Tutorial de Uso de CVS

Ismael Olea

Ignacio Arenaza

Historial de revisiones
Revisión $Revision: 1.3 $ $Date: 2002/10/08 16:54:29 $
Submitted.

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

Tabla de contenidos

1. Introducción

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

Para ser más concreto, CVS permite a un grupo de desarrolladores trabajar y modificar concurrentemente ficheros organizados en proyectos. Esto significa que dos o más personas pueden modificar un mismo fichero sin que se pierdan los trabajos de ninguna. Además, las operaciones más habituales son muy sencillas de usar.

Como añadido a lo anterior, CVS guarda todas las versiones antiguas de los ficheros. Esto permite recuperar en cualquier momento versiones anteriores a la actual.

Dado que trabaja con ficheros ASCII es igual de útil para trabajar con código fuente de programas o con toda clase de documentos siempre que su formato sea completamente de texto, como pueden ser ficheros sgml/html/xml.

Este documento es una referencia inmediata para trabajar con CVS. A poco que le saque partido necesitará consultar otro más extenso y detallado. Sin embargo el 80% o más de las acciones que desarrollan los usuarios de CVS están documentadas en este artículo.

Con CVS puede trabajarse de forma local (repositorio y copias de trabajo en el mismo sistema) o remota (el repositorio está en un sistema servidor y la copia local en otro que es cliente del primero).

En este artículo sólo prestaré atención al modo de trabajo remoto, que es el mas utilizado habitualmente en los proyectos de software libre, por su *modus operandi* habitual.

2. Terminología CVS

Para que no se pierda, un brevísimo vocabulario:

repositorio	Jerarquía de directorios alojada en el servidor CVS que contiene diferentes módulos a disposición de los usuarios.
módulo	Árbol de directorios que forma parte del repositorio. Cuenta con un nombre identificador gracias al cual podremos trabajar con él de forma selectiva.

3. Invocar a CVS

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

Un aspecto que debe tener en cuenta (sobre todo si este es el primer documento que lee sobre CVS) es que CVS tiene parámetros para cada una de sus órdenes. Para conocerlas tiene dos métodos: invocar CVS como **`cvs help`** o consultar la ayuda de su página del manual.

4. Configuración

Puede usar varios ficheros de configuración que CVS reconocerá y usará.

Entre otros, tenemos los siguientes:

~/.cvsignore

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

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

si los ficheros de este módulo son principalmente documentos escritos en (La)TeX, ya que las extensiones mostradas arriba corresponden a los diferentes ficheros temporales creados durante el procesado del fichero fuente original para obtener el fichero listo para impresión.
D1 Así evitamos *contaminar* el repositorio con ficheros que realmente no es necesario que estén controlados por CVS.

~/.cvsrc

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

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

Significa que queremos que cuando usemos la orden **update**, automáticamente y sin que tengamos que escribir nada más, se añadan los parámetros *-Pd* (borrar los directorios vacíos tras la actualización y crear los directorios del repositorio que no existan en la copia local). Algo similar ocurre con la orden **diff**.
D1 La última línea es especial, en el sentido de que no hay una orden **cvs**. Se trata de la forma en que se indican parámetros globales del propio CVS, no específicos de ninguna orden.

4.1. La autenticación

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

Las más utilizadas son vía **pserver** y vía **ssh** (aunque existen otras dos formas al menos: vía **rsh**, totalmente desaconsejada por motivos de seguridad, y vía Kerberos, que no es habitual por necesitar toda la infraestructura de seguridad de Kerberos).

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

4.1.1. ssh (OpenSSH)

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

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

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

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

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

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

4.1.2. pserver

Si no necesitamos que los datos que se intercambian con el servidor vayan cifrados y nos basta con una autenticacion relativamente segura, podemos utilizar el método pserver.

La ventaja de este método sobre el anterior es doble en este caso:

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

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

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

donde **`USUARIO`** es nuestro nombre de usuario, **`cvs.dominio.org`** es el servidor que aloja al repositorio y **`/var/lib/cvs`** es el directorio del servidor en el que está el repositorio.

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

```
$ cvs login
```

CVS le pedirá la contraseña del usuario que haya configurado en la variable de entorno mencionada arriba. Si la contraseña es correcta CVS guardará la información que necesita en el fichero `~/.cvspass` y no tendrá que volver a autenticarse.

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

```
$ cvs logout
```

4.1.3. Métodos combinados

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

5. Uso

5.1. Modo de uso

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

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

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

5.2. Descarga por primera vez del módulo

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

```
$ cd padre-de-directorio-donde-se-alojará-el-módulo
$ cvs checkout nombre-del-módulo
```

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

A partir de este momento no es necesario configurar las variables de entorno porque CVS sabe a qué repositorio pertenece el módulo con sólo examinar los subdirectorios `CVS`. No se debe modificar nunca esos subdirectorios a

mano. De lo contrario CVS perderá la pista de a que módulo pertenecen los ficheros, cuáles son las versiones de la copia local, etc.

5.3. Actualizar nuestra copia local desde el repositorio

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

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

5.4. Publicar nuestras modificaciones en el respositorio

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

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

Tras lo cual el sistema mostrará la pantalla de un editor de textos (el que tengamos configurado como nuestro favorito en la variable de entorno EDITOR) para que introduzcamos una descripción, lo más significativa posible, del conjunto de cambios realizados en el módulo desde el último **commit**.

5.5. Resolución de conflictos

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

El procedimiento es simple:

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

2. Editamos ese fichero y encontraremos marcas del tipo:

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

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

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

5.6. Añadir ficheros al módulo

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

Si quiere añadir un nuevo fichero o directorio al módulo CVS hay que seguir los siguientes pasos (ademas de crear o copiar el propio fichero al módulo, por supuesto):

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

pero si el fichero es binario hay que tener la precaución de hacer:

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

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

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

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

5.7. Eliminar ficheros del módulo CVS

Para eliminar un fichero del módulo CVS hay que hacer lo siguiente una vez borrado el fichero de la copia local de trabajo (se puede usar **cvs rm** como abreviatura):

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

En cambio, si queremos borrar físicamente los ficheros a la vez que los eliminamos del módulo deberemos usar:

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

6. Cómo configurar un servidor y cómo incorporar nuevos módulos al repositorio.

6.1. Configuración del servidor.

Para configurar un servidor CVS con acceso remoto hay básicamente tres formas (cuatro en las versiones recientes, pero no conozco ningún cliente de autenticación por medio de GSSAPI en el momento de escribir esto, asi que nos quedamos con los tres originales).

rsh/ssh	Esta primera opción implica tener una cuenta equivalente en el servidor CVS y no hace falta tocar ni `inetd.conf` ni ejecutar el servicio pserver. La seguridad de esta solución es nula a menos que se use ssh como sustituto de rsh.

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

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

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

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

(todo en una sola línea, por supuesto; aquí se muestra partido en dos por cuestiones de espacio). El valor `/var/lib/cvs` corresponde al directorio donde ubicaremos el repositorio en el servidor CVS y que denotaremos de ahora en adelante por la variable de entorno `$CVSROOT`.

