Perforce en el contexto del desarrollo de FreeBSD

Scott Long

scottl@FreeBSD.org

$FreeBSD: doc/es_ES.ISO8859-1/articles/p4-primer/article.sgml,v 1.3 2007/11/08 21:49:11 carvay Exp $

Table of Contents

1 Introducción

El proyecto FreeBSD utiliza el sistema de control de versiones **Perforce** para gestionar proyectos experimentales que todavía no están listos para ser incluidos en el repositorio principal de CVS.

1.1 Disponibilidad, documentación y recursos

Aunque que el producto **Perforce** es un producto comercial, el software cliente que se encarga de interactuar con el servidor se distribuye libremente. Pueden descargarse versiones binarias del mismo desde el sitio web de **Perforce**: http://www.perforce.com/perforce/loadprog.html.

Existe un cliente gráfico, pero la mayoría de la gente utiliza la aplicación de línea de órdenes, `p4`. Este documento trata sobre el uso de dicha herramienta para la línea de órdenes.

En http://www.perforce.com/perforce/technical.html encontrará documentación "online" detallada.

Se recomienda encarecidamente leer la "guía de usuario" y el "manual de Perforce". La aplicación **p4** dispone de una extensa ayuda "online" a la que puede accederse mediante la orden `p4 help`.

El servidor FreeBSD **Perforce** se encuentra en `perforce.freebsd.org`, puerto `1666`. Puede navegar por el repositorio desde http://perforce.freebsd.org. Ciertas partes del repositorio se exportan automáticamente hacia diversos servidores **CVSup**.

2 Los comienzos

El primer paso para utilizar **Perforce** consiste en obtener una cuenta en el servidor. Si ya dispone de una cuenta en `FreeBSD.org` entre en `freefall` y ejecute el siguiente comando utilizando una contraseña distinta del acceso de su FreeBSD o de cualquier otro mecanismo de autenticación SSH:

```
% /usr/local/bin/p4newuser
```

Por supuesto si no tiene una cuenta en `FreeBSD.org` necesitará coordinarse con su mentor.

El siguiente paso consiste en establecer las variables de entorno que necesita `p4` y en verificar que puede conectarse al servidor. Es necesario especificar la variable `P4PORT` para realizar cualquier operación. Dicha variable indica el servidor **Perforce** con el que se va a trabajar. En el caso del Proyecto FreeBSD, créela con el siguiente valor:

```
% export P4PORT=perforce.freebsd.org:1666
```

> **Note:** Los usuarios con acceso "shell" al "cluster" `FreeBSD.org` pueden querer encapsular el protocolo cliente-servidor de **Perforce** a través de un túnel SSH, en cuyo caso la variable de arriba debería establecerse al valor `localhost`.

El servidor FreeBSD también necesita que se establezcan las variables `P4PASSWD` y `P4USER`. Utilice el nombre de usuario y la contraseña anteriores del siguiente modo:

```
% export P4USER=nombre_de_usuario
% export P4PASSWD=contraseña
```

Compruebe que todo funciona mediante la siguiente orden:

```
% p4 info
```

A resultas de esta orden debería ver información referente al servidor. Si no es así compruebe que la variable `P4PORT` tiene el valor correcto.

3 Clientes

El sistema **Perforce** proporciona acceso al repositorio y mantiene el estado del cliente de forma individualizada. En términos de **Perforce**, un cliente es una especificación que asocia [1] ficheros y directorios desde el repositorio hasta la máquina local. Cada usuario puede poseer varios clientes, y cada cliente puede acceder a distintas partes del repositorio (incluso a varias partes que se solapan entre sí). El cliente también especifica el directorio raíz del árbol de directorios sobre el que se realiza la asociación y la máquina donde efectivamente está dicho árbol. Es por esto que si pretende trabajar en varias máquinas tendrá que usar varios clientes.

Puede acceder a los clientes mediante `p4 client`. Si se ejecuta esta orden sin argumentos aparece una plantilla del cliente dentro de un editor, permitiendo de esta forma crear un nuevo cliente. Los campos importantes de esta plantilla se explican a continuación:

`Client:`

Este es el nombre de la especificación del cliente. Puede ser cualquier cosa, pero debe ser una cadena única dentro del servidor **Perforce**. Suelen usarse nombres como *`nombre_de_usuario_nombre_de_máquina`*, que permite identificar fácilmente a los clientes cuando se navega por ellos. Por defecto hay ya un nombre, que se corresponde con el nombre de la máquina.

