Guía del sistema de control de versiones Subversion

Licenciatura en Ciencias de la Computación

Mariano Street, basado en un manual de Duilio Protti
Departamento de Ciencias de la Computación

28 de septiembre de 2018

Subversion (abreviado SVN) es un sistema de control de versiones. Se trata de una herramienta para ir registrando de forma organizada los cambios que uno hace a lo largo del tiempo sobre un conjunto de archivos. Además permite trabajar en proyectos, es decir, que múltiples usuarios trabajen sobre los mismos archivos, incluso al mismo tiempo. Para esto provee repositorios centralizados, lugares a donde uno sube los cambios que hace y de donde baja los cambios hechos por los demás.

Subversion es software libre[1] y lo desarrolla la Fundación Apache.

1. Repositorio de la LCC

En la Licenciatura en Ciencias de la Computación disponemos de un servidor de Subversion para ser usado en las diversas cátedras según cada una determine conveniente (téngase en cuenta que algunas materias usan otros medios). Se puede acceder al mismo desde las siguientes direcciones URL:

- `https://svn.dcc.fceia.unr.edu.ar/svn/lcc/` (acceso anónimo de solo lectura)
- `https://svn.dcc.fceia.unr.edu.ar/svn-no-anon/lcc/` (acceso autenticado de lectura y escritura)

Con el acceso anónimo, puede acceder a las áreas públicas del repositorio para lectura. Para además poder escribir (modificar archivos, crear nuevos, etc.) o poder acceder a algún área privada, necesita una cuenta de usuario en el servidor Kleene.

El servidor de Subversion está alojado en Kleene, uno de los servidores del Departamento de Ciencias de la Computación. Para obtener una cuenta, debe contactar al administrador de Kleene (Emilio López).

1.1. Esquema de directorios

El repositorio para las materias de la LCC se llama `lcc`. Tiene varios subdirectorios, uno por cada materia. Los nombres corresponden a los códigos de materias (como los estipula el plan de estudios): aquellos que empiezan con `R-` corresponden al plan 2010 y los que empiezan con `T-`, al plan 1995.

Cada directorio de materia, a su vez, se encuentra estructurado con los siguientes subdirectorios y permisos:

`Alumnos` (escriben y leen docentes y alumnos, nadie más lee ni escribe)

[1] El software libre es aquel que no restringe al usuario. Puede ser usado, modificado y distribuido con cualquier propósito. Para saber más, se recomienda leer el artículo enlazado al final de este documento.

`Catedra` (escriben y leen docentes, nadie más lee ni escribe)

`Public` (escriben y leen docentes, cualquiera puede leer, incluso anónimos)

Esquemáticamente:

```
<materia>
+-- Alumnos (docentes = rw, alumnos = rw)
+-- Catedra (docentes = rw, alumnos = )
+-- Public  (docentes = rw, todos = r)
```

Donde `<materia>` es un código de materia y los tres subdirectorios contenidos dentro tienen los permisos mencionados en paréntesis, de acuerdo con a qué grupo pertenezca el usuario que acceda al repositorio.

El directorio `Public` está destinado a que las cátedras hagan público material que quieran compartir con el mundo, como programas de la materia, etc. El directorio `Cátedra` es para uso interno de la cátedra, pues nótese que ni siquiera los alumnos pueden verlo, ni nadie de afuera del grupo `docentes`. El directorio `Alumnos` está destinado a ser usado por los alumnos para trabajar sobre, o entregar, código fuente que la cátedra use o evalúe durante el cursado. La organización interna del directorio `Alumnos` queda a discreción de la cátedra. Se puede incluso armar dentro de ella un esquema con restricciones de acceso especiales. Por ejemplo, se puede armar un directorio por cada grupo de trabajo designado en el cursado, y que un grupo de alumnos no pueda ver lo que hagan otros grupos de alumnos. Consúltese por esta y otras configuraciones con los administradores de Kleene.

2. Esquema de directorios para Arquitectura del Computador

Arquitectura del Computador es una materia que usa activamente el repositorio de Subversion. Su código es `R-222`.

Dentro del directorio `Alumnos`, hay un subdirectorio por cada año de dictado. Dentro de cada uno de estos, hay a su vez un subdirectorio por cada grupo de trabajo en las prácticas de la materia. Los nombres deben contener los apellidos de cada alumno integrante del grupo.

Cada grupo debe seguir las convenciones de Subversion para estructurar proyectos, que consisten en tener los siguientes directorios:

`tags:` etiquetas, se debe hacer una por cada entrega de una práctica; este es el único directorio obligatorio y el único que es inspeccionado por los docentes de la cátedra a la hora de corregir.

