvs_command** : **cvs -H** saca una lista de comandos disponibles, y **cvs -v** visualiza la información sobre la versión de CVS.

command_options

Opciones que son específicas para el comando.

command_args

Argumentos a los comandos.

Hay desafortunadamente una cierta confusión entre los **cvs_options** y los **command_options**. La opción **-l**, cuando está dado como **cvs_option**, afecta

solamente algunos de los comandos. Cuando se da como es un **command_option** tiene un diverso significado, y es validado por más comandos. Es decir no tome la clasificación antedicha demasiado seria.

4.1.2 ESTADO DE SALIDA DE CVS'S

CVS puede indicar al ambiente si tuvo éxito o falló fijando su estado de salida. La manera exacta de probar el estado de salida variará a partir de un sistema operativo a otro. Por ejemplo en un shell script de unix `$?' la variable será 0 si el comando pasado volvió a un estado de salida acertado, o mayor de 0 si el estado de salida indicó un incidente.

Si CVS es acertado, vuelve un estatus acertado; si hay un error, imprime un mensaje de error y vuelve un estatus del incidente. Una anomalía a esto es el comando del **diff** de los cvs. Volverá un estatus acertado si no encontró ninguna diferencia, o un estatus del incidente si había diferencias o si había un error. Porque este comportamiento no proporciona a ninguna buena manera de detectar errores, en el futuro es posible que el **diff** de los cvs será cambiado para comportarse como los otros comandos de CVS.

4.1.3 OPCIONES IMPLÍCITAS Y EL FICHERO DE ~/.CVSRC

Hay algunos **command_options** que se utilizan tan a menudo que puede ser que haya instalado un pseudónimo o algunos otros medios para cerciorarse que siempre se especifica esa opción. Un ejemplo, es realmente que mucha gente encuentra la salida

del valor por defecto del comando **diff** de ser muy dura de leer, y que los **diffs** o los **unidiffs** del contexto son mucho más fáciles de entender.

El fichero **~/.cvsrc** es una manera que se puede agregar opciones implícitas a los **cvs_commands** dentro de cvs, en vez de confiar en pseudónimos u otros **shell scriptes**.

El formato del fichero **~/.cvsrc** es simple. El fichero se busca para una línea que comience con el mismo nombre que el **cvs_command** que es ejecutado. Si se encuentra un emparejamiento, después el resto de la línea se dividiera en opciones separadas y se agrega a los argumentos del comando antes de cualquier opción de la línea de comando.

Si un comando tiene dos nombres, el nombre oficial, no necesariamente el que está usado en la línea de comando, será utilizado para corresponder contra el fichero. Si éste es el contenido del fichero **~/.cvsrc**:

Record diff - N update -u checkout - P

el comando **cvs checkout foo** tendría la opción **-P** agregada a los argumentos, así como **cvs co foo** .

Con el fichero del ejemplo de arriba, la salida **cvs diff foobar** está en formato del **unidiff**. **cvs diff -c foobar** proporcionará a **diffs** del contexto, como de costumbre. Consiguiendo " viejos " **diffs** del formato será levemente más complicado, porque el

diff no tiene una opción para especificar el uso del " viejo " formato, así que se necesitaría **cvs -f diff foobar**.

En lugar del nombre del comando se puede utilizar cvs para especificar opciones globales. Por ejemplo la línea siguiente dentro de **.cvsrc**

cvs - z6

hace a CVS utilizar el nivel 6 de la compresión.

4.1.4 OPCIONES GLOBALES

Las opciones disponibles **cvs_options** (a la izquierda de las cuales se dan cvs_command) son:

-- permitir - raíz = rootdir

Especifican el directorio legal de CVSROOT.

Las opciones de este comando se especifica en Anexos.

- d cvs_root_directory

Utilizan **cvs_root_directory** como el pathname del directorio de raíz del depósito. Reemplaza la configuración **$$CVSROOT** de la variable de entorno.

Las opciones de este comando se especifica en Anexos.

- H -- ayuda

Información del uso de visualización especifica **cvs_command**. Si no se especifica un nombre de comando, **cvs -H**, ayuda total de las visualizaciones para CVS, incluyendo una lista de otras opciones de ayuda. Las opciones de este comando se especifica en Anexos.

- variable = valor de s

Fijan a un utilizador variable.

Las opciones de este comando se especifica en Anexos.

- v -- versión

Versión de la visualización e información del **copyright** para CVS.

Las opciones de este comando se especifica en Anexos.

- z gzip-lebel

Instalan el nivel de comprensión. Solamente tienen un efecto en el cliente del CVS.

4.1.5 OPCIONES DE COMANDO COMÚN

Esta sección describe las opciones de comando que estan disponibles a través de algunos comandos de CVS. Estas opciones están dadas a la derecha del **cvs_command**.

No todos los comandos soportan todo de estas opciones; cada opción es solamente soportada por comandos donde este se hace notar. De cualquier modo, cuando un comando tiene una de estas opciones se puede casi siempre contar con el mismo comportamiento de la opción como en otros comandos. (Otras opciones de comando, las cuales están enlistadas con los comandos individuales, podrían tener diferente comportamiento de un comando del CVS a otro).

Advertencia: el comando **history** es una excepción; este soporta muchas opciones que chocan aún con estas opciones standares.

- D [date_spec]

Utiliza la revisión más reciente no la más tardía de **date_spec**. **[date_spec]** es un argumento simple, un dato de descripción especificando un dato en el pasado. La especificación es pegajosa cuando se lo usa para hacer una copia privada de un archivo de fuente; esto es, cuando se logra un archivo trabajado utilizando **- D**, el CVS graba los datos especificados, así más tarde se actualiza en el mismo directorio utilizando el mismo dato. La opción **- D** está disponible con los comandos **checkout**, **diff**, **export**, **history**, **rdiff**, **rtag**, y **update**.

Una amplia variedad de datos formateados son apoyados por el CVS. Unos de los más stándares son **ISO8601** (de la International Standars Organitation) y el standart **Internet e-mail** (especificado en RFC822 como rectificado por el RFC1123). Los datos del **ISO8601** tienen muchas variantes pero unos ejemplos son:

1972-09-24

1972-09-24 20: 05

En adición, las fechas son dejadas en el **Internet e-mail**, el CVS también deja que algunos de los campos sean omitidos. Por ejemplo:

24 Sep 1972 20: 05

24 Sep

Las fechas son interpretadas como son en la zona local de tiempo, a menos que una zona de tiempo específica sea determinada. Estos dos formatos de fecha son preferidos. De cualquier modo, el CVS corrientemente acepta una amplia variedad de formatos de fecha. Un formato semejante es mes/ día/ año. Esto podría confundir a las personas que están acostumbradas a tener el mes y día en otro orden; 1/ 4/ 96 es enero 4, no abril 1. Recordar citar el argumento a la señal **- D** así su presentación no interpretará espacios como argumentos separados.

Un comando utilizando la señal **- D** puede observarse así:

$ cvs diff – D "1 hour ago" [cvs.texinfo]

Otras opciones de este comando se especifica en Anexos.

- k kflag

Altera el proceso de omisión de palabras claves. La especificación **kflag** es pegajosa cuando se lo usa para crear una copia privada de una fuente de archivo; esto es,

cuando se utiliza esta opción con los comandos **checkout** o **update**, el CVS asocia el **kflag** seleccionado con el archivo, y continúa utilizándolo con futuros comandos **update** en el mismo archivo hasta que se especifique lo contrario.

La opción **- k** esta disponible con los comandos **add, checkout, diff, import** y **update**.

Las opciones de este comando se especifica en Anexos.

- m message

Utiliza mensajes para gran información, en lugar de invocar un editor. Disponible con los comandos siguientes: **add, commit** e **import**.

Las opciones de este comando se especifica en Anexos.

- r tag

Utiliza la revisión especificada por el argumento de etiqueta en lugar de la revisión de omisión **head**. También como etiquetas arbitrarias definidas con el comando **tag** o **rtag**, dos etiquetas especiales siempre son disponibles: **HEAD** se refiere a la versión más reciente disponible en el depósito, y **BASE** se refiere a la revisión última chequeada dentro del directorio trabajado corriente.

La especificación de la etiqueta es pegajosa cuando se usa con **checkout** o **update,** al hacer su propia copia del archivo: el CVS recuerda la etiqueta y continúa usandolo en

futuros comandos **update**, hasta que se especifique lo contrario. La etiqueta puede ser una etiqueta simbólica o numérica.

Especificando la opción global **-q** a lo largo con la opción de comando **-r** a menudo es útil, para suprimir los mensajes de advertencia cuando el archivo RCS no contiene la etiqueta especificada.

Advertencia: éste no es el mismo como todas las opciones **cvs - r**, que puede especificarse a la izquierda de un comando CVS! . La opción - **r** está disponible con los comandos **checkout, commit, diff, history, export, rdiff, rtag,** y **update.**
Las opciones de este comando se especifica en Anexos.

4.1.6 admin ADMINISTRACIÓN

- Requisitos: depósito, directorio trabajado.
- Cambios: depósito.
- Sinónimo: rcs

Este es la interface del CVS para combinar facilidades administrativas. Algunos de ellos tienen utilidades cuestionables para el CVS pero existen por propósitos históricos. Este comando trabaja recursivamente, debería ser utilizado con extremo cuidado.

Las opciones de este comando se especifica en Anexos.

4.1.7 checkout CHEQUEAR FUENTES REVISADAS PARA LA EDICIÓN

- Sinopsis: Chequear [opciones] modulos...
- Requiere: el depósito.
- Cambios: directorios trabajados.
- Sinónimos: co, get

Crea o actualiza un directorio de trabajo que contiene copias de los archivos de fuente especificado por módulos. Se debe ejecutar **checkout** antes de usar la mayoría de los comandos de CVS, la mayoría de ellos operan sobre el directorio de trabajo.

Los módulos o los nombres simbólicos son para algún recaudo de archivos y directorios de fuente, o trayectorias a directorios o archivos en el depósito. Los nombres simbólicos se definen en el archivo **modules**.

Depende de los módulos que se especifica, **checkout** puede crear recursivamente los directorios y los llenan con los archivos de fuente apropiados. Se puede editar entonces este archivo de fuente a cualquier tiempo (sin considerar si los otros desarrolladores de software editan sus copias propias de las fuentes); los actualiza para incluir nuevos cambios aplicado por otros al depósito de fuente; o comprometer su trabajo como un cambio permanente al depósito de fuente.

Nótese que **checkout** se usa para crear directorios. El directorio de alto nivel creado se agrega siempre al directorio donde **checkout** se invoca, y comúnmente tiene el mismo nombre como el módulo especificado. En el caso de un alias de módulo, el sub-directorio creado puede tener un nombre diferente, pero se puede estar seguro que será un sub - directorio, y que **checkout** muestre la trayectoria relativa que conduce al archivo como se extrae en su área privada de trabajo (a menos que se especifique la opción global -Q). Los archivos creados por **checkout** se crean como lectura-escritura, a menos que la opción **-r** de CVS se especifique.

Nótese que corriendo **checkout** en un directorio que estuvo ya construído previo un checkout es además permitido. Esto es similar a especificar la opción – **d** al comando **update** en el sentido de que nuevos directorios que ya han sido creados en el depósito aparecerán en su área de trabajo. Sin embargo, **checkout** toma un nombre de módulo y mientras que **update** toma un nombre de directorio.

También al usar **checkout** esta debe ser corrida desde el nivel tope del directorio (de donde se corrió originalmente checkout), antes de que haya sido corrido **checkout** actualizar un directorio existente, no hay que olvidar cambiar el directorio al nivel tope del directorio.

4.1.7.1 OPCIONES DEL COMANDO CHECKOUT

Estas opciones estándares son apoyadas por **checkout,** las mismas que se especifican en anexos.

4.1.7.2 EJEMPLOS DE CHECKOUT

Conseguir una copia del módulo **tc**:

$ cvs checkout tc

Conseguir una copia del módulo **tc** como se observó el día anterior:

$ cvs checkout - D yesterday tc

4.1.8 commit GUARDAR ARCHIVOS CHEQUEADOS DENTRO DEL DEPÓSITO

- Sinopsis: guardar [- lnRf] [- m 'log_messagé | - F archiva] [- r revission] [files...]
- Requiere: directorio trabajado, depósito.
- Cambios: el depósito.
- Sinónimo: ci

Se utiliza **commit** cuando se quiere incorporar cambios desde sus archivos de trabajo de fuente dentro de la fuente del depósito.

Si no se especifica archivos particulares para comprometer, todos los archivos en su directorio actual de trabajo se examinan. El **commit** es cuidadoso al cambiar en el depósito esos archivos que se ha cambiado realmente. Por falla (o si explícitamente especifica la opción -R), los archivos en subdirectorios se examinan y se guardan si ellos han cambiado; se puede usar la opción **-l** para limitar a **commit** a los directorios recientes solamente.

Commit verifica que los archivos selectos estén arriba de la fecha con las revisiones actuales en la fuente del depósito; se le notificará , y al salir si no están guardados, si cualquiera de los archivos especificados deben ser primero actualizados. **Commit** no llama al comando **update** para nosotros, sino que deja que se haga cuando el momento es correcto.

Cuando todo esta bien, un editor es invocado para permitir que entre en un mensaje de registro que se escribirá a uno o más registros de programa y poniendo el archivo RCS dentro del depósito. Este mensaje de registro puede recobrarse con el comando **log**; . Se puede especificar el mensaje de registro sobre la línea de comando la opción **-m message**, y así evitar la invocación del editor, o utilizar la opción **-F file** para especificar que el archivo de argumento contiene el mensaje de registro.

4.1.8.1 OPCIONES PARA GUARDAR

Estas opciones standard son apoyadas por **commit** y especificadas en anexos.

4.1.8.2 GUARDANDO

4.1.8.2.1 GUARDANDO A UNA RAMA

Se puede guardar una revisión de una rama (uno que tiene un número par de puntos) con la opción **-r**. Para crear una revisión de rama, se utiliza la opción **-b** de los comandos **rtag** o **tag**. Entonces, **checkout** o **update** puede usarse para basar sus

fuentes sobre la nueva rama creada. Desde este punto, sobre todo **commit** los cambios hechos dentro de estas fuentes trabajadas se agregarán automáticamente a una revisión de rama, por otro lado no daña el desarrollo de la línea principal de cualquier forma. Por ejemplo, si se tuvo que crear un parche a la versión 1.2 del producto, aunque la versión 2.0 esta ya desarrollada abajo, se podría hacer:

```
$ cvs rtag - b - r FCS1_2 FCS1_2_PATCH product_module
$ cvs checkout - r FCS1_2_PATCH product_module
$ cd product_module
[[ hack away ]]
$ cvs commit
```

Esto trabaja automáticamente desde la opción viscosa **-r**.

4.1.8.2.2 CREANDO LA RAMA DESPUÉS DE EDITAR

Si se ha trabajado sobre algún software sumamente experimental, basado en cualquier revisión y sucedió un chequeo la semana pasada. Si otros en su grupo les gustaría trabajar sobre este software con nosotros, pero sin perturbar la principal - línea de desarrollo, se podría guardar el cambio a una nueva revisión. Otros pueden entonces chequear el material experimental y utilizar todos los beneficios de la resolución de conflictos del CVS. El escenario se podría mirar así:

```
[[ hacked sources are present ]]
$ cvs tag - b EXPR1
$ cvs update - r EXPR1
```

```
$ cvs commit
```

El comando **update** hará la opción **-r EXPR1** sobre todos los archivos. Nótese que los cambios a los archivos nunca serán quitados por el comando **update**. El **commit** comprometerá automáticamente a la revisión correcta, porque la opción **-r** es viscosa. Se podría también hacer esto:

```
[[ hacked sources are present ]]
$ cvs tag - b EXPR1
$ cvs commit - r EXPR1
```

pero entonces, solamente esos archivos que fueron cambiados podrían tener la opción viscosa **-r EXPR1**. Si se le parte, y guarda sin especificar la opción **-r EXPR1**, algunos archivos pueden acabar accidentalmente sobre el tronco principal.

Para trabajar con nosotros sobre el cambio experimental, los otros harían simplemente:

```
$ cvs checkout - r EXPR1 whatever_module
```

4.1.9 diff—MUESTRA DIFERENCIAS ENTRE REVISIONES

- Sinopsis: diff [- lR] [format_options] [[- r rev1 | - D date1] [- r rev2 | - D date2]] [files...]
- Requiere: directorio trabajado, depósito.

- Cambios: nada.

El comando **diff** se usa para comparar revisiones diferentes de archivos. La acción por defecto es comparar sus archivos de trabajo con las revisiones en las que estaban basadas, e informar cualquier diferencias que se encuentran.

Si algunos nombres de archivo son dados, solamente esos archivos son comparados. Si algunos directorios son dados, todos los archivos debajo de ellos se compararán.

La condición de salida para **diff** es diferente que para otros comandos de CVS

4.1.9.1 OPCIONES DE diff

Estas opciones estándards son apoyadas por **diff** y especificadas en anexos.

4.1.9.2 EJEMPLOS Diff

La siguiente línea produce un **Unidiff** (-u flag) entre revisiones 1.14 y 1.19 de **backend.c**. Debido a la opción **-kk** ninguna palabra clave se sustituye, estas diferencias solamente dependen de que la sustitución de palabras claves son ignoradas.

$ cvs diff - kk - u - r 1.14 - r 1.19 backend.c

Supongamos que la rama experimental EXPR1 era basada en un conjunto de archivos etiquetados RELEASE_1_0. Para ver qué ha sucedido sobre esta rama, se puede usar lo siguiente:

$ cvs diff - r RELEASE_1_0 - r EXPR1

Un comando como puede ser usado para producir un contexto **diff** entre dos exoneraciones:

$ cvs diff - c - r RELEASE_1_0 - r RELEASE_1_1 > diffs

Si se mantiene los cambios completos, un comando como el siguiente justo antes de que se guarde los cambios puede ayudar a escribir para introducir el cambio completo.

Todas las modificaciones locales que aún no han sido guardadas se imprimirán.

$ cvs diff - u | less

4.1.10 export—FUENTES DE EXPORTACIÓN DESDE EL CVS, (SIMILAR A CHECKOUT)

- Sinopsis: export [- flNnR] [- r rev| - EL D fecha] [- k más sub] [- d dir] module...
- Requiere: el depósito.
- Cambios: el directorio actual.

Este comando es una variante de **checkout**; úselo cuando se quiera una copia de la fuente para el módulo sin directorios administrativos del CVS. Por ejemplo, se podría

usar **export** para preparar fuentes para enviar a otro sitio. Este comando requiere que se especifique una fecha o etiqueta (con -D o -r), así esto puede contar con la reproducción de la fuente que se envía a otros.

Nos gustaría frecuentemente usar **-kv** con **cvs export**. Esto ocasiona cualquier palabra clave que al ser expandida como en una importación de lo hecho en algún otro lugar no se perderá la palabra clave de la revisión de la información. Pero hay que ser consciente que no maneja un archivo binario de contenido de exportación correctamente.

4.1.10.1 OPCIONES DE export

Estas opciones standard son apoyadas por **export** y especificadas en anexos.

4.1.11 history—MUESTRA CONDICIÓN DE ARCHIVOS Y USUARIOS

- Sinopsis: history [- informa] [- banderas] [- opciones args] [files...]
- Requiere: el archivo `$CVSROOT/CVSROOT/history '
- Cambios: nada.

CVS puede guardar un archivo de historia que investiga el uso de cada comando **checkout , commit, rtag , update, y release**. Se puede usar **history** para mostrar esta información en diversos formatos. Registrando esto debe ser permitido crear el archivo **$CVSROOT/CVSROOT/history.**

Advertencia: history utiliza **-f, -l, -n,** y **–p** de manera que conflictúa con el uso normal al interior de CVS .

4.1.11.1 OPCIONES DE HISTORY

Varias opciones (mostradas arriba como -report) controlan qué clase de reporte es generado, estas están especificadas en anexos.

4.1.12 import—IMPORTA FUENTES DENTRO DEL CVS, USANDO RAMAS VENDEDORAS

- Sinopsis: import [- opcions] depósito vendortag releasetag...
- Requiere: El depósito, fuente distribución de directorio.
- Cambios: el depósito.

Usar **import** para incorporar una fuente entera de distribución desde la fuente externa en su directorio de depósito de fuente. Se puede usar este comando para la creación inicial de un depósito, y para la mayoría de actualizaciones del módulo desde la fuente de salida.

El argumento depósito da un nombre al directorio (o una trayectoria al directorio) bajo la raíz del directorio de CVS para depósitos; si el directorio no existió, importa lo crea.

Cuando se usa importación para actualizaciones a la fuente que se ha modificado en el depósito de fuente (desde una importación anterior), se notificará de cualquier archivo que están en conflicto en las dos de las ramas de desarrollo; usar **checkout - j** para reconciliar las diferencias, como las instrucciones importadas que se hace.

Si CVS decide ignorar un archivo, esto no lo importa y los imprime **I** seguidos por el nombre de archivo .

Si el archivo **$CVSROOT/CVSROOT/cvswrappers** existe, cualquier archivo cuyos nombres combinen las especificaciones en ese archivo se tratarán como paquetes y los filtradores apropiados se desempeñarán sobre el archivo/directorio antes de ser importados.

La fuente afuera es gravada en el primer nivel de la rama, por defecto 1.1.1. Las actualizaciones son las licencias de esta rama; por ejemplo, los archivos de la primera colección importada de la fuente será la revisión 1.1.1.1, entonces los archivos de la primera actualización importada será la revisión 1.1.1.2, y así sucesivamente.

Por lo menos tres argumentos se requieren. El depósito se necesito identificar el recaudo de fuente **.vendortag** es una etiqueta para la rama entera (p. ej., para 1.1.1). Se debe especificar también por lo menos un **releasetag** para identificar los archivos a las licencias creadas cada vez que se ejecute **import**.

Nota: Este **import** no cambia el directorio en que se lo invocó. En particular, este no instala ese directorio como directorio trabajado CVS; si se quiere trabajar con las fuentes impórtelos primero y entonces chequéelos en un directorio diferente

4.1.12.1 OPCIONES DE import

Estas opciones estándard son apoyadas por **import** y especificadas en anexos.

4.1.12.2 RENDIMIENTO DEL import

El **import** le mantiene informado de su progreso imprimiendo una línea para cada archivo, precedido por un carácter que indica la condición del archivo:

U file

El archivo ya existe en el depósito y no ha sido modificado localmente; una nueva revisión se ha creado (si es necesaria).

N file

El archivo es un nuevo archivo que se ha agregado al depósito.

C file

El archivo ya existe en el depósito pero no ha sido modificado localmente; se tendrá que combinar los cambios.

I file

El archivo está siendo ignorado

L file

El archivo es un nexo simbólico; **cvs import** ignora nexos simbólicos.

4.1.13 log—IMPRIME LA INFORMACIÓN TOTAL DE LOS ARCHIVOS

- Sinopsis: log [options] [files...]
- Requiere: el depósito, directorio trabajado.
- Cambios: nada.

Muestra información de registro para archivos. **Log** es usado para llamar el RCS de utilidad **rlog**. Aunque esto no es más cierto en las fuentes actuales, esta historia determina el formato del rendimiento y de las opciones, que no son bastantes en el estilo de otros comandos de CVS.