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

```
cvs: ALL
```

6.2. Clases de usuarios y tipo de acceso permitido.

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

- Adición/actualización de ficheros del repositorio (implica acceso en modo escritura). Aqui van los **commit**, **add**, **remove** y compañia.

- Actualizaciones en el cliente/creación de diferencias. Esto sólo implica acceso en modo lectura. Aqui van los **checkout**, **update**, **diff** y compañia.

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

El acceso en modo lectura no plantea problemas (salvo de carga de la maquina si los **checkout** / **diff** / **update** son masivos).

6.3. Creacion inicial del repositorio

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

1. Configurar `/etc/inetd.conf` para que lance el servidor pserver como se ha comentado arriba, en caso de usar el método de acceso pserver.

2. Definir una cuenta administrativa local para CVS (en el ejemplo se llamará cvsadm) que será quien administre CVS. Esta cuenta será la propietaria de los ficheros de control del repositorio y quien pueda crear nuevos módulos en el mismo. Aprovecharemos para crear un grupo llamado cvs para gestionar más cómodamente los permisos del repositorio y sus módulos.

   El usuario cvsadm deberá ser parte del grupo cvs. Sin embargo esta cuenta no tienen porque tener acceso local al sistema en caso de usar el método de acceso pserver.

3. Crear el repositorio en local en el servidor (en `$CVSROOT`), ejecutando las ordenes (si queremos colocar el respositorio en `/var/lib/cvs`):

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

   El repositorio conviene crearlo como root y tratarlo, desde el punto de vista de los permisos, como si fuera el directorio `/etc`. Todos sus ancestros sólo deberian ser escribibles por root. El propio `$CVSROOT` debería ser escribible por el administrador del servidor CVS (cvsadm, como hemos indicado más arriba). **NADIE MAS** debería tener permisos de escritura en dicho directorio.
   D1 Esto se hace asignando su propiedad al usuario cvsadm y haciendo al grupo cvs el grupo de `$CVSROOT`. Además, sería conveniente que `$CVSROOT` tuviera activado el bit SGID, para que los subdirectorios y ficheros que se creen debajo hereden el grupo principal del fichero o directorio (e indirectamente, por el ajuste del valor de umask que hace el propio CVS en modo pserver, el permiso de escritura para el grupo cvs y que tenga así control sobre todo lo que se introduzca allí). En resumen, `$CVSROOT` debería tener los permisos:

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

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

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

   (*) Esto sólo es necesario si se va a usar la caracteristica `loginfo` con el script **log.pl** o similares. En cualquier caso, no deberia ser problemático.

5. Incorporar al grupo cvs los diferentes usuarios locales que van a representar a los usuarios remotos de pserver. En concreto, en mi sistema, existen cvsadm, cvsuser y anoncvs, que representan respectivamente al administrador del servidor CVS, a los usuarios CVS con permiso de escritura en el repositorio (previa autenticación) y a los usuarios con acceso sólo lectura en modo anónimo.

   La idea de este grupo (como se ha comentado arriba) es poder asignar los permisos locales a los ficheros de forma que tanto el administrador como los usuarios tengan acceso a ellos.

6. Si se va a usar pserver, es muy importante que los usuarios que tengan que ver con CVS (cvsadm, anoncvs y cvsuser) **no puedan** tener acceso local al servidor (esto es, poner una contraseña no válida, un shell no válido, etc.)

7. Los módulos iniciales para cada proyecto (manual, como, programa, etc.) los debe crear el administrador del servidor CVS. Luego los puede usar cualquiera. Así garantizamos que los permisos son los adecuados, y que todo esta «controlado y en orden».

8. Para funcionalidades adicionales (envio de mensaje de corre cada vez que se hace un **commit**, actualizacion de un log de **commit** en el repositorio, publicacion en el web del estado del repositorio, etc.) el propio administrador del servidor CVS se puede encargar de todo. Todo lo necesario esta en la documentación oficial de CVS.

Todo lo anterior (especialmente el tema de permisos, usuarios necesarios, etc.) son conclusiones mías despues de leer la documentacion de CVS y su FAQ. No está indicado en ningún sitio que se deba hacer exactamente así, ya que sólo se dan consejos muy generales y bastante dispersos por toda la documentación. Con esto quiero decir que habría que hacer un análisis un poco más profundo sobre el posible impacto de seguridad.

6.4. Alta de usuarios

6.4.1. Para el método de acceso ssh

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

6.4.2. Para el método de acceso pserver

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

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

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

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

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

Esta funcionalidad es muy interesante, ya que nos permite crear únicamente una cuenta de usuario local en el servidor por cada tipo de acceso diferente necesario, y agregar después cuantas cuentas remotas queramos y mapearlas al usuario local correspondiente. Como el fichero donde se realiza el mapeo es `$CVSROOT/ROOT/passwd` y este fichero está disponible a través del propio CVS, el administrador del servidor CVS no necesita permisos de administrador global de la máquina donde este ubicado el repositorio para dar de alta nuevos usuarios de CVS. Nótese que en este caso, el fichero con las contraseñas viaja sin cifrar por la red, con lo cual se recomienda usar el método de acceso ssh para este tipo de operaciones).

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

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

(*) Si fuese el primer usuario del repositorio, el fichero `passwd` no existirá, y por tanto será necesario usar la opción *`-c`* de la orden **htpasswd**. De igual modo, habrá que indicar a CVS que existe un nuevo fichero llamado `passwd` para que lo tenga en cuenta y lo añada al repositorio al hacer el **commit**.

(**) Si se desea la funcionalidad de mapear el usuario remoto a un usuario local concreto, se puede editar el fichero a mano y añadir el tercer campo, separándolo del segundo por ':'.

(***) Para que esto funcione es necesario haber incluido el nombre del fichero `passwd` en el fichero `$CVSROOT/checkoutlist` previamente. (no se aconseja por motivos de seguridad a menos que el acceso administrativo a CVSse realice por medio de ssh, como se ha comentado más arriba y se ha hecho en el ejemplo mostrado).

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

```
cvsadm
hermes-team
```

```
lucasiano
```

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

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

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

6.5. Añadir nuevos módulos al repositorio

Para añadir nuevos modulos al respositorio, la operación debe ser llevada a cabo por el administrador del mismo. Para ello se usa la orden **cvs import** que le indica a CVS que debe crear una copia en el respositorio del conjunto de ficheros del directorio actual.