`trunk:` árbol de desarrollo principal, es donde el grupo trabaja normalmente hasta llegar a un estado listo para entregar.

`branches:` ramas, por ejemplo para implementar funcionalidades específicas sin afectar el árbol de desarrollo principal.

Estos tres directorios pueden ser hijos directos del directorio de cada grupo, o puede en cambio haber un directorio por cada práctica y aparecer entonces estos tres repetidos para cada práctica. Es decir, con la primera opción se tendría:

```
<grupo>
+-- branches
+-- tags
+-- trunk
```

Y con la segunda, algo como lo siguiente:

```
<grupo>
+-- práctica1
    +-- branches
    +-- tags
    +-- trunk
+-- práctica2
    +-- branches
    +-- tags
    +-- trunk
```

Una desventaja de la primera es que obliga luego a separar por práctica dentro de cada uno de los tres directorios. Consulte a los docentes de la cátedra cuál organización es más recomendable.

3. Esquema de directorios para Sistemas Operativos 2

Sistemas Operativos 2 es otra materia que usa activamente el repositorio de Subversion. Su código es `R-412`.

La organización dentro del directorio `Alumnos` es muy similar a la de Arquitectura del Computador. La diferencia es que se pide que en el directorio de cada grupo no se haga un subdirectorio por cada práctica sino que necesariamente se use un `tags` (y opcionalmente un `trunk` o un `branches`) común para todas ellas. Esto va de la mano con la metodología de desarrollo incremental de Nachos.

4. Buenas prácticas en el uso del repositorio

No haga "commits" de funcionalidad parcial.

Esto es, usted está trabajando y en un momento decide cortar. Es mala práctica decir "bueno, subo lo que tengo hasta acá al repo, como copia de respaldo". ¡Mal! Solo tiene sentido versionar algo que tenga sentido querer volver a recuperar en un futuro. Por ejemplo, si se está implementando una funcionalidad X en un programa, solo se querrá volver al punto anterior a agregar X, o al punto luego de que se haya agregado, pero nunca a un punto intermedio donde el programa no anduviese. Si se quiere hacer una copia de respaldo de la funcionalidad parcial, usar un medio local (por ejemplo, un pendrive) o almacenamiento de red (por ejemplo, Nextcloud), pero nunca el repositorio de versionado, pues desperdicia espacio y además al trabajar en grupo es una muy mala práctica, pues el resto del equipo al hacer un `update` tendrá una copia local de un programa que no corra y quizá ni siquiera compile. En el ambiente de programación esta regla se conoce como: *el repo nunca se deja roto.*

No agregue al repositorio archivos generados automáticamente.

A menos que se quiera poner en un directorio público un archivo generado automáticamente por un programa para que otros lo vean y no deban tomarse el trabajo de reconstruirlo, y porque además no se quiera ofrecer la posibilidad de cambiarlo o se quiera ocultar su código fuente (práctica para nada recomendable), en todo otro caso, no es buena práctica subir al repositorio archivos binarios que se puedan generar a partir de otras fuentes. Es decir, si se tiene un archivo `.tex` o `.sxw` a partir del cual se genere un PDF, se debe subir el `.tex` o el `.sxw`, y no el PDF generado. El PDF uno lo puede generar en su copia local a partir de las fuentes que baje del repositorio. Hacerlo de otra manera genera una sobrecarga innecesaria en el repositorio y también quizá en las personas que trabajen con uno, pues podría forzarlos a hacer innecesariamente actualizaciones de archivos autogenerados.

Tenga en cuenta el licenciamiento del material que agregue.

Considere que el repositorio es de acceso público y localizado en una universidad pública.

5. Cómo usar el repositorio con el comando `svn`

Para acceder al repositorio, hay que hacer uso de un cliente de Subversion. En esta sección se mostrará cómo usar el cliente oficial de línea de comandos (el comando `svn`). Hay diversos clientes disponibles, algunos ofrecen interfaces gráficas (por ejemplo, TortoiseSVN) o de otros tipos.

1. Instale el cliente de Subversion. Para el comando `svn`, las distribuciones de GNU/Linux suelen ofrecer en sus repositorios un paquete llamado `subversion` o `svn`. Este se puede instalar con el gestor de paquetes que corresponda. Por ejemplo, en Debian o Ubuntu, puede usar el comando `apt` para instalarlo:

   ```
   $ sudo apt install subversion
   ```

2. El comando `svn` ofrece diversas acciones que el usuario puede pedirle realizar. Cada vez que se lo ejecuta, se le indica alguna acción. Antes que nada, hay que conocer la más importante de todas: `help`. Esta ofrece ayuda al usuario sobre las acciones disponibles, qué argumentos requiere cada una y qué opciones hay disponibles.