El rendimiento incluye la ubicación del archivo RCS, la revisión **head** (la última revisión sobre el tronco), todos los nombres simbólicos (etiqueta) y algunas otras cosas.

Para cada revisión, el número de revisión, el autor, el número de líneas added/deleted y el mensaje de registro son inpresos. Todo el tiempo son mostrados en Coordinated Universal Time (UTC). (Otras partes de CVS se imprimen a veces en la zona de tiempo local).

Advertencia: **log** utiliza **-R** en una forma que conflictúa con el uso normal al interior de CVS.

4.1.13.1 OPCIONES log

Por defecto, **log** imprime toda la información que está disponible. Todas otras opciones restringen el rendimiento y están especificadas en anexos.

4.1.14 rdiff --FORMATO DE PARCHE ('patch') diffs Entre Liberaciones

- rdiff [- flags] [- V vn] [- r t| - D d [- r t2| - D d2]] modules...
- Requiere: el depósito.
- Cambios: nada.
- Sinónimo: remiendo

Construye un formato de parche, que archivan entre dos liberaciones, este puede ser alimentado directamente dentro del programa patch para traer una liberación antigua con la nueva liberación. (Este es uno de los pocos comandos CVS que opera directamente desde el depósito, y no requiere un chequeo previo). El rendimiento de **diff** es enviado al dispositivo de rendimiento stándard .

Se puede especificar (usando las opciones normales -r y -D) cualquier combinación de uno o dos revisiones o fechas. Si solamente una revisión o fecha es especificada, el archivo de parche refleja diferencias entre esa revisión o fecha y las revisiones actuales en el RCS de archivo.

Nótese que si el software afectado libera, este contiene más de un directorio, entonces puede ser necesario especificar la opción **-p** al comando **patch** cuando se parcha a las fuentes viejas, para que el parche sea capáz de encontrar los archivos que se ubican en otros directorios.

4.1.14.1 OPCIONES rdiff

Estas opciones estándards son apoyadas por **rdiff** y se encuentran especificadas en anexos.

4.1.14.2 EJEMPLOS rdiff

Supongamos que se recibe correo desde **foo@bar.com** pidiendo una actualización desde la liberación 1.2 a 1.4 del compilador **tc**. No se tiene tales parches a mano, pero con CVS esto puede ser fácilmente arreglado con un comando como este:

```
$ cvs rdiff - c - r FOO1_2 - r FOO1_4 tc | \
$$ Mail - s 'The patches you asked for ' foo@bar.com
```

Supongamos que se ha liberado 1.3, y bifurcado una rama llamada R_1_3fix para arreglar virus. R_1_3_1 corresponde a la liberación 1.3.1, que se hizo hace algún tiempo. Ahora, se quiere ver cuánto desarrollo se ha hecho sobre la rama. Este comando puede usarse:

```
$ cvs patch - s - r R_1_3_1 - r R_1_3FIX module - name
cvs rdiff: Diffing module - name
File ChangeLog,v changed from revision 1.52.2.5 to 1.52.2.6
File foo.c,v changed from revision 1.52.2.3 to 1.52.2.4
File bar.h,v changed from revision 1.29.2.1 to 1.2
```

4.1.15 release--INDICA QUE UN MÓDULO NO ES TAN EXTENSO EN SU USO

- release [- d] directories...
- Requiere: Directorio trabajado.
- Cambios: Directorio trabajado, historia de registro.

Este comando es principal para ahorrar cancelación del efecto del **cvs checkout**. Desde que CVS no cierra los archivos, no es estrictamente necesario usar este comando. Se puede siempre simplemente borrar su directorio de trabajo, si así se quiere; pero se arriesga perder cambios que se puede haber olvidado, y dejado de rastrear en el archivo **history** del CVS que ha abandonado sus chequeos.

Utilizar **cvs release** para evitar estos problemas. Este comando chequea que cambios no guardados estén presentes; que se está ejecutándolo inmediatamente desde arriba en

un directorio CVS; y que el depósito registrado para sus archivos es al igual que el depósito definido en el módulo de la base de datos.

Si todas estas condiciones son ciertas, **cvs release** deja un registro de su ejecución (dando testimonio de su intencionalmente abandono del chequeo) en el total **history** del CVS.

4.1.15.1 OPCIONES release

El comando **release** soporta una opción de comando:

- d

Borra sus copias de trabajo del archivo si la liberación sucede. Este parámetro no es dado por los archivos que permanecerán en el directorio de trabajo.

Advertencia: El comando **release** borra todos los directorios y archivos recursivamente. Esto tiene el lado muy serio en efecto que cualquier directorio que se ha creado dentro de sus fuentes revisadas, y no se añade al depósito (usando el comando add) que serán silenciosamente borrados aún si este no está vacío!

4.1.15.2 RENDIMIENTO DEL release

Antes **release** liberaba sus fuentes que se imprimirá en una - línea de mensaje para cualquier archivo que no está a la fecha.

Advertencia: Algunos nuevos directorios que se ha creado, pero no agregado al CVS la jerarquía del directorio con el comando **add** se ignorarán silenciosamente (y borrados, si es especificado -d), aún si ellos contienen archivos.

A file

El archivo se ha agregado a su copia privada de las fuentes, pero no se ha comprometido al depósito. Si se borra su copia de las fuentes este archivo se perderá.

M file

El archivo se modifica en su directorio de trabajo. Allí puede también ser una nueva revisión más dentro del depósito.

? file

El archivo está en su directorio de trabajo, pero no corresponde a nada en el depósito de fuente, y no es en la lista de archivos de CVS para ignorar. Si se quitan las fuentes de trabajo, este archivo se perderá.

4.1.15.3 EJEMPLOS DE release

Libere el módulo, y borre su copia de trabajo local de los archivos.

```
$ cd ..
cvs release.
$ cvs release - d tc
```

Se tiene **0** archivos alterados en este depósito.

Esta seguro que quiere liberar (y borrar) el módulo **tc**:

y

$

4.1.16 rtag—AÑADIR UNA ETIQUETA SIMBÓLICA AL MÓDULO

- rtag [- falnR] [- b] [- d] [- r tag | - Ddate] symbolic_tag modules...
- Requiere: el depósito.
- Cambios: el depósito.
- Sinónimo: rfreeze

Se puede usar este comando para asignar etiquetas simbólicas a particular, explícitamente revisiones de fuente especificada en el depósito. **rtag** trabaja directamente sobre los contenidos del depósito (y no requiere previo chequeo). Al utilizar **tag** por otro lado, se basa la selección de revisiones sobre los contenidos de su directorio de trabajo.

Si se intenta usar un nombre de etiqueta que ya existe, el CVS se quejará y no sobreescribirá esa etiqueta. Utilice la opción **-F** para forzar el nuevo valor de etiqueta.

4.1.16.1 OPCIONES rtag

Estas opciones estándards son apoyadas por **rtag** y especificadas en anexos.

4.1.17 tag—AÑADE UNA ETIQUETA SIMBÓLICA A VERSIONES REVISADAS DE ARCHIVOS

- tag [- lR] [- b] [- c] [- d] symbolic_tag [files...]
- Requiere: directorio trabajado, depósito.
- Cambios: el depósito.
- Sinónimo: freeze

Utilice este comando para asignar etiquetas simbólicas a las versiones más cercanas del depósito a sus fuentes de trabajo. Las etiquetas se aplican inmediatamente al depósito, como con **rtag**, pero las versiones se abastecen implícitamente por las grabaciones del CVS de la historia de su archivo trabajado en lugar de ser aplicadas explícitamente.

El uso para las etiquetas es registrar una instantánea de las fuentes actuales cuando la fecha de software se congela de un proyecto que llega. Como los virus son arreglados después de que la fecha se congela, solamente estas fuentes cambiadas que están para ser parte de la liberación necesitan ser reetiquetadas.

Las etiquetas simbólicas están principalmente, permanentemente registradas. Dichas revisiones de archivos fueron utilizados en la creación de una distribución de software. Los comandos **checkout** y **update** permiten que se extraiga una copia exacta de una liberación etiquetada a cualquier tiempo en el futuro, sin considerar si los archivos se han cambiado, agregados, o quitados desde que la liberación fue etiquetada.

Este comando puede también ser utilizado para borrar una etiqueta simbólica, o para crear una rama. Si se intenta utilizar un nombre de etiqueta que ya existe, CVS se quejará y no sobreescribirá esa etiqueta. Use la opción **-F** para forzar el nuevo valor de etiqueta.

4.1.17.1 OPCIONES tag

Estas opciones estándards son apoyadas por **tag** y especificadas en anexos.

4.1.18 update—TRAE UN ÁRBOL DE TRABAJO EN LA SINCRONIZACIÓN CON EL DEPÓSITO

- update [- AdflPpR] [- d] [- r tag| - D date] files...
- Requiere: el depósito, directorio trabajado.
- Cambios: directorio trabajado.

Después que se tiene corrido el chequeo para crear su copia privada de fuente desde el depósito común, los otros desarrolladores continuarán cambiando la fuente central. De vez en cuando, cuando es conveniente en su proceso de desarrollo, se puede usar el comando **update** desde el interior de su directorio de trabajo para reconciliar su trabajo con cualquier revisión aplicado al depósito de fuente desde su último chequeo o actualización.

4.1.18.1 OPCIONES DE update

Estas opciones estándar están disponibles con **update** y especificadas en anexos.

4.1.18.2 RENDIMIENTO DEL update

update y **checkout** guardan el informe del progreso al imprimir una línea para cada archivo, precedido por un carácter indicando la condición del archivo

u file

El archivo fue traído a la fecha con respecto al depósito. Esto se hace para cualquier archivo que existe en el depósito pero no en la fuente, y para archivos que no han cambiado pero que no están en las versiones más recientes disponibles en el depósito.

P file

Como la **U**, pero el servidor del CVS envía un parche en vez de un archivo entero.

Estas dos partes realizan la misma cosa.

A file

El archivo se ha agregado a su copia privada de las fuentes, y se agregará al depósito de fuente cuando se corra **commit** sobre el archivo. Este es un recordatorio que el archivo necesita ser guardado.

R file

El archivo ha sido quitado de su copia privada de las fuentes, y se quitará desde el depósito de fuente cuando se corra **commit** sobre el archivo. Esto es un recordatorio de que el archivo debe ser guardado.

M file

El archivo se modifica en su directorio de trabajo. **M** puede indicar uno de los dos estados de un archivo que se está trabajando: si no hay modificaciones en el archivo en el depósito, entonces el archivo permanece como se lo vio por última vez; o si hay modificaciones en el depósito también las hay en la copia, pero estas fueron combinadas exitosamente, sin conflictos, en el directorio trabajado. El CVS imprimirá algunos mensajes si este combina el trabajo, y una copia de respaldo de su archivo de trabajo (como se miró antes de ejecutar update) se hará. El nombre exacto de este archivo es impreso mientras **update** corre.

C file

Un conflicto fue detectado mientras se trató de combinar sus cambios al **file** con cambios de la fuente del depósito. **File** (la copia en su directorio de trabajo) es ahora el resultado de intentar combinar las dos revisiones; una copia inmodificable de su archivo está en su directorio de trabajo, con el nombre **.# file. revision** donde **revision** es la revisión desde donde empezó el archivo modificado. (Nótese que algunos sistemas automáticamente purgan archivos que comienzan con `.#' si ellos no han sido accedidos por unos pocos días. Si no se intentan guardar una copia de su archivo original, esto es una idea muy buena para renombrarlos.).

? file

File es un directorio trabajado, pero no corresponde a nada en el depósito de fuente, y no está en la lista de archivos de CVS para ignorarlo.

4.2 REFERENCIA RÁPIDA A LOS COMANDOS DE CVS

Este apéndice describe cómo invocar a CVS, con las referencias a donde cada comando o característica se describe detalladamente.

Un comando de CVS aparece así:

cvs order [global_options][command_options] [command_args]

OPCIONES GLOBALES:

-- permitir-raíz = rootdir

Especifique el directorio legal de **CVSROOT** (servidor solamente) (no en CVS 1,9 y más antiguo). Los parámetros que se pueden utilizar están especificados en anexos.

- H -- ayuda

Imprime un mensaje de ayuda. Sus opciones se especifican en anexos.

- variable = valor de s

Fija a un utilizador variable.

- tempdir de T

Pone los ficheros temporales en **tempdir**.

- t

Ejecuta el rastro de CVS.

- v-- versión

Versión de la visualización e información del copyright para CVS.

- W

Hace los ficheros de trabajo nuevos de lectura/grabación.

- x

Cifre toda la comunicación (cliente solamente).

- gzip-nivel de z

Fija el nivel de la compresión (cliente solamente).

Modos de la extensión de la palabra clave:

```
- kkv $$Id:  file1, v 1,1 1993/12/09 03:21:13 Joe Exp $ -

kkvl $$Id:  file1, v 1,1 1993/12/09 03:21:13 Joe Exp $ harry - kk

$$Id$ - el kilovoltio file1, v 1,1 1993/12/09 03:21:13 Joe
```

Exp - ko

Ninguna extensión

- KB

Ninguna extensión, el fichero es binario

Palabras claves :

```
$$Author: Joe $ $$Date: 1993/12/09 03:21:13 $ $$Header: /
home/files/file1, v 1,1 1993/12/09 03:21:13 Joe Exp $ harry $$Id:
file1, v 1,1 1993/12/09 03:21:13 Joe Exp $ harry $$Locker: $ harry
$$Name: snapshot_1_14 $ $$RCSfile: file1, v $ $$Revision: 1,1 $: /
home/files/file1, v $: Exp $ $$Log: file1, revisión 1,1 de v $
1993/12/09 revisión inicial de 03:30:17 Joe
```

COMANDOS, OPCIONES DEL COMANDO, Y ARGUMENTOS DEL COMANDO:

- **add [Opciones] [Ficheros ...]**

Agrega un file/directory nuevo.

- K kflag

Fija la extensión de la palabra clave.

- m msg

Fija la descripción del fichero.

- **admin [Opciones] [Ficheros ...]**

Administración de los ficheros de historia en el depósito. Sus parámetros se especifican en anexos.

- **anote [Opciones] [Ficheros ...]**

Muestra la revisión pasada donde cada línea fue modificada. Sus parámetros se especifican en anexos.

- **Chekout module [Opciones] ...**

Consigue una copia de las fuentes. Sus parámetros se especifican en anexos.

- **confide [Opciones] [Ficheros ...]**

Controla los cambios en el depósito. Sus parámetros se especifican en anexos.

- **diff [Opciones] [Ficheros ...]**

Muestra las diferencias entre las revisiones. Además de las opciones mostradas en anexos, valida una variedad amplia de opciones para controlar el estilo de la salida, por ejemplo **-c** para los **diffs** del contexto.

- **correct [Opciones] [Ficheros ...]**

Consigue corregir un fichero que se está observando

- action

Especifica las acciones para el reloj temporal, donde están las acciones **correct, unedit,** confident, all, o none.

- l

Local; ejecútese solamente en directorio de trabajo actual.

- R

Funciona recurrentemente (valor por defecto).

- **editors [opciones] [ficheros ...]**

Se conoce quién está corrigiendo un fichero observado.

- l

Local; ejecútese solamente en directorio de trabajo actual.

- R

Funciona recurrentemente (valor por defecto).

- **export modules [de las Opciones] ...**

Exporta los ficheros de CVS. Sus parámetros se especifican en anexos.

- **history [Opciones] [Ficheros ...]**

Muestra la historia del acceso del depósito. Sus parámetros se especifican en anexos.

- **import [Opciones] depósito vendedor - tag liberador - tags ...**

Importa los archivos dentro del CVS, usando ramas vendedoras. Sus opciones se especifican en anexos.

- **log [Opciones] [Archivos ...]**

Imprime la historia de información de los archivos. Sus opciones se especifican en anexos.

- **rdiff [opciones] módulos ...**

Muestra diferencias entre las liberaciones. Sus opciones se especifican en anexos.

- **release [opciones] directorio**

Indica que un directorio no está más en uso.

- d

Borra un directorio determinado.

- **Remove [opciones] [archivos ...]**

Remueve una entrada del depósito.

- l

Local; corre solamente con un directorio trabajado reciente.

- R

Opera recursivamente (omisión).

- **rtag [Opciones] etiqueta módulos ...**

Agregua una etiqueta simbólica al módulo. Sus opciones se especifican en anexos.

- **status [opciones] archivo ...**

Muestra el estado de información en un directorio de trabajo.

- l

Local; corre únicamente en directorios trabajados recientes.

- R

Opera recursivamente (omisión).

- v

Incluye información de etiqueta para el archivo.

- **tag [opciones] tag [archivo ...]**

Agregua una etiqueta simbólica a una versión revisada de archivos. Sus opciones se especifican en anexos.

- **unedit [Opciones] [archivos ...]**

Desace un comando editor.

- a actions

Especifica acciones para observar temporalmente, en **action** está **edit, unedit, commit, all, o none**.

- l

Local; corre únicamente en el directorio de trabajo actual.

- R

Opera recursivamente (omisión).

- **update [opciones] [archivos ...]**

Trae un árbol de trabajo en sincronización con el depósito. Sus opciones se especifican en anexos.

- **watch [on|off|add|remove] [opciones] [archivo ...]**

on/ off: prende/apaga lecturas, únicamente de archivos chequeados. add/remove: agrega o quita, notificación sobre acciones.

- a actions

Especifica acciones para ver temporalmente, donde actions es **edit, unedit, commit, all, o none.**

- l

Local; corre únicamente en directorio de trabajo actual.

- R

Opera recursivamente (omisión).

- **watchers [opciones] [archivos ...]**

Ver quien está observando un archivo.

- l

Local; corre únicamente en el directorio de trabajo actual.

- R

Opera recursivamente (omisión).

4.3 MANUAL DE REFERENCIA PARA LOS FICHEROS ADMINISTRATIVOS

Dentro del depósito, en el directorio **$CVSROOT/CVSROOT,** hay un número de ficheros de apoyo para CVS. Se puede utilizar CVS en una manera limitada sin cualquiera de ellos, pero si se instalan correctamente pueden ayudarnos a hacer la vida más fácil. El más importante de estos ficheros es **modules clasifíe,** que define los módulos dentro del depósito.

4.3.1 EL FICHERO DE LOS MÓDULOS

El fichero **modules** registra sus definiciones de los nombres para las colecciones del código de fuente.

CVS utilizará estas definiciones si se utiliza CVS para poner al día el fichero de los módulos.

El fichero **modules** puede contener las líneas en blanco y los comentarios (líneas que comienzan con #') así como definiciones del módulo. Las líneas largas se pueden continuar en la línea siguiente especificando un backslash (\ ) como el carácter pasado en la línea.

Hay tres tipos básicos de módulos: alias módulos, módulos regulares, y módulos del signo "&". La diferencia entre ellos es la manera que asocian ficheros en el depósito a los ficheros en el directorio de funcionamiento. En todos los ejemplos siguientes, el depósito a nivel superior contiene un directorio llamado **first-dir**, que contiene dos ficheros, **file1** y **file2**, y un directorio **sdir**. **first-dir/sdir** contiene un fichero **sfile**.

4.3.1.1 ALIAS MÓDULOS

Alias módulos son la clase más simple de módulo:

mname - pseudónimos ...

Esto representa la manera más simple de definir un **mname** del módulo. Los indicadores **-a** la define como simple alias: CVS tratará cualquier uso del **mname** (como argumento del comando) como si la lista de los pseudónimos de los nombres hubiera sido especificada en lugar de otro. Los pseudónimos pueden contener otros nombres o los caminos del módulo. Cuando se utiliza los caminos en pseudónimos, la

comprobación crea todos los directorios intermedios en el directorio de funcionamiento, apenas como si el camino habría sido especificado explícitamente en los argumentos de CVS.

Por ejemplo, si el fichero de los módulos contiene:

amodule - first-dir

entonces los dos comandos siguientes son equivalentes:

cvs co first-dir de $ del amodule del co de los cvs de $

y cada uno proporcionarían la salida por ejemplo:

comprobación de los cvs: Puesta al día de la first-dir

comprobación de los cvs de U first-dir/file1 U first-dir/file2:

Puesta al día de first-dir/sdir U first-dir/sdir/sfile

4.3.1.2 MÓDULOS REGULARES

mname dir [Opciones] [Ficheros ...]

En el caso más simple, esta forma de definición del módulo reduce a **mname dir**. Esto define todos los ficheros en **dir** del directorio como **mname** del módulo. El **dir** es un camino relativo (de $$CVSROOT) a un directorio de la fuente en el depósito de la

fuente. En este caso, en la comprobación, un solo directorio llamado **mname** se crea como directorio de funcionamiento; no se utiliza ningún nivel intermedio del directorio por valor, por defecto, incluso si el **dir** era un camino que implicaba varios niveles del directorio.

Por ejemplo, si un módulo se define:

regmodule first-dir

Entonces el **regmodule** contendrá los ficheros a partir de **first-dir**:

cvs update cvs co regmodule

$: Puesta al día de la comprobación de los cvs de U regmodule/file1

U regmodule/file2 del regmodule: Puesta al día de regmodule/sdir U

regmodule/sdir/sfile $

Por los ficheros, especificar en la definición del módulo después del **dir**, se puede seleccionar ficheros determinados de **dir** del directorio. Aquí está un ejemplo:

regfiles first-dir/sdir sfile

Con esta definición, conseguir el módulo de los **regfiles** creará un solo directorio de funcionamiento **regfiles** contiene el fichero enumerado, que viene de un directorio más profundo en el depósito de la fuente de CVS:

regfiles U regfiles/sfile $ del co de los cvs de $

4.3.1.3 MÓDULOS DEL SIGNO "&"

Una definición del módulo puede referirse a otros módulos incluyendo **& module** en su definición.

mname [opciones] ... &module

Entonces conseguir el módulo crea un subdirectorio para cada módulo, en el directorio que contiene el módulo. Por ejemplo, si los módulos contienen

ampermod &first-dir

Entonces una comprobación creará un directorio del ampermod que contenga un directorio llamado **first-dir**, que en vueltas contiene todos los directorios y ficheros que viven allí. Por ejemplo:

$ cvs co ampermod

Creará los ficheros siguientes:

ampermod/first-dir/file1 ampermod/first-dir/file2

ampermod/first-dir/sdir/sfile

4.3.1.4 EXCEPTO DIRECTORIOS

Un módulo del pseudónimo puede excluir directorios determinados de otros módulos usando una marca de exclamación (!) antes del nombre de cada directorio que se excluirá.

Por ejemplo, si el fichero de los módulos contiene:

exmodule - un!first-dir/sdir primer-dir

entonces controlando fuera del módulo **exmodule** llegará fuera de todo **first-dir** excepto cualquier fichero en el subdirectorio **first-dir/sdir**.

4.3.1.5 OPCIONES DEL MÓDULO

Los módulos regulares o los módulos del signo **&** pueden contener las opciones, que proveen la información adicional referente al módulo.

- d name

Nombra al directorio de funcionamiento algo con excepción del nombre del módulo.

- e prog

Especifica un **prog** del programa para ejecutarse siempre que los ficheros en un módulo se exporten. El **prog** se ejecuta con un solo argumento, el nombre del módulo.

- i prog

Especifica un **prog** del programa para ejecutarse siempre que los ficheros en un módulo estén confiados. El **prog** se ejecuta con un solo argumento, el **pathname** completo del directorio afectado en un depósito de fuente.