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

```
cvs import nombre-módulo etiqueta-vendedor etiqueta-versión
```

Donde:

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

7. Bibliografía

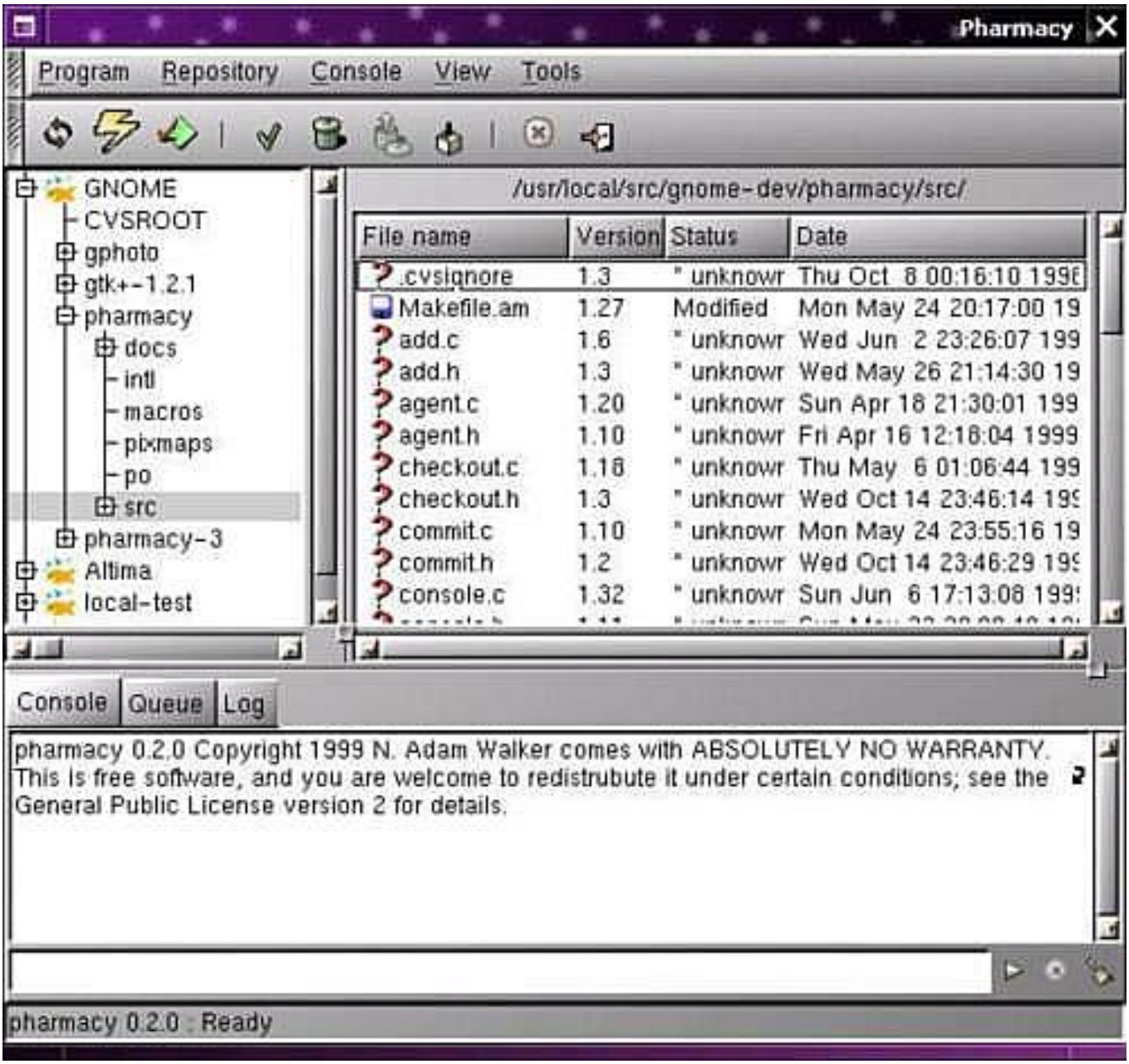
Por un lado hay un completísimo fichero info dedicado a CVS, que es la documentación oficial de mismo. Si usa GNU/Linux es muy probable que ya lo tenga instalado en su sistema.

Por otro lado en http://cvsbook.red-bean.com/ está disponible otro libro documentando CVS.

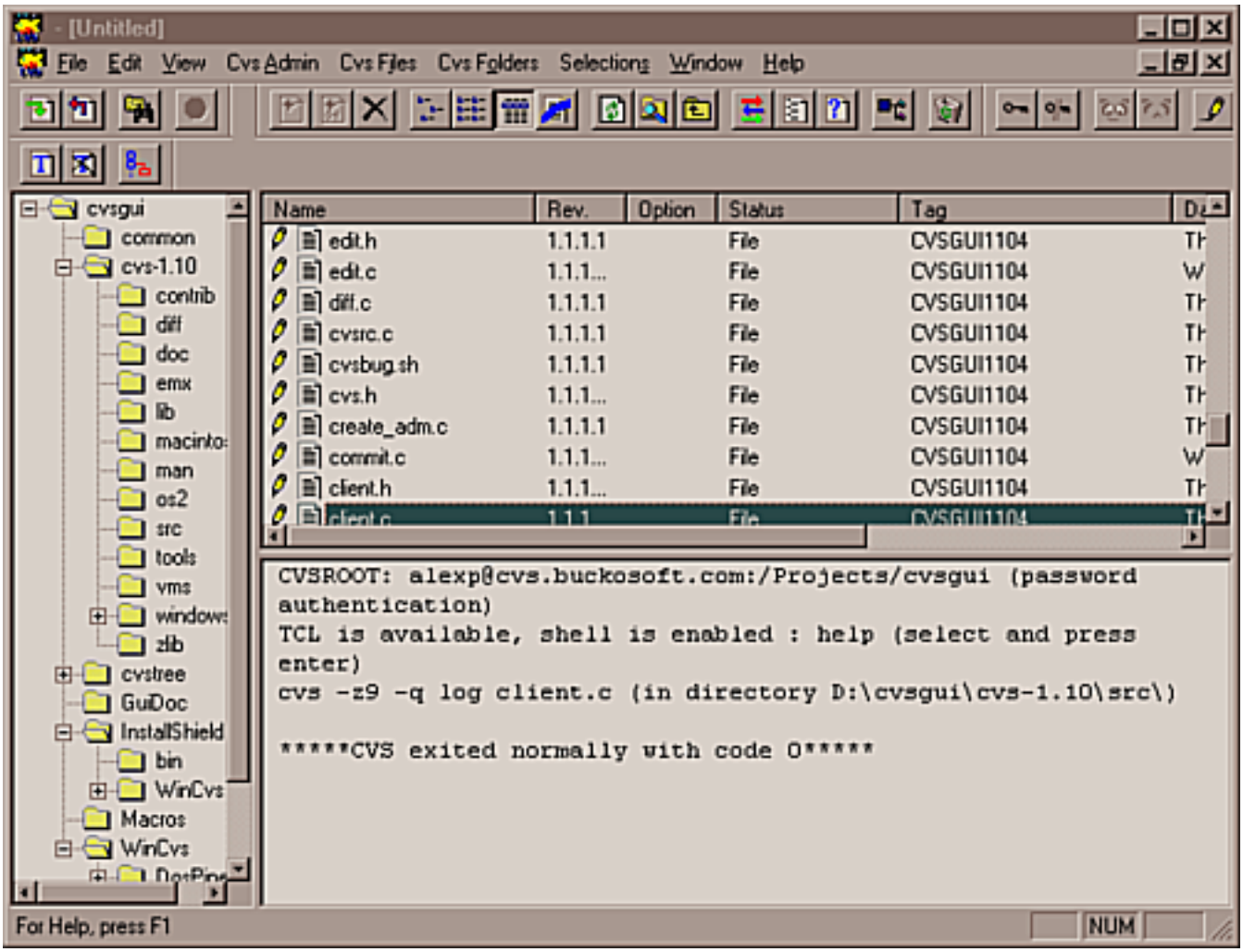
8. Interfaces de usuario para CVS

He aquí algunas interfaces de usuario para CVS, en mayor o menor estado de desarrollo. Todas ellas se pueden encontrar en http://freshmeat.net/.

