Arquitectura de sistemas

Abelardo Pardo

University of Sydney
School of Electrical and Information Engineering
NSW, 2006, Australia
Autor principal del curso de 2009 a 2012

Iria Estévez Ayres

Damaris Fuentes Lorenzo

Pablo Basanta Val

Pedro J. Muñoz Merino

Hugo A. Parada

Derick Leony

Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática
Avenida Universidad 30, E28911 Leganés (Madrid), España

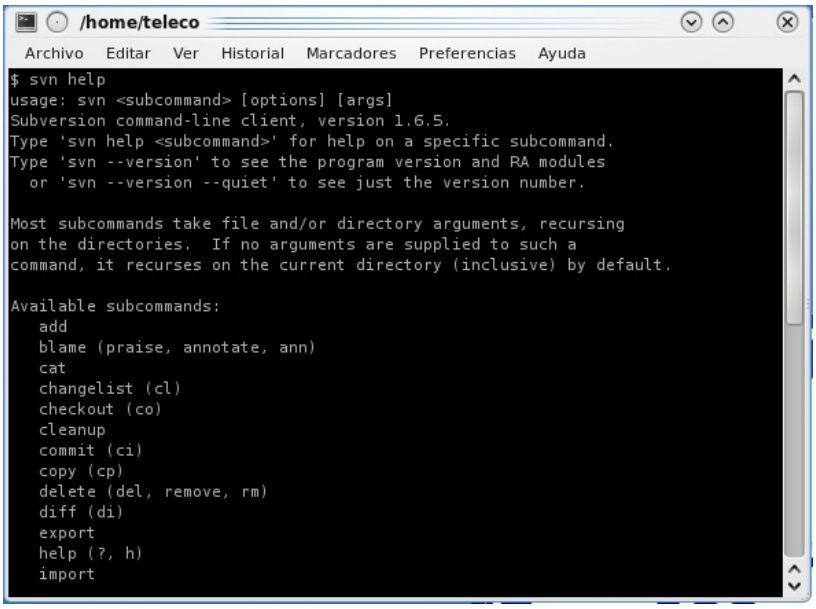
Capítulo 13. El controlador de versiones Subversion

Tabla de contenidos

Subversion es un "controlador de versiones", esto es, una aplicación para guardar y compartir entre varios usuarios múltiples copias de un conjunto de directorios y ficheros en un depósito central. Para poder utilizar esta aplicación se necesitan dos programas.

El primero de ellos es el "servidor remoto" que contiene la copia central de los datos y se ejecuta en un ordenador al que te conectas a través de la red. El segundo programa es el "cliente", una aplicación que ejecutas en tu ordenador. Se conecta con el depósito remoto e intercambia los ficheros en ambos sentidos, del depósito a tu carpeta local para obtener nuevas versiones, y de tu carpeta al depósito para enviar tus cambios.

Abre una ventana de comandos en tu equipo de trabajo y ejecuta el comando **svn help**, se mostrará un mensaje similar al que aparece en la siguiente figura.



El programa cliente se invoca siempre con el nombre (**svn**) seguido de la acción que queremos ejecutar, y el lugar del depósito sobre el que queremos hacer esa operación.