- o prog

Especifica un **prog** del programa para ejecutarse siempre que los ficheros en un módulo se controlen hacia fuera. El **prog** se ejecuta con un solo argumento, el nombre del módulo.

- s status

Asigna un estatus al módulo. Cuando el fichero del módulo se imprime con **cvs checkout –s** los módulos se clasifican según el estatus del módulo, y segundo según el nombre del módulo. Esta opción no tiene ningún otro significado. Se puede utilizar esta opción para varias cosas además del estatus: por ejemplo, enumerar a la persona que es responsable de este módulo.

- t prog

Especifica un **prog** del programa para ejecutarse siempre que los ficheros en un módulo se marquen con etiqueta con **rtag**. Funcionamientos del **prog** con dos argumentos: el nombre del módulo y la etiqueta simbólica especificaran al **rtag**. No se ejecuta cuando se ejecuta la etiqueta. Se encontrará generalmente que el **taginfo** es una solución mejor.

- u prog

Especifica un **prog** del programa para ejecutarse siempre que **cvs update** se ejecute del directorio a nivel superior del módulo controlado -hacia fuera. El **prog** se ejecuta con un solo argumento, el camino al depósito de la fuente para este módulo.

4.3.2 EL FICHERO DE LOS cvswrappers

El fichero **cvswrappers** define la escritura que será ejecutada en un fichero cuando su nombre corresponde con una expresion regular. Hay dos escrituras que se pueden ejecutar en un fichero o un directorio. Una escritura se ejecuta en el **file/directory** antes de ser controlado en el depósito (esto se denota con indicador - t) y el otro cuando el fichero se controla fuera del depósito (esto se denota con el indicador de -f). Las opciones **-t /-f** no trabajan con CVS **client/server**.

Los **cvswrappers** también tienen una opción **-m** para especificar la metodología de la fusión que debe ser utilizada cuando un fichero no-binario es actualizado. La FUSIÓN significa el comportamiento generalmente de CVS: intente combinar los ficheros. COPIE los medios que la actualización de los cvs rechazará a los ficheros de fusión, como también hace para los ficheros especificados como binario con **-kb**. CVS proveerá del utilizador las dos versiones de los ficheros, y requiere al utilizador usando mecanismos fuera de CVS, para insertar cualquier cambio necesario.

ALERTA: No utilice la COPIA con CVS 1,9 o anterior -- tales versiones de CVS copiarán una versión del archivo sobre los otros, limpiando los contenidos previos. La opción **-m wrappers** únicamente afecta el comportamiento cuando la combinación es hecha en **update**; esto no afecta cuando los archivos son almacenados.

El formato básico del archivo **cvswrappers** es:

```
        wildcard    [option value][option value]...

where option is one of
- f       from cvs filter              value: path to filter
- t       to cvs filter                value: path to filter
- m       update methodology           value: MERGE or COPY
- k       keyword  expansion           value: espansion mode

       and value is a single-quote delimeted value.

       *.nib    - f 'unwrap %s ' - t 'wrap %s %s ' - m 'COPY'
       *.c      - t 'indent %s %s '
```

El ejemplo de arriba de un archivo **cvswrappers** dice que todos los archivos que terminan con un **.nib** deben filtrarse con el programa **wrap** antes de comprobar el archivo en el depósito. El archivo debería filtrarse aunque el programa **unwrap** cuando el archivo es chequeado fuera del depósito. El archivo **cvswrappers** también afirma que una metodología DE COPIA deberían ser usados cuando se estén actualizando los archivos en el depósito.

El último ejemplo de línea dice que al final todos los archivos con **.c** debería filtrarse con **indent** antes de verificar en el depósito. Diferente del ejemplo previo, ningún filtrador del archivo **.c** se ha hecho cuando éste es chequeado fuera del depósito. El filtrador **- t** es llamado con dos argumentos, el primero es el nombre del archivo/ directorio para filtrar y el segundo es el **pathname** a donde el archivo filtrado resultante debería ponerse.

El filtrador **- f** es llamado con un argumento, el cual es el nombre del archivo a filtrar. El resultado final de este filtro será un archivo en el directorio de usuarios que ellos pueden trabajar como ellos normalmente lo hacen.

Nótese que los aspectos **-t / -f** no manejan convenientemente una porción de operaciones de CVS'S: determinando cuando los archivos son modificados. CVS querrá todavía que un archivo (o directorio) exista, y usará este tiempo de modificación para determinar si un archivo es modificado. Si CVS erróneamente piensa un archivo es inmodificable, se puede forzar esto para chequear el archivo de cualquier manera especificando la opción **-f** de **cvs commit**.

Para otro ejemplo, el comando siguiente importa un directorio, tratando archivos cuyo nombre termina en **.exe** como binario:

cvs import - I ! - W "*.exe - k 'b'" first- dir vendortag reltag

4.3.2.1 EL commit SOPORTA ARCHIVOS

La opción **-I** en el archivo **Modules** puede usarse para correr un programa seguro cuando los archivos son guardados. Los archivos descritos en esta sección proveen otrss maneras más flexibles para correr programas cuando alguna cosa es guardada.

Hay tres clases de programas que pueden correrse sobre **commit**. Ellos son especificados en archivos en el depósito, como descritos más adelante. La siguiente parte resume el archivo nombrado y el propósito de los programas correspondientes.

commitinfo

El programa es responsable de estar chequiando que el **commit** sea permitido. Si este existe con una salida non - zero dice que el **commit** será abortado.

verifymsg

El programa especificado es utilizado para evaluar el mensaje de registro, y posiblemente verifica que este contenga todos los campos requeridos. Esta es muy útil en combinación con el archivo `rcsinfó, que puede retener una plantilla de mensaje de registro.

editinfo

El programa especificado es usado para editar el mensaje de registro, y posiblemente verificar que este contenga todos los campos requeridos. Esta es muy útil en

combinación con el archivo `rcsinfó, que puede retener una plantilla de mensaje de registro.

loginfo

El programa especificado es llamado cuando el commit está completo. Este recibe el mensaje de registro y alguna información adicional y puede almacenar el mensaje de registro en un archivo, o envía esto a personas apropiadas, o quizás pegarlo a un grupo de noticias local.

4.3.2.2 LA SINTAXIS COMÚN

Los archivos administrativos como **commitinfo, loginfo, rcsinfo, verifymsg,** etc., todos tienen un formato común. El propósito de los archivos son descritos despúes. La sintaxis común se describe aquí. Cada línea contiene lo siguiente:

- Esto es una expresión regular básica en la sintaxis usada por GNU emacs.
- Un whitespace separator—uno o más espacios y/o lengüetas.
- Un nombre de archivo o línea de comando de plantilla.

Las líneas vacías se ignoran. Estas líneas comienzan con el carácter #' son tratadas como comentarios. Las líneas largas desafortunadamente no pueden romperse en dos partes de ninguna forma. La primera expresión regular que equipara el nombre de directorio actual en el depósito es usado. El resto de la línea es usada como un nombre de archivo o línea de comando como apropiado.

4.3.3 commitinfo

El archivo **commitinfo** define programas para ejecutar cuando **cvs commit** está para ejecutarse. Estos programas son usados para chequeos pre - guardar para verificar que los archivos modificados, agregados y quitados son realmente preparados para ser guardados. Esto podría usarse, por ejemplo, para averiguar que los archivos cambiados estén conformes a los standares del local para la práctica de codificación.

Como se ha mencionado anteriormente, cada línea en el archivo **commitinfo** consiste de una expresión regular y una plantilla de línea. La plantilla puede incluir un nombre de programa y algún número de argumentos que se desea abastecerlo. La trayectoria completa a la actual fuente de depósito es añadido a la plantilla, seguido por los nombres de archivo de cualquier archivos involucrados en el commit (archivos agregados, quitados, y modificados).

La primera línea con una expresión regular que equipara la trayectoria relativa al módulo será usada. Si el comando devuelve a la salida non - zero nos dice que el **commit** será abortado.

Si el nombre de depósito no equipara ninguna de las expresiones regulares en este archivo, la línea **DEFAULT** es usada, si esta es especificada.

Todas las ocurrencias del nombre **ALL** aparecen como una expresión regular que son usadas en adición al primer conjunto de expresión regular o el nombre **DEFAULT**.

Nota: Cuando CVS esta accediendo a un depósito remoto, **commitinfo** debe ser corrido en el lado remoto (es decir, servidor), no en el lado del cliente.

4.3.4 VERIFICANDO LOS MENSAJES TOTALES

Una vez que se ha entrado en un mensaje de registro, se puede evaluar ese mensaje para chequear su contenido específico, tal como un virus identificado (ID). Se usa el archivo **verifymsg** para especificar un programa que es usado para averiguar el mensaje de registro. Este programa podría ser un manuscrito simple que verifica que la entrada del mensaje contiene los campos requeridos.

El archivo **verifymsg** es frecuentemente muy útil junto con el archivo **rcsinfo,** que puede usarse para especificar una plantilla de mensaje de registro. Cada línea en el archivo **verifymsg** consiste de una expresión regular y una plantilla de línea de comando. La plantilla debe incluir un nombre de programa, y puede incluir cualquier número de argumentos. La trayectoria completa de la actual plantilla de mensaje de registro el archivo es añadido a la plantilla.

Una de las cosas que debe notarse es que la palabra clave **ALL** no es apoyada. Si más de un conjunto de línea se encuentra, la primera línea es usada. Esto puede ser útil para especificar un manuscrito de comprobación por omisión en un módulo, y entonces sobremontarlo en un subdirectorio.

Si el nombre de depósito no equipara ninguna de las expresiones regulares en este archivo, la línea **DEFAULT** es usada, si esta es especificada.

Si el manuscrito de comprobación sale con una salida non - zero nos dice que el commit es abortado.

Nótese que el manuscrito de comprobación no puede cambiar el mensaje de registro; esto puede meramente aceptarlo o rechazarlo.

El siguiente es un pequeño ejemplo de un archivo **verifymsg**, junto con el correspondiente archivo **rcsinfo**, la plantilla de mensaje de registro y un manuscrito de comprobación. Nosotros comenzamos con la plantilla de mensaje de registro. Nosotros queremos siempre registrar un **virus - id** de número sobre la primera línea del mensaje de registro. El resto del mensaje de registro es el texto libre. La plantilla siguiente se encuentra en el archivo **/usr/cvssupport/tc.template**.

BugId:

El manuscrito **/usr/cvssupport/bugid.verify** se usa para evaluar el mensaje de registro.

```
#!/bin/sh
#
#     bugid.verify filename
#
# Verify that the log message contains a valid bugid
# on the first line.
#
if head -1
<
```

```
$1 | grep '^BugId:[ ]*[0-9][0-9]*$'
>
/dev/null; then
   exit 0
else
   echo "No BugId found."
   exit 1
fi
```

The `verifymsg' file contains this line:

```
^tc    /usr/cvssupport/bugid.edit
```

The `rcsinfo' file contains this line:

```
^tc    /usr/cvssupport/tc.template
```

4.3.5 editinfo

La apariencia **editinfo** se ha dado como obsoleto. Para colocar un editor por defecto para mensajes de registro usa la variable de ambiente EDITOR o la opción global **-e**.

Si se quiere asegurar que todos los mensajes de registro miran de la misma manera, se puede usar el archivo `editinfó para especificar un programa que es usado para editar el mensaje de registro. Este programa podría ser un hábito - hecho editor que siempre impone un estilo seguro del mensaje de registro, o quizá una coraza de manuscrito simple de que llama un editor, y verifica que el mensaje entero contenga los campos requeridos.

Si ningún conjunto de línea se encuentra en el archivo **editinfo**, en lugar de esto el editor especifica en la variable de ambiente **$CVSEDITOR** sea usada. Si esta variable no es colocada, en lugar de esta entonces la variable de ambiente **$ EDITOR** es usada.

Si esta variable no es colocada, una omisión será usada.

El archivo **editinfo** es frecuentemente muy útil junto con el archivo **rcsinfo**, que puede usarse para especificar una plantilla de mensaje de registro.

Cada línea en el archivo **editinfo** consiste de una expresión regular y una plantilla de línea de comando. La plantilla debe incluir un nombre de programa, y puede incluir cualquier número de argumentos.

Una de las cosas que debe notarse es que la palabra clave **ALL** no es apoyada. Si más de un conjunto de línea es encontrada, la primera línea es usada.

Esto puede ser útil para especificar una omisión del manuscrito editado en un módulo, y entonces sobreescribirlo en un subdirectorio.

Si el nombre de depósito no equipara ningunas de las expresiones regulares en este archivo, la línea DEFAULT es usada, si es especificada. Si la edición del manuscrito existe con una salida non - zero, nos dice que el guardar es abortado.

Nota: cuando CVS accede a un depósito remoto, o cuando las opciones **-m** o **-F** de cvs **commit** son usadas, **editinfo** no se consultará. No hay buen ambiente de trabajo para esto; use en su lugar **verifymsg**.

4.3.5.1 EJEMPLO DE editinfo

Lo siguiente es un ejemplo grotesco de un archivo **editinfo,** junto con el correspondiente archivo **rcsinfo**, la plantilla de mensaje de registro y un manuscrito de editor. Nosotros comenzamos con la plantilla de mensaje de registro. Nosotros queremos siempre registrar un **virus - id** de número sobre la primera línea del mensaje de registro. El resto de mensaje de registro es el texto libre. La plantilla siguiente se encuentra en el archivo **/usr/cvssupport/tc.template**.

BugId:

El manuscrito **/usr/cvssupport/bugid.edit** se usa para editar el mensaje de registro.

```
#!/bin/sh
#
#     bugid.edit filename
#
# Call $EDITOR on FILENAME, and verify that the
# resulting file contains a valid bugid on the first
# line.
if [ "x$EDITOR" = "x" ]; then EDITOR=vi; fi
if [ "x$CVSEDITOR" = "x" ]; then CVSEDITOR=$EDITOR; fi
$CVSEDITOR $1
until head -1|grep '^BugId:[ ]*[0-9][0-9]*$'
<
```

```
$1
do  echo -n  "No BugId found.  Edit again? ([y]/n)"
   read ans
   case ${ans} in
      n*) exit 1;;
   esac
   $CVSEDITOR $1
done
```

The `editinfo' file contains this line:

```
^tc    /usr/cvssupport/bugid.edit
```

The `rcsinfo' file contains this line:

```
^tc    /usr/cvssupport/tc.template
```

4.3.6 loginfo

El archivo **loginfo** se usa para controlar donde **cvs commit** envía información de registro. La primera entrada sobre una línea es una expresión regular que se prueba contra el directorio que los cambios están siendo hechos, para referirse al **$CVSROOT**.

Si un conjunto se encuentra, entonces lo restante de esta línea es un programa de filtro que debe esperar la información total sobre su aporte estándard. Si el nombre de

depósito no equipara ningunas de las expresiones regulares en este archivo, la línea DEFAULT es usada, si es especificada.

Todas las ocurrencias del nombre **ALL** aparece como una expresión regular que son usadas en adición al primer conjunto de expresión regular o DEFAULT. El primer conjunto de expresión regular es usado.

El usuario puede especificar un formato lineal como parte del filtro. La línea está compuesta de un **%** seguida por un espacio, o seguida por un carácter simple de formato, o seguida por un conjunto de caracteres de formato rodeados por { y } como separadores. Los caracteres de formato son:

s

Nombre de archivo

V

Número de versión antigua (chequeo anterior)

v

Número de versión nueva (chequeo posterior)

Todos los otros caracteres que aparecen en una línea de formato expande para un campo vacío.

Por ejemplo, algunas líneas de formato válidas son **%, %s, %{s}, y %{sVv}** .

El rendimiento será una línea de símbolos separados por espacios. Para la compatibilidad atrasada, el primero en venir será el nombre del depósito. El resto de los traídos será una lista delimitada por comas de la información requerida en la cadena de formato. Por ejemplo, si **/u/src/master** es el depósito, **%{sVv}** es la línea de formato, y tres archivos (ChangeLog, Makefile, foo.c) son modificados, el rendimiento podría ser:

/ u/src/master ChangeLog,1.1,1.2 Makefile,1.3,1.4 foo.c,1.12,1.13

Como otro ejemplo, **%{}** significa que solamente el nombre del depósito se generará.

4.3.6.1 EJEMPLO DE loginfo

El siguiente archivo **loginfo**, junto con una leve coraza de manuscrito más adelante, añade todos los mensajes de registro al archivo **$CVSROOT/CVSROOT/commitlog,** y guardar cualquiera de los archivos administrativos (dentro del directorio CVSROOT) son registrados también en **/usr/adm/cvsroot-log**. Guarda al directorio **prog1** y envía a ceder .

```
ALL             /usr/local/bin/cvs-log $CVSROOT/CVSROOT/commitlog $USER
^CVSROOT        /usr/local/bin/cvs-log /usr/adm/cvsroot-log
^prog1          Mail -s %s ceder
```

The shell-script `/usr/local/bin/cvs-log' looks like this:

```
#!/bin/sh
(echo "----------------------------";
 echo -n $2"  ";
 date;
 echo;
 cat)
>
>
 $1
```

4.3.6.2 MANTENIENDO UNA COPIA REVISADA

Esto es frecuentemente útil para mantener un árbol de directorio que contiene los archivos que corresponden a la última versión en el depósito.

La manera para hacer esto es habiendo invocado a **loginfo cvs update**. Haciendo así en la manera simple causará un problema con las seguridades, entonces el **cvs update** debe ser corrido en el fondo. Aquí esta un ejemplo para unix (esto debería estar sobre una línea):

```
^cyclic-pages           (date; cat; (sleep 2; cd
/u/www/local-docs;
 cvs -q update -d)
&
)
>
>
```

```
 $CVSROOT/CVSROOT/updatelog 2
>
&
1
```

Esto ocasionará chequeos a los directorios del depósito que comienzan con **cyclic–pages** para actualizar el árbol chequeado en **/u/www/local - docs**.

4.3.7 rcsinfo

El archivo **rcsinfo** puede usarse para especificar una forma para editar cuando se llenó el registro guardado. El archivo **rcsinfo** tiene una sintaxis similar a los archivos **verifymsg, commitinfo** y **loginfo**. Diferente a los otros archivos la segunda parte no es una línea de comando de plantilla. En lugar de esto, la siguiente parte de la expresión regular debería ser un pathname completo del archivo que contiene la plantilla de mensaje de registro.

Si el nombre de depósito no equipara ninguna de las expresiones regulares en este archivo, la línea DEFAULT es utilizada, si esto es especificado.

Todos las ocurrencias del nombre **ALL** aparecen como una expresión regular que son usadas al primer conjunto de expresión regular o **DEFAULT**.

La plantilla de mensaje de registro se usará como un mensaje de registro por defecto. Si se especifica un mensaje de registro con **cvs commit - m message** o **cvs commit - f file** el mensaje total sobremontará la plantilla.

Cuando CVS accede a un depósito remoto, los contenidos de **rcsinfo** en el momento en que un directorio es primero chequeado especificará una plantilla la cual entonces no cambia. Si el editor **rcsinfo** o esas plantillas, se puede necesitar chequear un nuevo directorio de trabajo.

4.3.8 IGNORANDO ARCHIVOS POR MEDIO DE cvsignore

Hay nombres seguros de archivo que frecuentemente ocurren dentro la copia de trabajo, pero que si no se quiere poner debajo del control de CVS. Los ejemplos son todos los archivos de objeto que se consigue mientras se compila las fuentes. Normalmente, cuando se corre **cvs update**, este imprime una línea para cada archivo este encontrará que esto no se conoce.

CVS tiene una lista de archivos (o un molde de nombres de archivo sh(1)) que debería ignorar mientras esta corriendo **update, import** y **release**. Esta lista se construye de la siguiente manera.

- La lista es inicializada para incluir ciertos modelos de nombre de archivo: nombres asociados con administración de CVS, o con otros sistemas comunes de control de fuente; los nombres comunes para archivos parche, archivos objeto, archivos archivados, y archivos de respaldo del editor; y otros nombres que son comúnmente artefactos de utilidades variadas. Actualmente, la lista de omisión de modelos de nombre de archivo ignorado es:

- RCS SCCS CVS CVS.adm
- RCSLOG cvslog.*
- tags TAGS
- .make.state .nse_depinfo
- *~ #* .#* ,* _$* *$
- *.old *.bak *.BAK *.orig *.rej .del-*
- *.a *.olb *.o *.obj *.so *.exe
- *.Z *.elc *.ln
- core

- La lista del pre - depósito en el **$CVSROOT/CVSROOT/cvsignore** es añadido a la lista, si este archivo existe.
- La lista del pre - usuario en **.cvsignore** en el directorio de casa es añadido a la lista, si este existe.
- Cualquier entrada en el variable de ambiente **$CVSIGNORE** es añadido a la lista.
- Cualquier opción **-I** dadas por CVS es añadido.
- Como CVS cruza mediante sus directorios, los contenidos de cualquier **.cvsignore** serán añadidos a la lista. Los modelos encontrados en **.cvsignore** son solamente válidos para el directorio que los contiene, no para ningún sub - directorio.

En cualquiera de los 5 lugares enumerados arriba, una sola exclamación marca (!) aclara la lista ignorada. Esto puede usarse si se quiere almacenar cualquier archivo que normalmente es ignorado por CVS.

Especificar **-I !** para **cvs import** importará todo, que es generalmente lo que se quiere hacer si se importa archivos desde una distribución o cualquier otra fuente que se

conoce que no contiene ningún archivo extraño. Sin embargo, mirando las reglas de arriba se puede ver que hay un vuelo en el argumento; si la distribución contiene algún archivo **.cvsignore**, entonces los modelos de estos archivos serán procesados aún cuando **-I !** se especifique. Solamente el área de trabajo es removida a los archivos **.cvsignore** en orden para hacer el importe.

Nótese que la sintaxis de los archivos ignore consisten de una serie de líneas, cada una de las cuales contiene una lista de espacio separado de nombres de archivo. Esto no ofrece una vía limpia para especificar los nombres de archivo los cuales contienen espacios, pero se puede usar un área de trabajo como **foo? bar** para conjugar un archivo llamado **foo bar.**

4.3.9 EL ARCHIVO history

El archivo **$CVSROOT/CVSROOT/ history** es usado para registrar información del comando **history**. Este archivo debe ser creado para que funciones en **login**. Esto es hecho automáticamente si el comando **cvs init** es usado para establecer el depósito.

El formato de archivo del archivo **history** se documenta unicamente en comentarios en el CVS de código original, pero generalmente deberían usarse programas del comando **cvs history** para accesar este de cualquier manera.

4.3.10 EXPANSIÓN EN ADMINISTRACIÓN DE ARCHIVOS

A veces al escribir un archivo administrativo, se desearía que el archivo sea capáz de conocer diversas cosas basado en el ambiente del CVS que está corriendo. Hay varios mecanismos para hacer esto.

Para encontrar el directorio en casa del usuario que corre CVS (desde la variable de medio ambiente HOME), el usar ~ seguido por / o el fin de la línea. Asimismo para el directorio de casa del usuario, usar ~**user**.

Estas variables son expandidas sobre la máquina de servidor, y no consiguen ninguna expansión razonable si **pserver** está en uso; por lo tanto variables de usuario puede ser una elección mejor para personalizar el comportamiento basado en el usuario que corre CVS.