`Description:`

Este campo suele consistir en un breve texto descriptivo que ayude a identificar al cliente.

`Root:`

Se trata del directorio local que actuará como directorio raíz para todos los ficheros dentro de la asociación en el cliente. Debe ser una localización única dentro del sistema de ficheros que no se solape con otros ficheros o clientes **Perforce**.

`Options:`

La mayoría de las opciones por defecto son correctas y válidas para todo el mundo, aunque suele ser recomendable comprobar que estén activadas las opciones `compress` y `rmdir` y que no tienen el prefijo `no`. Los detalles de cada una de estas opciones están en la documentación de **Perforce**.

`LineEnd:`

Este parámetro gestiona las conversiones CR-LF y debe dejarse tal cual salvo que sus necesidades específicas requieran cambiarlo.

`View:`

Aquí es donde están las asociaciones de ficheros servidor-a-local. El valor por defecto es:

`//depot/... //`*`cliente`*`/...`

Esto asociará por completo el repositorio **Perforce** al directorio `Root` del cliente. *NO USE ESTE VALOR POR DEFECTO*. El repositorio de FreeBSD es enorme e intentar asociarlo y sincronizarse con dicho repositorio tardará muchísimo y consumirá enormes recursos. Asocie sólamente la sección del repositorio en la que va a trabajar. Por ejemplo, hay un árbol para el proyecto smpng en `//depot/projects/smpng`. Una asociación en ese caso sería algo así:

`//depot/projects/smpng/... //`*`cliente`*`/...`

Los `...` deben tomarse literalmente tal cual están. Es un dialecto de **Perforce** para decir "este directorio y todos los ficheros y directorios por debajo de él.".

Una "vista" (View) puede contener múltiples asociaciones. Vamos a suponer que quiere asociar los árboles de SMPng y de NFS. Su "View" sería algo así:

```
//depot/projects/smpng/... //cliente/smpng/...
   //depot/projects/nfs/... //cliente/nfs/...
```

Recuerde que *`cliente`* es el nombre del cliente que se especificó en la sección `Client`, pero en la sección `View` además se utiliza para resolver al directorio especificado en la sección `Root`.

También tenga en cuenta que el mismo fichero o directorio no puede asociarse más de una vez dentro de una única vista. La orden del siguiente ejemplo no es correcta y producirá resultados imprevistos:

```
//depot/projects/smpng/... //cliente/smpng-esto/...
   //depot/projects/smpng/... //cliente/smpng-lo_otro/...
```

Las "vistas" son la parte compleja del proceso de aprendizaje de **Perforce**, así que no tenga miedo de hacer tantas preguntas como estime oportunas.

Puede listar los clientes existentes mediante `p4 clients`. Puede listarlos sin que sufran modificaciones mediante `p4 client -o` *`nombre_cliente`*.

Siempre que se interactue con ficheros en **Perforce** la variable de entorno `P4CLIENT` debe contener al nombre del cliente que se está utilizando, es decir:

```
% export P4CLIENT=nombredemicliente
```

Fíjese en que las asociaciones del cliente en el repositorio no son exclusivos; varios clientes pueden estar asociados en la misma zona del respositorio. Esto permite el trabajo en equipo sobre el mismo código, permitiendo que distintas personas accedan y modifiquen la misma parte del respositorio.

4 Sincronizaciones

Una vez definida la especificación del cliente y una vez establecida la variable de entorno `P4CLIENT`, el siguiente paso consiste en recuperar los ficheros para el cliente en cuestión desde el servidor hasta la máquina local. Esto se realiza con `p4 sync`, el cual indica a **Perforce** que sincronice los ficheros locales con los del repositorio. La primera vez que se ejecuta descargará todos los ficheros. Las siguientes ejecuciones sólo descargarán aquellos ficheros que hayan cambiado desde la última ejecución de la orden. Gracias a esto es posible sincronizar sus fuentes con las de otras personas con las que esté trabajando.