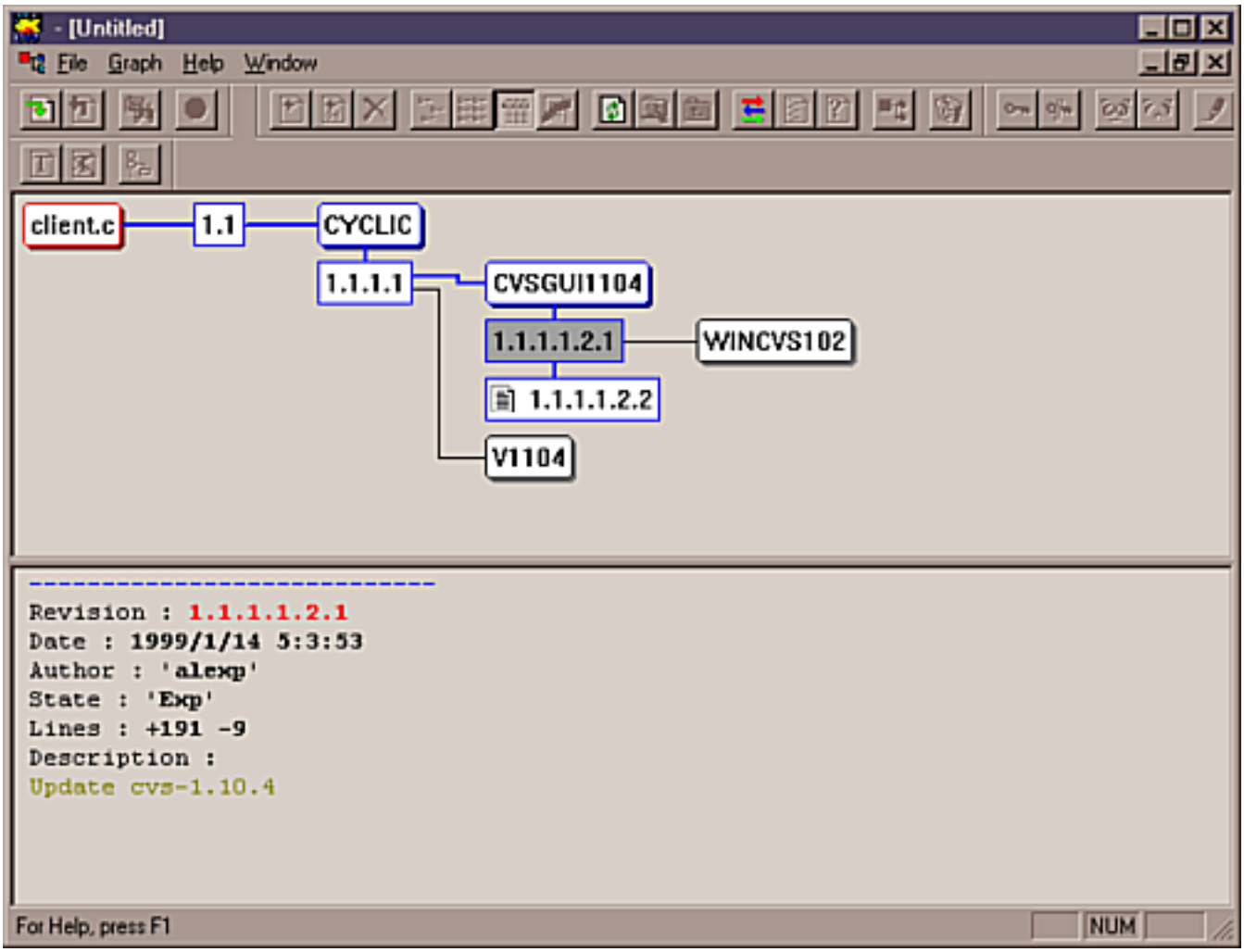
- pharmacy: Una interfaz GNOME para CVS, disponible en http://pharmacy.sourceforge.net/ [http://pharmacy.sourceforge.net/]. Se encuentra aun en un estado de desarrollo bastante temprano y no se actualiza desde hace casi un año en el momento de escribir esto. Aquí se puede ver un pantallazo del aspecto de su pantalla principal:


Pharmacy
Program
Repository
Console
View
Tools
GNOME
CVSROOT
gphoto
gtk+-1.2.1
pharmacy
docs
intl
macros
pixmaps
po
src
pharmacy-3
Altima
local-test
/usr/local/src/gnome-dev/pharmacy/src/
File name
Version
Status
Date
.cvsignore 1.3 * unknowr Thu Oct 8 00:16:10 199
Makefile.am 1.27 Modified Mon May 24 20:17:00 19
add.c 1.6 * unknowr Wed Jun 2 23:26:07 199
add.h 1.3 * unknowr Wed May 26 21:14:30 19
agent.c 1.20 * unknowr Sun Apr 18 21:30:01 199
agent.h 1.10 * unknowr Fri Apr 16 12:18:04 1999
checkout.c 1.16 * unknowr Thu May 6 01:06:44 199
checkout.h 1.3 * unknowr Wed Oct 14 23:46:14 19
commit.c 1.10 * unknowr Mon May 24 23:55:16 19
commit.h 1.2 * unknowr Wed Oct 14 23:46:29 19
console.c 1.32 * unknowr Sun Jun 6 17:13:08 199
Console
Queue
Log
pharmacy 0.2.0 Copyright 1999 N. Adam Walker comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistrubute it under certain conditions; see the
General Public License version 2 for details.
pharmacy 0.2.0 : Ready

