Subversion

Nadir SOUALEM

1 Requirements

- Linux Users subversion client svn 1.6.5 or higher
- Windows users subversion client Tortoise 1.6.6 or higher

2 What is Subversion ?

Source control or *version control* is an important element of cooperationg develoopement. Subversion (svn) is an open source revision control system. It allows developers to track all versions of all files and directories. You can retrieve at any time backward and forward versions of your files and folder.
Svn can operate through networks(Apache or svnserve) which allows it to be used by people on different computers.

3 Subversions's feature and terminology

3.1 Repository

Repository is the central place in which all subversion transactions are kept. This location is a server where you can track all revision numbers. Here is the structure of a repository:

`conf/ dav/ db/ format hooks/ locks/`

conf A directory containing configuration files

dav A directory provided to mod_dav_svn for its private housekeeping data

db The data store for all of your versioned data

format A file that contains a single integer that indicates the version number of the repository layout

hooks A directory full of hook script templates (and hook scripts themselves, once you've installed some)

locks A directory for Subversion's repository lock files, used for tracking accessors to the repository

3.2 Working copy

A Subversion *working copy* is a revision copy of the repository tree on your local system. It is the copy you work with. The structure of a working copy is the structure of code project, it is not a database like the repository; remember it is a copy of revision. You can have several working copies.

3.3 Checkout

This operation is the first step to get a working copy from the repository into your local file system.

3.4 Update

An *update* brings your working copy up-to-date with the *HEAD* revision(last revision).

3.5 Commit

A *commit* operation send changes from your working copy to the repository.

4 Basic Work Cycle

You will start using a Subversion repository by doing a *checkout* of your project. Checking out a repository creates a working copy of it on your local machine. This copy contains the *HEAD* of the Subversion *repository*.
The typical work cycle looks like this:

- *update* your working copy
- Make changes
- Examine and test your changes
- *update* and merge others changes into your working copy
- *commit* your changes

5 Checkout a *working copy*

5.1 Access method to the repository

To Check-out a *working copy*, you must know the access method to the server.

file:/// direct repository access on local disk

http:// Access via WebDAV protocol

https:// Access via WebDAV protocol with SSL encryption

svn:// Access via custom protocol to an svnserve server

svn+ssh:// Access via custom protocol to an svnserve server through an SSH tunnel

For exemple, Gforge Inria repositories are accessible via **https://** or **svn+ssh**.
Note: Access via **svn+ssh** is faster than **https://**
I suppose here that the server is located in

`scm.gforge.inria.fr/svn/hydrolab/`

and the code project is in the *trunk* directory.

5.2 Linux

```
svn checkout svn+ssh://login@scm.gforge.inria.fr/svn/hydrolab/trunk .
```

or

```
svn checkout --username login
https://login@scm.gforge.inria.fr/svn/hydrolab/trunk .
```

The dot in end-of-line means *here*, i.e. in the current directory. you can replace it by a directory for example:

```
svn checkout --username login
https://login@scm.gforge.inria.fr/svn/hydrolab/trunk my/dir/
```