Las operaciones de sincronización sólo atañen a aquellos ficheros cuyas modificaciones han sido transmitidas a **Perforce**. Si se modifica o borra un fichero en local sin informar de ello al servidor la ejecución de un "sync" no reflejará dichos cambios. No obstante, la ejecución de `p4 sync -f` sincrozará incondicionalmente todos los ficheros, sin que importe su estado. Esto resulta útil para solucionar problemas cuando se cree que el árbol pueda haber sufrido algún tipo de corrupción.

Puede sincronizarse parte del árbol o del cliente especificando una ruta relativa a la orden "sync". Por ejemplo, para sincronizar sólo el directorio `ufs` del proyecto `smpng` ejecute lo siguiente:

```
% cd raizdelproyecto/smpng
% p4 sync src/sys/ufs/...
```

El uso de rutas locales relativas funciona en muchas otras órdenes `p4`.

5 Ramas

Una de las características más interesantes de **Perforce** es la posibilidad de crear ramas. Las ramas son muy sencillas de crear y también resulta muy fácil mover cambios entre distintas ramas (como se verá más adelante). Las ramas también nos permiten realizar trabajos muy experimentales dentro de un entorno de "sandbox", sin necesidad de tener que preocuparnos por las colisiones con otros usuarios o por desestabilizar el árbol principal. Además, las ramas proporcionan el aislamiento necesario frente a los errores que se cometen cuando se aprende a manejar el sistema **Perforce**. Vistas estas ventajas es lógico que cada proyecto disponga de su propia rama y en FreeBSD recomendamos encarecidamente este esquema. También se recomienda la aplicación frecuente de los cambios realizados.

El repositorio **Perforce** (conocido como el "depósito", o "depot" en la jerga de **Perforce**) es un único árbol plano. Se accede a cada fichero a través de una sencilla ruta bajo el directorio `//depot`, tanto si se trata de un fichero de nueva creación como si proviene de una ramificación. Esto supone una gran diferencia con respecto a sistemas como CVS, donde cada rama se encuentra en la misma ruta que su rama padre. En **Perforce** el servidor mantiene las relaciones entre los ficheros padre e hijo, pero los ficheros en sí están bajo sus propias rutas.

El primer para para crear una rama consiste en crear una especificación de rama. Es similar a la especificación de un cliente, pero se crea mediante la orden `p4 branch` *`nombre_de_rama`*.

Veamos los campos más importantes:

`Branch`

> El nombre de la rama. Puede ser cualquier nombre, pero debe ser único en el repositorio. La convención que se usa en FreeBSD es *`nombre_de_usuario_nombre_del_proyecto`*.

`Description`

> Puede poner aquí un texto simple que describa la rama.

`View`

> Esto es la asociación de la rama. En lugar de asociar desde el "depósito" hacia la máquina local como una asociación de cliente, se crea una asociación entre la rama padre y la rama hija dentro del "depósito". Por ejemplo, puede querer crear una rama del proyecto smpng. La asociación resultaría en algo parecido a esto:
>
> `//depot/projects/smpng/... //depot/projects/mi-super-smpng/...`
>
> O puede crear una rama totalmente nueva a partir de las fuentes de FreeBSD:
>
> `//depot/vendor/freebsd/... //depot/projects/mi-nuevo-proyecto/...`
>
> Esto asociará el HEAD del árbol de FreeBSD a su nueva rama.

La creación de la especificación de rama únicamente graba la especificación en sí misma dentro del servidor. No modifica el "depósito" ni cambia ningún fichero. El directorio que se declara en la rama permanece vacío en el servidor hasta que se comience a llenar.

Para rellenar la rama primero debemos editar el cliente con la orden `p4 client` y asegurarnos de que el directorio de rama está asociado en el cliente. Puede ser necesario añadir una línea `View` como esta:

`//depot/projects/mi-nuevo-proyecto/... //`*`micliente`*`/mi-nuevo-proyecto/...`

El siguiente paso consiste en ejecutar `p4 integrate`, como se describe en la siguiente sección.