   Se ejecuta así:

   ```
   $ svn help
   ```

3. Descargue la versión más reciente del repositorio de la LCC. La forma más general de hacerlo sería:

   ```
   $ svn checkout https://svn.dcc.fceia.unr.edu.ar/svn/lcc/
   ```

   Sin embargo, no es recomendable hacerlo así. El repositorio entero es grande y por lo tanto demoraría mucho en descargarse. Además si se quiere trabajar en una sola materia, no resulta útil descargar el material de todas las otras también.

   Puede bajar solo un subdirectorio del repositorio, indicando una ruta más específica. Por ejemplo, para descargar solo el directorio de Arquitectura del Computador:

   ```
   $ svn checkout https://svn.dcc.fceia.unr.edu.ar/svn/lcc/R-222/
   ```

   O incluso puede descargar solo lo que corresponda a los alumnos de su año:

   ```
   $ svn checkout \
       https://svn.dcc.fceia.unr.edu.ar/svn/lcc/R-222/Alumnos/2018/
   ```

   (La barra invertida ("\") sirve para partir un comando en múltiples líneas.)

   Cuanto más específica sea la ruta, menor será la cantidad de archivos que se tengan que descargar y por lo tanto más rápido terminará el "checkout".

   Note que en los ejemplos anteriores se usó la ruta para acceso anónimo. Usando esta, podrá acceder al contenido de los subdirectorios `Alumnos` y `Public`, pero no podrá hacer ningún cambio. Si planea luego subir cambios, debe usar en cambio la ruta de acceso autenticado:

   ```
   $ svn checkout https://svn.dcc.fceia.unr.edu.ar/svn-no-anon/lcc/...
   ```

   Este paso pedirá su usuario y contraseña de Kleene. Si quiere evitar tener que ingresar como entrada su nombre de usuario cada vez, puede la opción `--username <usuario>` al ejecutar el comando, donde `<usuario>` es su nombre de usuario. Por ejemplo:

```
$ svn checkout https://svn.dcc.fceia.unr.edu.ar/svn/lcc/ \
    --username mstreet
```

El "checkout" creará un directorio con los contenidos del repositorio.

4. Consulte el estado actual de su copia de trabajo:

```
$ svn status
```

Si recién hizo el "checkout" y nada más, entonces "status" no mostrará ninguna salida. Esto quiere decir que no hay cambios pendientes de subir al servidor, está todo sincronizado.

5. Modifique un archivo cualquiera y luego cree un archivo nuevo. Si consulta el estado nuevamente (con `svn status`, obtendrá una salida como la siguiente:

```
M  <modificado>
?  <nuevo>
```

donde `<modificado>` y `<nuevo>` son los nombres de los archivos modificado y creado, respectivamente.

6. Si desea agregar un archivo al repositorio, por ejemplo `resultado_parcial.txt`, asegúrese de estar dentro del directorio generado por "checkout" o en uno de sus subdirectorios, y entonces ejecute:

```
$ svn add resultado_parcial.txt
```

Si consulta el estado nuevamente, aparecerá la siguiente línea en la salida:

```
A  resultado_parcial.txt
```

7. Para borrar un archivo, está la acción `remove`. Por ejemplo:

```
$ svn remove documento.pdf
```

En `svn status` aparecerá:

```
D  documento.pdf
```

Tenga en cuenta que si borra un archivo por los medios tradicionales (por ejemplo con el comando `rm` o con un administrador de archivos gráfico), este cambio no quedará registrado en Subversion. En tal caso, la línea de estado será diferente:

```
!  documento.pdf
```

8. Para recuperar un archivo borrado, puede usar `update` pasando como argumento la ruta al archivo. Ejemplo:

```
$ svn update documento.pdf
```

Esto vuelve a descargar del servidor la última versión del archivo. Al consultar el estado, verá que ya no aparece `documento.pdf`. Está sin cambios, como en un principio.

9. Una vez que esté a gusto con los cambios, debe aplicarlos al repositorio, esto es lo que se llama "commit":

   ```
   $ svn commit
   ```

   Este paso pedirá su usuario y contraseña.

