



TECHNICAL REPORT

Asier Roo Mendiguchia
ICHEC System Administrator



Version Control :: Using Subversion on ICHEC's systems

Introduction

Subversion is a Version Control System designed to help you keep track of all your code changes when editing your applications. It can be an invaluable tool to quickly undo undesired changes or keep separate software branches.

This quick tutorial should guide you through basic configuration and most useful commands, while further official documentation and man pages will be necessary to make the most of your experience with Subversion.

Creating a repository

Repositories are the central places where we will keep all our versioned data. Let's start by creating our own repository inside our work folder:

```
$ svnadmin create /home/me/svn/
```

version control

Introduction	1
Creating a repository	1
Importing your projects	2
Working with subversion	3
Working cycle & commands	4
Working remotely	5

If we have a look inside, we will see a collection of directories and databases. We will not work directly with these files.

```
$ ls /home/me/svn/
conf/  dav/  db/  format  hooks/  locks/
README.txt
```



To administer the repository that has been created see:
<http://www.eecs.harvard.edu/~brchen/svn/svn-book-html-chunk/svn.reposadmin.html>

If you are or have been a CVS user in past, you may be interested in Subversion for CVS users:
<http://www.eecs.harvard.edu/~brchen/svn/svn-book-html-chunk/svn.forcv.html>

Importing your projects

It is important to note that Subversion does not have the concept of a "project". The repository is a virtual versioned file-system (only deals with directories and files), a large tree that can hold anything you wish. You may prefer to store only one project per repository or projects by placing into different directories.

To allow Subversion to understand where it should take your project data from, you will need to create a directory structure in line with its expectations and populate it with your files.

#1 Create a temporary directory to work in your home directory:

```
$ mkdir /home/me/tmp/myproject
```

#2 In this directory, Subversion will expect the following three directories by convention:

```
/home/me/tmp/myproject/  
    branches/  
    tags/  
    trunk/
```

#3 Initially you will work under the 'trunk' folder. Copy your project's source files and directories to the 'trunk' folder:

```
$ cp -r ~/code ~/tmp/myproject/trunk
```




And check that everything was copied as expected:

```
$ ls ~/tmp/myproject/trunk/code/  
    foo.c  
    bar.c  
    Makefile  
    ...
```

#4 Now let's make your repository store the tree of data we have copied using the 'import' command:

```
$ svn import /home/me/tmp/myproject file:///home/me/svn/ -m "initial import"  
Adding      /home/me/tmp/myproject/branches  
Adding      /home/me/tmp/myproject/tags  
Adding      /home/me/tmp/myproject/trunk  
Adding      /home/me/tmp/myproject/trunk/code/foo.c  
Adding      /home/me/tmp/myproject/trunk/code/bar.c  
Adding      /home/me/tmp/myproject/trunk/code/Makefile  
...  
Committed revision 1.
```

Where:

-  `/home/me/tmp/myproject` Temporary directory where your project data is kept in svn format
-  `file:///home/me/svn/` Where your svn repository is located
-  `-m "initial report"` A text comment you can add as a reference

Note: If you wish you can now delete the temporary directory; all the data is still kept in the repository. In the repository directory, your files are stored inside database files, you will not find versions of them. At the top of the repository's imaginary file-system there is a directory named 'origProject'



In this tutorial we will look at the simplest case however you can check the following link for further discussion about how to organise your data.
<http://svnbook.red-bean.com/en/1.4/svn.reposadmin.planning.html>

Working with Subversion for the first time

#1 Create a new temporary directory where you will start working with a copy of the data contained in your repository:

```
$ mkdir /home/me/work
```

#2 Then get your files from the repository: this action can be performed by using 'checkout' command:

```
$ svn checkout file:///home/me/svn/ myproject
```

#3 Now check that your files are there and you can start working them:

```
$ ls /home/me/work/myproject/trunk/code/
    foo.c
    bar.c
    Makefile
    ...

$ vi /home/me/work/myproject/trunk/code/foo.c

$ svn mkdir /home/me/work/myproject/trunk/code/newdir/
```

#4 It is recommended to check your working copy with your repository's copy just in case somebody else working with you has modified some part of your code:

```
$ svn update
```

#5 Finally you will have to commit your changes: this will result in a new revision number of your repository tree.

```
$ svn commit -m "Updated modifications for whatever improvement."
```



When updating your working copy, you should carefully check that there are no conflicts with other changes made by your co-workers.

For further details on how to merge differences check:
<http://svnbook.red-bean.com/nightly/en/svn.branchmerge.html>


```
$ svn update
Conflict discovered in 'foo.c'.
Select: (p) postpone, (df) diff-full, (e) edit,
        (h)elp for more options : p
C foo.c
Updated to revision 2.

$ ls -l
foo.c
foo.c.mine
foo.c.r1
foo.c.r2
```

#6 In this example we will go for the simplest case and we will imagine that only our copy is the right one, overwriting all the changes that others made:

```
$ svn resolve --accept working foo.c
Resolved conflicted state of 'foo.c'

$ svn commit -m "My working copy is OK."
```

Working cycle & commands 	
Update your working copy	svn update
Make changes.	svn add svn delete svn copy svn move
Examine your changes	svn status svn diff
Possibly undo some changes	svn revert
Resolve conflicts (merge others' changes)	svn update svn resolve
Commit your changes	svn commit
For a detailed guide to svn command line client commands check: http://svnbook.red-bean.com/nightly/en/svn.ref.html	

Working remotely



You are probably working with ICHEC resources remotely and your project could be in a development phase: so you have to make many changes in your code, and you need to compile and execute frequently using the system's development queues. You could use Subversion to keep your repository in your ICHEC home directory, and use a checked-out copy on your workstation where you work with your favourite editor.

Advantages

- Easy to decide which version of your code you want to compile and execute on ICHEC's compute resources.
- When editing locally, only differences will be sent through the network so you can test your new code quickly.

Setup

Assuming you already created your repository with all the data on ICHEC's systems, you can access your repository by typing:

```
$ svn list svn+ssh://myAcct@host.example.com//home/me/svn/code/  
harry@host.example.com's password: *****  
  
foo  
bar  
baz  
...
```

Where:

- | | | |
|---|------------------|---|
| ➔ | svn+ssh | Subversion client is invoking a local ssh process |
| ➔ | host.example.com | Connecting to this host |
| ➔ | myAcct | The user you want to authenticate with |
| ➔ | /home/me/svn/ | Path to repository in your home directory |

To avoid being asked your password every time you work with your remote repository you can use 'ssh-agent' to hold your public keys used for public key authentication with ssh (generate your keys using ssh-keygen see: <http://www.ichec.ie/FAQ.php#16>):

```
$ ssh-agent bash  
$ ssh-add  
Identity added: /home/myself/.ssh/id_rsa (/home/myself/.ssh/id_rsa)
```

You can now start working inside this bash environment with Subversion.