6 Integraciones

"Integración" es el término que se utiliza en **Perforce** para describir la acción de mover cambios desde una parte del "depósito" a otra. Se suele realizar junto con las órdenes creación y mantenimiento de ramas. Una integración es necesaria cuando se quiere rellenar inicialmente una rama y cuando se quieren mover cambios realizados en la rama padre hacia la rama hija, o de la la rama hija a la padre. Un caso muy común es la integración periódica desde el árbol original de FreeBSD hacia la rama hija propia del usuario. El servidor **Perforce** mantiene el estado de los cambios en cada rama y sabe cuándo hay cambios que pueden integrarse de una rama a otra.

La forma más común de hacer una integración se muestra en la siguiente orden:

```
% p4 integrate -b nombrederama
```

nombrederama es el nombre que se ha dado a la especificación de rama, tal y como se explicó en la sección anterior. Esta orden indica a **Perforce** que busque cambios en la rama padre que todavía no se hayan aplicado a la rama hija. En base a los cambios encontrados se prepara un listado de diferencias a aplicar. Si la integración se realiza por primera vez sobre una rama (por ejemplo cuando se realiza una operación de rellenado inicial) los ficheros de la rama padre simplemente se copiarán en la ubicación en la rama hija de la máquina local.

Una vez que la operación de integración ha finalizado se debe ejecutar `p4 resolve`, que aplicará los cambios y resolverá posibles conflictos. Los conflictos puede surgir debido a cambios que se solapan al encontrarse tanto en fichero de la rama padre como en la copia del fichero de la rama hija. Normalmente no suelen aparecer conflictos y **Perforce** puede calcular rápidamente cómo unir los cambios. Para ejecutar una operación de resolución ("resolve") utilice las siguientes órdenes:

```
% p4 resolve -as
% p4 resolve
```

La primera invocación indica a **Perforce** que una automáticamente los cambios y que acepte aquellos ficheros que no den conflictos. La segunda invocación permite inspeccionar cada fichero con conflictos y resolver de forma manual dichas incompatiblidades.

Una vez hecha la integración de los ficheros llega el momento de aplicar los cambios al repositorio. Para ello se emplearemos la orden `p4 submit`, cuyo uso se explica en la siguiente sección.

7 Aplicación de cambios en el repositorio

Los cambios que se han realizado en local se deben aplicar en el contenido del servidor **Perforce** para mayor seguridad frente a pérdidas y para que otras personas puedan acceder a dichos cambios; esto se hace con la orden `p4 submit`. Cuando se ejecuta esta orden se abre una plantilla ("submit template") en el editor. FreeBSD dispone de una platilla personalizada, de la que a continuación se explican los campos más importantes:

```
Description:
        <enter description here>
        PR:
        Submitted by:
        Reviewed by:
        Approved by:
        Obtained from:
        MFP4 after:
```

es decir

```
Descripción:
       <Introduzca una descripción>
       PR:
       Enviado por:
       Revisado por:
       Aprobado por:
       Obtenido de:
       MFP4 tras:
```

Se considera una buena práctica proporcionar al menos dos o tres frases que describan los cambios entregados. Debería declarar aquí qué hacen dichos cambios, por qué se han hecho de esa forma o qué problemas intenta resolver con ellos. También conviene explicar qué APIs cambian y qué otros efectos secundarios pueden tener. Este texto debe sustituir a la línea `<enter description here>` que aparece en la plantilla. Debe recubrir las líneas y comenzar cada línea con una tabulación. Las etiquetas de más abajo son específicas de FreeBSD y puede eliminarlas si no resultan útiles o apropiadas en su contexto.

```
Files:
```

Este campo se rellena automáticamente con todos los ficheros que el cliente etiquetó en el servidor con estados de adición, borrado, integración o edición. Le aconsejamos que revise esta lista y elimine de ella los ficheros que todavía no esten listos.

Una vez guardada la sesión de su editor tiene lugar la entrega de los datos al servidor. Esto significa que las copias locales de los ficheros entregados se enviarán al servidor. Si algo va mal durante este proceso se cancelará la entrega y se avisará al usuario de que la entrega se ha convertido en una lista de cambios que deben corregirse y reenviarse. Las entregas son atómicas, es decir, si un fichero falla la entrega se cancela en su totalidad.

