The PracTEX Journal, 2007, No. 3
Article revision 09/04/2008, revised version

LaTeX Document Management with Subversion, rev: 25

Uwe Ziegenhagen

Email latex@ziegenhagen.info
Website http://www.uweziegenhagen.de
Address Humboldt-Universität zu Berlin, Germany
Center for Applied Statistics and Economics

Abstract From the single-author composition of a Bachelor thesis to the creation of a book by a team there are many occasions, where version management of a document may be helpful. With the aim of overcoming the shortcomings of CVS (Concurrent Version System) the Subversion version control system was implemented.

In this article I will describe the Subversion setup on Windows and Linux systems, the elementary steps of document management and various LaTeX packages working hand in hand with Subversion.

Note: The original version of this article was published in the PracTeX Journal issue 3/2007 and is available through http://www.tug.org/pracjourn/2007-3/ziegenhagen/

1 CVS versus Subversion

Contrary to CVS the versioning scheme of Subversion does not refer to single files anymore but to a whole tree of files. Each revision number n refers to the state of the repository after the n-th commit. When we speak about a file in revision 4 we mean the file in the state of revision 4.

The revision numbers of a single file may even have gaps if it had not been changed on every commit to the repository. Table 1 illustrates an example: Up to revision 4 all files have been changed before each commit, so the revision of the repository and the revision numbers of the files are equal. Before the commit to revision 5 only chapter1.tex is modified however the whole repository receives the revision number 5, before the commit to version 6 all files were modified again and therefore have the revision number 6.

Table 1: Gaps in Subversion revisions

Revision 4	Revision 5	Revision 6
thesis.tex:4	thesis.tex:4	thesis.tex:6
preamble.tex:4	preamble.tex4	preamble.tex:6
chapter1.tex:4	chapter1.tex:5	chapter1.tex:6

On each checkout from a Subversion repository the highest revision number of each file will be checked out which is smaller or equal to the desired revision number. Subversion stores a second copy of each file in a special directory (`.svn`) on each checkout, update and commit.

Although the required space on the hard-disk doubles there are certain advantages, especially when the repository is on a remote server: Local changes can be viewed without access to the network and on the commit of a file Subversion has only to send the changed parts whereas CVS calculates the changes on the server and has to send the whole file on each commit. Commits are atomic, which means a change to a file is either completely stored or not stored at all. Thus network issues or concurrent commits cannot lead to an inconsistent status.

2 Installation

There are different options for the installation of Subversion. One can either use svnserve [3] or install Subversion as an Apache 2 module, which uses WebDAV[1].

In this article I will focus on the latter option by installing Subversion as an Apache2 module, since the integration into Apache 2 provides a few interesting features such as the possibility of browsing through repositories using a web browser and the use of the Apache authentification mechanisms.

1. *Web-based Distributed Authoring and Versioning*, a set of extensions to the HTTP protocol which allows users to collaboratively edit and manage files on remote web servers.

2.1 Windows XP

2.1.1 Apache Setup

Binary versions of Apache 2 are available from [1], however I usually prefer to use a WAMP[2]-solution provided by apachefriends.org. We extract the xampp.zip[3] to e.g. `C:/xampp` and start the Apache server using `xampp-control.exe`. When we open `http://localhost` in a web browser the XAMPP start page should show up as depicted in Figure 1.

Figure 1: Screenshot of xampp starting page

As we assume that only the local computer will be allowed to access the webserver it is necessary to secure the machine against access from outside. For details please see the respective chapter in the Apache documentation [7].

2. Windows-Apache-MySQL-PHP
3. current version at printtime: 1.6.6a

2.1.2 Subversion

We download Subversion[4] from [2] and extract all files from the zip archive to e.g. `C:/Program Files/Subversion`. After adding the path to the `C:/Program Files/Subversion/bin` directory to the PATH environment variable from Windows, we can call `svn help` from the commandline to check if our installation is working. In the next step we copy `mod_authz_svn.so` and `mod_dav_svn.so` from the `subversion/bin` directory to the Apache modules directory and overwrite older versions of these files if necessary.

In the final step we enable WebDAV and the Subversion module by adding

- `LoadModule dav_svn_module modules/mod_dav_svn.so` and
- `LoadModule authz_svn_module modules/mod_authz_svn.so`

to the `httpd.conf` in the Apache `/conf` directory. Before we restart Apache to load the modules we need to make further adjustments to this file. We create a root directory for all our repositories (e.g. `c:/allMyRepositories`) and add the code from Listing 1 to `httpd.conf`:

```
1 <Location /svn>
2 DAV svn
3
4 SVNParentPath c:/allMyRepositories
5 </Location>
```

Listing 1: Setup code for the Windows repository root

From the command line we change to the `c:/allMyRepositories` directory and create our first repository by executing `svnadmin create firstSample`.

If we now open `http://localhost/svn/firstSample/` in a browser we should see an empty directory listing with the headline `Revision 0: /`. The basic installation of Subversion is now done, however we can achieve a much more convenient way of handling repositories by installing TortoiseSVN.