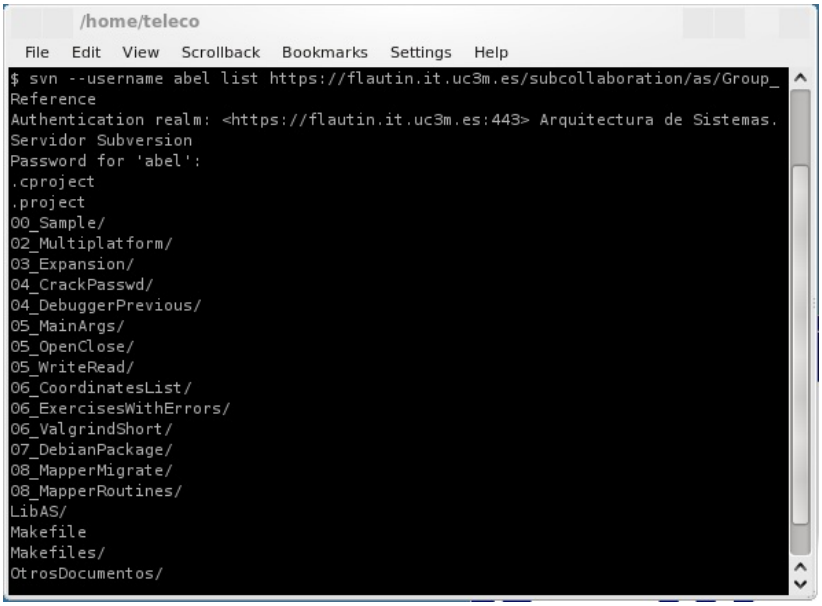
13.1. Conexión con el depósito remoto

Las operaciones más frecuentes son dos: obtener la última versión de los ficheros del depósito remoto, y mandar los cambios que has hecho en tus ficheros al depósito remoto. Pero antes de ejecutar estos comandos se puede ver el contenido del depósito utilizando la orden "list" de la siguiente forma:

```
svn --username [TU NIA] list [URL DEL DEPÓSITO]
```

Reemplaza "`[TU NIA]`" por tu NIA (9 dígitos sin el sufijo **@alumnos.uc3m.es**), y "`[URL DEL`

DEPÓSITO]" por la URL que se te ha mandado por correo electrónico al crearte tu espacio en el servidor. El comando pide a continuación código y clave. Debes utilizar los mismos datos que al entrar en CampusGlobal. A continuación se muestra por pantalla el contenido del directorio especificado en el depósito remoto tal y como se muestra en la siguiente pantalla.



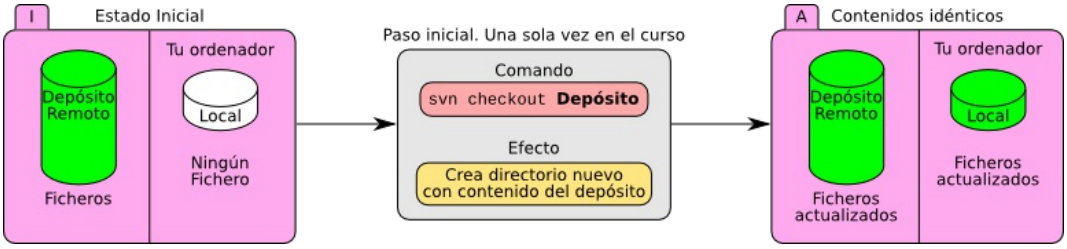
En caso de que este comando no se ejecute correctamente, debes revisar tu conectividad de red, tu nombre de usuario/clave (de CampusGlobal) y verificar que son correctas.

13.2. Descarga la primera copia: checkout

El primer paso para trabajar con un directorio gestionado por Subversion es obtener la primera copia de todos los ficheros del depósito remoto. Esta operación se ejecuta con la orden "**checkout**" seguido de la URL en la que está el depósito, tal y como se muestra en el siguiente ejemplo:

```
svn --username [TU NIA] checkout [URL DEL DEPÓSITO]
```

De nuevo debes reemplazar "[TU NIA]" por tu NIA (9 dígitos sin el sufijo **@alumnos.uc3m.es**), y "[URL DEL DEPÓSITO]" por la URL que se te ha mandado por correo electrónico al crearte tu espacio en el servidor. Tras la ejecución verás que se ha creado en tu carpeta actual una subcarpeta con nombre igual al último sufijo de la URL del depósito. La siguiente figura muestra esta operación de manera gráfica.

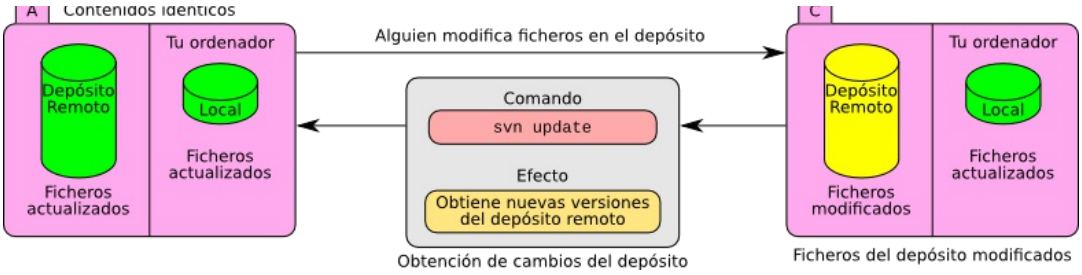


13.3. Descarga nuevas versiones del depósito central: update

Cuando se comparten ficheros entre varios usuarios (o se mantienen varias copias en diferentes máquinas) es posible que el servidor central tenga una copia más reciente que los almacenados en tu ordenador. En este caso es necesario "actualizar" nuestra copia para que tenga los cambios que hay en el depósito. Esta operación se invoca con la orden **update** tal y como se muestra en el siguiente ejemplo:

```
svn --username [TU NIA] update
```

