

## Curso de subversion

### Introducción a los sistemas de control de versiones

Carlos Hernando  
chernando@acm.org

ACM Facultad de Informática  
Universidad Politécnica de Madrid

17 de noviembre de 2005  
Curso svn



## Contenido

### Introducción a los sistemas de control de versiones

Problemas  
Conceptos generales  
subversion

### Uso de subversion

Creación de un repositorio  
Ciclo habitual de desarrollo con subversion  
Administración básica



## Problemas comunes

### Situación:

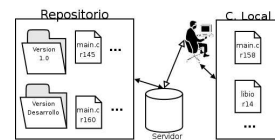
- ▶ Proyecto complejo.
- ▶ Varios programadores.
- ▶ Separados físicamente.

### Problemas:

- ▶ Mantenerse al día.
- ▶ Recordar que hemos hecho.
- ▶ Trabajar en varias versiones.



## Conceptos generales



**Repositorio** Servidor, encargado de mantener nuestro proyecto.

**Copia local** Copia del desarrollador.

**Versión** Estado asociado a una clave, normalmente un número.

**Rama** Línea de desarrollo del proyecto.



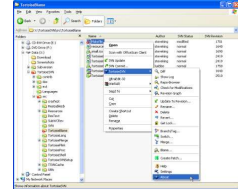
## Características de subversion

- ▶ Sistemas de acceso URL.
- ▶ Trabajar sin conexión permanente.
- ▶ Implementado sobre una base de datos.
  - ▶ Curioso sistema de etiquetas y ramas.
  - ▶ La versión aumenta con cada cambio.
- ▶ Mayor comodidad respecto a CVS.



## Programas

- ▶ Versión oficial: <http://subversion.tigris.org/>
- ▶ Versión Windows:  
<http://tortoissvn.tigris.org/>



## Creación de un repositorio

### Sintaxis

```
svnadmin create RUTA
```

- ▶ El comando de administración es svnadmin.
- ▶ Hemos de especificar una ruta local.
- ▶ Solamente creamos el repositorio.

### Example

```
svnadmin create /srv/svn
```



## Incorporando contenidos en el nuevo repositorio

### Sintaxis

```
svn import REPOSITORIO
```

- ▶ Debemos realizar el import en el directorio en el que residen los contenidos que queremos incorporar.
- ▶ La ruta depende del método de acceso.

### Example

```
svn import file:///srv/svn  
svn import svn+ssh:///srv/svn
```



## Organización de un proyecto

### Estructura de directorios

```
/
|--> /trunk
|--> /branches
\--> /tags

/
|--> /proyectoA
|   |--> trunk
|   |--> ...
|--> /proyectoB
|   \--> trunk
```

- trunk** Rama de desarrollo principal.
- branches** Raíz de las ramas del proyecto.
- tags** Raíz de las etiquetas del proyecto.



## Obteniendo nuestra copia local

### Sintaxis

`svn checkout REPOSITORIO [DIRECTORIO]`

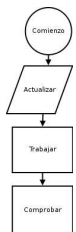
- El repositorio se queda fijado.
- El directorio en el que bajar la copia es opcional.
- A partir de aquí disponemos de una copia local.

### Example

```
svn checkout http://servidor/svn/curso
svn checkout svn://servidor/svn/proyecto/trunk curso
```



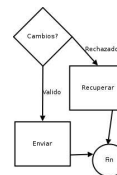
## El día a día



- Actualizar la copia local:  
**Sintaxis**  
`svn update`
- Revisar que hemos cambiado:  
**Sintaxis**  
`svn status`
- Revisar los cambios realizados:  
**Sintaxis**  
`svn diff [FICHERO]`



## El día a día (y 2)



- Elegir que hacemos:
  - Rechazar los cambios:  
**Sintaxis**  
`svn revert [FICHERO]`
  - Enviar al repositorio:  
**Sintaxis**  
`svn commit [FICHERO]`



## Trabajando con ficheros

### Sintaxis

```
svn add FICHEROS | DIRECTORIOS
svn remove FICHEROS | DIRECTORIOS
svn move ORIGEN DESTINO
svn copy ORIGEN DESTINO
svn mkdir DIRECTORIO
```

- ▶ Si añadimos un directorio todo su contenido será a su vez añadido.
- ▶ Los ficheros movidos y copiados heredan el historial de versiones.
- ▶ Todos los cambios se hacen efectivos en el momento del commit.



## Conflictos

### El problema

```
C src/main.c
```

Pasos para resolverlo:

1. Revisar las diferentes versiones:
  - ▶ Corrigiendo el fichero principal.
  - ▶ Trabajando con los ficheros .rN, .rM y .mine
2. Arreglar el fichero principal y ejecutar

### Sintaxis

```
svn resolved [fichero]
```



## Hooks, mejorando nuestro repositorio

### Directorio hooks

```
post-commit.tmpl post-lock.tmpl ...
pre-commit.tmpl pre-lock.tmpl ...
```

- ▶ Podemos añadir comprobaciones antes de admitir un *commit*.  
Por ejemplo: comprobar y corregir la indentación.
- ▶ Podemos añadir acciones después de un *commit*.  
Por ejemplo: enviar un mensaje a una lista de correo.
- ▶ Los scripts deben ser ejecutables por el usuario que hace el *commit*.
- ▶ Las plantillas terminan en .tmpl, los scripts no tienen terminación.



## Problemas de permisos (svn+ssh:// y fi le://)

### El problema:

1. Usuario A: *commit*.
2. Usuario A crea nuevos ficheros en la bdd.
3. Usuario A completa correctamente.
4. Usuario B: *commit*.
5. Usuario B es incapaz de escribir en los ficheros de A.
6. *commit* rechazado y repositorio inestable.

### Solución preventiva:

- ▶ Los usuarios han de pertenecer al mismo grupo.
- ▶ Fijar una máscara permisiva:
  - ▶ Obligar a los usuarios a fijar su máscara manualmente.
  - ▶ Hacer un wrapper para svn.



## Resumen

- ▶ Conocimientos adquiridos:
  - ▶ Conceptos y terminología de los sistemas de control de versiones.
  - ▶ Nivel medio de uso diario de subversion.
  - ▶ Administración básica de repositorios personales.
- ▶ Por tratar:
  - ▶ Métodos de acceso: `http://` y `svn://`
  - ▶ Trabajar con *branches* y *tags*.
  - ▶ Propiedades de los ficheros.
  - ▶ Trabajo sucio con la base de datos del repositorio.