5.3 Windows

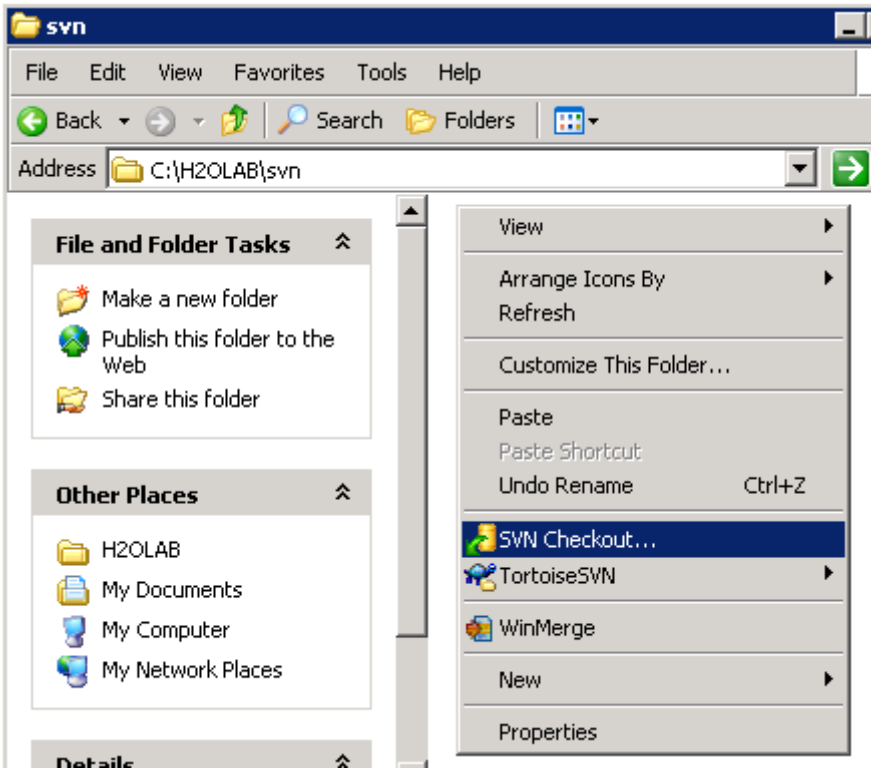

Figure 1: Checkout operation

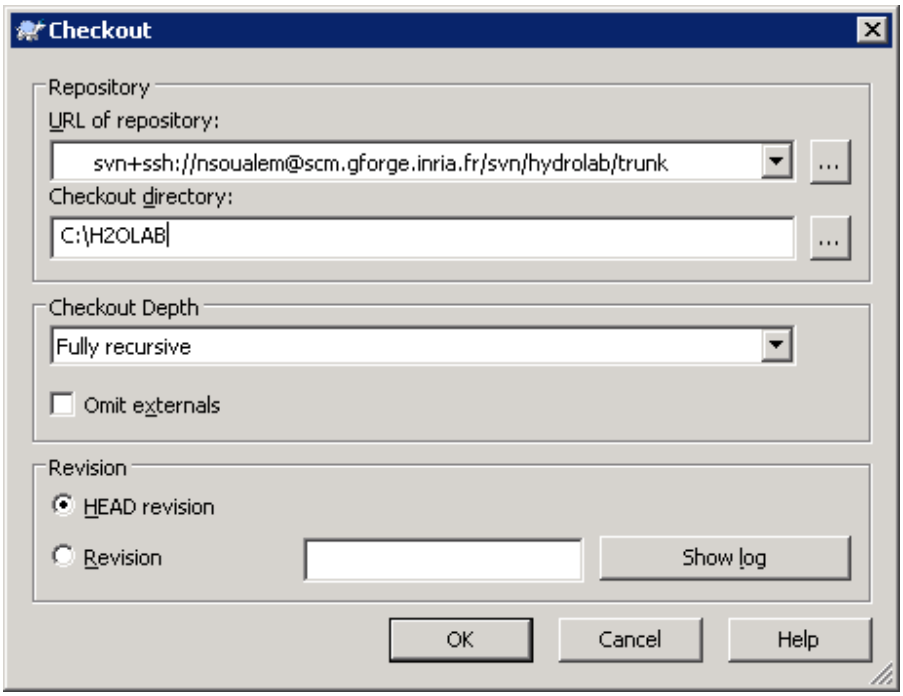

Figure 2: Inria Gforge Repository using **svn+ssh**

6 Make changes: Add, Move, Delete

6.1 Linux

To add a file or a directory

```
svn add filename
svn add directory_name
```

To move them:

```
svn mv filename filename_destination
svn mv directory_name directory_name_destination
```