2.1.3 TortoiseSVN

TortoiseSVN[5] [4] is a free Subversion client, implemented as a Windows shell extension. It features a multilingual interface with Windows Explorer integration,

4. current version at printtime: 1.4.6
5. current version at printtime: 1.4.8

its icon overlays show immediately which files/folders have been changed and need to be committed to the repository. The installation is straightforward, after rebooting the computer we find various entries in the context menu to manage our repositories. Besides there are more clients available, for example RapidSVN (Windows, Unix/Linux) and SVNcommander (Linux).

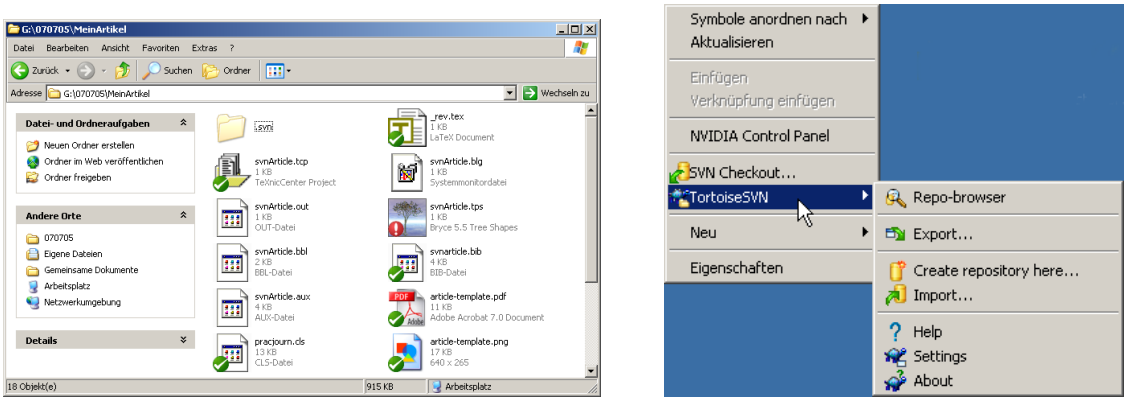

Figure 2: Screenshot of a Working Directory with TortoiseSVN installed and context menu of TortoiseSVN 1.4.6.

2.2 Linux (Ubuntu 7.10)

The installation on a Linux system is much easier than the installation on Windows. Using `sudo apt-get install` or the Synaptic package manager we install the following packages:

- apache2.2-common and apache2-utils
- libapache2-svn
- subversion

Additional packages are selected automatically by the package management tool. After the installation of these packages the last remaining step is to make the necessary adjustments to `/etc/apache2/sites-available/default` and to set the access rights for this directory via `chmod -R 770 /home/uwe/repositoryRoot` as depicted in Listing 2.

```
<Location /svn>
DAV svn

SVNParentPath /home/uwe/repositoryRoot
</Location>
```

Listing 2: Setup code for the Linux repository root

2.3 svnserve under Linux

I recently checked the svnserve daemon under Ubuntu 7.x, the setup procedure was as following: Using `apt-get install subversion` I downloaded and installed the latest subversion package (For most of these commands `sudo` may be necessary when working with Ubuntu).

The next step is to create the necessary folders by `mkdir /srv/svn` and `mkdir /srv/svn/repos` and `svnadmin create /srv/svn/repos/test` and to create a new user named 'svn', as it may be a potential security risk to have the svnserve process run with root privileges: `useradd -d /home/svn -m svn`. Listing 3 shows the necessary commands to set the rights.

```
addgroup subversion          # creates a group 'subversion'
adduser svn subversion       # add user svn to group 'subversion'
chgrp -R subversion /srv/svn/ # set group 'subversion' for all files
chown -R svn /srv/svn/repos # change the owner of the repos dir
chmod -R o-rwx /srv/svn/     # no one else is allowed to do anything
chmod -R g+rw /srv/svn/      # grant write access to group members
chmod -R g+s /srv/svn/       # allow logs to be written
```

Listing 3: Linux commands for setting the access rights (to be changed later, allow write access for everybody)

Next we create a repository by `svnadmin create /srv/svn/repos/test` and edit the `subversion.conf` in the `conf` subdirectory as shown in Listing 4. We will temporarily allow anonymous writing and reading. We will set up the correct access rights in a later step.

```
[general]
### These options control access to the repository for unauthenticated
### and authenticated users. Valid values are "write", "read",
### and "none". The sample settings below are the defaults.
anon-access = write
auth-access = write
```

Listing 4: Commands for setting the rights

The next step is to modify the `iptables` firewall settings by running `iptables -A INPUT -p tcp -dport 3690 -j ACCEPT` from the commandline. From this moment iptables should let subversion communication pass the filters. Using the `svn` user we start the `svnserve` daemon now using `svnserve -d -listen-host <ip>-r /srv/svn/repos`. `netstat -tulpen` should bring some output similar to the one in Listing 5.

```
root@:~# netstat -tulpen
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State User Inode PID/
    Program name
tcp       0      0 <ip>:3690 0.0.0.0:* LISTEN 1000 615571 22047/svnserve
tcp       0      0 0.0.0.0:22 0.0.0.0:* LISTEN 0 615506 21997/sshd
```

Listing 5: Using `netstat` to check svnserve

If we run now `svn co svn://<IP>/test` or use TortoiseSVN there should come a message indicating that revision 0 was successfully checked out. What if you receive an error instead? If the connection fails there is high probability that the iptables configuration is faulty and needs to be checked. If the connection is made but subversion complains about denied access, check the `subversion.conf` settings. If anything else fails just drop the error message into Google.