Uno puede querer saber sobre diversas partes de información interna de CVS. Una variable interna de CVS tiene la sintaxis **${ variable }**, donde comienza la variable con una carta y consiste en caracteres alfanuméricos y _ . Si el carácter sigue a la variable es un caracter no-alfanumérico a excepción de _ , y { y } puede omitirse. Las variables internas de CVS son:

CVSROOT

Este es el valor de la raíz de CVS en uso.

RCSBIN

En CVS 1.9.18 y más antiguo, este especificó el directorio donde CVS buscaba el programa RCS.

CVSEDITOR , VISUAL, EDITOR

Todos estos expanden al mismo valor, que es el editor que CVS está usando.

USER

Nombre de usuario del usuario que está corriendo CVS (sobre la máquina del servidor de CVS).

Si se quiere pasar un valor a los archivos administrativos que el usuario quien está corriendo CVS puede especificar, usar una variable de usuario. Para expandir una variable de usuario, el archivo administrativo contiene **${= variable }**. Para colocar una variable de usuario, especificar la opción global **-s** a CVS, con el argumento variable = value. Este puede ser particularmente útil para especificar esta opción por medio de **.cvsrc**.

Por ejemplo, si se quiere que el archivo administrativo se refiera al directorio de prueba se podría crear una variable de usuario TESTDIR. Entonces si CVS se invocó como:

cvs - s TESTDIR=/work/local/tests

y el archivo administrativo contiene **sh ${=TESTDIR}/runtests**, entonces esta cadena se expande a **sh /work/local/test/runtests.**

Todas otras cadenas que contienen $ son reservadas; no hay manera para cotizar un carácter $ para que $ represente esto mismo.

4.3.11 CVSROOT/config Configuración De Archivo

El archivo administrativo **config** contiene diversas miscelaneas instaladas las cuales afectan el comportamiento de CVS. La sintaxis es ligeramente diferente de otros archivos administrativos. Las variables no son expandidas. Líneas que comienzan con # son consideradas comentarios. Otras líneas consisten de una palabra clave =, y un valor.

Nótese que esta sintaxis es muy estricta. Las lengüetas o espacios extraños no son permitidos. Actualmente las palabras claves definidas son:

RCSBIN= bindir

Para CVS 1.9.12 mediante 1.9.18, se instala diciendo al CVS que busque programas RCS en el directorio **bindir**. Las versiones actuales de CVS no corren programas de RCS; para que compatiblemente esta instalación sea aceptada, pero esto no hace nada.

SystemAuth= value

Si **value** está yes , entonces **pserver** debería verificar para usuarios en la base de datos del sistema de usuarios, si no se establece en **CVSROOT/passwd**. Si esto es no,

entonces todos los usuarios de **pserver** deben existir en **CVSROOT/passwd**. La omisión es yes.

PreservePermissions= value

Permite apoyar para guardar un archivo de dispositivo especial, enlace simbólico, permisos de archivo y propiedad en el depósito. El valor por omisión es no.

TopLevelAdmin= value

Modifica el comando **checkout** para crear un directorio CVS a un nivel alto del nuevo directorio de trabajo, además de directorios CVS creados dentro de directorios verificados. El valor por omisión es no. Esta opción es útil si se encuentra desempeñando muchos comandos a un nivel alto del directorio de trabajo, más bien que en uno de los chequeos fuera de subdirectorios. El directorio CVS creado allí significará que no tiene que especificar **CVSROOT** para cada comando. Si también provee un lugar para el archivo **CVS/Template**.

4.4 TODAS LAS VARIABLES DE ENTORNO QUE AFECTAN A CVS

Esta es una lista completa de todas las variables de entorno que afecten a CVS.

$$CVSIGNORE

Una lista whitespace-separada de los modelos del nombre de archivo de que CVS no debe hacer caso.

$$EDITOR, $$CVSEDITOR

Especifica el programa para utilizar los mensajes del registro de la grabación durante esto se confía. **$$CVSEDITOR** reemplaza **$$EDITOR**.

Si **$$RCSBIN** no se fija, y no se compila ningún camino en CVS, entonces lo utilizará para intentar encontrar todos los programas y aplicaciones.

$$HOME, $$HOMEPATH, $$HOMEDRIVE

Localizan el directorio donde **.cvsrc** clasifica, y se buscan otros ficheros. En Windows NT, el sistema fijará HOMEDRIVE, por ejemplo a **d:** y HOMEPATH, por ejemplo a **\joe**. En Windows 95, se necesitará probablemente fijar HOMEDRIVE y HOMEPATH

$$CVS_RSH

Especifica el programa externo con el cual CVS conecta, cuando **:ext:** especifica el método de acceso.

$$CVS_SERVER

Se lo utiliza en modo de servidor de cliente al tener acceso a un depósito alejado usando RSH. Especifica el nombre del programa para comenzar en la cara del servidor al tener acceso a un depósito alejado usando RSH. El valor por defecto es cvs .

$$CVS_PASSFILE

Se lo utiliza en modo de servidor de cliente al tener acceso al servidor de la conexión de los cvs . El valor por defecto es **$HOME/.cvspass**.

$$CVS_CLIENT_PORT

Se lo utiliza en modo de servidor de cliente al tener acceso al servidor vía Kerberos.

$$CVS_RCMD_PORT

Se lo utiliza en modo de servidor de cliente. Si está fijado, especifica el número de acceso que se utilizará al tener acceso al RCMD en la cara del servidor. (no utilizado actualmente para los clientes de Unix).

$$CVS_CLIENT_LOG

Se lo utiliza para poner a punto solamente en modo del servidor de cliente. Si está fijada, todo envía al servidor se registra en **in $$CVS_CLIENT_LOG** y todo envía del servidor se registra en **out $$CVS_CLIENT_LOG**.

$$CVS_SERVER_SLEEP

Utilizado solamente para poner a punto la cara del servidor en modo del servidor de cliente. Si está fijado, se tendrá retardos al comienzo del proceso del servidor la cantidad especificada en segundos de modo que se pueda asociar a ella con un depurador.

$$COMSPEC

Utilizado bajo OS/2 solamente. Especifica el nombre del intérprete del comando y omite CMD.EXE.

MPDIR, MP, EMP

Directorio en el cual los ficheros temporales están situados. El servidor de CVS utiliza TMPDIR . Algunas partes de CVS utilizarán siempre **/tmp**' (vía la función del tmpnam proporcionada por el sistema). En Windows NT, se utiliza TMP.

El programa de la corrección que es utilizado por el cliente de CVS utiliza TMPDIR , y si no se fija, lo utiliza **/tmp**. Observe que si su servidor y cliente son CVS que se ejecuta 1,9,10 o más adelante, CVS no invocará un programa externo de la corrección.

4.5 COMPATIBILIDAD ENTRE LAS VERSIONES DE CVS

El formato del depósito es el ir compatible de nuevo a CVS 1.3. Si se hace que las copias de CVS 1,6 o más antiguas y se desea utilizar las características opcionales de la comunicación del revelador.

El formato del directorio de funcionamiento es el ir compatible de nuevo a CVS 1.5. Cambió entre CVS 1.3 y CVS 1.5. Si se ejecuta CVS 1.5 o una versión más reciente en un directorio de funcionamiento controlado hacia fuera con CVS 1.3, CVS lo convertirá, pero ir de nuevo a CVS 1.3 que se necesita controlar fuera de un nuevo directorio de funcionamiento con CVS 1.3.

El protocolo alejado es el ir interoperable de nuevo a CVS 1.,5. pero no más lejos (1.5 era el primer desbloqueador oficial con el protocolo alejado, pero algunas versiones más antiguas pudieron todavía flotar alrededor). Sin embargo en muchos casos se necesita aumentar el cliente y el servidor para aprovecharse de nuevas características y bugfixes .

4.6 LOCALIZACIÓN DE AVERÍAS

4.6.1 LISTA PARCIAL DE LOS MENSAJES DE ERROR

Aquí está una lista parcial de los mensajes de error que se puede ver de CVS. No es un lista CVS completa pero es capaz de imprimir muchos mensajes de error, con las partes que ellas dan a menudo por el sistema operativo, pero la intención es enumerar el campo común y/o los mensajes de error potencialmente confusos.

Los mensajes son alfabéticos, pero el texto introductorio por ejemplo **cvs update:** no se considera en pedirlos. En algunos casos la lista incluye los mensajes impresos por viejas versiones de CVS.

comando de los cvs : la autorización falló: acceso rechazado ordenador principal del servidor:

Esto es una respuesta genérica al intentar conectar con un servidor del pserver que elija no proporcionar una razón específica para negar la autorización. Controle que el

username y la palabra de paso especificada estén correctos y que el **CVSROOT** especificado esté permitido **inetd.conf root**.

fichero : línea : La aserción ' texto ' falló

El formato exacto de este mensaje puede variar dependiendo de su sistema.

comando de los cvs : conflicto: el fichero quitado fue modificado por el segundo partido

Este mensaje indica que usted quitó un fichero, y algún otro lo modificó. Para resolver el conflicto, lo primero que se debe hacer es **cvs add file**. Si es que desea, mirar la otra modificación para decidir si todavía desea quitarla. Si no se desea quitarla, entonces pare aquí. Si desea quitarla, proceda con **cvs remove file** y confirme su retiro.

No puede cambiar permisos en directorio temporal

Operación no permitida

Este mensaje ha estado sucediendo de una manera no reproducible, ocasionalmente cuando ejecutamos el testsuite client/server, en Linux 3,0,3 y 4,1. No se ha podido calcular fuera de qué causas, (o si es en una sola máquina determinada!). Si el problema ocurre en otros unix, **Operation not permitted** probablemente se debe leer **Not owner** o lo que el sistema en la pregunta utiliza para el error de **unix EPERM**.

No puede abrir CVS/cEntries para la lectura: Ningún tal fichero o directorio

Esto indica generalmente un error interno de CVS, y se puede dirigir como con otros fallos de funcionamiento de CVS. Generalmente hay un workaround la naturaleza exacta de la cual dependería de la situación pero de la cual esperanzadamente podría ser calculado hacia fuera.

cvs [init abortado]: no puede abrir VCS/cRoot: Ningún tal fichero o directorio

Este mensaje es inofensivo. Con tal que no sea acompañado por otros errores, la operación ha terminado con éxito. Este mensaje no debe ocurrir con versiones actuales de CVS, sino que se lo indica aquí para la ventaja de CVS 1,9 y más antiguo.

cvs [comprobación abortada]: no puede retitular el fichero del fichero a CVS /, al fichero : Argumento inválido

Este mensaje ha estado intermitentemente sucediendo con CVS 1,9 en Solaris 2,5. La causa es desconocida.

cvs [comando abortado]: no puede encender el servidor vía rcmd

Este, desafortunadamente, es un mensaje de error algo no específico que CVS 1.9 imprimirá si usted está ejecutando al cliente de CVS. Las versiones actuales de CVS deben imprimir un mensaje de error mucho más específico.

ci: fichero ,v: mala línea de salida del diff: Ficheros binarios - y / tmp/T2a22651 diferencia

CVS 1,9 o una versión más vieja este mensaje tratará de llegar a un fichero binario si RCS no está instalado correctamente.

Comprobación de los cvs: no podía controlar fuera de fichero

Con CVS 1,9, puede significar que el programa **co** (parte de RCS) volvió un incidente. Debe ser precedida por otro mensaje de error, no obstante se ha observado sin otro mensaje de error y la causa bien no se entiende. Con la versión actual de CVS, que no ejecuta el co , si este mensaje ocurre sin otro mensaje de error, él es definitivamente un fallo de funcionamiento de CVS.

cvs [conexión abortada]: no podía descubrir el directorio casero

Esto significa que usted necesita fijar las variables de entorno que CVS utiliza, localizar su directorio casero.

Actualización de los cvs: no podía combinar la revolución de la revisión del fichero : Ningún tal fichero o directorio

En CVS 1,9 y en una versión más antigua este mensaje era un problema que encontraba el programa del rcsmerge.

Cerciórese de que esté en su path , o auméntelo a una versión actual de CVS, que no requiere un programa externo del rcsmerge.

cvs [actualización abortada]: no podía parchear el fichero : Ningún tal fichero o directorio

Esto significa que había un problema que encontraba el programa de la corrección. Cerciórese de que esté en su path. Observe que a pesar de estos aspectos el mensaje no se está refiriendo si puede encontrar el fichero. Si el cliente y el servidor está ejecutando una versión actual de CVS, no hay necesidad de un programa externo de la corrección y no se debe temer de este mensaje.

Actualización de los cvs: no podía parchear el fichero ; refetch de la voluntad

Esto significa que para cualquier razón el cliente no podía aplicar una corrección que el servidor envió. El mensaje no es nada tratado alrededor, porque la inhabilidad de aplicar la corrección retarda solamente algunas cosas y no tiene ningún efecto en lo que hace el CVS.

Fallo del servidor inesperado

Hay un fallo de funcionamiento conocido en el servidor para CVS 1,9,18 y más antiguo que pueda causar esto. Esto puede producir al utilizar la opción global -t. Fue

fijada por el cambio de Andy Piper el 14 de nov de 1997 a **src/filesubr.c**. Si el mensaje aparece , se puede apenas probablemente revisar la operación que falló.

Extremo del fichero del servidor

La causa más común para este mensaje es si se está utilizando un programa externo del rsh y se produjo un error. En este caso el programa del rsh debe haber impreso un mensaje, que aparecerá antes del mensaje antedicho.

Los cvs confían: Ejecutar mkmodules

Esto significa que su depósito está instalado para una versión de CVS antes de CVS 1.8. Al usar CVS 1,8 o más adelante, el mensaje antedicho será precedido de:

Los cvs confían: Reconstrucción de la base de datos del fichero administrativo

Si se prodece ambos mensajes, la base de datos se está reconstruyendo dos veces, que es innecesaria pero inofensiva. Si se desea evitar la duplicación, y usted no tiene ninguna versión de CVS 1.7 o anterior en uso, borre - **i mkmodules** en cada lugar que aparece en su fichero de los módulos.

Autor que falta

Esto puede suceder típicamente si se creó un fichero de RCS con su username fijado para vaciar.

La voluntad de CVS, crea un fichero ilegal de RCS sin el valor para el campo del autor. La solución es cerciorarse de que su username está fijado a un valor no vacío y reconstruir el fichero de RCS.

Lo importante es que hay un directorio llamado CVS pero no contiene los archivos administrativos que el CVS colocó en un directorio de CVS. El problema es que se creó un directorio de CVS por un mecanismo algo diferente del CVS, entonces la respuesta es simple, utilice otro nombre que CVS. Si no, esto indica un CVS virus.

Rcs error: Opción desconocida: - x, v/

Este mensaje será seguido por un uso del mensaje para RCS. Significa que se tiene una versión vieja de RCS (probablemente suministrada con su sistema operativo). El CVS sólo trabaja con RCS versión 5 y despúes de esta.

Cvs (server aborted): recibió signo de la conexión caida

Este mensaje parece ser causado por un virus atrapado fuerte en el CVS o en los sistemas,. Esto parece pasar sólo después de que el comando de CVS se ha completado, y se podría sólo ignorar el mensaje.

¡Too many arguments ! (¡Demasiados argumentos!)

Se imprime este mensaje típicamente por la escritura **log.pl** que está en el directorio del **contrib** en la fuente de distribución del CVS.

En algunas versiones del CVS, el **log.pl** ha sido parte de una falla en la instalación del CVS. La escritura **log.pl** es llamado desde el archivo administrativo **loginfo.** Revisar que los argumentos aprobados en el **loginfo** hagan juego con la versión de **log.pl**. En particular, el **log.pl** del CVS 1.3 y los más viejos esperan que el archivo **logfile** como un argumento considerando que el **log.pl** del CVS 1.5 y los más nuevos esperan que este sea especificado con una opción **-f**. Por supuesto, si no requiere log.pl se puede sólo comentarlo fuera del **loginfo**.

cvs commit: Arriba-de la-fecha verifica la falla por archivo

Esto significa que alguién más ha realizado un cambio desde la última vez que realizo una actualización del cvs. Así antes del procedimiento con el **cvs commit** se requiere una actualización del cvs. El CVS unirá los cambios que se realizaron y los cambios que la otra persona hizo. Si estos no se detectan algún conflicto se reportará en **M cacErrCodes.h** y se estará listo para que el cvs se encargue. Si descubre conflictos estos imprimirá un mensaje así, **C cacErrCodes.h**, y se requiere resolver manualmente el conflicto.

Usage : diff3 [exEX3 [- i| - m] [L label1- L label3]] file1 file2 file3 (solamente uno de los [exEX3] es permitido)

Esto indica un problema con la instalación del **diff3** y **rcsmerge**. Específicamente el rcsmerge se compiló para buscar GNU **diff3**, pero este está encontrando es su lugar **unix diff3**. El texto exacto del mensaje variará dependiendo del sistema. La solución

más simple es actualizar a una versión presente de CVS, que no cuenta con rcsmerge externo o programas **diff3**.

Warning: respuestas irreconocibles " text ' desde el servidor del cvs

Si el texto contiene una respuesta válida seguido por un carecter de retorno extra (en muchos sistemas esto permitirá que la segunda parte del mensaje sobreescriba la primera parte), entonces probablemente significa que se está utilizando el método de acceso : **ext:** con una versión de rsh, las cuales por omisión no proveen datos transparentes. En tales casos probablemente se querrá tratar : **server:** en lugar de : **ext:**. Si el texto es es alguna otra cosa, esto significaría un problema con el servidor del CVS. Vuelva a chequear la instalación con las instrucciones para instalar el servidor del CVS.

CVS commit: Warning: (advertencia) falla en la sesión del editor

Esto significa que el editor que el CVS está utilizando salidas con un estatus de salida cero. Algunas versiones de **vi** , harán esto aún cuando no haya problemas con el archivo de edición. Si es así, señala el ambiente variable CVSEDITOR para una pequeña escritura como:

```
# !/ bin/ sh
vi $*
exit 0
```

4.6.2 PROBLEMAS PARA REALIZAR UNA CONEXIÓN A UN SERVIDOR DEL CVS

Esta sección concierne qué hacer si tiene un problema para hacer una conexión a un servidor del CVS. Si se está corriendo la línea de comandos cvs el cliente de la línea Windows, primero se clasifica al cliente al CVS 1.9.12 o versiones siguientes. El error reportado en las primeras versiones provee mucha menos información sobre cuál fue el problema. Si el cliente no está en Windows, el CVS 1.9 debería ser bueno.

Si los mensajes de error no son suficientes para perrseguir el problema, los próximos pasos dependen principalmente en cuales métodos de acceso se utilizarán.

: ext:

Tratar de correr el programa **rsh** desde la línea de comando. Por ejemplo: **rsh servername cvs- v** debería imprimir la información de la versión de CVS. Si éste no trabaja, se necesita arreglarlo antes de que se preocupe por los problemas del CVS.

: server:

No se necesita una línea de comando del programa **rsh** para utilizar este método de acceso, pero si se tiene un programa **rsh** alrededor, este podría ser útil como una herramienta de la puesta a punto de programa. Siguiendo las direcciones dadas por : **ext:**

: pserver:

Una buena herramienta de la puesta a punto de programa es **telnet servername 2401**. Después de conectado, envía cualquier texto (por ejemplo foo seguido por retorno). Si CVS trabaja correctamente, responderá con

Cvs pserver aborted: bat auth protocol start: foo

Si éste falla en el trabajo, entonces asegurese que el inetd esté trabajando correctamente. Cambie la invocación en inetd.conf para correr el programa echo instalado por el cvs. Por ejemplo:

2401 stream tcp nowait root /bin/ echo echo hello

Después de hecho el cambio e instruir al inetd para re-leer su archivo de configuración, el "telnet servername 2401" debe mostrarle el texto hola y entonces el servidor debe cerrar la conexión. Si éste no trabaja, necesita arreglarlo antes de que se preocupe por problemas del CVS. En el sistema AIX, el sistema a menudo tendrá su propio programa que trata de usar el puerto 2401. Ésto es que el problema de AIX en el sentido de que el puerto 2401 es registrado para utilizarlo con CVS.

CAPITULO V

INSTALACIÓN E IMPLANTACIÓN

5.1. INSTALACIÓN E IMPLANTACIÓN DE UN CVS EN WINDOWS NT

La instalación e implementacion de un Sistema de Control de Versiones para la P.U.C.E.S.A, tiene por objetivo que la Universidad promueva y desarrolle sus proyectos bajo el CVS, logrando de esta manera tener un verdadero control sobre los proyectos.

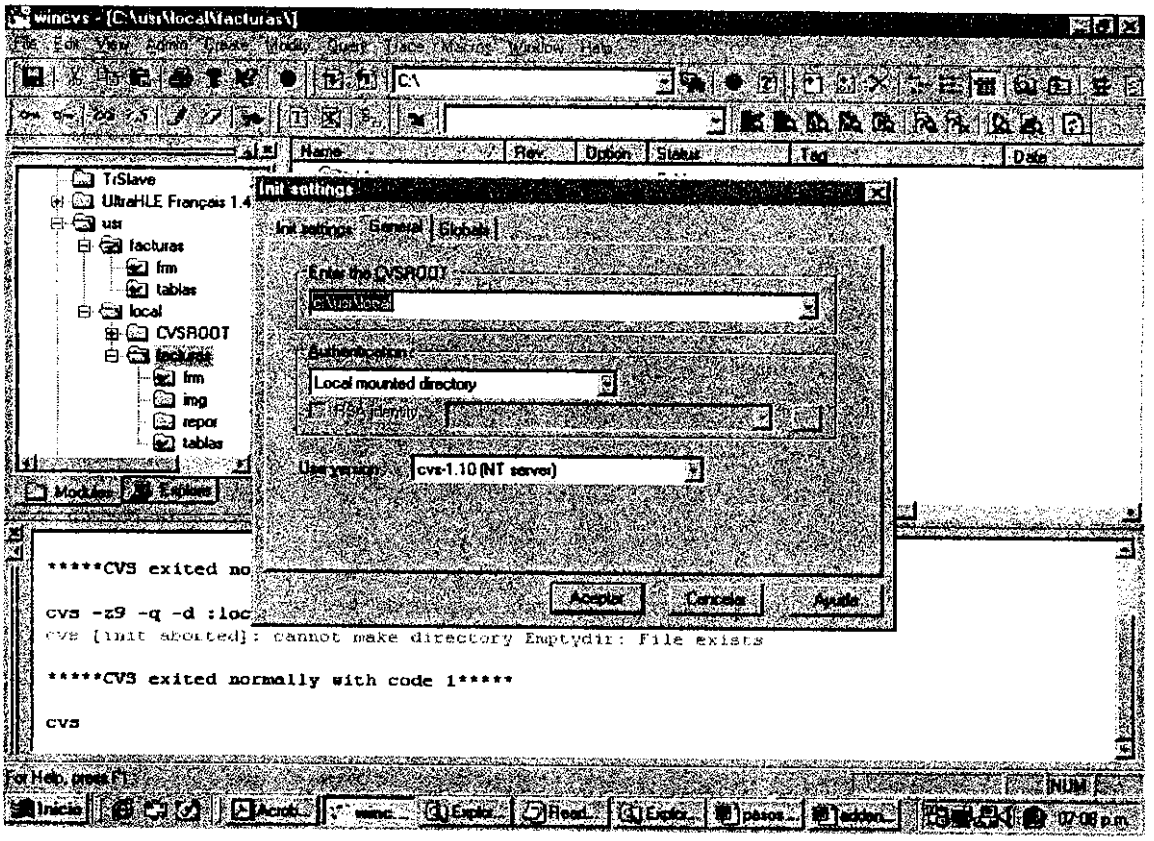
Para realizar esta la instalación se realizan los siguientes pasos:

1. Se debe descomprimir los siguientes archivos cvs-1.10.8 nt.tar.gz - cvs-nserver 1.10.7.1-1.10.8.1.dif.gz, los mismos que se pueden obtener en la página http://www.cvsnt.org

2. Copiar los archivos ejecutables ntservice.exe y cvs.exe en el directorio del sistema en, nuestro caso WINNT, estos ejecutables estarán en la carpeta que se realizó la descompresión.