Los cambios efectuados en el servidor no pueden cancelarse una vez hechos, pero sí que pueden cancelarse si, dentro aún del editor, se sale de él sin cambiar el texto del campo `Description`. **Perforce** se quejará la primera vez que intente salir y le devolverá al editor. Si sale por segunda vez el editor cancelará la operación. Devolver el repositorio al estado anterior a un cambio ya efectuado es un proceso muy complicado y no hay un procedimiento estándar, por lo que depende del caso concreto.

8 Edición

En el servidor se almacena y mantiene el estado de cada fichero del cliente. Para evitar colisiones entre distintas personas trabajando al mismo tiempo en el mismo fichero **Perforce** presta atención a qué ficheros están abiertos en modo de edición, y utiliza esa información para poder gestionar posteriormente las operaciones de entrega, las sincronizaciones y las integraciones.

Para abrir un fichero para editarlo utilice `p4 edit` de la siguiente forma:

```
% p4 edit nombredefichero
```

Esto marca el fichero en el servidor con el estado de edición, lo que permite entregar el fichero posteriormente una vez realizados los cambios oportunos, o lo etiqueta como de tratamiento especial cuando se está efectuando una operación de integración o sincronización. Tenga en cuenta que la edición no es exclusiva en **Perforce**. Varias personas pueden tener el mismo fichero en estado de edición (será informado de ello si es necesario cuando ejecute

`edit`), pero podrá entregar sus cambios incluso cuando haya otras personas que tengan ese fichero en estado de edición.

Cuando alguien entregue un cambio de un fichero que usted esté editando necesitará cotejar sus modificaciones con las de la otra u otras personas para poder aplicar correctamente sus modifaciones al repositorio. La forma más sencilla de hacerlo es ejecutar `p4 sync` o `p4 submit` y dejar que el programa encuentre algún conflicto, y a continuación ejecutar `p4 resolve` para "resolver" manualmente los conflictos y aceptar los cambios de la otra persona en su copia del fichero. Hecho esto, utilice `p4 submit` para aplicar sus cambios en el repositorio.

Si posee un fichero abierto para su edición y quiere descartar los cambios y devolverlo a su estado original ejecute `p4 revert` de la siguiente forma:

```
% p4 revert nombredefichero
```

Esto resincroniza el fichero con el contenido del servidor y elimina en el servidor el atributo de edición para ese fichero. Se perderá cualquier cambio que haya hecho en local. Esto resulta muy útil cuando se han efectuado una serie de cambios en un determinado fichero y se decide posteriormente que no se desean aplicar dichos cambios en el servidor.

Cuando se sincroniza un fichero se marca como sólo lectura en el sistema de ficheros. Aunque se pueden sobreescribir fácilmente dichos permisos se aplican para recordar al usuario de una forma educada que para ello se debe utilizar `p4 edit`. Los ficheros modificados en local pero que no están en estado de edición pueden sobreescribirse al ejecutar `p4 sync`.

9 Cambios, descripciones e historial

Puede ver el historial de cambios realizados al "depósito" de **Perforce** puede consultarse mediante `p4 changes`. Esta orden proporciona una breve descripción de cada cambio, quién la realizó y cúal es el número de modificación. Si lo que se quiere son los detalles de un cambio en concreto utilice `p4 describe` *`numero_de_cambio`*. Esta orden proporciona el "log" y los "diffs" de dicho cambio. Normalmente se utilizan las opciones `-du` o `-dc` para generar "diffs" unificados o contextuales, respectivamente, en lugar del formato del "diff" nativo.

`p4 filelog` *`nombre_de_fichero`* muestra el historial de un fichero, incluyendo todas sus modificaciones, integraciones y ramas que contenga.

10 "diffs"

Existen dos formas de generar "diffs" de ficheros en **Perforce**, bien entre cambios locales que todavía no se han entregado o bien entre dos árboles (o dentro de una misma rama) del "depósito". Estos "diffs" se generan mediante órdenes distintas, `diff` y `diff2`:

`p4 diff`

> Ese comando genera un "diff" entre los cambios locales y los cambios de ficheros en estado de edición. Los parámetros `-du` y `-dc` permiten crear "diffs" unificados o contextuales, respectivamente. También se puede establecer la variable `P4DIFF` para que apunte a un "diff" local. Le recomendamos encarecidamente usar esta orden para revisar sus cambios antes de aplicarlos en el servidor.