To remove them:

```
svn del filename
svn del directory_name
```

6.2 Windows

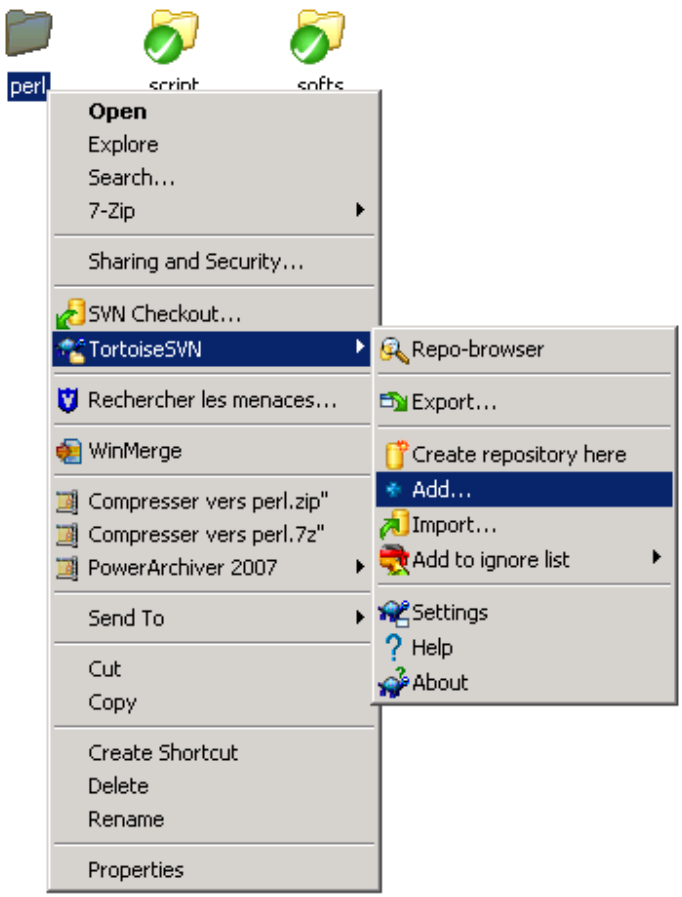

Figure 3: Add file or directory

Figure 4: Add file or directory

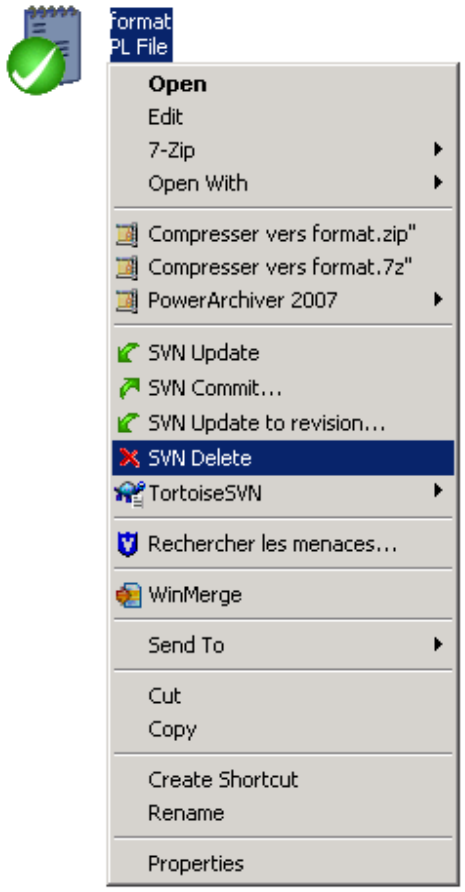

Figure 5: Add file or directory

7 What changes have been made ?

Subversion provides tools for reviewing what changes have been made

diff to see the differences between your working copy and the last update or checkout you have made

status gives a report on which files have been modified, which are scheduled to be added or deleted etc ...

revert will cancel your modifications

log to see whose edited file or directory tree

Note: status, **diff** and **revert** can be used without any network access. This makes it easy to manage your changes-in-progress when you are somewhere without a network connection.