   Se abre un editor de texto para que usted introduzca una descripción de los cambios hechos. Se trata de un pequeño (o si las circunstancias lo ameritan, no tan pequeño) texto legible por humanos que se asociará al conjunto de cambios que está por subir al servidor. Es muy recomendable escribir una descripción que refleje con claridad qué es lo que cambió, ya que esto luego sirve para navegar fácilmente por el historial de cambios.

   Si la descripción va a ser muy corta, se puede evitar abrir un editor de texto usando la opción `-m` más un pequeño texto como argumento:

   ```
   $ svn commit -m 'Descripción corta'
   ```

   Nótese que las comillas son necesarias en caso de que la descripción contenga espacios u otros caracteres especiales.

10. Para ver el historial de cambios, se usa `log`:

    ```
    $ svn log
    ```

11. Para ver las diferencias que irían en el próximo "commit":

    ```
    $ svn diff
    ```

12. Uno puede no solo descargar la última versión un archivo particular del servidor con `update`, como ya se mostró, sino también actualizar todo el repositorio a la vez y obtener todos los "commits" nuevos:

    ```
    $ svn update
    ```

    Esto es necesario para obtener los cambios hechos por otros usuarios en la copia local de uno.

13. Se pueden crear directorios con `mkdir`. Ejemplo:

    ```
    $ svn mkdir práctica2
    ```

14. Se pueden copiar archivos de forma eficiente bajo la supervisión de Subversion:

    ```
    $ svn copy <origen> <destino>
    ```

    donde `<origen>` y `<destino>` son las rutas que correspondan.

    Con esto, `<destino>` automáticamente queda bajo control de versiones, es decir, agregado en el repositorio (aunque no necesariamente subido al servidor).

    `copy` es útil para realizar entregas, copiando el contenido de `trunk` a un subdirectorio nuevo de `tags`.

15. De forma similar, con `move` se pueden mover archivos:

```
$ svn move <origen> <destino>
```

16. **Importante.** Por defecto, al hacer "checkout" con la ruta de acceso autenticado, las credenciales de acceso que se usen (es decir, el nombre de usuario y la contraseña) quedan guardadas en el sistema. Esto permite que las próximas ejecuciones que requieran autenticación no pidan los datos nuevamente al usuario.

    Si trabaja en una máquina pública (como los laboratorios de la facultad), preferirá que esto no quede guardado. Para esto hay dos opciones.

    a) Una es agregar, en los comandos `svn` respectivos, la opción `--no-auth-cache`. Esto es solo necesario en aquellos comandos que se comuniquen con el servidor; por ejemplo, hace falta para `svn checkout` pero no para `svn add`. Un ejemplo de uso de `svn co` de esta forma sería:

    ```
    $ svn checkout https://... --username mstreet --no-auth-cache
    ```

    b) Otra es que, una vez se haya terminado de trabajar con Subversion, se elimine el directorio donde se guardan las credenciales. En GNU/Linux, se trata del directorio `.subversion` en el directorio personal (tenga en cuenta que, al empezar el nombre con un punto, se trata de un directorio oculto).

    Se lo puede borrar tanto con un administrador de archivos gráfico como por línea de comando:

    ```
    $ rm -r ~/.subversion
    ```

5.1. Abreviaturas

Muchas acciones de Subversion permiten abreviaturas, con lo cual es lo mismo escribir `svn co` que `svn checkout`, por ejemplo. Algunas abreviaturas útiles se listan a continuación:

- `svn ci` (`svn commit`)
- `svn co` (`svn checkout`)
- `svn cp` (`svn copy`)
- `svn di` (`svn diff`)
- `svn h` (`svn help`)
- `svn mv` (`svn move`)
- `svn rm` (`svn remove`)
- `svn st` (`svn status`)
- `svn up` (`svn update`)

Para ver más abreviaturas, use `svn help`.

5.2. Estados

Al hacer `svn status`, por cada archivo cambiado de alguna forma en el árbol local con respecto a la última versión en el servidor, se muestra una línea con un caracter que indica el tipo de cambio. Los caracteres más importantes y sus significados son:

`A` agregado

`M` modificado

`D` borrado

`?` sin información (no agregado al repositorio)

`!` faltante

`C` conflicto

6. Enlaces útiles

- Antigua guía por Duilio Protti: `https://dcc.fceia.unr.edu.ar/services/svn-howto.html`
- Sitio web oficial de Subversion: `https://subversion.apache.org/`
- Libro desarrollado por la comunidad de Subversion: `http://svnbook.red-bean.com/`
- Sitio web oficial de la Fundación Apache: `https://apache.org/`
- Artículo sobre el software libre: `http://www.gnu.org/philosophy/free-sw.es.html`