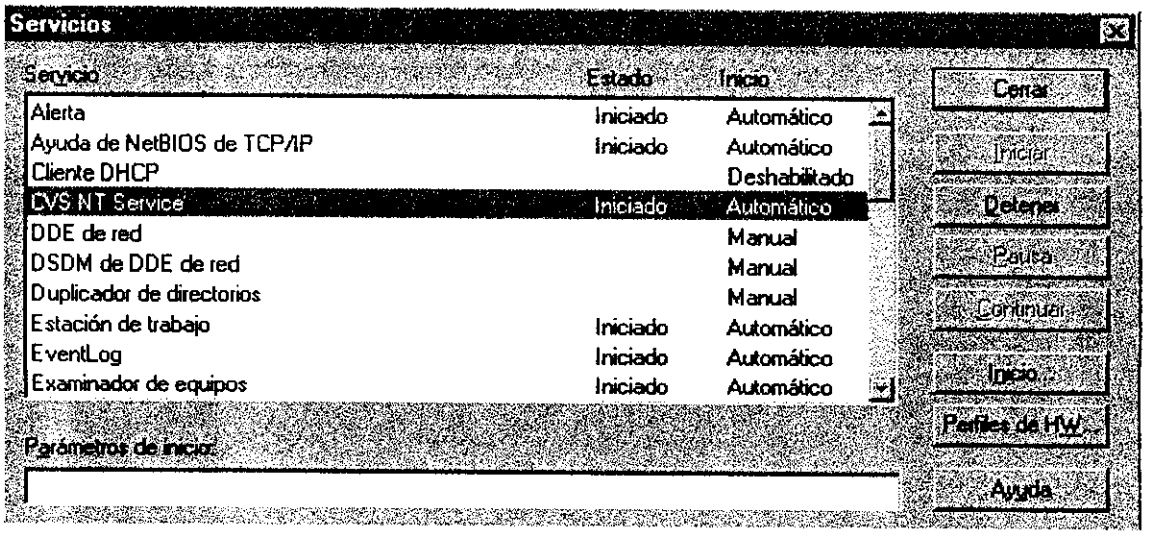
3. Instalamos el WINCVS 1.2 ó 1.3b disponibles en la página http://sourceforge.net y además se deberá instalar el TCL programa que contienen macros de ejecución de comandos en la página: http://aspn.activestate.com/ASPN/Downloads/ActiveTcl/.

4. Ingresamos al WinCVS y creamos nuestro nuevo repositorio **cvs –d : local: <cvsroot>**, de esta forma:



5. Para inicializar el servicio de **CVS NT SERVICE,** debemos hacer los siguiente: MenúInicio-ejecutar y en la ventana de ejecución de comandos digitamos ntservice –i <cvsroot>; a continuación ejecutamos tambien "**net star cvs** "para inicializar el servicio de CVS, cabe indicar que este servicio utiliza en puerto 2401.

6. Para detener este servicio de CVS utilizamos el comando **"net stop cvs"**. O también podemos ingresar al panel de control y escogemos servicios en el cuál aparecerá CVS NT SERVICE, y procedemos a detener el servicio.



Si se necesita cambiar el repositorio principal se deberá cambiar el registro del sistema ejecutando el comando **regedit,** el cvs crea la siguiente entrada de registro: HKEY_LOCAL_MACHINE\Software \CVS\Pserver\Repository0.

En esta entrada de registro se deberá modificar el path a su nuevo cvsroot.

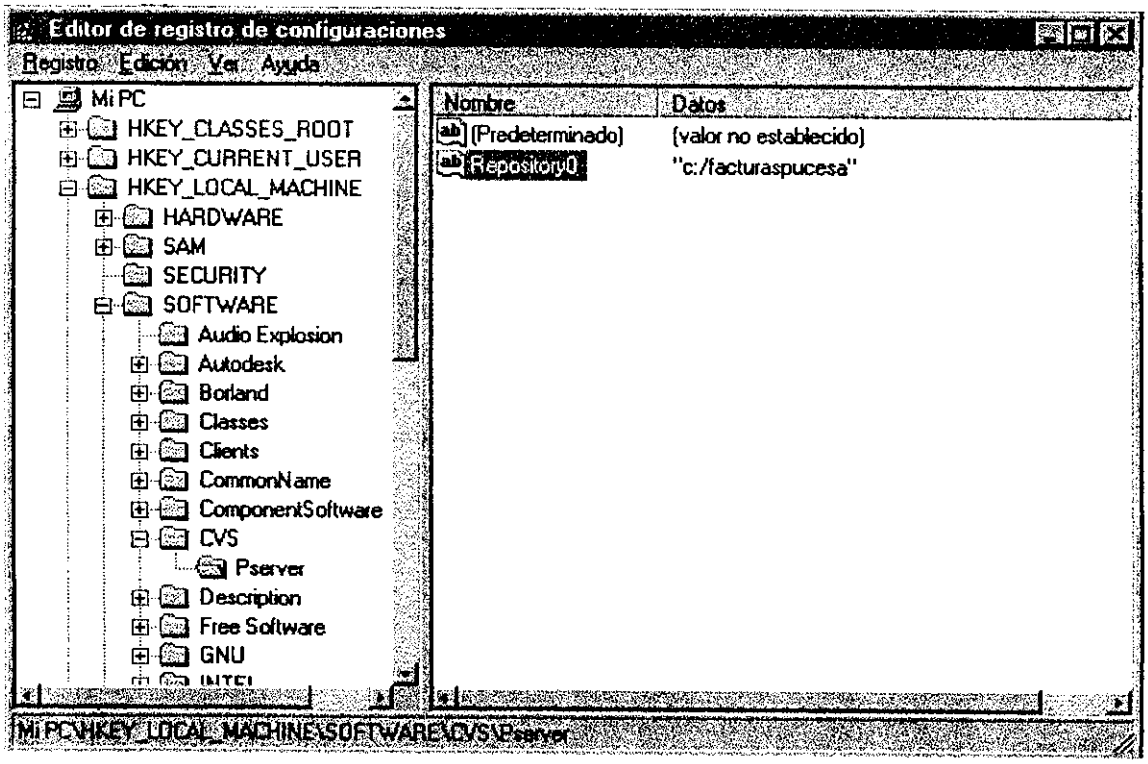
Recuerde que para trabajar con este nuevo path debe reiniciar el servicio con los pasos anteriores en los cuales se inicia el CVS NT SERVICE.

Estos son unos ejemplos como se muestra el registro con varios repositorios, siendo el Repository0 el principal.

Clave del Registro

En **HKEY_LOCAL_MACHINE\Software\CVS\Pserver**

Key	Type	Description
Repository0	String	First repository
Repository1	String	Second repository
Repository63	String	63rd repository



5.2. INSTALACIÓN DE LOS CLIENTES DEL CVS

Para realizar la instalación de los clientes del CVS, que pueden tener sistema operativo Windows 95, 98 o Me, debemos realizar los siguientes pasos en cad una de las estaciones de trabajo que van a tener acceso al CVS :

1. Se necesita instalar el Wincvs Ver. 1.2 o 1.3b, disponibles en la página http://sourceforge.net, además se debe considerar que se tenga una versión actualizada de las librerías Comctl32.dll, Comdlg32.dll.

2. Debemos Instalar el Tcl/Tk 8.3.2 para Windows, programa que contiene macros de ejecución de comandos, el mismo que funciona conjuntamente con el Wincvs, los instaladores están disponibles en: http://dev.scriptics.com/software/tcltk/8.3.html .

3. Copiar el archivo cvs2ntslib.dll que se encuentra en el directorio en el cual fue instalado el Wincvs, en la carpeta Windows\System, este archivo sirve para que se puedan comunicar con el repositorio que se encuentra instalado en el Windows NT, en el modo :ntserver:

4. El archivo cvs2ntslib.dll debe ser nombrado o removido de la carpeta que se realizo la instalación en Wincvs.

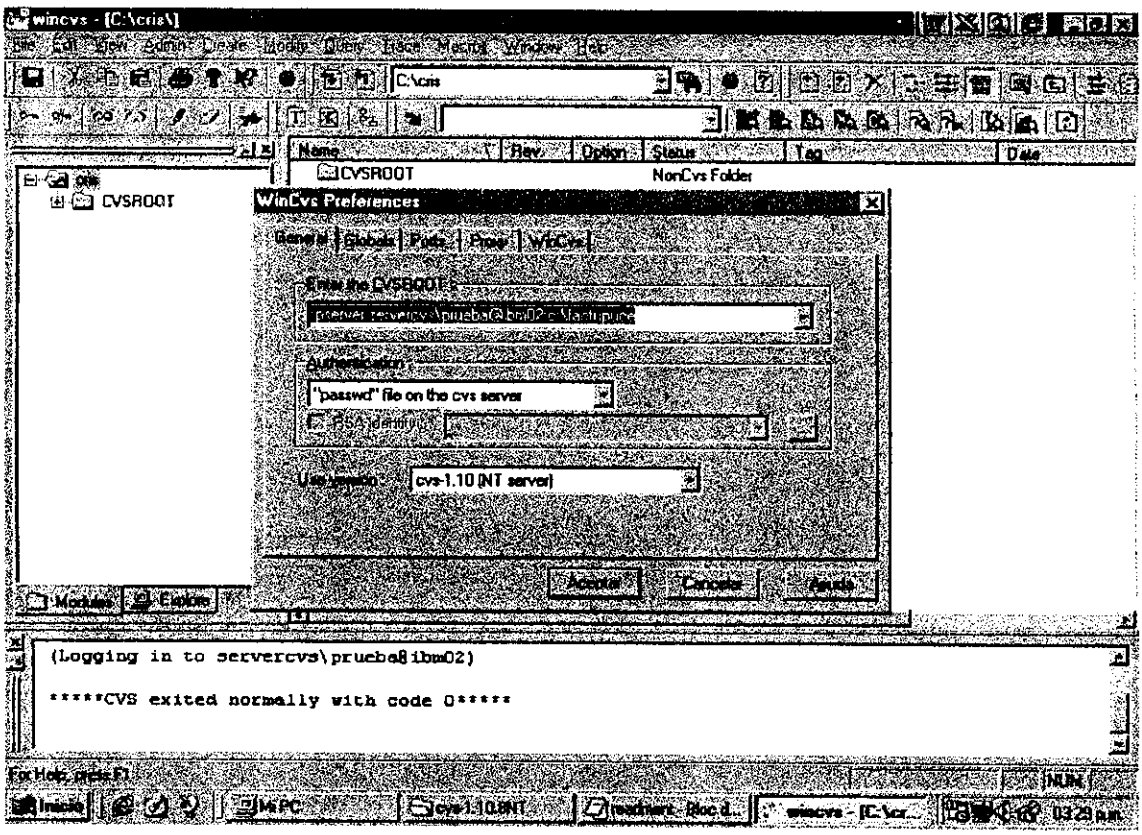
5. Una vez realizado los pasos anteriores, ejecutamos el Wincvs en la estación de trabajo, y debemos configurar el Wincvs, para que pueda acceder al CVS NT SERVICE inicializado en el Servidor.

Configuración del Wincvs en las estaciones de Trabajo:

- Acedemos al cuadro de diálogo de preferencias CTRL+F1, o por el menu desplegable escogemos admin- preferencias.

- Si el commando se ejecutó correctamente obtendremos el siguiente mensaje:

  - `*****CVS exited normally with code 0*****`



5.3. DESARROLLO DE SOFTWARE UTILIZANDO CVS

La aplicación desarrollada tiene por objetivo implementar un Sistema de Facturación para la P.U.C.E.SA., realizado bajo en control de CVS, con el cuál podemos tener un control exacto de: facturas, cuentas por cobrar, análisis de vencimientos, etc, siendo muy útil para el mejor desenvolvimiento del Departamento Financiero.

5.3.1. MÓDULOS CONTENIDOS

El Sistema de Facturación consta de los siguientes Módulos:

- **Alumnos**.- en esta módulo se puden ingresar, consultar, obtener un informe de los alumnos de la Universidad.
- **Rubros de Facturas**.- se ingresan, modifican, consultan todos los rubros de facturas.
- **Facturación**.- en este módulo, podemos realizar las facturas para la matriculación de los estudiantes de la Universidad.
- **Cuentas por Cobrar**.- en este módulo maneja todo lo que se refiere a cobros, análisis de vencimientos de cobros, consultas, debitos de cheques protestados.
- **Parámetros del Sistema**.- este módulo contiene todos los valores que son paramétricos en cada unos de los módulos anteriores, como por ejemplo: tabla de IVA, tipos de documentos.
- **Usuarios**.- nos permite crear un nuevo usuario de acceso al Sistema de Facturación.

Codificación y Diseño de Interface del Sistema de Facturación Ver Anexos # 2

5.3.2 ALCANCE DEL SISTEMA.

Para establecer la utilidad y alcance del sistema podemos detallar lo siguiente:

a.) El sistema, automatiza completamente la Facturación que se realiza en el departamento financiero de la P.U.C.E.S.A.

b.) Podemos tener un control exacto de las cuentas por cobrar, realizando un análisis de vencimientos de los documentos pendientes de cobro.

c.) Además se puede tener un control de todos los ingresos o depósitos bancarios efectuados por los alumnos.

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

Al llegar a la culminación de este trabajo, obteniendo conocimientos necesarios para un buen desarrollo, podemos concluir diciendo:

- Sin duda alguna, Internet se ha convertido en una parte necesaria de nuestra vida cotidiana. Para muchos de nosotros ha pasado a ser la forma más rápida de obtener información esencial para realizar nuestras actividades con éxito, de esta forma logramos construir una investigación positiva del Sistema de Control de Versiones.

- Gracias a la existencia del software libre GNU, que nos permite desarrollar sistemas con costos reducidos en distintos campos; las herramientas, guías y el sistema en sí, se pueden obtenerlos de manera gratuita.

- Por medio de esta investigación logramos obtener conocimientos básicos de las ventajas que nos ofrece un Sistema de Control de Versiones, que en la actualidad los verdaderos desarrolladores de software lo están aplicando ya que la clave es mejorar la cantidad y calidad en el control de proyectos teniendo formas poderosas para administrar tareas y comunicarse.

- Además logramos comprender, analizar y aplicar los comandos y herramientas que nos ofrece un Sistema de Control de Versiones, empleando de esta forma tales conocimientos para realizar su instalación en la P.U.C.E.S.A

- Podemos concluir que el Sistema de Control de Versiones, puede ser instalado en multiples plataformas, lo cuál nos permite desarrollo de proyectos y de software.

- Este trabajo nos permite diseñar un sistema de facturación el mismo que se desarrolló bajo el control de versiones, de esta forma se facilitará el servicio y atención a todos los alumnos de las diferentes escuelas de la P.U.C.E.S.A. en el departamento financiero.

- El proyecto nos permite interactuar con el uso y manejo del CVS, y así aprovechar de sus bondades en proyectos y programas existentes en el Centro de Cómputo de la Escuela de Ingeniería de Sistemas de la P.U.C.E.S.A, demostrando de esta forma su alcance e importancia de su utilización.

- Finalmente podemos decir que nuestros objetivos planteados inicialmente han sido cumplidos adquiriendo nuevos conocimientos y aplicándolos satisfactoriamente.

6.2. RECOMENDACIONES

- Se recomienda que el uso del sistema de facturación para el Departamento Financiero de la P.U.C.E.S.A. sea manejado adecuadamente con la finalidad de obtener un mayor desarrollo del mismo con resultados satisfactorios.

- Manejar los futuros proyectos que se desarrollen en la P.U.C.E.S.A. con el Sistema de Control de Versiones

- Aplicar el Sistema de Control de Versiones a los proyectos que se han desarrollado en la P.U.C.E.S.A

- Esta investigación es bastante amplia, por tal motivo se la recomienda como fuente de ayuda para quienes necesiten conocer y aplicar estos conocimientos, debido a la poca información y la falta de conocimiento de existencia de este software que existe en nuestro medio acerca de este tema.

- Para la aplicación correcta de un Sistema de Control de Versiones es necesario tener indicios de programación para mayor facilidad.

- Para un desenvolvimiento adecuado del Sistema de Control de Versiones se necesita como mínimo un computador instalado Windows 9x/Nt/Me con suficiente memoria RAM y el espacio de disco duro según la plataforma y el proyecto en la cuál se vaya a trabajar.

- Se recomienda difusión a los estudiantes de la escuela de Sistemas de la P.U.C.E.S.A. sobre la existencia e importancia de este software ya que su utilización llega más allá de lo que nos rodea.

6.3 ANEXOS

6.3.1 Guía de comandos

Parámetros que se utilizan con los siguientes comandos:

-- permitir - raíz = rootdir

- a

Autentifica toda la comunicación entre el cliente y el servidor. Tiene solamente un efecto en el cliente de CVS. En fecha de escritura, esto se pone en ejecución solamente cuando usa una autentificación de la conexión de GSSAPI previene ciertas clases de ataques que implica que se secuestre la conexión activa del TCP. Permitiendo la autentificación no el cifrado.

- b bindir

En CVS 1,9,18 y más antiguo, esto especifica que los programas de RCS están en el directorio del bindir. Las versiones actuales de CVS no ejecutan programas de RCS; para la compatibilidad se valida esta opción, pero no hace nada.

- T tempdir

Utilizan el **tempdir** como el directorio donde se localizan los ficheros temporales. Reemplaza la configuración del **MPDIR** la variable de entorno y cualquier pre-compilador del directorio. Este parámetro se debe especificar como pathname absoluto.

- **- d cvs_root_directory**

- e editor

Utilizan el editor para incorporar la información del registro de la revisión. Reemplaza la configuración del **$$CVSEDITOR** y **$$EDITOR** de las variables de entorno.

- f

No lee el fichero **~/.cvsrc**. Esta opción se utiliza lo más a menudo posible debido a el no-non-orthogonality del conjunto de la opción de CVS. Por ejemplo, **cvs log** la opción **-N** no tiene una opción correspondiente para girar la visualización. Si se tiene **-N** en **~/.cvsrc** la entrada para **log**, se puede necesitar utilizar **-f** para mostrar los nombres de la etiqueta.

- **- H -- ayuda**

Información del uso de la visualización especifica **cvs_command** (pero no ejecute realmente el comando). Si no se especifica un nombre del comando, **cvs -H** ayuda total de las visualizaciones para CVS, incluyendo una lista de otras opciones de ayuda.

- l

No registra **cvs_command** en la historia del comando, para la información sobre historia del comando.

- n

No cambia ningun fichero. Procuran ejecutar **cvs_command**, para publicar solamente informes; no quite, no ponga al día, ni combine ningún fichero existente, o cree cualquier fichero nuevo. Observe que CVS no producirá necesariamente la misma salida que fuera **-n**. En algunos casos la salida estará igual, pero en otros casos CVS saltará algo del proceso que habría sido requerido para producir la misma salida.

- Q

Provoca que el comando sea realmente reservado; el comando generará solamente la salida para los problemas serios.

- q

Provoca que el comando sea realmente reservado; los mensajes informativos, tales como informes de la repetición a través de subdirectorios, se suprimen.

- r

Hacen ficheros de trabajo nuevos inalterables. El mismo efecto como si **$$CVSREAD** se fija la variable de entorno. El valor por defecto es hacer ficheros de trabajo escribibles, a menos que los relojes estén encendidos.

- **- variable = valor de s**

  - t

  Ejecución del programa de rastreo; visualizan los mensajes que muestran los pasos de progresión de la actividad de CVS.

- **- v -- versión**

  - W

  Hacen ficheros de trabajo nuevos de lectura/grabación. Reemplaza la configuración **$$CVSREAD** de la variable de entorno. Los ficheros son de lectura/grabación creado por valor, por defecto, a menos que **$$CVSREAD** se fije o **-r** se dé.

- **date_spec**

  - f

  Cuando se especifica una fecha en particular o se etiqueta a los comandos del CVS, ellos normalmente ignoran los archivos que no contienen la etiqueta (o no existían previo a la fecha) que se especificó. Utilice la opción **- f** si se quiere que los archivos se recuperen aún cuando no sean parte de la etiqueta o de la fecha. (se usará La revisión más reciente del archivo). **-f'** está disponible con estos comandos: **annotate, checkout, export, rdiff, rtag,** y **update.**

  Advertencia: El guardar y remover comandos tienen además una opción **- f**, pero esta tiene un diferente comportamiento para esos comandos.

- **kflag**

  - l

  Local; corre solamente en directorios trabajados corrientes, en lugar de recursar a través de subdirectorios.

  Advertencia: ¡ésto no es lo mismo que todas las opciones **cvs - l**, la cual se puede especificar a la izquierda del comando cvs!. Disponible con los siguientes comandos: **annotate**, **checkout** , **commit**, **diff**, **edit**, **editors**, **export**, **log**, **rdiff**, **remove**, **rtag**, **status**, **tag**, **unedit**, **update**, **watch**, y **watchers**.

- **message**

  - n

  No corre ningun programa **checkout/ commit/ tag program**. (un programa puede ser especificado para correr cada una de estas actividades, en la base de datos de los módulos, esta opción lo desvia).

  Advertencia: ¡ésto no es lo mismo como todas las opciones **cvs -n**, la cual se puede especificar a la izquierda de un comando cvs! Disponible con los comandos **checkout**, **commit**, **export**, y **rtag**.

  - P

  Recorta los directorios vacíos.

- p

Conduce los archivos recuperados del depósito a un rendimiento estándar, en lugar de escribirlos en el directorio corriente. Disponible con los comandos **checkout** y **update**.

- R

Procesa directorios recursivamente. Esto es por omisión. Disponible con los comandos siguientes: **annotate, checkout, commit, diff, edit, editors, export, rdiff, remove, rtag, status, tag, unedit, updat, watch,** y **watchers.**

- **tag**

- W

Especifica los nombres de archivo que deberían ser filtrados. Puede usarse esta opción repetidamente. El **spec** puede ser un nombre de patrón de archivo del mismo tipo que puede especificarse en el archivo **.cvswrappers**. Disponible con los siguientes comandos: **import**, y **update**.

- **Opciones del [admin]**

- A oldfile

No podría trabajar junto con el CVS. Añade el acceso listo del **oldfile** para el acceso listo del archivo RCS.

- a logins

Podría no trabajar junto con CVS. Añade los nombres del **login** que aparecen en la coma seprando la lista **logins** para el acceso listo del archivo RCS.

- b [rev]

Coloca la omisión de rama al **rev**. En el CVS, normalmente no se manipula ramas omitidas; las etiquetas pegajosas son la mejor forma para decidir en cual rama se quiere trabajar. Hay una razón para correr el **cvs - admin - b:** revertir la versión del vendedor cuando se utiliza ramas vendedoras. No hay espacio entre - **b** y su argumento.