7.1 Linux

diff

Diff in current directory, of a file or directory

```
svn diff
svn diff filename
svn diff directory_name
```

status

Status of current directory, file or directory

```
svn status
svn status filename
svn status directory_name
```

revert

Revert current directory, file or directory

```
svn revert
svn revert filename
svn revert directory_name
```

log

Log of the current directory, file or directory

```
svn log
svn log filename
svn log directory_name
```

7.2 Windows

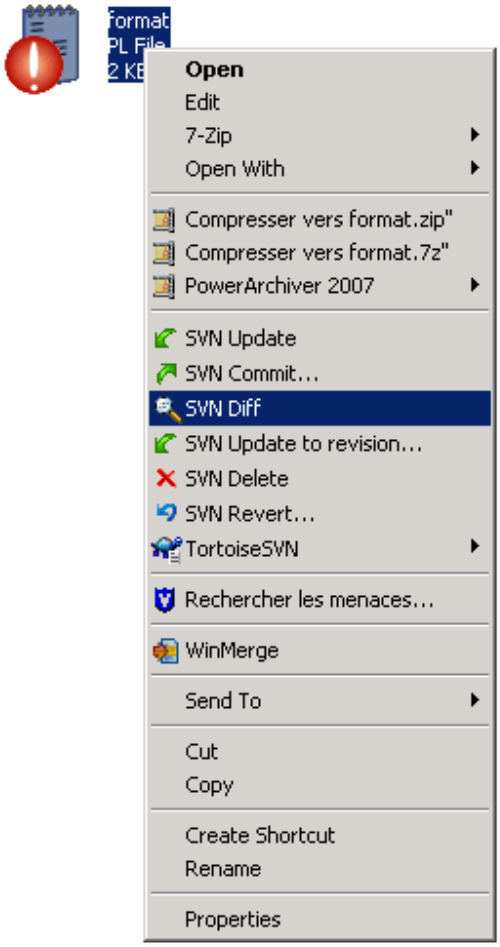

Figure 6: Add file or directory

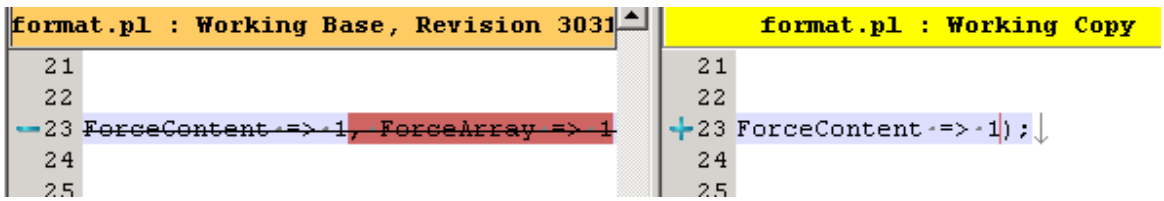

Figure 7: Add file or directory

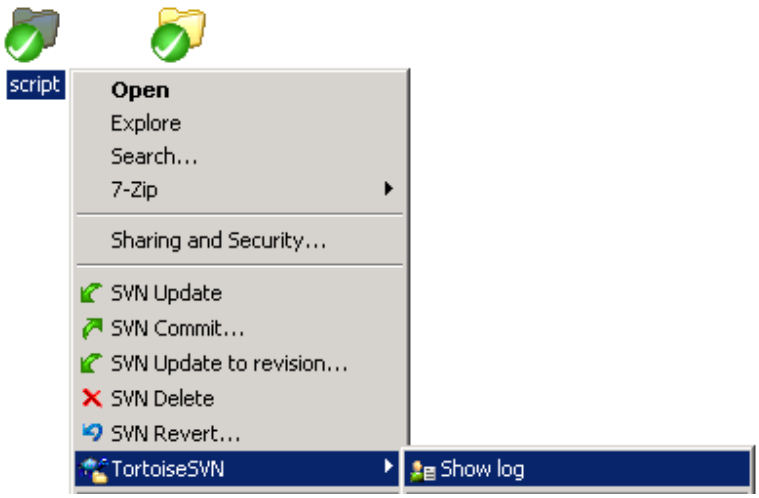

Figure 8: Add file or directory

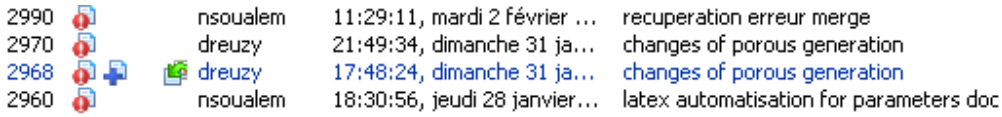

Figure 9: Add file or directory

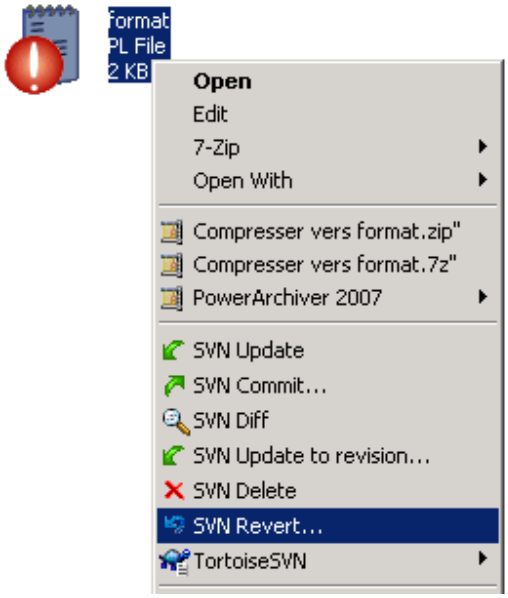

Figure 10: Add file or directory

8 Update your *working copy*

When someone uses the commit command to push their changes up to the server, you want to pull those changes down to your local machine. It is an *update*.

8.1 Linux

Update the current directory, a file or directory

```
svn update
svn update filename
svn update directory_name
```

8.2 Windows

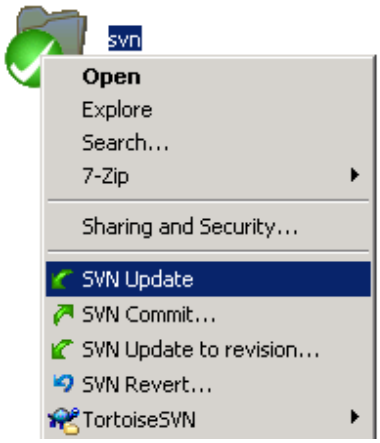

Figure 11: Add file or directory

9 Resolving potential conflicts

Someone else commits a modified version main.cpp to the repository. We suppose you have been working on the same file, and the changes from the server overlapped with your own. Suppose you run an update, you will obtain something like:

```
U  Launcher.cpp
C  main.cpp
Updated to revision 2612.
```

`C` stands for conflict. It means that subversion places a conflict marker. For every conflicted file, Subversion places three extra files in your working copy:

filename.mine this is your file as it existed in your working copy before you updated your working copythat is, without conflict markers. This file has your latest changes in it and nothing else

filename.rOLDREV this is the file that was the BASE revision before you updated your working copy. That is, the file that you checked out before you made your latest edits

filename.rNEWREV this is the file that your Subversion client just received from the server when you updated your working copy. This file corresponds to the HEAD revision of the repository

OLDREV is the revision number of the file in your *.svn* directory and **NEWREV** is the revision number of the repository *HEAD*.
If you get a conflict, you need to do one of three things:

- Merge the conflicted text by hand, by examining and editing the conflict markers within the file
- Copy one of the temporary files on top of your working file
- Run a svn revert *main.cpp* to throw away all of your local changes.

9.1 Linux

9.2 Windows

10 Commit your changes

When you make changes to files in your Working Copy, Commit will push those changes into the Repository.