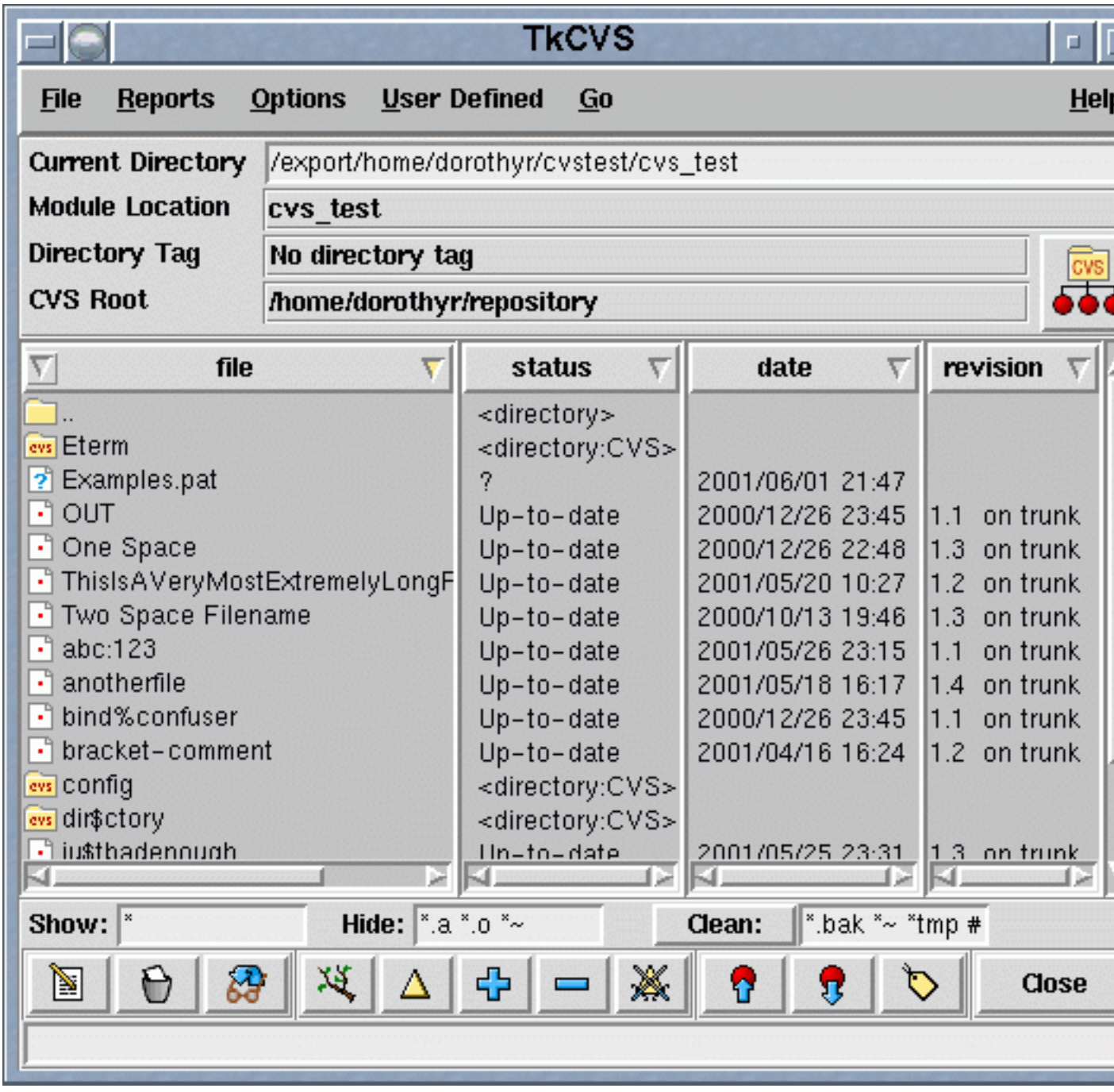
• cvsgui: Una interfaz multiplataforma para CVS escrita en C++ (anteriormente conocida como gCVS), disponible en http://sourceforge.net/projects/cvsgui/ [http://sourceforge.net/projects/cvsgui]. Su pantalla principal,



y el visor de ramas y versiones de los ficheros:
D1



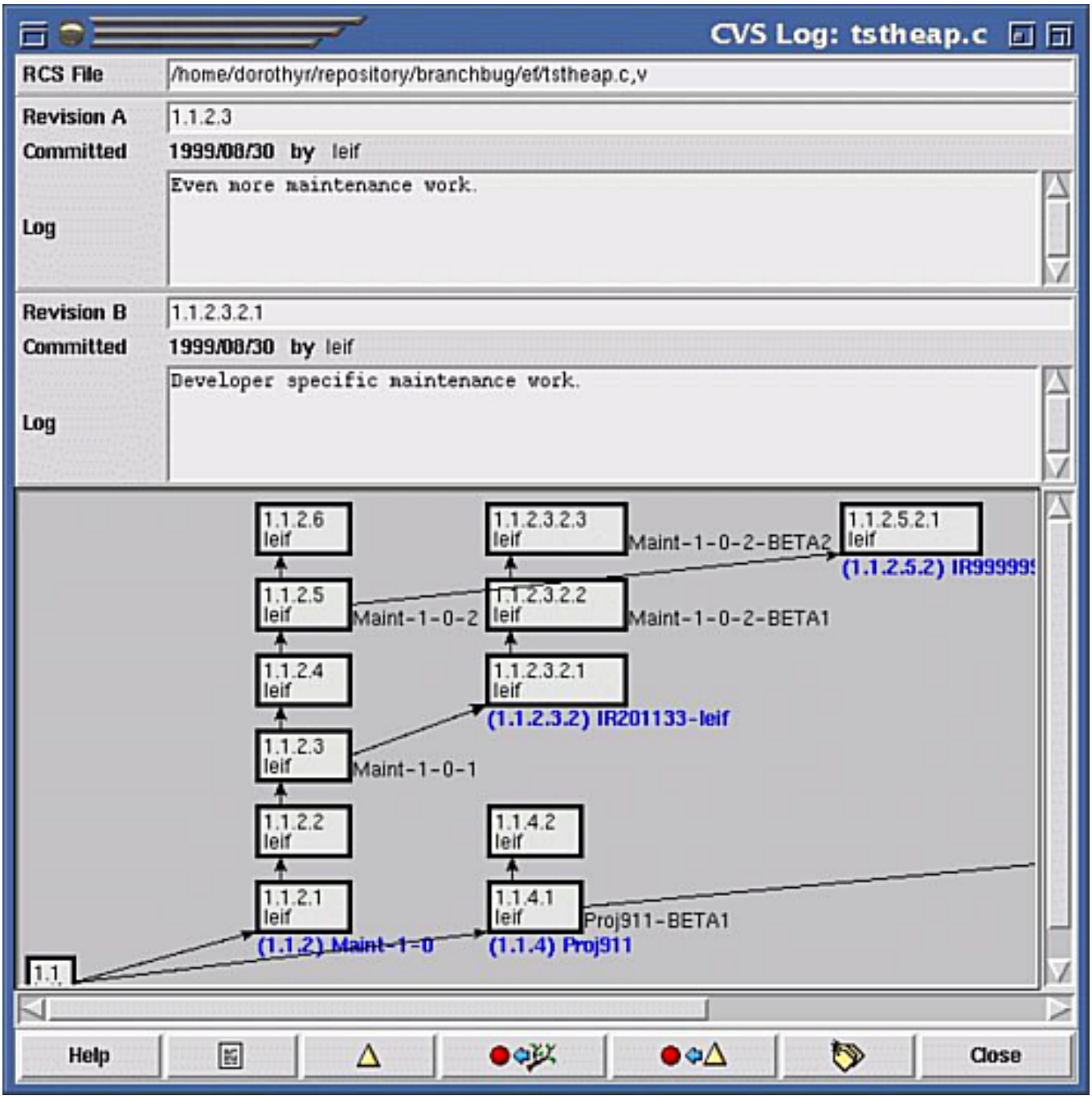
- tkcvs: Interfaz gráfica para CVS escrita en Tcl/Tk muy establecida y estable, disponible en http://www.twobarleycorns.net/tkcvs.html [http://www.twobarleycorns.net/tkcvs.html]. El pantallazo de su ventana principal,