- c string

Fija los comentarios guía a la cadena. El comentario guía no es utilizado por versiones corrientes del CVS o RCS 5.7.

- e [logins]

No podría trabajar junto con CVS. Borra los nombres del **login** que aparecen en la lista **logins** separados por coma del acceso listo del archivo del RCS. El **logins,** es omitido borra la lista de acceso completamente.

- I

Corre interactivamente, aun si la entrada del standar no es un terminal. Esta opción no trabaja con el cliente/ servidor del CVS.

- k [subst]

Fija la omisión de la substitución de la palabra clave al **subst**. Dando una explícita opción **- k** para el **cvs update**, **cvs export**, o **cvs checkout** atropellar esta omisión.

- l [rev]

Asegura la revisión con un número **rev**. Si una rama es dada, asegura la última revisión de la rama. Si se omite **rev**, asegura la última revisión sobra la rama omitida. No hay espacio entre rama **- l** y su argumento. Esto puede ser utilizado en relación con el manuscrito **rcslock.pl** en el directorio **contrib** de la fuente de distribución de CVS proporcionando revisiones reservadas (donde solo un usuario puede revisar un archivo dado a la vez).

- L

Coloca seguros a la reserva. La reserva asegurada significa que el propietario de un archivo RCS no está exento de asegurarse para chequeos.

- m [rev]: [msg]

Reemplaza los mensajes **log** de la revisión **rev** con **msg**.

- N name[: [rev]]

Actúa como **- n**, exceptúa sobrepasar a cualquier asignación previa de nombre. Para su uso con ramas mágicas.

- n name[: [rev]]

Asocia el nombre simbólico **name** con la rama o revisión **rev**. Es por otro lado mejor utilizar **cvs tag** o **cvs rtag** . Elimina el nombre simbólico si **:** y **rev** son omitidos; por otro lado, imprime un mensaje de error si **name** ya está asociado con otro número. Si **rev** es simbólico, esta es expandida a la asociación anterior. Un **rev** consiste de un número de rama seguida por la colocación de un **.** (un punto) de la revisión última corriente en la rama **A :** (dos puntos) con la colocación de un **rev** vacío para la revisión última corriente sobre la rama de omisión, normalmente el tronco. Por ejemplo **cvs admin - n name:** asociado **name** con la revisión última corriente de todos los archivos RCS; esto contrasta con **cvs admin - n name : $** la cual asocia **name** con la revisión del número extraído desde las cadenas de palabras clave en los correspondientes archivos trabajados.

- o range

Elimina (saca datos) las revisiones dadas por **range**. Nótese que este comando puede ser silencioso y peligroso a menos que se conozca exactamente qué se está haciendo. Si es corto en disco está opción podría ayudar. Si se elimina diferentes revisiones de las que se planeó, pueden ser debido a falta de cuidado o un virus en el CVS, no hay oportunidad de corregir el error antes de que la revision sea eliminada. Especifique el **range** en una de las maneras siguientes:

rev1:: rev2

Colapse todas las revisiones entre **rev1 y rev2**, para que CVS solamente almacenará las diferencias asociadas que van desde rev1 a rev2, sin pasos intermedios. Por ejemplo después de - o 1.3:: 1.5 se puede recuperar revisión 1.3, revisión 1.5, o las diferencias logradas desde 1.3 a 1.5, pero no la revisión 1.4, o las diferencias entre 1.3 y 1.4. Otros ejemplos: - o 1.3:: 1.4 y - o 1.3:: 1.3 no tiene efecto, porque no hay revisiones intermedias para remover.

:: rev

El colapso revisa entre el principio de la rama que contiene **rev** y el mismo **rev**. El punto de la rama y **rev** quedan intactos. Por ejemplo, - o:: 1.3.2.6 anula la revisión 1.3.2.1, la revisión 1.3.2.5, y todo lo que sté entre ellas, pero dejar 1.3 y 1.3.2.6 intactos.

Rev ::

Este colapso revisa entre **rev** y el fin de la rama que contiene **rev**. La revisión **rev** es dejada intacta pero la cabeza de la revisión es anulada.

rev

Anula la revisión **rev**. Por ejemplo - o 1.3 es equivalente a o 1.2:: 1.4.

rev1: rev2

Anula las revisiones desde **rev1** a **rev2**, inclusive, en la misma rama. No se podrá recuperar **rev1** o **rev2** o cualquiera de las revisiones que estén entre ellas. Por ejemplo el comando cvs admin - oR_1_01: R_1_02 . Es raramente útil.

Esto significa anular las revisiones de arriba , e incluso, la etiqueta R_1_02. ¡Pero cuidado ! Si hay archivos que no han cambiado entre R_1_02 y R_1_03 el archivo tendrá la misma revisión numérica, número asignado a las etiquetas R_1_02 y R_1_03. No solamente será imposible recuperar R_1_02; ¡R_1_03, tendrá también que ser restaurado de las cintas!

En la mayoría de casos se querrá especificar **rev1:: rev2** a cambio.

: rev

Anula revisiones del principio de la rama que contiene rev desde arriba e incluye rev.

rev:

Anula revisiones desde revisión rev, incluyendo el mismo rev, al final de la rama que contiene rev. Ninguna de las revisiones que serán borradas podrán tener ramas o seguros. Si algunas de las revisiones que son borradas tienen nombres simbólicos, y se especifica una de las sintaxis **::**, entonces el CVS dará un error y no anulará ninguna de las revisiones. Si verdaderamente quiere anular los nombres simbólicos y las revisiones, primero hay que anular los nombres simbólicos con **cvs tag - d**, entonces correr **cvs admin - o**. Si se especifica la sintaxis **non- ::**, entonces CVS anulará las revisiones pero dejará los puntos de los nombres simbólicos a revisiones no existentes. Debido a que la manera que CVS maneja ramas **rev** no se especifica simbólicamente si es

una rama. Asegúrese de que nadie tenga chequeos de una copia de la revisión que eliminó.

- q

Corre calladamente; no imprime diagnósticos.

- s state [: rev]

Util con CVS. Coloca los atributos del estado de la revisión **rev** a **state**. Si **rev** es una rama numérica, se asume la última revisión sobre esa rama. Si **rev** es omitido, se asume la última revisión en la rama omitida. Cualquier identificado es aceptable para **state**. Un útil instalador de states, **Exp** (por experimentar), **Stab** (por estable), y **Rel** (por liberar). Pero por presunción, el estado de una nueva revisión es instalado a **Exp** cuando este es creado. El estado es visible en el rendimiento desde **cvs log**, y en las palabras claves **$ Log$** y **$ State$**. Nótese que CVS usa el **dead state** para su propio propósito; para llevar un archivo a o desde el **dead state** (estado muerto) utilice los comandos como **cvs remove** y **cvs add**, no **cvs admin - s**.

- t[file]

Util con el CVS. Escribe textos descriptivos de los contenidos del llamado archivo dentro del archivo, anulando el texto existente. La vía del nombre del archivo podría no empezar con -. Los textos descriptivos pueden ser vistos en el rendimiento desde el **cvs log**. Puede no haber espacio entre **-t** y su argumento. Si el archivo es omitido, se obtiene el texto desde la entrada estándar, terminado por el final del archivo o por una línea que contiene . (un punto).

- t- string

Similar a **-t file**. Escribe textos descriptivos desde el **string** dentro del archivo RCS, anulando el texto existente. No puede haber espacio entre **-t** y su argumento.

- U

Coloca seguro a lo no estricto. Lo No-estricto asegurado significa que el dueño de un archivo no requiere asegurar una revisión para chequearla.

- u [rev]

No asegurar la revisión con número **rev**. Si una rama es dada, no asegurar la última revisión en esa rama. Si se omite **rev**, quitar el último seguro sostenido por el llamador. Normalmente, solamente el asegurador de una revisión lo abriría. Alguien más que desasegurara una revisión rompería el seguro. Ésto causa un mensaje de correo para ser enviado al asegurador original. El mensaje contiene un comentario solicitado desde el violador. El comentario es terminado por el fin de un archivo o por una línea que contiene . (un punto). No puede haber espacio entre **-u** y su argumento.

- **Opciones del comando checkout**

- D date

Usa la revisión más reciente y no las anteriores a la fecha. Esta opción es viscosa, e implica **-P**.

- f

Es útil únicamente con las opción **-D date** o **-r tag**. Si ningún juego o revisión combinada es encontrada, recupera la revisión más reciente (en vez de ignorar el archivo).

- k kflag

Procesa palabras claves según **kflag**. Esta opción es viscosa; las futuras actualizaciones de este archivo en este directorio de trabajo usarán el mismo **kflag** . El comando de condición puede inspeccionarse para ver las opciones viscosas.

- l

Local; corre únicamente en el directorio de trabajo actual.

- n

No corre ningún programa **checkout**.

- P

Recorta directorios vacíos.

- p

Conduce archivos a la salida estándard.

- R

Chequea los directorios recursivamente. Esta opción es solamente por falla.

- r tag

Utiliza revisión de etiqueta . Esta opción es viscosa, e implica **-P**.

Además de esos, se puede usar estas opciones especiales de comando con checkout

.

- A

Resetea cualquier etiqueta pegajosa, fechas, u opciones **-k**.

- c

Copia los archivos de módulo, clasifica, al rendimiento estándard, en vez de crear o modificar cualquiera de los archivos o directorios en su directorio de trabajo.

- d dir

Crea un directorio llamado dir para los archivos de trabajo, en vez de usar el nombre de módulo. En general, utilizar este parámetro es equivalente a usar **mkdir dir**; **cd dir** seguido por el comando **checkout** sin la opción **-d**. Sin embargo hay una excepción importante. Es muy conveniente cuando se chequea un artículo único tener la salida, que aparezca un directorio que no contiene directorios intermedios vacíos. En este caso único, el CVS trata de acortar vías de nombres para evitar esos directorios vacíos. Por ejemplo, dado un módulo **foo** que contiene el archivo **bar.c**, el comando **cvs co – d dir foo** crea el directorio **dir** y colocará **bar.c** en su lugar. Similarmente, dado un módulo **bar** que tiene un subdirectorio **baz** en donde hay un archivo **quux.c**, el comando **cvs - d dir co /baz** crea el directorio **dir** y colocará en su lugar **quux.c**. Utilizando la opción **-N** se eliminará este comportamiento. Dando las mismas definiciones de los módulo de arriba, **cvs**

co - N - d dir foo creará los directorios **dir/foo** y colocará en su lugar **bar.c**, mientras **cvs co - N - d dir bar/baz** creará los directorios **dir/bar/baz** y colocará en su lugar **quux.c**.

- N

Solamente es útil junto con **-d dir**. Con esta opción, el CVS no "acorta" los caminos del módulo en su directorio de trabajo cuando se chequea un módulo simple

- s

Como **-c**, pero incluye el status de todos los módulos, y los clasifica por la línea de status

- **Opciones de commit**

- l

Local; corre solamente con el directorio de trabajo actual.

- n

No corre cualquier programa de módulo.

- R

Guarda directorios recursivamente. Esto es por omisión.

- r revision

Guarda la revisión. La revisión puede ser de cualquier rama, o una revisión de la rama principal que es más alta que cualquier revisión de número resistente. No se puede guardar una revisión específica sobre una rama.

Commit además soporta estas opciones:

- F file

Lee el mensaje completo desde el archivo, en vez de invocar un editor.

- f

Forza al CVS a guardar una nueva revisión aún si no se ha hecho cualquier cambio del archivo. Si la revisión actual del archivo es 1.7, entonces los siguientes dos comandos son equivalentes:

```
$ cvs commit - f
file
$ cvs commit - r 1.8
file
```

La opción **-f** inutiliza la recursión (es decir, implica -l). Para forzar CVS a guardar una nueva revisión para todos los archivos en todos los subdirectorios, se debe usar **-f , - R** .

- m message

Utiliza el mensaje como el mensaje completo, en vez de invocar un editor.

- **Opciones de Diff**

- D date

Se utiliza la revisión más reciente y no la anterior a la fecha .

- k kflag

Procesa palabras claves según **kflag**.

- l

Local; corre únicamente en el directorio de trabajo actual.

- R

Examina directorios recursivamente. Esta opción es por omisión.

- r tag

Compara con la revisión **tag**. Encera, uno o dos opciones **-r** que pueden estar presentes. Sin la opción **-r**, el archivo de trabajo se comparará con la revisión en la que estaba basada. Con la opción **-r**, esa revisión será comparada con su archivo de trabajo actual.

Con dos opciones **-r** estas dos revisiones serán comparadas (y su archivo de trabajo no afectará el resultado de cualquier forma). Una o ambas opciones **-r** pueden ser reemplazadas por una opción **-D date**, como de describió arriba.

- **Opciones de export**

- D date

Usar la revisión más reciente y no la anterior a la fecha .

- f

Si no es encontrada la revisión en conjunto, recobra la revisión más reciente (en vez de ignorar el archivo).

- l

Local; corre únicamente en el directorio de trabajo actual.

- n

No corre ningún programa **checkout**.

- R

Exporta directorios recursivamente. Esto es por omisión.

- r tag

Usa revisión **tag**.

Además, estas opciones (que son comunes a checkout y export) se apoyan también:

- d dir

Crea un directorio llamado dir para los archivos de trabajo, en vez de usar el nombre del módulo.

- k subst

Coloca a modo de palabras claves de expansión

- N

Unicamente es útil junto con **-d dir**.

- **Opciones de history**

- c

Reporta cada momento en el que **commit** fue utilizado (es decir, cada vez que el depósito se modificó).

- e

Equivale para especificar **-x** con todos los tipos de registro. Por supuesto, **-e** incluirá también tipos de registro que se agregan en una futura versión de CVS; si se escribe un manuscrito el cual puede simplemente manejar tipos de registro, se querrá especificar **-x** .

- m módule

Informa sobre un módulo particular. (Se puede significativamente usar -m más de una vez en la línea de comando.)

- o

Informa los módulos revisados.

- T

Informa sobre todas las etiquetas.

- x type

Extrae un conjunto particular de tipos de registro **type** desde la historia del CVS. Los tipos son indicados por letras simples, las cuales podrían especificarse en combinación.

Comandos recientes tienen un tipo único de registro:

F	Elimina
O	Chequea
E	Exporta
T rtag	Uno de los cuatro tipos de registro pueden resultar de **update:**
U	Un archivo de trabajo se copió desde el depósito.
W	La copia de trabajo de un archivo se borró durante la actualización (porque se pasó del depósito).
Uno de los tres tipos de registro resultó desde **commit**:	
A	Un archivo se agregó por primera vez.
M	Un archivo se modificó.
R	Un archivo se quitó.

Las opciones mostradas como **-flags** limitan o expanden el informe sin requerir argumentos de opción:

- a

Muestra datos de todos los usuarios (la falla es que muestra datos solamente de los usuarios que ejecutaron history).

- l

Muestra solamente las últimas modificaciones

-w

Muestra solamente los registros para modificaciones hechas desde el mismo directorio de trabajo donde **history** es ejecutado.

Las opciones mostradas como **- options args** limitan el informe basado en un argumento:

- b str

Muestra datos que regresan a un registro que contiene la cadena **str** en cualquiera del nombre del módulo, el nombre de archivo, o la trayectoria del depósito.

- D date

Muestra datos desde **date**. Esto es ligeramente diferente del uso normal de **-D date**, el cual selecciona las revisiones más nuevas más viejas de **date**.

- p repository

Muestra datos de una fuente de depósito en particular (se puede especificar varias opciones -p en la misma línea de comando).

- r rev

Muestra registros referiéndose a revisiones desde la revisión o la llamada etiqueta **rev** que aparece en archivos RCS individuales. Cada archivo RCS es buscado desde la revisión o etiqueta.

- t tag

Muestra registros desde la etiqueta **tag** que fue la última en sumarse al archivo **history**. Esto difiere de la opción **-r** de arriba en que lee únicamente el archivo de **history**, no los archivos RCS, y es mucho más rápido.

- u name

Muestra registros de los nombres de usuario .

- **Opciones de import**

- m message

Utiliza **message** como una información total en lugar de invocar un editor. Hay las opciones especiales adicionales siguientes:

- k subs

Indica el modo de expansión de la palabra clave deseada.

Esto instalado aplicará a todos los archivos creados durante la importación, pero no a cualquier archivos que anterioramente existieron en el depósito.

- I name

Especifica nombres de archivo que deben ignorarse durante la importación. Se puede usar esta opción repetidamente. Para evitar ignorar cualquier archivos en total (aún esos ignorados por omisión), especifique **-I !.name** puede ser el nombre de un archivo modelo del mismo tipo que puede especificar en el archivo **.cvsignore**.

- W espec

Especifica los nombres de archivo que deben filtrarse durante la importación. Se puede usar esta opción repetidamente. **Espec** puede ser el nombre de un archivo modelo del mismo tipo que se puede especificar en el archivo **.cvswrappers**.

- **Opciones de log**

- b

Imprime la información sobre la revisiones de la rama omitida, normalmente la rama más alta en el tronco.

- d date

Imprime la información sobre revisiones con un chequeo en el rango dado por el punto y coma separado la lista de fechas. Los formatos de fecha aceptados son esos

aceptados por la opción **-D** para muchos otros comandos de CVS. Las fechas pueden combinarse en rangos como se indica a continuación:

d1 < d2

d2 > d1

Seleccione la revisión que fue depositada entre **d1** y **d2.**

< d

d >

Seleccione todas las revisiones fechadas en **d** o anteriores.

d <

> d

Seleccione todas las revisiones fechadas en **d** o luego.

d

Seleccione la simple, fechada en **d** o anteriores.

Los caracteres > o <, pueden ser seguidos por = para indicar un rango que incuye en lugar de un rango que excluye. Nótese que el separador es un punto y coma (;).

- h

Imprime solamente el nombre del archivo RCS, nombre del archivo en el directorio trabajado, encabezado, rama de omisión, lista de accesos, seguros, nombres simbólicos y sufijos.

- l

Local; corre únicamente en el directorio de trabajo actual. (Por omisión está corre recursivamente).

- N

No imprime la lista de etiquetas para este archivo. Esta opción puede ser muy útil cuando se sitúa usando muchas etiquetas.

- R

Imprime únicamente el nombre del archivo RCS.

- r revisions

Imprime la información sobre las revisiones de la lista separada por la coma de las revisiones y rangos. La siguiente tabla explica los rangos de formatos disponibles:

rev1 : rev2

Revisar **rev1** a **rev2** (las cuales deben estar sobre la misma rama).

: rev

Revisar desde el inicio de las rama incluyendo hasta **rev**.

rev:

Revisar empezando con **rev** a el final de la rama conteniendo **rev**.

branch

Un argumento que está en una rama que significa todas las revisiones en esa rama.

branch1 : branch2

Un rango de ramas que significa todas las revisiones de las ramas en ese rango.

Branch

La última revisión en la rama .

Una **-r** vacía sin revisiones significa la última revisión en una rama de omisión, normalmente el tronco. No puede haber espacio entre la opción **-r** y su argumento.

- s states

Imprime la información sobre las revisiones cuyos atributos estatales combinan uno de los estados dados en una lista de estados separados por la coma .

- w logins

Imprime la información sobre las revisiones registradas por usuarios con nombres completos apareciendo en la lista **login** separada por una coma. Si **logins** se omite, el usuario login se presume. No puede haber espacio entre la opción **-w** y su argumento.

Log

Imprime la intersección de las revisiones seleccionadas con las opciones **-d**, **-s**, y **-w**, cruzado con la unión de las revisiones seleccionadas por **-b** y **-r**.

- **Opciones de rtag**

- D date

Etiqueta las revisiones más recientes y no las más anteriores a la fecha .

- f

Solamente es útil con los parámetros **-D date** o **-r tag**. Si el conjunto o las combinaciones de revisión no es encontrada, utiliza la revisión más reciente (en vez de ignorar el archivo).

- F

Sobreescribe una etiqueta existente del mismo nombre sobre una revisión diferente.

- l

Local; corre únicamente en directorio de trabajo actual.

- n

No corre ningún programa de etiqueta que se especificó con la opción **-t** dentro del archivo del modulo.

- R

Etiqueta directorios recursivamente. Esto es por omisión.

- r tag

Solamente etiqueta esos archivos que contienen **tag**. Esto puede usarse para renombrar una etiqueta. Etiqueta solamente los archivos identificados por la etiqueta vieja, entonces los borra, dejando en la nueva etiqueta exactamente los mismos archivos como la etiqueta vieja.

Además de las opciones comunes de arriba , estas opciones son disponibles:

- a

Utiliza la opción **-a** para tener una vista del **rtag** en el **Attic** para remover archivos

que contienen la etiqueta especificada. La etiqueta es removida desde estos archivos,

que son convenientes para reutilizar una etiqueta simbólica como desarrollo contínuo (y los archivos se logran remover desde la distribución que viene de arriba).

- d

Borra la etiqueta en vez de crearla. En general, las etiquetas (frecuentemente los nombres simbólicos de las distribuciones del software) no deberían ser removidas, pero la opción **-d** esta disponible como principal para remover completamente los nombres simbólicos obsoletos si se necesita (como podría ser el caso de una liberación Alpha, o si se equivocó en un módulo).

- **Opciones de tag**

- F

Sobreescribe una etiqueta existente del mismo nombre sobre una revisión diferente.

- l

Local; corre únicamente en el directorio de trabajo actual.

- R

Etiqueta directorios recursivamente. Esto es por omisión.

Dos opciones especiales son disponibles:

- b

Hace de la etiqueta una etiqueta de rama, permitiendo un concurrente desarrollo aislado. Este es muy útil para crear un parche para la liberación previa a la distribución de software .

- c

Chequea que todos los archivos que están para ser etiquetados sean inmodificables. Esto puede usarse para asegurar que se puede reconstruir los contenidos actuales de archivo.

- d

Borra una etiqueta. Si se utiliza **cvs tag - d symbolic_tag**, la etiqueta simbólica que se especificó es borrada en vez de ser agregada.

Advertencia: Debe estar muy seguro de su terreno antes de que se borre una etiqueta; hacer esto permanentemente desecha alguna información histórica, que puede resultar más tarde de valor.

- **Opciones de update**

- D date

Usa la revisión más reciente y no la anterior a la fecha.

- f

Solamente útil con las opciones **-D date** o **-r tag**. Si el conjunto de revisiones no es encontrada, recupera la revisión más reciente (en vez de ignorar el archivo).

- k kflag

Procesa las palabras claves según **kflag**. Esta opción es viscosa; las futuras actualizaciones de este archivo en este directorio de trabajo usarán el mismo **kflag**. El **status** puede inspeccionarse para ver las opciones viscosas.

- l

Local; corre únicamente con directorios de trabajo actual.

- P

Recorta directorios vacíos.

- p

Conduce archivos al rendimiento estándard.

- R

Actualiza directorios recursivamente (omisión).

- r rev

Recobra **revision/tag rev**. Esta opción es viscosa, e implica **-P**.

Estas opciones especiales son también disponibles con **update**.

- A

Resetea cualquier etiqueta viscosas, fechas, u opciones **-k**

- d

Crea cualquier directorio que existe en el depósito si ellos están extraviados del directorio de trabajo. Normalmente, **update** actúa únicamente sobre directorios y archivos que ya estaban enrolados en el directorio trabajado. Esto es útil para actualizar directorios que se crearon en el depósito desde el chequeo inicial; pero tiene un efecto desafortunado.