10.1 Linux

Commit the current directory, a file or directory

```
svn commit
svn commit filename
svn commit directory_name
```

10.2 Windows

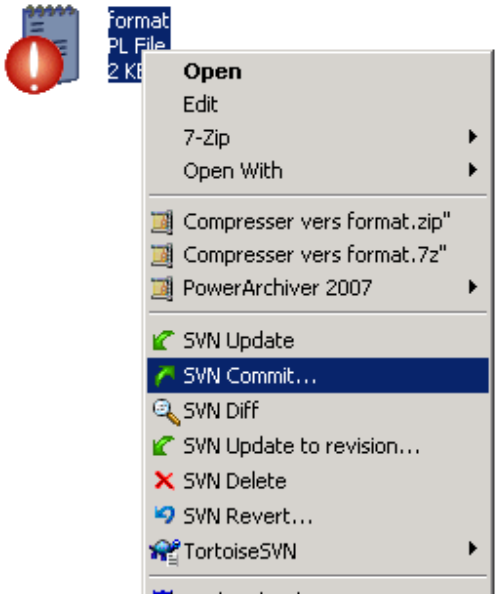

Figure 12: Add file or directory

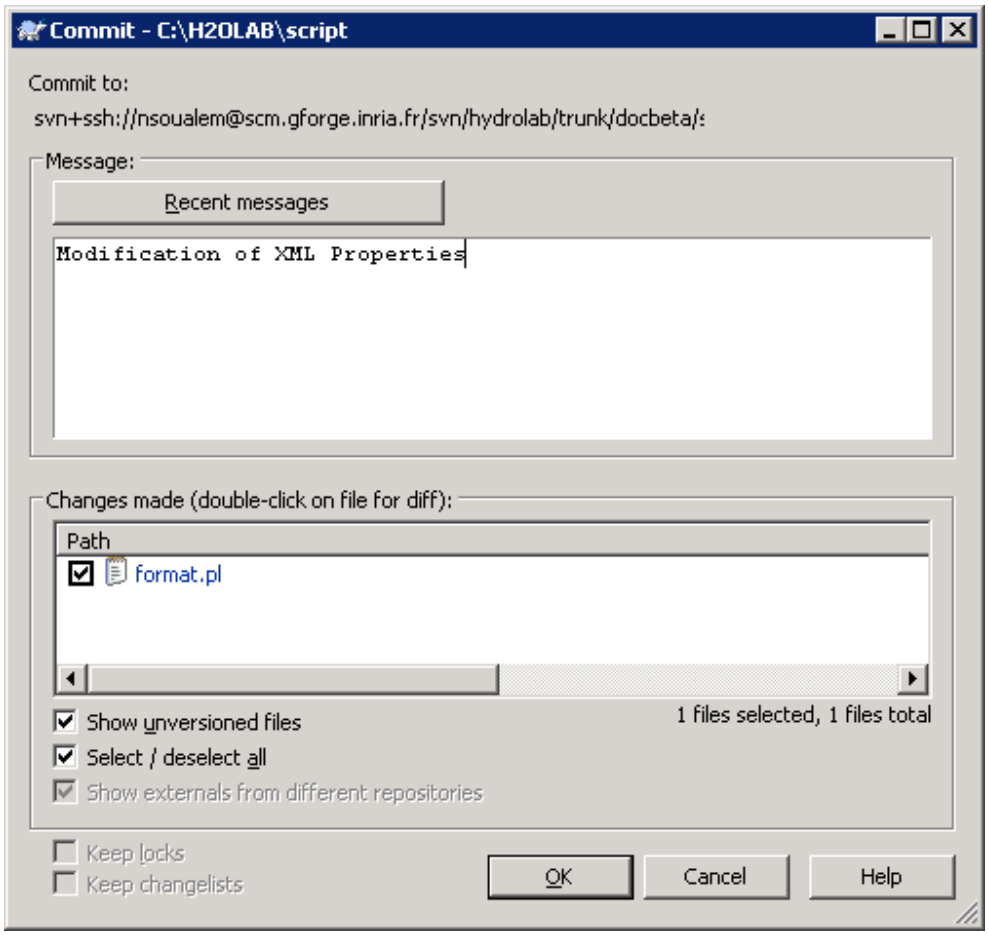

Figure 13: Add file or directory