TkCVS
File
Reports
Options
User Defined
Go
Help
Current Directory
/export/home/dorothyr/cvstest/cvs_test
Module Location
cvs_test
Directory Tag
No directory tag
CVS Root
/home/dorothyr/repository
CVS
file
status
date
revision
..
<directory>
Eterm
<directory:CVS>
Examples.pat
?
2001/06/01 21:47
OUT
Up-to-date
2000/12/26 23:45
1.1 on trunk
One Space
Up-to-date
2000/12/26 22:48
1.3 on trunk
ThisIsAVeryMostExtremelyLongF
Up-to-date
2001/05/20 10:27
1.2 on trunk
Two Space Filename
Up-to-date
2000/10/13 19:46
1.3 on trunk
abc:123
Up-to-date
2001/05/26 23:15
1.1 on trunk
anotherfile
Up-to-date
2001/05/18 16:17
1.4 on trunk
bind%confuser
Up-to-date
2000/12/26 23:45
1.1 on trunk
bracket-comment
Up-to-date
2001/04/16 16:24
1.2 on trunk
config
<directory:CVS>
dir$ctory
<directory:CVS>
Show:
*
Hide:
*.a *.o *~
Clean:
*.bak *~ *tmp #
Close

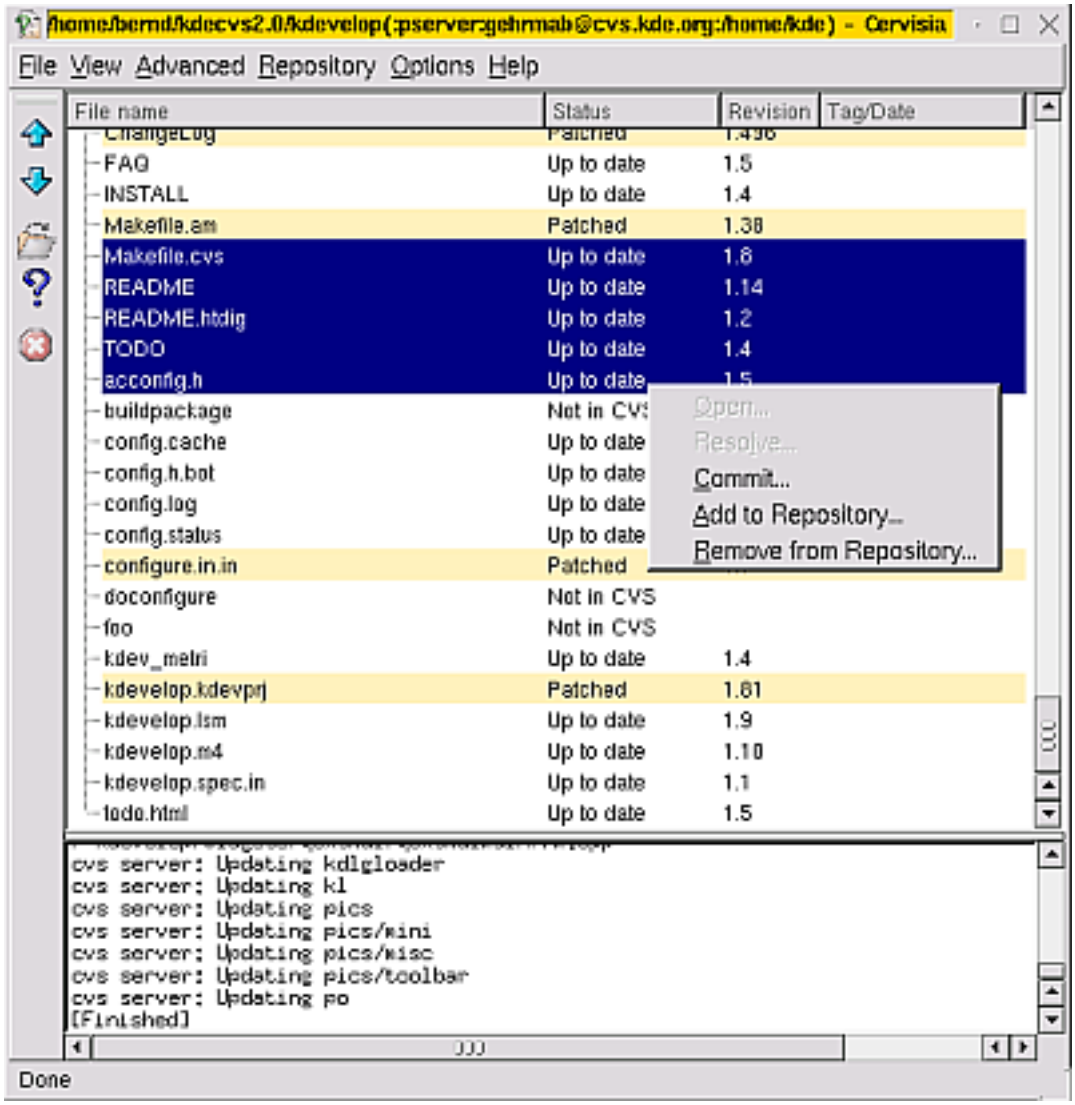
el navegador de módulos disponibles en el repositorio,
D1


Module Browser
File
Reports
Checkouts
Patch
Tag
Help
Module
Vendor
CVSROOT
/home/dorothyr/repository
Current Directory
/export/home/dorothyr/cvstest/cvs_test
Tag A
HEAD
Tag B
Module
Information
CVSROOT
CVS Administrative Files
Examples
Vendor merge example
Local
Local code used for testing Merge
Vendor
Vendor code used for testing Merge
bin
Bin
branchbug
Electric Fence
cheatsheets
Unix How-To
chocolate
Top Chocolate
cocoa
Alias for chocolate/truffle/cocoa3
composite
Two sub-modules
cvs_test
Junk for testing CVS
html
Some HTML Files
inscan
html inscan-29713.html help.gif
Close