Este comando se debe ejecutar desde una carpeta que contiene ficheros gestionados por Subversion (de ahí que no haga falta especificar la URL del depósito remoto). Lo típico es comenzar una sesión de trabajo con este comando para cerciorarnos de que tenemos la versión más actualizada de los ficheros. La siguiente figura muestra la evolución de las copias locales y remotas cuando se aplica el comando **update**.



Cuando se ejecuta este comando se actualizan todos los ficheros de las subcarpetas de forma recursiva. Para cada fichero que procesa (los que no tienen cambios se ignoran), se imprime una letra en la primera columna de la pantalla cuyo significado es:

- A: "Añadido". El fichero ha sido añadido en la remota y por tanto se ha añadido también a la local.
- D: "Borrado". El fichero ha sido borrado en la copia remota, y por tanto, también ahora en la local.
- U: "Actualizado". Se ha actualizado la copia local con una más reciente del depósito remoto.
- E: "Ya existente". Hay un fichero nuevo creado en la copia local, pero que no ha sido incorporado a subversion, y se quiere descargar otro de igual nombre desde la copia remota. Lo recomendable es borrar la copia local para que la remota se descargue.
- G: "Mezclado". La copia remota tenía cambios, y la local también, pero se han aplicado ambos sin problema y se tiene ahora una copia local, todavía modificada, pero con los cambios del depósito remoto.
- C: "Conflicto". Se han detectado cambios en la copia local y remota de un fichero, y no se pueden mezclar automáticamente. El fichero local se modifica para incluir las dos versiones del cambio una a continuación de la otra para que el usuario lo resuelva. Hay que editar el fichero, buscar la marca "<<<<<" y mezclar a mano las dos versiones.

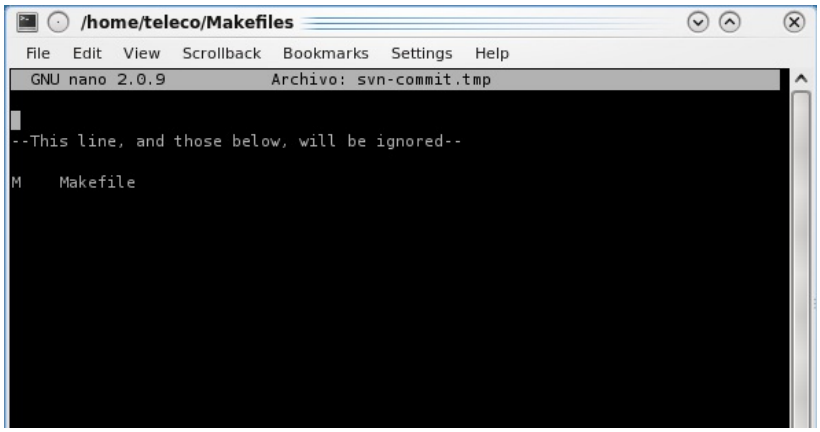
Para evitar los conflictos que surgen al mezclar cambios, se pueden seguir dos reglas sencillas: evitar que dos personas trabajen a la vez en el mismo fichero, y en caso de que esto no sea posible, enviar a menudo los cambios al depósito y descargarse las últimas versiones más a menudo para que los cambios puedan ser mezclados de forma automática. Si tras actualizar tu copia local se ha producido algún conflicto, lee la sección 13.9 para ver en detalle cómo proceder.

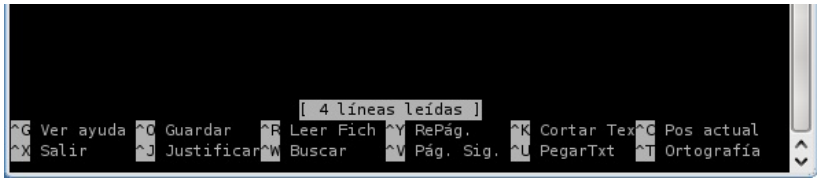
13.4. Envío de cambios al depósito central: commit

Cuando modificas algún fichero, tu copia local y la remota tienen datos diferentes. En este caso los cambios en tu copia local se deben incorporar al depósito remoto mediante la operación **commit**. Posiciona el intérprete en la carpeta en la que están los cambios que quieres enviar y ejecuta la orden "`commit`" tal y como se muestra en el siguiente ejemplo.

```
svn --username [TU NIA] commit
```

Si quieres mandar los cambios de sólo algunos ficheros, puedes añadir sus nombres separados por espacios a continuación de `commit`. Si no pones ningún nombre, el comando se aplica a la carpeta en la que se ejecuta. A continuación aparece un editor para que introduzcas un comentario tal y como se muestra en la siguiente figura.