- I name

Ignora archivos cuyos nombres se combinan con **name** (en el directorio de trabajo) durante la actualización. Se puede especificar **-I** más de una vez en la línea de comando para especificar algunos archivos para ignorarlos. Utilice **-I !** para evitar ignorar algunos archivos del total.

- W espec

Especifica nombres de archivo que deberían ser filtrados durante la actualización. Se puede usar esta opción repetidamente. El **espec** puede ser un modelo de nombre de archivo del mismo tipo que puede ser especificado en el archivo **.cvswrappers**.

- **Opciones de rdiff**

- D date

Usa la revisión más reciente y no anterior a la fecha

- f

Si la combinación de revisión no es encontrada, recobra la revisión más reciente (en vez de ignorar el archivo).

- l

Local; no desciende subdirectorios.

- R

Examina directorios recursivamente. Esta opción es por omisión.

- r tag

Utiliza revisiones **tag**.

Además de las opciones de arriba, estas opciones están disponibles:

- c

Usa el formato de contexto **diff**. Este es formato de omisión.

- s

Crea un reporte de cambios en lugar de un **parche**. El resumen incluye información sobre archivos que fueron cambiados o sumados entre las liberaciones. Se envía al dispositivo estándard de rendimiento. Esto es útil para encontrarlos, por ejemplo, cuales son los archivos que han cambiado entre dos fechas o revisiones.

- t

Un **diff** de dos revisiones tope es enviado al dispositivo estándard de rendimiento. Este es muy útil para observar cuál fue el último cambio del archivo.

- u

Utiliza el formato **unidiff** para el contexto **diffs**. Esta opción no esta disponible si su **diff** no soporta el formato **unidiff**.

- **-- permitir-raíz = rootdir**

- a

Autentifica toda la comunicación (cliente solamente) (no en CVS 1,9 y más antiguo).

- b

Especifica la localización de RCS (CVS 1,9 y más antiguo).

- d root

Especifica el **CVSROOT.**

- e editor

Corrije los mensajes con el **editor**.

- f

No lee el fichero **~/.cvsrc**.

- **- H -- ayuda**

- l

No se abre una sesión del fichero de **VCScRcOcOcT/history**.

- n

No cambia ningun fichero.

- Q

Es realmente reservado.

- q

Es algo reservado.

- r

Hace los ficheros de trabajo nuevos inalterables.

- **admin [opciones] [ficheros ...]**

- b[ ] revolution

Ramificación determinada del valor por defecto.

- c string

Arranque de cinta determinado del comentario.

- K subst

Fija la substitución de la palabra clave.

- l[ ] revolution

Bloquea la revolución de la revisión , o la última revisión.

- m : msg revolution

Substituye el mensaje del registro de la revolución de la revisión por msg .

- o range

Revisiones de la cancelación del depósito.

- q

Funcionamiento reservado; no imprime el diagnóstico.

- s [: revolución] state

Fija el estado.

- t

Fija la descripción del fichero de la entrada de información estándar.

- t file

Fija la descripción del fichero.

- t string

Fija la descripción del fichero a la cadena .

- u[ ] revolution

Abre la revolución de la revisión, o la última revisión.

- **anote [opciones] [ficheros ...]**

- D date

Anota la revisión más reciente que fechan .

- f

Utiliza la revisión principal si **tag/date** no se ha encontrado.

- l

Local; ejecútese solamente en el directorio de trabajo actual.

- R

Funciona recurrentemente (valor por defecto).

- r tag

Anota la etiqueta de la revisión .

- **Chekout module [opciones] ...**

- A

Reajusta cualquier **tags/date/options** pegajosos.

- c

Provoca sacar la base de datos del módulo.

- D date

Controla fuera de las revisiones en fecha (es pegajoso).

- d dir

Controla hacia fuera en **dir**.

- f

Utiliza la revisión principal si **tag/date** no se ha encontrado.

- j revolution

Fusiona en cambios.

- k kflag

Utiliza la extensión de la palabra clave del **kflag**.

- l

Local; ejecútese solamente en directorio de trabajo actual.

- N

" no acorta " los caminos del módulo si **- d** se especifica.

- n

No ejecuta el programa del módulo (si cualquiera).

- P

Controla directorios vacíos.

- p

Controla fuera de ficheros a la salida estándar (evita viscosidad).

- R

Funciona recurrentemente (valor por defecto).

- r tag

Etiqueta de la revisión de la comprobación (es pegajoso).

- s

Realiza lo mismo que la opción - **c**, pero incluye estatus del módulo.

- **confide [opciones] [ficheros ...]**

- F file

Lee el mensaje del registro del fichero .

- f

Forza al fichero para ser confiado; invalida la repetición.

- l

Local; ejecútese solamente en directorio de trabajo actual.

- m msg

Utiliza los **msg** como mensaje del registro.

- n

No ejecuta el programa del módulo (si cualquiera).

- R

Funciona recurrentemente (valor por defecto).

- r revolution

Confía a la revolución .

- **diff [opciones] [ficheros ...]**

- D date1

Revisión de **Diff** para la fecha contra fichero de trabajo.

- D date2

Diff rev1 / date1 contra **date2**.

- l

Local; ejecútese solamente en directorio de trabajo actual.

- N

Incluye los **diffs** para los ficheros agregados y quitados.

- R

Funciona recurrentemente (valor por defecto).

- r rev1

Revisión de **Diff** para **rev1** contra fichero de trabajo.

- r rev2

Diff rev1 / date1 contra **rev2**.

- **export modules [de las opciones] ...**

- D date

Controla fuera de revisiones en fecha .

- d dir

Controla hacia fuera en dir .

- f

Utiliza la revisión principal si **tag/date** no se ha encontrado.

- k kflag

Utiliza la extensión de la palabra clave del **kflag**.

- l

Local; ejecútese solamente en directorio de trabajo actual.

- N

" no acorta " los caminos del módulo si - **d** se ha especificado.

- n

No ejecuta el programa del módulo (si cualquiera).

- P

Controla directorios vacíos.

- R

Funciona recurrentemente (valor por defecto).

- r tag

Etiqueta de la revisión de la comprobación (es pegajoso).

- **history [opciones] [ficheros ...]**

- a

Todos los utilizadores (el valor por defecto es uno mismo)..

- b str

De nuevo a expediente con el **str** en el módulo **/file/repos field**

- c

Informa sobre archivos guardados(modificados).

- D date

Desde la fecha .

- e

Informa sobre todos los tipos de registro.

- l

Ultima modificación (guarda o modifica el reporte).

- m module

Informa sobre el módulo (repetible).

- n module

En el módulo .

- o

Informa sobre módulos chequeados.

- r rev

Desde la revisión **rev**.

- T

Produce reportes sobre todas LAS ETIQUETAS.

- t tag

Desde la etiqueta grabada colocada en el archivo de historia.

- u user

Para usuario de **user** .

- w

Trabaja con directorios que deben combinarse.

- z zone

Rinde para el tiempo zona zona .

- **import [opciones] depósito vendedor - tag liberador - tags ...**

- b bra

Importa a la rama vendedora **bra**.

- d

Usa el tiempo de modificación del archivo como el tiempo de importe.

- k kflag

Coloca por omisión un modo de sustitución de palabras claves.

- m msg

Usa **msg** para el mensaje total.

- I ign

Más archivos para ignorar (! para resetear).

- W espec

Más envolturas.

init

Crea un depósito de CVS si no existe.

- **log [opciones] [archivos ...]**

- d date

Especifica fechas (d1 < d2 para rangos, d para antes de la última).

- h

Solamente imprime cabeceras, títulos.

- l

Local; corre únicamente en directorios trabajados actualmente.

- N

No enumera etiquetas.

- R

Solamente imprime nombres de archivos RCS

- r revs

Solamente enlista revisiones **revs**.

- s states

Solamente enlista revisiones con status especificados.

- t

Solamente imprime cabeceras, títulos y describe textos.

- w logins

Solamente enlista revisiones chequeadas por **logins** especificadas.

login

Mueve palabras claves para servidores autentificados.

logout

Remueve palabras claves almacenadas para servidores autentificados.

- **rdiff [opciones] módulos ...**

- c

Contextúa el rendimiento de un formato **diff** (omisión).

- D date

Selecciones, revisiones en fecha .

- f

Usa título de revisión si la etiqueta/fecha no se encuentra.

- l

Local; corre únicamente en directorios trabajados recientes.

- R

Opera recursivamente (omisión).

- t

Supera dos **diffs** - último cambio hecho al archivo.

- **rtag [opciones] etiqueta módulos ...**

- a

Limpia la etiqueta de un archivo quitado que de otra manera sería etiquetado.

- b

Crea una rama llamada **tag**.

- D date

Etiqueta revisiones como de **date**.

- d

Borra la etiqueta determinada.

- F

Mueve la etiqueta si ya existe.

- f

Forza un conjunto de revisión de titulo, si etiqueta/fecha no se encuentra.

- l

Local; corre únicamente en directorio trabajado reciente.

- n

Ninguna ejecución de programa **tag**.

- R

Opera recursivamente (omisión).

- **tag [opciones] tag [archivo ...]**

- b

Crea una rama llamada **tag**.

- D date

Etiqueta revisiones como de **date**.

- d

Borra una etiqueta determinada.

- F

Mueve la etiqueta si ya existe.

- f

Forza un conjunto de revisión de título si etiqueta/fecha no se encuentra.

- l

Local; corre únicamente en el directorio de trabajo actual.

- n

Ninguna ejecución del programa **tag**.

- R

Opera recursivamente (omisión).

- r tag

Etiquetea etiqueta existente **tag**.

- **update [opciones] [archivos ...]**

- A

Resetea cualquier etiqueta viscosa /fechas/opciones.

- D date

Chequea revisiones como de **date** (es viscosa).

- d

Crea directorios.

- f

Usa revisiones de cabecera si etiqueta/fecha no se encontro.

- I ign

Más archivos para ignorar (! para resetear).

- j rev

Combina cambios.

- k kflag

Usa expansión de palabra clave **Kflag**.

- l

Local; corre únicamente en el directorio de trabajo actual.

- P

Purga directorios vacios.

- p

Chequea archivos para el rendimiento estándard (evita rigidéz).

- R

Opera recursivamente (omisión).

- r tag

Chequea revisiones **tag** (es viscosa).

GLOSARIO DE TERMINOS

CVS	Sistema de Control de Versiones
RCS	Sistema de Control de Revisiones
SSH	SSH CLIENTE es un programa para anotar dentro una maquina remota y para ejecutar comandos en una maquina remota. El es planeado para reemplazar rlogin y rsh, y proporcionar comunicaciones encriptadas seguras entre dos untrust-ed servidores sobre una red insegura.
RSH	Protocolo de Acceso remoto al CVS
JCVS	Es una interfaz grafica para clientes de CVS escritos en Java
Kerberos	Protocolo de seguridad para acceso al CVS
TCL	TCL programa que contienen macros de ejecución de comandos
GSSSAPI	El GSSAPI es una interfase genérica para los sistemas de seguridad de red como el **Kerberos**

Licencia Pública General de GNU

Versión 2, Junio 1991

59 Temple Place, Suite 330, Boston, MA 02111 USA

Preámbulo

Las licencias que cubren la mayor parte del software están diseñadas para quitarle a usted la libertad de compartirlo y modificarlo. Por el contrario, la Licencia Pública General de GNU pretende garantizarle la libertad de compartir y modificar software libre, para asegurar que el software es libre para todos sus usuarios. Esta Licencia Pública General se aplica a la mayor parte del software de la Free Software Foundation y a cualquier otro programa cuyos autores se comprometen a utilizarla. (Existe otro software de la Free Software Foundation que está cubierto por la Licencia Pública General de GNU para Bibliotecas). Si quiere, Ud. también puede aplicarla a sus propios programas.

Cuando hablamos de software libre, estamos refiriéndonos a libertad, no a precio. Nuestras Licencias Públicas Generales están diseñadas para asegurarnos de que Ud. tenga la libertad de distribuir copias de software libre (y cobrar por ese servicio si quiere), de que reciba el código fuente o que pueda conseguirlo si lo quiere, de que pueda modificar el software o usar fragmentos de él en nuevos programas libres, y de que sepa que puede hacer todas estas cosas.

Para proteger sus derechos necesitamos hacer algunas restricciones que prohiban a cualquiera negarle a usted estos derechos o pedirle que renuncie los derechos. Estas restricciones se traducen en ciertas obligaciones que le afectan si Ud. distribuye copias del software, o si lo modifica.

Por ejemplo, si usted distribuye copias de uno de estos programas, ya sea gratuitamente o por cobrar, Ud. debe dar a todos los recibidores todos los derechos que Ud. tiene. Ud. debe asegurarse de que ellos también reciban o puedan conseguir el código fuente. Y debe mostrarles estas condiciones de manera que ellos pueden conocer sus derechos. Nosotros protegemos sus derechos con estas dos medidas: (1) Ponemos el software bajo copyright y (2) le ofrecemos esta licencia que le da permiso legal a copiar, distribuir y/o modificar el software.

También, para la protección tanto de cada autor como la de nosotros mismos, queremos asegurar que todos entiendan que no hay garantía alguna para este software libre. Si alguien más ha modificado y distribuido el software, queremos que sus recibidores sepan que no es el original, de manera que cualquier problema que otro introduzca no afecte la reputación de los autores originales.

Finalmente, cualquier programa libre se ve constantemente amenazado por los patentes de software. Deseamos evitar el peligro de que los redistribuidores de un programa libre individualmente obtengan patentes, con el efecto de convertir el programa en un programa propietario. Para evitar esto, hemos dejado claro que cualquier licencia de patente debe ser conseguido para a el uso libre de todos o no debe conseguirse.

Los términos y condiciones exactas para la duplicación, distribución y modificación se elaboran a continuación.

TÉRMINOS Y CONDICIONES PARA LA DUPLICACIÓN, DISTRIBUCIÓN Y MODIFICACIÓN

1. Esta licencia se aplica a cualquier programa u otra obra que contenga un aviso de parte del propietario del copyright diciendo que se puede distribuir bajo los términos de esta Licencia General Pública. En adelante, ``Programa" se refiere a cualquier dicho programa u obra, y ``obra basada en el Programa" quiere decir ya sea el programa o cualquier obra derivada de él bajo las leyes de copyright. Es decir, una obra que contenga el Programa o una porción del mismo, ya sea literal o con modificaciones y/o traducido a otro idioma. (De aquí en adelante se incluye la traducción sin limitación en el término ``modificación".) Se dirige a cada licenciatario como ``Ud."

   Esta licencia no cubre otras actividades fuera de la duplicación, distribución y modificación; éstas están fuera de su alcance. El acto de ejecutar el Programa no está restringido, y los datos que resultan de su uso están cubiertos solamente cuando constituyen una obra basada en el Programa, independientemente del hecho de haber sido producido por la ejecución del programa. El caso de que sea así o no depende de qué es lo que hace el Programa.

2. Ud. puede hacer y distribuir copias literales del código fuente del Programa tal como Ud. lo recibió, en cualquier medio, con tal de que publique en cada copia, de manera visible y apropiada, un aviso sobre el copyright y repudiación de garantía; mantenga intactos todos los avisos que refieren a esta Licencia y la ausencia de garantía; y proporcione a cualquier otro recibidor del Programa una copia de esta Licencia junto con el Programa.

   Ud. puede cobrar un honorario por el acto fisico de transferir una copia, y a opción suya puede ofrecer protección de garantía a cambio de un honorario.

3. Ud. puede modificar su copia o copias del Programa o cualquier porción del mismo, y así formar una obra basada en el Programa, y duplicar y distribuir dichas modificaciones u obra bajo los términos de la antedicha apartado 1, con tal de que también cumpla con todas las siguientes condiciones:

   a. Ud. debe hacer que los ficheros modificados lleven avisos indicando que Ud. ha cambiado los ficheros, con la fecha de cualquier cambio.

   b. Ud. debe hacer que cualquier obra que distribuya o publique que contenga o sea derivada del Programa o de una parte del mismo, ya sea en su integridad o en parte, sea licenciada en su integridad sin costo a todas las terceras partes bajo los términos de esta licencia.

   c. Si el programa modificado normalmente lee órdenes interactivamente al ejecutarse, usted debe hacer que, al iniciar dicho uso interactivo en la

manera más habitual, el programa muestre un mensaje incluyendo un aviso apropiado de copyright y un aviso de que no hay garantía (o diciendo que Ud. ofrece una garantía) y que los usuarios pueden redistribuir el programa bajo estas condiciones y avisando al usuario como ver esta Licencia. (Excepción: si el Programa mismo es interactivo pero no suele mostrar un mensaje de este tipo, entonces no se requiere que su obra basada en el Programa muestre un mensaje.)

Estos requisitos son aplicables a la obra modificada en su integridad. Si secciones identificables de dicha obra no están derivadas del Programa y pueden ser razonablemente consideradas obras independientes y separadas en sí, entonces esta Licencia y sus términos no se aplican a esas secciones cuando Ud. los distribuye como obras separadas. Pero cuando distribuye las mismas secciones como parte de una integridad que es una obra basada en el Programa, la distribución del todo debe ser bajo los términos de esta Licencia, cuyos permisos para otros licenciatarios se extienden a la integridad de la obra y por consiguiente a todas y cada una de sus partes, sin considerar de quien la haya escrito.

Por lo tanto, no es la intención de este apartado reclamar derechos o disputar sus derechos sobre obras escritas enteramente por Ud. Más bien, la intención es de ejercer el derecho de controlar la distribución de obras derivadas o colectivas basadas en el Programa.

Adicionalmente, la simple agregación de otra obra no basada en el Programa junta con el Programa (o con una obra basada en el Programa) en un volumen de un medio de almacenamiento o de distribución no extiende el alcance de esta Licencia a la otra obra.

4. Ud. puede duplicar y distribuir el Programa (o una obra basada en ella, bajo el apartado 2) en forma de código objeto o ejecutable bajo los términos de los antedichos apartados 1 y 2, con tal de que también haga uno de los siguientes:

   a. Acompañarlo con el código fuente completo correspondiente en una forma legible por máquina, el cual debe ser distribuido bajo los términos de los antedichos apartados 1 y 2 en un medio habitualmente utilizado para el intercambio de software; o

   b. Acompañarlo con una oferta por escrito, válida por un mínimo de tres años, de proporcionar a cualquier tercera parte por un honorario que no exceda del costo de fisicamente realizar la distribución de los fuentes, una copia completa en forma legible por máquina del código fuente correspondiente, el cual debe ser distribuido bajo los términos de los antedichos apartados 1 y 2 en un medio habitualmente utilizado para el intercambio de software; o

   c. Acompañarlo con la información que Ud. recibió en cuanto a la oferta de distribución del código fuente correspondiente. (Esta alternativa sólo es permitido para distribución no comercial y solamente si Ud. recibió el

Programa en forma de código objeto o ejecutable con dicha oferta de acuerdo con el subapartado b anterior.)

El "código fuente" de una obra significa la forma preferida de la obra para hacer modificaciones a la misma. Para una obra ejecutable, el "código fuente completo" quiere decir todo el código fuente para todos los módulos que contiene, más cualesquier ficheros asociados de definición de interfaz, más los scripts que se utilizan para controlar la compilación e instalación del ejecutable. Sin embargo, como una excepción especial, el código fuente distribuido no necesita incluir algo que normalmente se distribuye (ya sea en forma de código fuente o en forma binaria) con los componentes principales (compilador, núcleo, etc.) del sistema operativo con el cual el ejecutable funciona, a no ser que dicho componente mismo acompañe el ejecutable.

Si la distribución del ejecutable o código objeto se hace al ofrecer acceso para copiarlo de un lugar designado, entonces el ofrecer acceso equivalente para copiar el código fuente del mismo lugar cuenta como distribución del código fuente, aunque no se exija a terceras partes que copien el código fuente junto con el código objeto.

5. Ud. no puede copiar, modificar, sublicenciar o distribuir el Programa excepto de la manera expresamente previsto por esta licencia. Cualquier intento de copiar, modificar, sublicenciar o distribuir el Programa de otra manera es inválido y terminará sus derechos bajo esta Licencia automáticamente. Sin embargo, otras

partes que hayan recibido copias o derechos de Ud. bajo esta Licencia no perderán sus derechos mientras dichas partes sigan en pleno cumplimiento.

6. Dado que no lo ha firmado, Ud. no está obligado a aceptar esta licencia. Sin embargo, no hay nada más que le dé permiso para modificar o distribuir el Programa o sus obras derivadas. Estas acciones son prohibidas por la ley a no ser que Ud. acepte esta Licencia. Por lo tanto, al modificar o distribuir el Programa (o cualquier obra basada en el Programa), Ud. indica su aceptación de esta Licencia para hacerlo, y de todos sus términos y condiciones sobre la duplicación, distribución o modificación del Programa u obras basadas en él.

7. Cada vez que Ud. redistribuye el Programa (o cualquier obra basada en el Programa), el que lo recibe automáticamente recibe una licencia del licenciante original para copiar, distribuir o modificar el Programa, sujeto a estos términos y condiciones. Ud. no puede imponer al recibidor ninguna restricción adicional sobre el ejercicio de los derechos concedidos en la presente. Ud. no es responsable de hacer que terceras partes cumplan con esta Licencia.

8. Si como consecuencia de un fallo judicial o de una alegación de infracción de patente o por cualquier otra razón (no limitándose a cuestiones de patentes), se le imponga a Ud. condiciones (ya sea por una orden judicial, acuerdo o de otra manera) que contradigan las condiciones de esta Licencia, éstas no le eximen de las condiciones de esta licencia. Si Ud. no puede distribuirlo de manera que

satisfaga simultáneamente sus obligaciones bajo esta licencia y cualquier otra obligación perteneciente, entonces en consecuencia Ud. no puede distribuir el programa. Por ejemplo, si una licencia de patente no permite a todos los que reciban copias de Ud., ya sea directamente o indirectamente, redistribuir del Programa libre de regalías, entonces la única manera en que Ud. puede cumplir tanto con ella como con esta Licencia sería de abstenerse del todo de la distribución del Programa.

Si cualquier parte de este apartado es considerado inválido o imposible de hacer cumplir bajo alguna circunstancia particular, el resto del apartado debe aplicarse y el apartado en su integridad debe aplicarse en otras circunstancias.

No es el propósito de este apartado inducirlo a infringir algún patente u otro derecho de propiedad o a alegar contra la validez de algún derecho reclamado; este apartado tiene el único propósito de proteger la integridad del sistema de distribución de software libre, el cual se pone en práctica mediante licencias públicas. Muchas personas han hecho contribuciones generosas a la amplia gama de software distribuida mediante este sistema, confiando en la aplicación uniforme de dicho sistema. Depende del autor o donador decidir si está dispuesto a distribuir software mediante algún otro sistema y un licenciatario no puede imponer esa elección.

Este apartado pretende hacer abundantemente claro lo que se cree ser una consecuencia del resto de esta Licencia.

9. Si la distribución y/o el uso del Programa está restringido en ciertos países debido a patentes o interfaces bajo copyright, el titular original del copyright que pone el Programa bajo esta Licencia puede añadir una limitación geográfica explícita a la distribución, excluyendo dichos países, de manera que la distribución quede permitido solamente en o entre países no así excluidos. En dicho caso, esta Licencia incorpora la limitación así como si estuviera escrita en el cuerpo de esta Licencia.