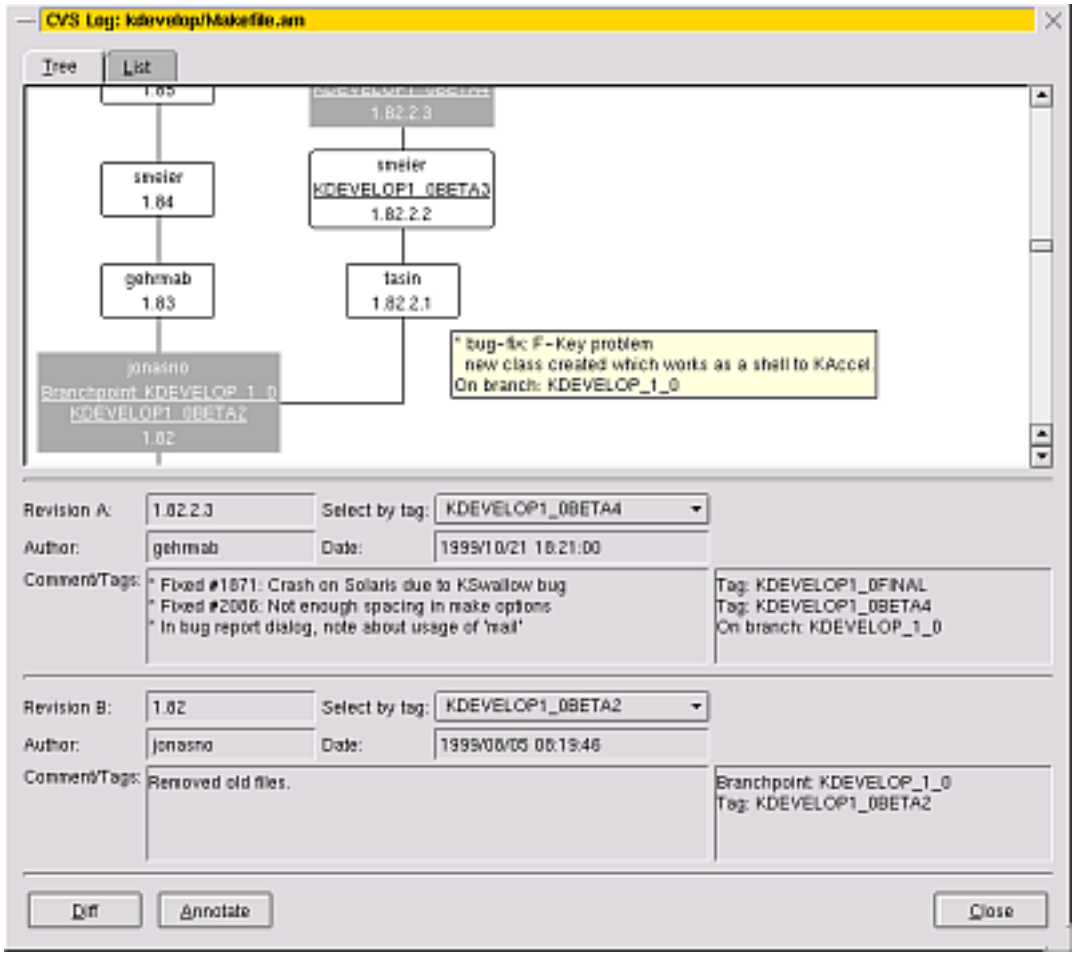
y el visor de ramas y versiones de los ficheros:


CVS Log: tstheap.c
RCS File
/home/dorothyr/repository/branchbug/ef/tstheap.c,v
Revision A
1.1.2.3
Committed
1999/08/30 by leif
Log
Even more maintenance work.
Revision B
1.1.2.3.2.1
Committed
1999/08/30 by leif
Log
Developer specific maintenance work.
1.1.2.6 leif
1.1.2.3.2.3 leif
Maint-1-0-2-BETA2
1.1.2.5.2.1 leif
(1.1.2.5.2) IR99999
1.1.2.5 leif
Maint-1-0-2
1.1.2.3.2.2 leif
Maint-1-0-2-BETA1
1.1.2.4 leif
1.1.2.3.2.1 leif
(1.1.2.3.2) IR201133-leif
1.1.2.3 leif
Maint-1-0-1
1.1.2.2 leif
1.1.4.2 leif
1.1.2.1 leif
1.1.4.1 leif
Proj911-BETA1
(1.1.2) Maint-1-0
(1.1.4) Proj911
1.1
Help
Close

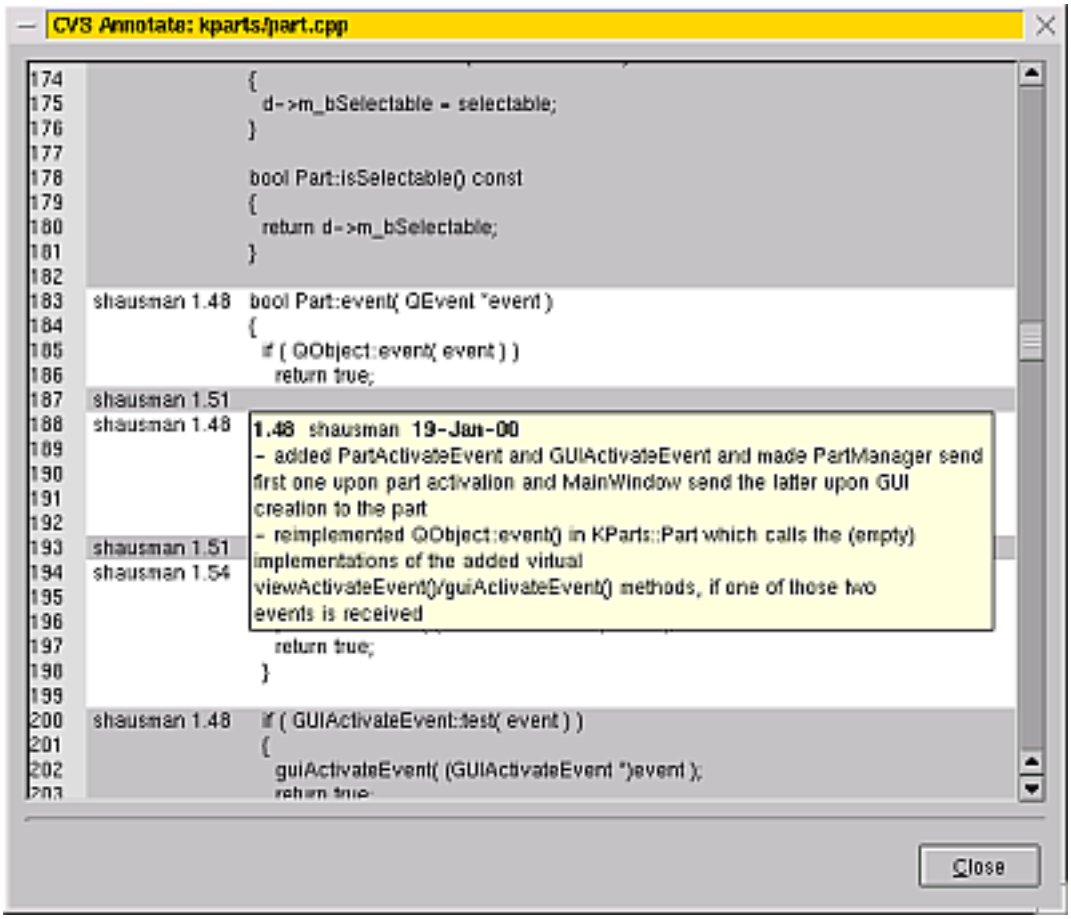
• cervisia: Interfaz gráfica KDE para CVS, disponible en http://cervisia.sourceforge.net/ [http://cervisia.sourceforge.net/]. El pantallazo de su ventana principal,