No es necesario que pongas la fecha, la hora ni los ficheros que has modificado, esta información la añade Subversion automáticamente. Debes incluir un texto en el que expliques al resto de personas que comparten la carpeta, en qué consisten tus cambios.

Para mayor agilidad, y para evitar la salida por pantalla del editor de texto, puedes incorporar el mensaje directamente al realizar el `commit` de la siguiente manera:

```
svn --username [TU NIA] commit -m "Un mensaje cualquiera"
```

13.5. Comprobar el estado de los ficheros: status

El comando **status** nos muestra, el estado de la copia local de cada fichero especificado en la línea de comandos, o de una carpeta con sus subcarpetas. De manera parecida a lo que hace el comando **update**, este comando muestra el estado mediante unas letras al comienzo de la línea (por ejemplo: A, C, D, I, M, R, X para añadido, en conflicto, borrado, ignorado, modificado, reemplazado, sin controlar). Consultar el convenio para estos símbolos mediante el comando:

```
svn help status
```

13.6. Añadir ficheros al control de versiones: add

El control de versiones sólo gestiona aquellos ficheros o carpetas que se añadan con el comando **add**. Si creas un nuevo fichero en tu copia local, será ignorado por Subversion hasta que ejecutes este comando tal y como se muestra en el siguiente ejemplo.

```
svn --username [TU NIA] add [FICHERO]
```

Debes reemplazar "[FICHERO]" por el nombre de los ficheros o carpetas que quieras añadir. Esta operación sólo añade el fichero a la gestión del control de versiones en la copia local. Este cambio no se transmite al depósito hasta que se ejecuta la orden **commit**.

13.7. Otras operaciones con Subversion

Subversion permite más operaciones sobre los ficheros que controla. A continuación se mencionan algunas de ellas. Para una descripción detallada, consultar el manual de la aplicación.

- Ignorar ficheros. Se pueden apuntar aquellos ficheros que deben ser ignorados por Subversion. Esto se consigue asignando los nombres de estos ficheros a la propiedad `svn:ignore` mediante el comando **propset**. Por ejemplo, el comando

  ```
  svn propset svn:ignore 'f1.o f2.o f3.o' .
  ```

  marca los ficheros `f1.o`, `f2.o` y `f3.o` para que Subversion los ignore en todas las operaciones.

- Historia. El comando **log** muestra el histórico de versiones y comentarios de un fichero o carpeta.

- Diferencias entre dos versiones. Esta operación compara dos versiones de un fichero, una de ellas puede ser la copia local. Por ejemplo, los comandos:

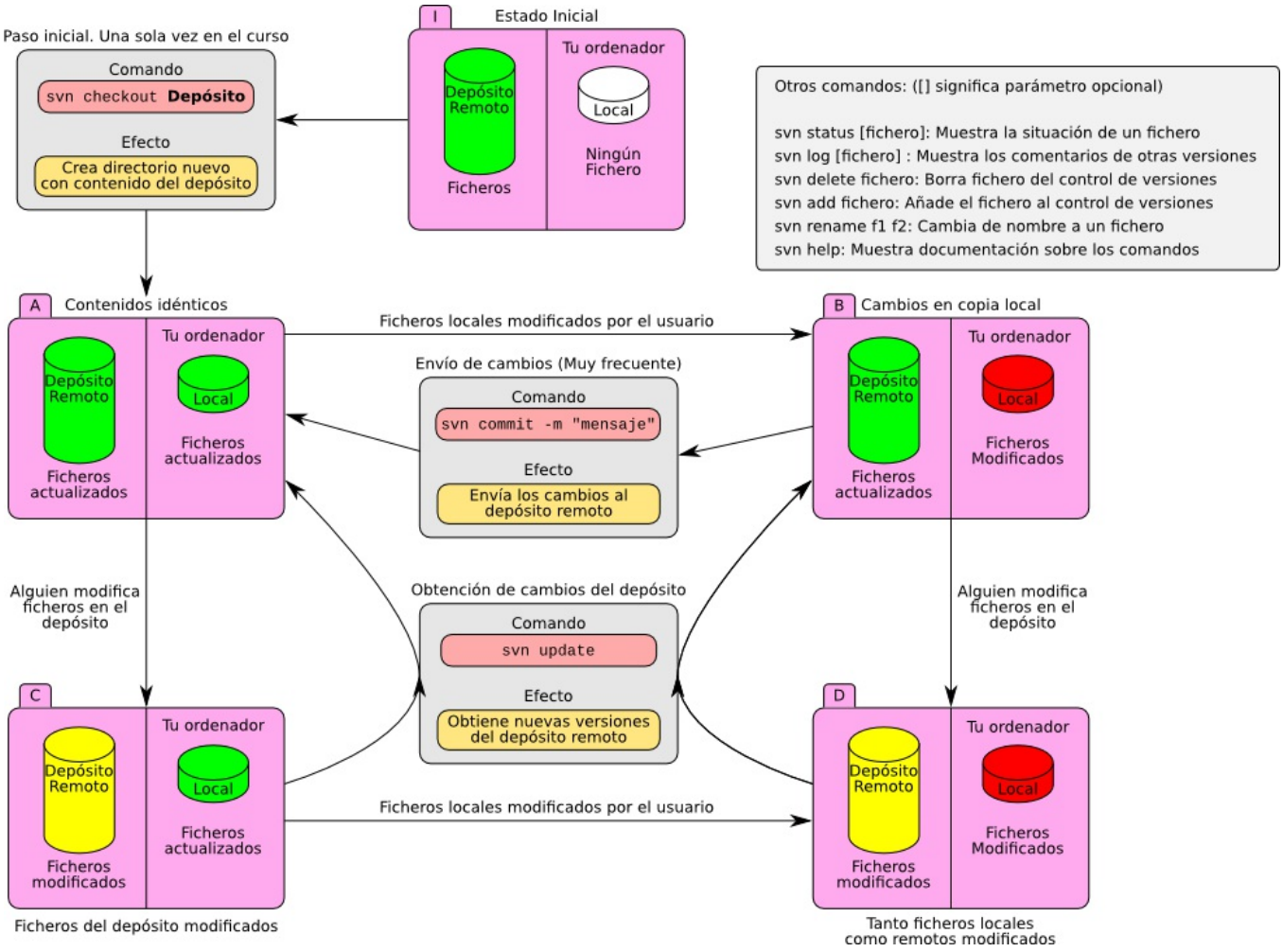
  ```
  svn diff f.c
  svn diff -r1534:1535 f.c
  ```

  muestran primero las diferencias entre la versión local de `f.c` y la copia en el depósito, y a continuación las diferencias entre las versiones 1534 y 1535 de ese fichero.

13.8. Resumen de las operaciones de Subversion

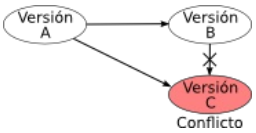
La siguiente figura muestra las principales operaciones de Subversion y la transición entre los diferentes estados. Fíjate que si tienes una versión modificada de ficheros locales, y que también

están modificados en el depósito remoto, primero has de ejecutar el comando **update** y a continuación **commit** para volver a tener el contenido idéntico al del depósito.



13.9. Resolución de conflictos con kdiff3

Supongamos que en el depósito remoto de Subversion existe una versión que denominaremos A de un fichero. Un miembro del equipo hace unos cambios en una zona del fichero y envía estos cambios al depósito con el comando **commit**. A esta nueva versión le llamaremos B. Al mismo tiempo, en tu copia local, que es idéntica a la versión A del depósito remoto, haces unos cambios en la misma zona del fichero que tu compañero. Cuando intentas mandar estos cambios al depósito, la operación marca el fichero con la letra "C", se ha encontrado un conflicto (ver sección 13.4). Subversion ha intentado mezclar los cambios de tu compañero con los tuyos, pero no ha sido capaz, con lo que deja el fichero modificado y preparado para que tú resuelvas esta mezcla. La siguiente figura ilustra esta situación.