10. La Free Software Foundation puede publicar versiones modificadas y/o nuevas de la Licencia Pública General de vez en cuando. Dichas versiones nuevas serán similares en espíritu a la versión presente, pero pueden ser diferentes en detalles para abarcar nuevos problemas o situaciones.

    A cada versión se le dará un número de versión que lo distingue de otras. Si el Programa especifica un número de versión que se le aplica y "cualquier versión posterior", Ud. tiene la opción de cumplir con los términos y condiciones ya sea de esa versión o de cualquier versión posterior que publique la Free Software Foundation. Si el Programa no especifica un número de versión de esta Licencia, Ud. puede escoger cualquier versión que la Free Software Foundation haya en algún momento publicado.

11. Si Ud. desea incorporar partes del Programa en otros programas libres cuyos condiciones para la distribución son diferentes, escriba al autor pidiendo permiso. Para software cuyo titular del copyright es la Free Software

Foundation, escriba a la Free Software Foundation; en veces hacemos excepciones para esto. Nuestra decisión será guiada por las dos metas de preservar el estado libre de todos los derivados de nuestro software libre y de promover que se comparta y reutilice el software en general.

AUSENCIA DE GARANTÍA

12. DEBIDO A QUE EL PROGRAMA SE LICENCIA LIBRE DE CARGAS, NO HAY GARANTÍA ALGUNA SOBRE EL PROGRAMA, EN LA MEDIDA PERMITA POR LAS LEYES APLICABLES. LOS TITULARES DEL COPYRIGHT Y/U OTRAS PARTES PROVEEN EL PROGRAMA "TAL Y COMO ESTÁ" SIN GARANTÍA DE NINGUNA CLASE, YA SEA EXPRESA O IMPLÍCITA, INCLUYENDO SIN LIMITACIÓN LAS GARANTÍAS IMPLÍCITAS DE COMERCIABILIDAD Y APTITUD PARA UN PROPÓSITO ESPECÍFICO, EXCEPTO CUANDO LO CONTRARIO SEA DECLARADO POR ESCRITO. TODO EL RIESGO EN CUANTO A LA CALIDAD Y ACCIÓN DEL PROGRAMA LO ASUME UD. SI EL PROGRAMA SE COMPRUEBA DEFECTUOSO, UD. ASUME TODO EL COSTO DE TODO SERVICIO, REPARACIÓN O CORRECCIÓN QUE SEA NECESARIO.

13. NINGÚN TITULAR DE COPYRIGHT NI OTRA PARTE QUE PUEDA MODIFICAR Y/O REDISTRIBUIR EL PROGRAMA SEGÚN SE PERMITE

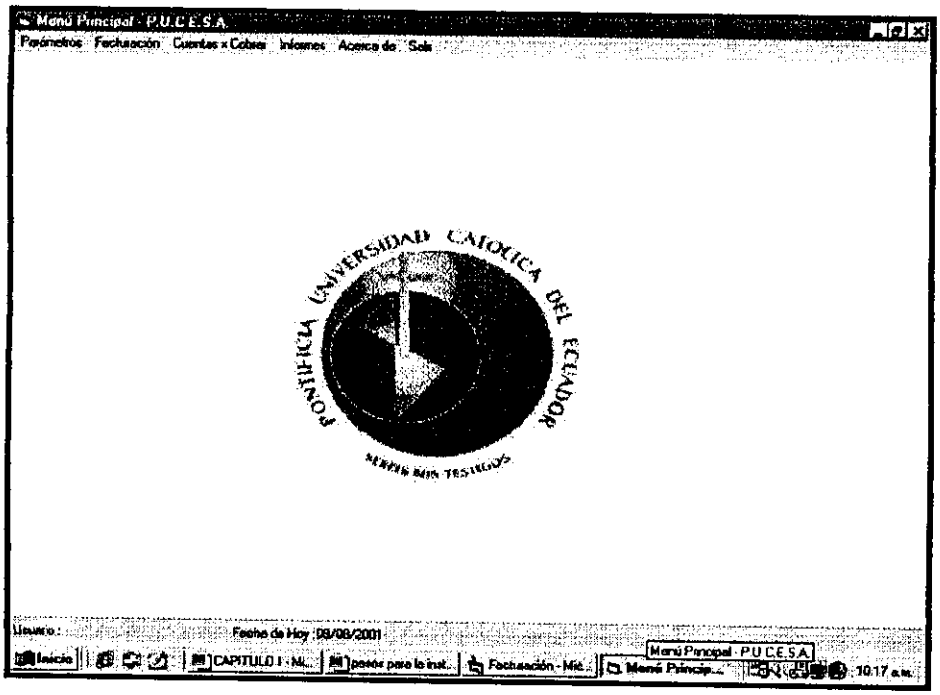
EN ESTA LICENCIA SERÁ RESPONSABLE ANTE UD. JAMÁS POR PERJUICIOS, INCLUYENDO CUALQUIER PERJUICIO GENERAL, ESPECIAL, INCIDENTAL O CONSECUENTE DEBIDO AL USO O LA IMPOSIBILIDAD DE PODER USAR EL PROGRAMA (INCLUYENDO SIN LIMITACIÓN LA PÉRDIDA DE DATOS O QUE DATOS SE VUELVAN INCORRECTOS O PÉRDIDAS SOSTENIDAS POR UD. O POR TERCERAS PARTES O LA IMPOSIBILIDAD DEL PROGRAMA A OPERAR CON ALGÚN OTRO PROGRAMA), A NO SER QUE LEYES APLICABLES LO REQUIERAN O HAYA SIDO ACORDADO POR ESCRITO, AUNQUE DICHO TITULAR U OTRA PARTE HAYA SIDO AVISADO DE LA POSIBILIDAD DE TALES PERJUICIOS.

FIN DE LOS TÉRMINNOS Y CONDICIONES

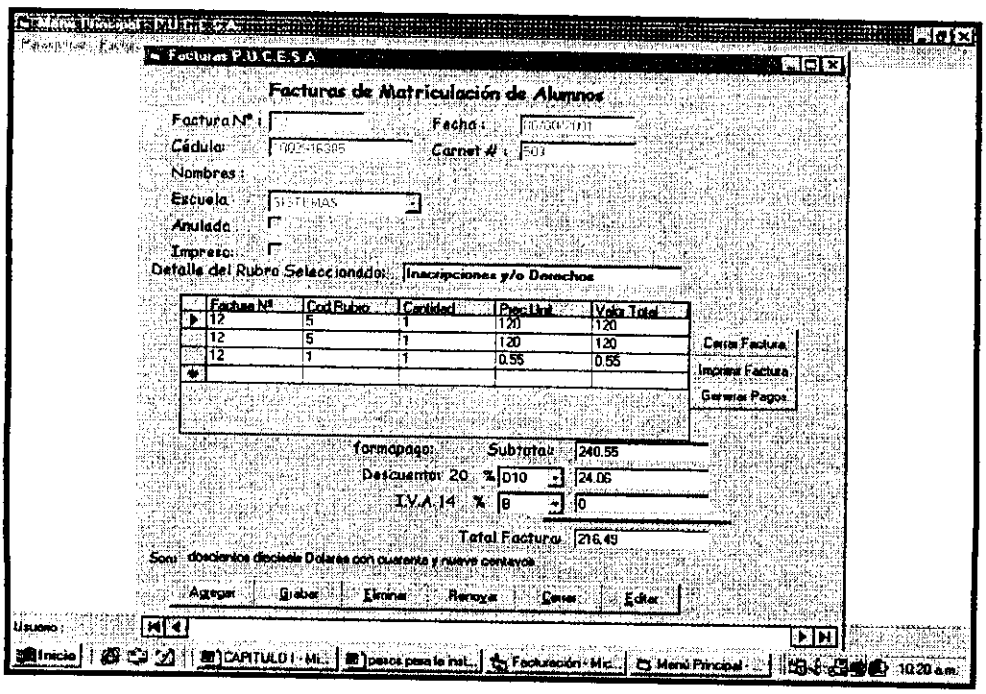
6.4 Anexos

6.4.1 Interface programa de facturación y de Wincvs

Menu Principal del Sistema de Facturación Para la PUCESA



Ingreso de Facturas:



CODIFICACIÓN:

```
Dim WithEvents adoPrimaryRS As Recordset
Dim mbChangedByCode As Boolean
Dim mvBookMark As Variant
Dim mbEditFlag As Boolean
Dim mbAddNewFlag As Boolean
Dim mbDataChanged As Boolean

Private Sub Form_Load()
  Dim db As Connection
  Set db = New Connection
  db.CursorLocation = adUseClient
  db.Open "PROVIDER=MSDASQL;dsn=Facturapucesa;uid=;pwd=;"

  Set adoPrimaryRS = New Recordset
  adoPrimaryRS.Open "select codigor,nombrer,valor from rubros Order by codigor",
db, adOpenStatic, adLockOptimistic

  Dim oText As TextBox
  'Enlaza los cuadros de texto con el proveedor de datos
  For Each oText In Me.txtFields
    Set oText.DataSource = adoPrimaryRS
  Next
  mbDataChanged = False
End Sub

Private Sub Form_Resize()
  On Error Resume Next
  lblStatus.Width = Me.Width - 1500
  cmdNext.Left = lblStatus.Width + 700
  cmdLast.Left = cmdNext.Left + 340
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
  If mbEditFlag Or mbAddNewFlag Then Exit Sub

  Select Case KeyCode
    Case vbKeyEscape
      cmdClose_Click
    Case vbKeyEnd
      cmdLast_Click
    Case vbKeyHome
      cmdFirst_Click
    Case vbKeyUp, vbKeyPageUp
      If Shift = vbCtrlMask Then
        cmdFirst_Click
```

```
    Else
      cmdPrevious_Click
    End If
   Case vbKeyDown, vbKeyPageDown
    If Shift = vbCtrlMask Then
      cmdLast_Click
    Else
      cmdNext_Click
    End If
  End Select
End Sub

Private Sub Form_Unload(Cancel As Integer)
  Screen.MousePointer = vbDefault
End Sub

Private Sub adoPrimaryRS_MoveComplete(ByVal adReason As
ADODB.EventReasonEnum, ByVal pError As ADODB.Error, adStatus As
ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)
  'Esto mostrará la posición de registro actual para este Recordset
  lblStatus.Caption = "Registro: " & CStr(adoPrimaryRS.AbsolutePosition)
End Sub

Private Sub adoPrimaryRS_WillChangeRecord(ByVal adReason As
ADODB.EventReasonEnum, ByVal cRecords As Long, adStatus As
ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)
  'Aquí se coloca el código de validación
  'Se llama a este evento cuando ocurre la siguiente acción
  Dim bCancel As Boolean

  Select Case adReason
  Case adRsnAddNew
  Case adRsnClose
  Case adRsnDelete
  Case adRsnFirstChange
  Case adRsnMove
  Case adRsnRequery
  Case adRsnResynch
  Case adRsnUndoAddNew
  Case adRsnUndoDelete
  Case adRsnUndoUpdate
  Case adRsnUpdate
  End Select

  If bCancel Then adStatus = adStatusCancel
End Sub

Private Sub cmdAdd_Click()
```

```
  On Error GoTo AddErr
  With adoPrimaryRS
    If Not (.BOF And .EOF) Then
      mvBookMark = .Bookmark
    End If
    .AddNew
    lblStatus.Caption = "Agregar registro"
    mbAddNewFlag = True
    SetButtons False
    botones (True)
    txtFields(0).SetFocus
    txtFields(2).Text = 0
  End With

  Exit Sub
AddErr:
  MsgBox Err.Description
End Sub

Private Sub cmdDelete_Click()
  On Error GoTo DeleteErr
  With adoPrimaryRS
    .Delete
    .MoveNext
    If .EOF Then .MoveLast
  End With
  Exit Sub
DeleteErr:
  MsgBox Err.Description
End Sub

Private Sub cmdRefresh_Click()
  'Esto sólo es necesario en aplicaciones multiusuario
  On Error GoTo RefreshErr
  adoPrimaryRS.Requery
  Exit Sub
RefreshErr:
  MsgBox Err.Description
End Sub

Private Sub cmdEdit_Click()
  On Error GoTo EditErr

  lblStatus.Caption = "Modificar registro"
  mbEditFlag = True
  SetButtons False
  botones (True)
  txtFields(0).Enabled = False
```

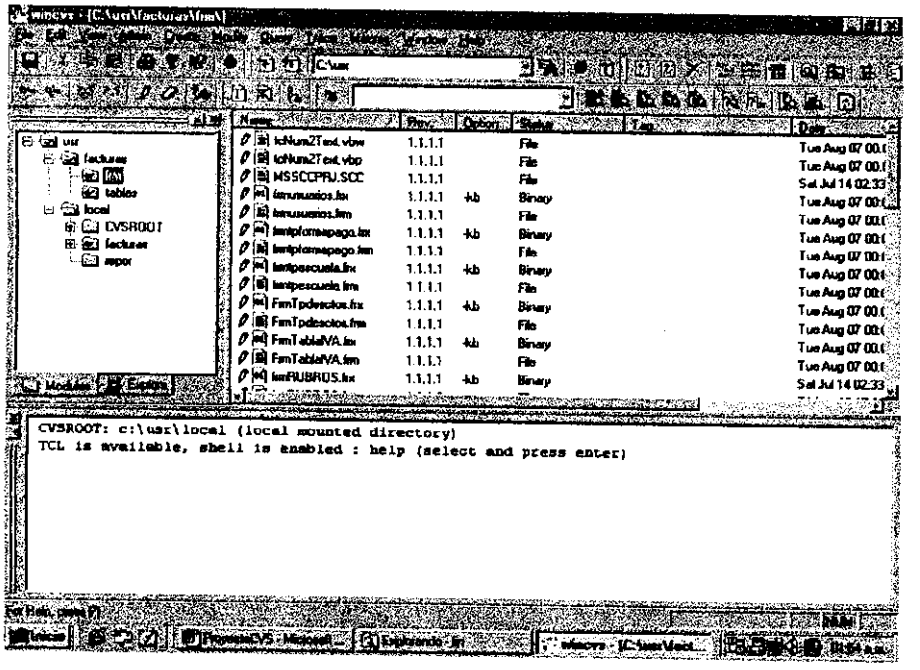
```
  txtFields(0).BackColor = &H80000000
  txtFields(1).SetFocus
  Exit Sub

EditErr:
  MsgBox Err.Description
End Sub
Private Sub cmdCancel_Click()
  On Error Resume Next

  SetButtons True
  mbEditFlag = False
  mbAddNewFlag = False
  adoPrimaryRS.CancelUpdate
  If mvBookMark > 0 Then
    adoPrimaryRS.Bookmark = mvBookMark
  Else
    adoPrimaryRS.MoveFirst
  End If
  mbDataChanged = False
  botones (False)
End Sub

Private Sub cmdUpdate_Click()
  On Error GoTo UpdateErr

  adoPrimaryRS.UpdateBatch adAffectAll

  If mbAddNewFlag Then
    adoPrimaryRS.MoveLast          'va al nuevo registro
  End If

  mbEditFlag = False
  mbAddNewFlag = False
  SetButtons True
  mbDataChanged = False
  botones (False)
  Exit Sub
UpdateErr:
  MsgBox Err.Description
End Sub

Private Sub cmdClose_Click()
  Unload Me
End Sub

Private Sub cmdFirst_Click()
  On Error GoTo GoFirstError
```

```
  adoPrimaryRS.MoveFirst
  mbDataChanged = False

  Exit Sub

GoFirstError:
  MsgBox Err.Description
End Sub

Private Sub cmdLast_Click()
  On Error GoTo GoLastError

  adoPrimaryRS.MoveLast
  mbDataChanged = False

  Exit Sub

GoLastError:
  MsgBox Err.Description
End Sub

Private Sub cmdNext_Click()
  On Error GoTo GoNextError

  If Not adoPrimaryRS.EOF Then adoPrimaryRS.MoveNext
  If adoPrimaryRS.EOF And adoPrimaryRS.RecordCount > 0 Then
    Beep
     'ha sobrepasado el final; vuelva atrás
    adoPrimaryRS.MoveLast
  End If
  'muestra el registro actual
  mbDataChanged = False

  Exit Sub
GoNextError:
  MsgBox Err.Description
End Sub

Private Sub cmdPrevious_Click()
  On Error GoTo GoPrevError

  If Not adoPrimaryRS.BOF Then adoPrimaryRS.MovePrevious
  If adoPrimaryRS.BOF And adoPrimaryRS.RecordCount > 0 Then
    Beep
    'ha sobrepasado el final; vuelva atrás
    adoPrimaryRS.MoveFirst
  End If
```

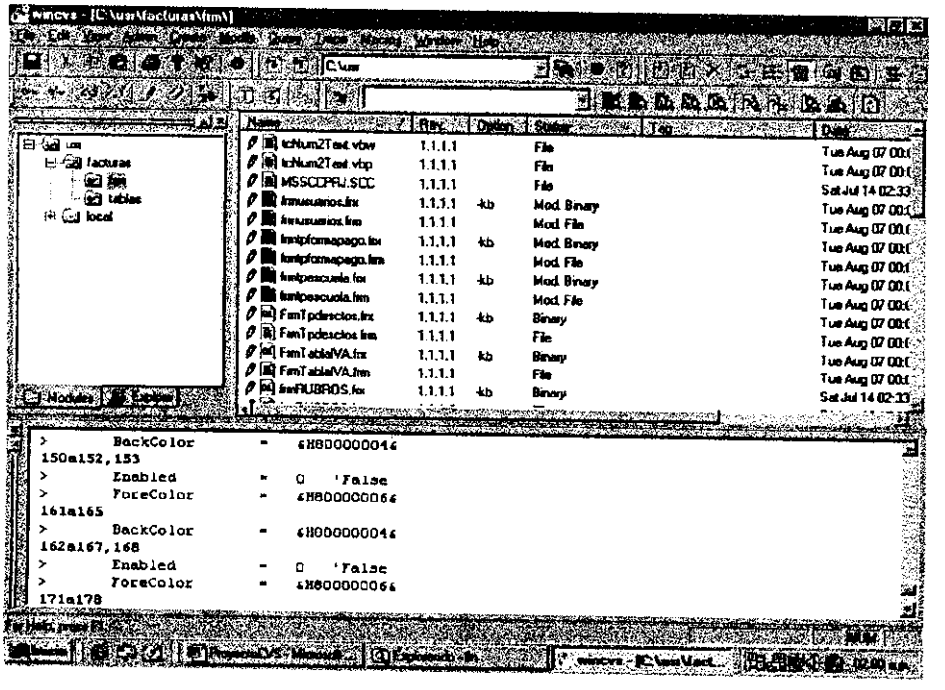
```
  'muestra el registro actual
  mbDataChanged = False

  Exit Sub

GoPrevError:
  MsgBox Err.Description
End Sub

Private Sub SetButtons(bVal As Boolean)
  cmdAdd.Visible = bVal
  cmdEdit.Visible = bVal
  cmdUpdate.Visible = Not bVal
  cmdCancel.Visible = Not bVal
  cmdDelete.Visible = bVal
  cmdClose.Visible = bVal
  cmdRefresh.Visible = bVal
  cmdNext.Enabled = bVal
  cmdFirst.Enabled = bVal
  cmdLast.Enabled = bVal
  cmdPrevious.Enabled = bVal
End Sub
Private Sub botones(estado As Boolean)
 If estado = True Then
    For i = 0 To 2
      txtFields(i).Enabled = True
      txtFields(i).BackColor = &H80000005
    Next i
  Else
    For i = 0 To 2
      txtFields(i).Enabled = True
      txtFields(i).BackColor = &H80000000
    Next i

End If


End Sub
```

VISUALIZACION DE INTERFACE DEL WINCVS

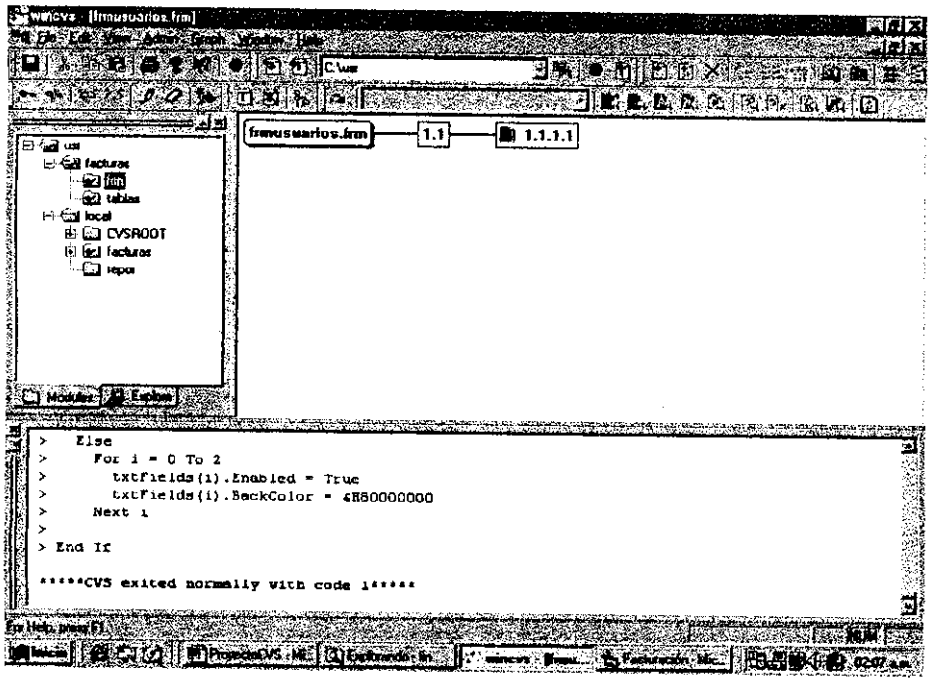
Interface de control del Sistema de Facturación con el Wincvs 1.2. para Windows 9x



Pantalla de cómo de muestras las modificaciones que se van realizando en el proyecto



Visualizando la gráfica de las ramas



BIBLIOGRAFIA

DIRECCIONES DE INTERNET

- **Manual de CVS**
  http://www.loria.fr/~molli/cvs-index.html

- **Manual Software de CVS**
  http://gnu.org/manual/cvs/index.html

- **Manual para utilizar y administrar la versión 1,11 de CVS.**
  http://www.cvshome.org/

- **Guía de desarrollo para Wuf-CVS, una herramienta (front end) para un sistema de control de versiones.**
  http://cuba.xs4all.nl/~tim/scvs/

- **Kerberos**
  http://www.wntmag.com/atrasados/1998/16_ene98/articulos/Especial1.htm

- **Software Wincvs 1.2, 1.3b4**
  http://sourceforge.net

- **TCL – Command Macros**
  http://aspn.activestate.com/ASPN/Downloads/ActiveTcl/.

- **Cvs para Windows NT**
  http://www.cvsnt.org

- **Ayuda y links de CVS**
  http://www.cyclic.com

- **Información, Links, Dowload de CVS**
  http://www.wincvs.org

- **Anomino de CVS, Ayuda de SSH**
  http://www.openbsd.org

- **Ayuda de Visual Basic**
  http://www.msdn.com

- **Información y ejemplos de Programación en Visual Basic**
  http://guille.costasol.net/indice.asp

LIBROS UTILIZADOS:

- **Mastering Database Programming with Visual Basic 6, Evangelos Petroutsos.**