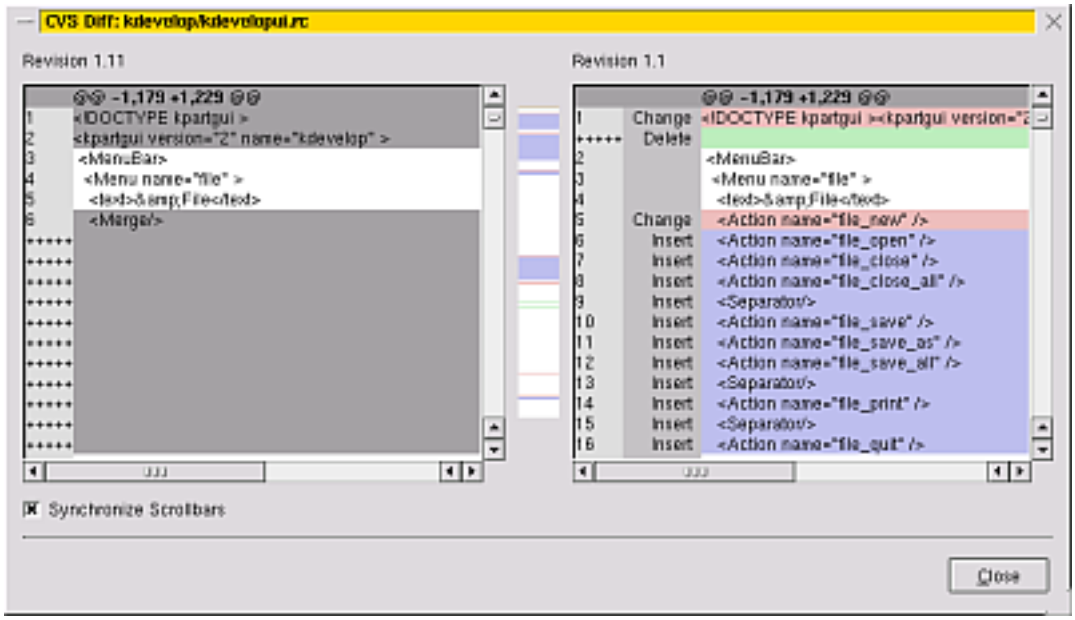
el visor de ramas y revisiones de los ficheros:
D1



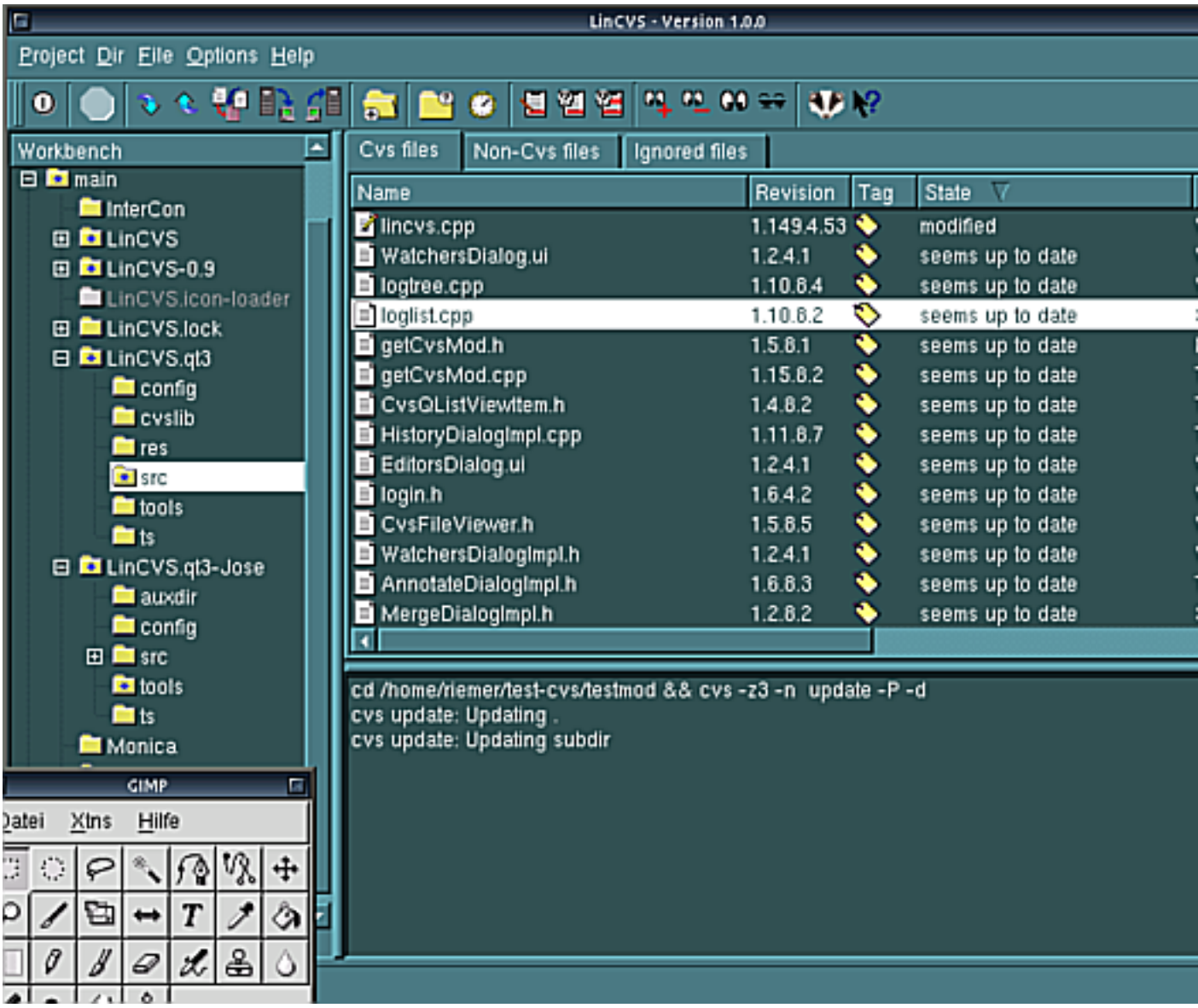
el visor de anotaciones



y el visor gráfico de diferencias:



• lincvs: Interfaz gráfica para CVS que usa la biblioteca grafica Qt, disponible en http://www.lincvs.org/. Su pantalla principal:



- PCL-CVS: extensión de (X)Emacs que permite manipular ficheros gestionados con CVS de forma automática y transparente, disponible como parte de XEmacs, y como parte del propio CVS.

9. Otros recursos CVS interesantes

- La página del propio CVS que ahora está en http://www.cvshome.org/.
- cvs2cl.pl: que es una herramienta para crear ficheros Changelog al estilo GNU y que puede encontrarse en http://www.red-bean.com/cvs2cl/ [http://www.red-bean.com/cvs2cl/]

10. Otros recursos CVS más avanzados

- cvsadmin: Herramienta para administrar las cuentas de un repositorio, disponible en http://www.cooptel.qc.ca/~limitln/cvsadmin/ [http://www.cooptel.qc.ca/~limitln/cvsadmin/]
- cvsauth: conjunto de parches para CVS que sirven para autenticar usuarios sin tener que ejecutar en el servidor CVS como root. Añade además cifrado SSL en las conexiones pserver tradicionales. Disponible en http://cvsauth.sourceforge.net/ [http://cvsauth.sourceforge.net/]

11. Licencia