Supongamos que el fichero en el que ha aparecido el conflicto se llama `writenumbers.c`. Tras crear la versión C (copia local modificada) y enviar los cambios al depósito con el comand **commit** se nos informa de que el fichero está sin actualizar (hay una nueva versión). Al actualizar con el comando **update** se produce el conflicto y se ofrecen varias posibilidades a lo que respondemos con la letra "p" para posponer la resolución. Si consultamos el estado del fichero con el comando **status**, vemos que refleja el conflicto. Además, en el directorio se han creado tres ficheros adicionales: `writenumbers.c.mine`, y dos ficheros más con el sufijo "r" seguido de un número (que es el número de versión). La siguiente figura muestra esta situación.

```
$ svn status writenumbers.c
C       writenumbers.c
$ ls writenumbers.c*
writenumbers.c  writenumbers.c.mine  writenumbers.c.r151  writenumbers.c.r153
$
```

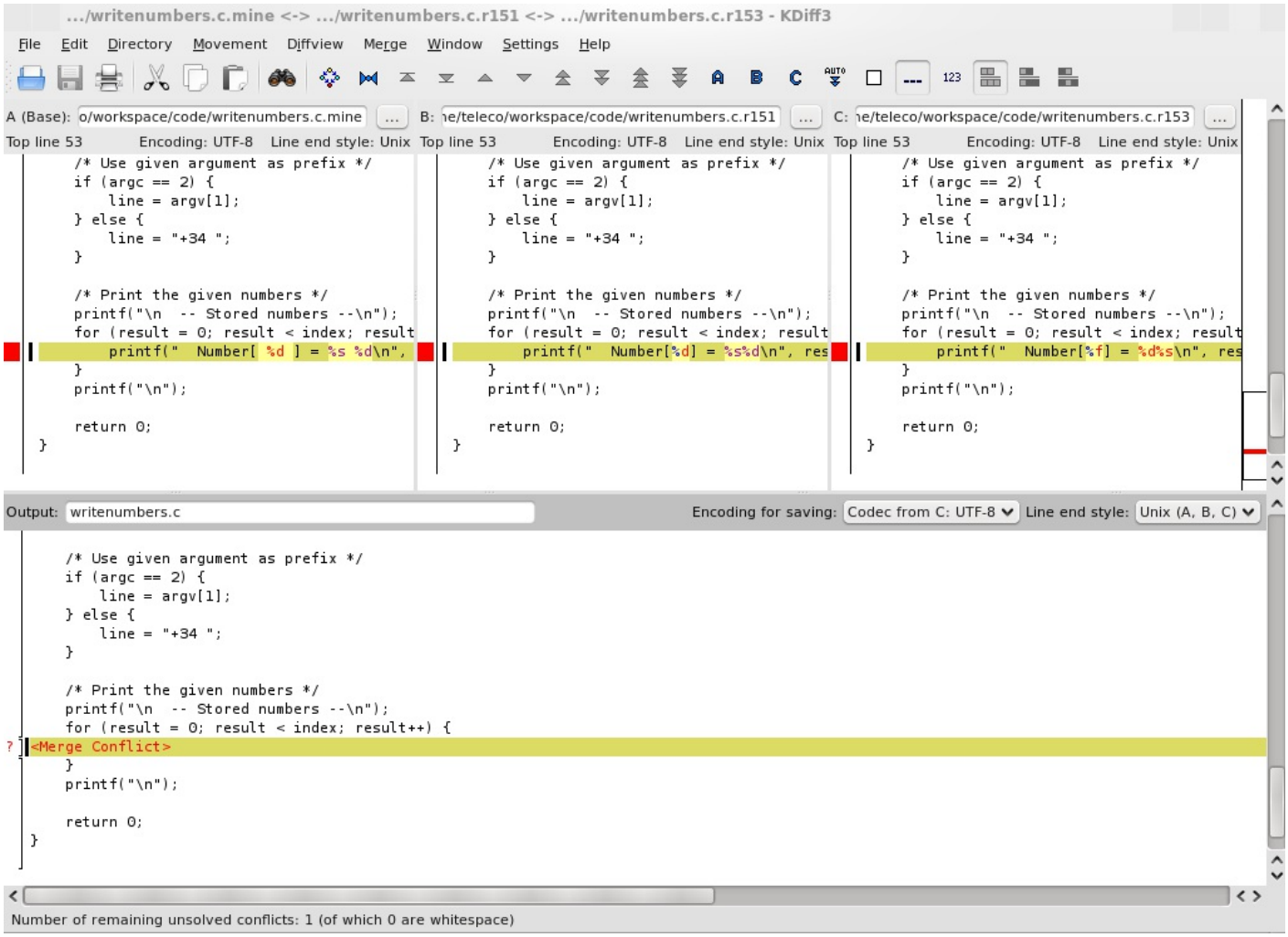
El fichero `writenumbers.c.mine` contiene la versión con tus modificaciones locales (versión C), y los dos ficheros con el sufijo "r" seguido de un número son las versiones A y B del depósito remoto. El fichero con número de versión menor es el más antiguo.

fichero con numero de version menor es el mas antiguo.