`p4 diff2`

Esta orden crea un "diffs" entre ficheros dados en el "depósito", o entre ficheros especificados en una especificación de rama. La operación tiene lugar en el servidor, así que la variable `P4DIFF` no surte ningún efecto, aunque las opciones `-du` y `-dc` sí pueden usarse. Las dos formas de esta orden son:

```
% p4 diff2 -b nombrederama
```

y

```
% p4 diff2 //depot/ruta1 //depot/ruta2
```

En todos los casos los "diffs" se muestran en la salida estándar. Por desgracia **Perforce** usa un formato de "diffs" que resulta ser ligeramente incompatible con las herramientas Unix estándar `diff` y `patch`. La utilización de la variable `P4DIFF` para que apunte al verdadero diff(1) puede paliar este problema, o al menos en ciertos casos, puesto sólo funciona con la orden `p4 diff`. La salida de `diff2` debe procesarse para que sea de alguna utilidad (la opción `-u` de `diff2` producirá "diffs" unificados que serán *más o menos compatibles*, pero no esto no incluye ficheros nuevos o borrados. Este "script" puede serle de utilidad para este "proceso necesario": http://people.freebsd.org/~scottl/awkdiff.

11 Añadir o eliminar ficheros

La integración de una rama hará que se añadan ficheros existentes en el servidor en su árbol, pero quizás sea necesario añadir nuevos ficheros o eliminar alguno de los ya existentes. Para añadir ficheros no tiene más que crear el fichero y ejecutar `p4 add` de la siguiente forma:

```
% p4 add nombredefichero
```

Si quiere añadir un árbol completo de ficheros ejecute:

```
% find . -type f |xargs p4 add
```

Al ejecutar `p4 submit` se copiarán los ficheros al "depósito" del servidor. Es muy importante añadir sólo ficheros y no directorios. Si se añade explícitamente un directorio, **Perforce** lo tratará como fichero, lo cual seguramente no es lo que usted tenía previsto.

Borrar un fichero es igualmente sencillo mediante `p4 delete`:

```
% p4 delete nombredefichero
```

Esta orden marcará el fichero para que sea borrado del "depósito" la siguiente vez que se ejecute una entrega. También borrará la copia local del fichero, así que sea cauteloso cuando la use.

Por supuesto que borrar un fichero no significa que se borre realmente del repositorio.

Los ficheros borrados se pueden "resucitar" mediante la sincronización con una versión previa. La única forma de borrar de forma permanente un fichero es mediante la orden `p4 obliterat`. Dicha orden es irreversible y costosa, así que sólo está al alcance del personal que administra el repositorio.

12 El trabajo con "diffs"

Algunas veces puede ser necesario aplicar un "diff" al árbol de **Perfoce** que provenga de otra aplicación. Si se trata de un "diff" de gran tamaño y que afecta a muchos ficheros, puede resultar tedioso ejecutar manualmente `p4 edit`

sobre cada fichero. Hay un truco para hacerlo de una forma más sencilla. En primer lugar, asegúrese de que no hay ficheros abiertos en su cliente y de que su árbol está sincronizado y actualizado a la última versión. A continuación aplique sus cambios mediante las herramientas habituales, y forzando los permisos de los ficheros en caso de ser necesario. Después ejecute lo siguiente:

```
% p4 diff -se ... |xargs p4 edit
% p4 diff -sd ... |xargs p4 delete
% find . -type f |xargs p4 add
```

La primera orden le dice a **Perforce** que busque los ficheros que han cambiado, incluso si no están abiertos. La segunda orden le dice a **Perforce** que busque los ficheros que no existen en la máquina local pero que sí están en el servidor. La tercera orden intenta añadir todos los ficheros que están en local. Es un método de fuerza bruta, pero funciona bien porque **Perforce** sólo añadirá los ficheros que le resulten desconocidos. El resultado de estas órdenes es un conjunto de ficheros abiertos para edición, borrado o para ser añadidos, según el caso. Hecho esto solo nos queda ejecutar `p4 submit` para entregar los cambios.