El programa **kdiff3** es un editor preparado para mezclar el contenido de de estas tres versiones en una versión definitiva. El programa se invoca de la siguiente forma:

```
kdiff3 writenumbers.c.mine writenumbers.c.r151 writenumbers.c.r153 -o writenumbers.c
```

Como argumentos del comando se dan los tres ficheros (versiones A, B y C) y con la opción **-o** se selecciona el fichero en el que guardar el resultado. El editor muestra en tres ventanas los tres ficheros y en la parte inferior el fichero resultante. La siguiente figura muestra esta herramienta editando este conflicto.



Con la flechas de la parte superior se puede mover entre conflictos (en este caso sólo hay uno). Para cada conflicto se puede seleccionar el contenido de cada una de las tres versiones (botones A, B y C de la parte superior) para incluir en el fichero resultante, o directamente escribir en este fichero el texto correcto. Una vez se termina se salva el fichero final sin conflictos. Tras esta operación, el estado del fichero todavía sigue mostrando el conflicto. Esto es porque Subversion requiere que se confirme la resolución de un conflicto de forma explícita mediante el comando **resolved**, tal y como se muestra en el siguiente ejemplo:

```
svn resolved writenumbers.c
```

Tras este comando, el fichero está listo para ser enviado al depósito centrar con el comando **commit**.

13.10. Preguntas de autoevaluación

Comprueba con estas preguntas que has entendido cómo funciona **Subversion**.

1. Quieres bajarte la copia de tu repositorio por primera vez en tu portátil. Tu NIA es 100011111 y el repositorio está localizado en `https://flautin.it.uc3m.es/as/NIA_100011111`. ¿Cuál sería el comando correcto?

   - `svn --username 100011111 list`

https://flautin.it.uc3m.es/as/NIA_100011111

- `svn --username 100011111 download https://flautin.it.uc3m.es/as/NIA_100011111`
- `svn --username 100011111 checkout https://flautin.it.uc3m.es/as/NIA_100011111`
- `svn --username 100011111 commit https://flautin.it.uc3m.es/as/NIA_100011111`

2. Una vez bajada la copia de tu repositorio, modificas uno de sus ficheros, `file.c`, y quieres subir esa modificación al repositorio remoto. ¿Cuál sería el comando correcto?

- `svn --username 100011111 add file.c`
- `svn --username 100011111 upload file.c`
- `svn --username 100011111 update file.c`
- `svn --username 100011111 commit file.c`

3. Una vez bajada la copia de tu repositorio, creas un fichero nuevo, new.c y quieres subirlo al repositorio remoto. ¿Cuál sería la secuencia de comandos correcta?

- `svn --username 100011111 add new.c`

  `svn --username 100011111 commit new.c`
- `svn --username 100011111 copy new.c`

  `svn --username 100011111 update new.c`
- `svn --username 100011111 commit new.c`

  `svn --username 100011111 update new.c`
- `svn --username 100011111 status new.c`

  `svn --username 100011111 commit new.c`

4. Tras crear y subir el fichero `new.c` al repositorio remoto, tu compañero de prácticas tiene acceso a ese repositorio remoto, así es que:

- Si en su momento tu compañero se bajó una copia del repositorio remoto antes de que tú crearas y subieras el archivo, no necesita hacer nada más para ver el nuevo fichero `new.c`; éste se habrá bajado automáticamente a su copia local.
- Aunque bajara una copia del repositorio remoto, necesita actualizar su repositorio local (comando `update`) para ver el nuevo fichero.
- Aunque bajara una copia del repositorio remoto, necesita actualizar el repositorio con el fichero (comando `add new.c`) para que exista también en su copia local.
- Ninguna de las anteriores.

13.11. Bibliografía de apoyo

- ***Branching and Merging***: "Version Control with Subversion - The Official Guide and Reference Manual " by Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato, 83-98.