13 Cambiar nombres de ficheros

Perforce no dispone de una forma predefinida de cambiar nombres a ficheros o de moverlos a otra parte del árbol. Si se copia el fichero en cuestión a una nueva ubicación mediante `p4 add`, y posteriormente `p4 delete` en la versión anterior, se obtiene algo muy parecido a lo que se quería, pero tiene el inconveniente de que no se preserva el historial de cambios de ese fichero. Esto puede perjudicar futuras integraciones entre padres e hijos. Hay otro método más recomendable, que consiste en efectuar una integración dentro del mismo árbol y de una sola vez. Veamos un ejemplo:

```
% p4 integrate -i ficheroprevio ficheronuevo
% p4 resolve
% p4 delete ficheroprevio
% p4 submit
```

La integración fuerza a **Perforce** a mantener un registro de las relaciones entre los nombres antiguos y los nuevos, lo cual será muy útil en futuras integraciones. La opción `-i` indica que se trata de una integración “sin base”, es decir, que no existe un historial de ramas al que recurrir en la integración. Este parámetro tiene sentido en el presente ejemplo, pero no debería utilizarse en integraciones basadas en ramas.

14 Interacciones entre el CVS de FreeBSD y Perforce

Los repositorios de **Perforce** y de CVS de FreeBSD están completamente separados. No obstante, los cambios que se producen en CVS se reflejan casi en tiempo real en **Perforce**. Cada 2 minutos se pregunta al servidor de CVS sobre cambios realizados en la rama HEAD, y dichos cambios se entregan a **Perforce** dentro del árbol `//depot/vendor/freebsd/...`. De este modo este árbol permite la ramificación y la integración de proyectos derivados. Cualquier proyecto que implique la modificación del código fuente de FreeBSD debería tener este árbol como su rama padre (o rama “abuela”, dependiendo de las necesidades concretas de cada proyecto), y deberían tener lugar integraciones periódicas y sincronizaciones para que el árbol esté en consonancia con el desarrollo de FreeBSD y evitar conflictos en la medida de lo posible.

El puente entre CVS y **Perforce** es de un sólo sentido; los cambios del CVS se reflejarán en **Perforce**, pero los cambios en **Perforce** no se reflejarán en el CVS. Si es necesario, se pueden exportar partes del repositorio de

Perforce al **CVSup** y que así se puedan distribuir. Por favor, contacte con los administradores de **Perforce** de FreeBSD si ese es su caso.

15 Funcionamiento sin conexión de red

Uno de los inconvenientes de **Perforce** es que supone que siempre es posible acceder al servidor a través de la red. La mayoría de los estados, el historial y los metadatos se almacenan en el servidor y no existe mecanismo alguno para replicar el servidor como los hay en CVS/**CVSup**. Es posible ejecutar un servidor proxy, pero solamente ayuda un poco si se quiere trabajar sin conexión al servidor.

La mejor forma de trabajar sin conexión de red es comprobando que el cliente no tiene ningún fichero abierto y que está totalmente sincronizado antes de dejar de estar conectado. Cuando se edite un fichero se deberán cambiar manualmente los permisos a lectura-escritura. Cuando vuelva a estar conectado ejecute la orden que se mostraba en la Section 12 para identificar automáticamente los ficheros que se han editado, añadido o eliminado. Es bastante común encontrarse con la sorpresa de que **Perforce** ha sobreescrito un fichero modificado en local que no se abrió en modo edición, así que tenga especial cuidado con esto.

16 Consideraciones finales para el “Google Summer of Code”

La mayoría de los proyectos de FreeBSD dentro del programa “Google Summer of Code” están en `//depot/projects/soc2005/`*`nombre_del_proyecto`*`/...` en el servidor FreeBSD de **Perforce**.

Entre las responsabilidades del mentor del proyecto está seleccionar un nombre adecuado para dicho proyecto y hacer que el estudiante sea capaz de trabajar con **Perforce**.

El acceso al servidor FreeBSD de **Perforce** no implica pasar a ser miembro de la comunidad de committers del CVS de FreeBSD, aunque animamos de todo corazón a todos los estudiantes que consideren la posibilidad de unirse al proyecto cuando estén listos para ello.

Notes

1. Este término, que también puede traducirse como asociar o asignar, suele aparecer en la jerga de la administración de sistemas como “mapear”.