As mentioned above security issues have not been touched so far, so we need to block access from unauthenticated users in the next step. We set the rights that way, that only authenticated users may access the repository and uncomment the line with the `password-db`, see Listing 6.

```
[general]
### These options control access to the repository for unauthenticated
### and authenticated users. Valid values are "write", "read",
### and "none". The sample settings below are the defaults.
anon-access = none
auth-access = write
### The password-db option controls the location of the password
### database file. Unless you specify a path starting with a /,
### the file's location is relative to the conf directory.
### Uncomment the line below to use the default password file.
password-db = passwd
```

Listing 6: Restricting the use of the repository

Listing 7 shows the `password-db` file created by subversion with one user added, note that the passwords are saved in unencrypted plain text. Please also note that the communication itself is still unencrypted so this way of accessing the repositories should be used only in trusted environments. Fo untrusted networks one can tunnel all subversion traffic through SSH, for details please see the manual.

```
### This file is an example password file for svnserve.
### Its format is similar to that of svnserve.conf. As shown in the
### example below it contains one section labelled [users].
### The name and password for each user follow, one account per line.

[users]
# harry = harryssecret
# sally = sallyssecret
andreas=mypassword
```

Listing 7: Example `password-db` file

2.4 Repository Usage

2.4.1 Adding files to the repository

To fill the repository we created during the installation we create a empty directory (all files in this directory will be imported to the repository) which we

populate with a small LaTeX document (article-template.tex):

```
\documentclass{article}
\begin{document}

Hello World!

\end{document}
```

Listing 8: A simple LaTeX file

Using the the command line (`svn import http://localhost/svn/firstSample/ -m "import"`) or the TortoiseSVN context menu we can now import the file using our URL for the repository `http://localhost/svn/firstSample/` and use "import" as comment. Apache will list now `Revision 1: /` when we browse the repository (see Figure 3. To work with the files we need check them out into a `working directory`. The files in the working directory are the files we edit, all future commits are made from this directory.

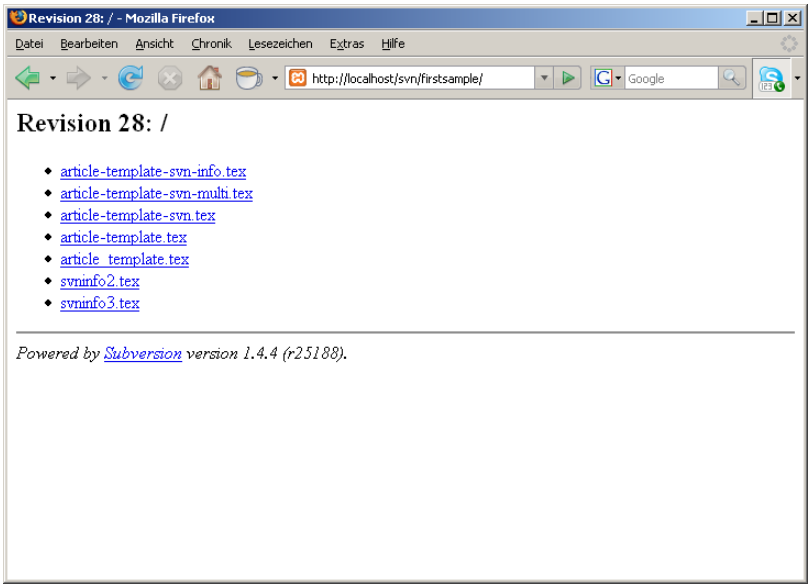

Figure 3: Repository browsing with Apache and Subversion Module

2.4.2 Backing and Restore

The introduction of Subversion as a version control system cannot be valued high enough in cases of damaged harddrives or stolen computers. However, of equal importance is the backup of the Subversion repositories. The important Subversion commands are `svnadmin dump` and `svnadmin load`. For details on those commands see the Subversion manual [6].

2.4.3 Copying, Moving and Deleting Files

`svn move`

3 Integration with LaTeX

To integrate the Subversion metadata in our LaTeX files we need to include the keywords in the LaTeX-file and tell Subversion to expand them. The following list contains the available keywords and their description:

Date (LastChangedAt) date and time of last check-in
Revision: (LastChangedRevision) the number of the revision
Author: (LastChangedBy) name of the submitting author
HeadURL: the URL of this file
Id: a summary of the above keywords

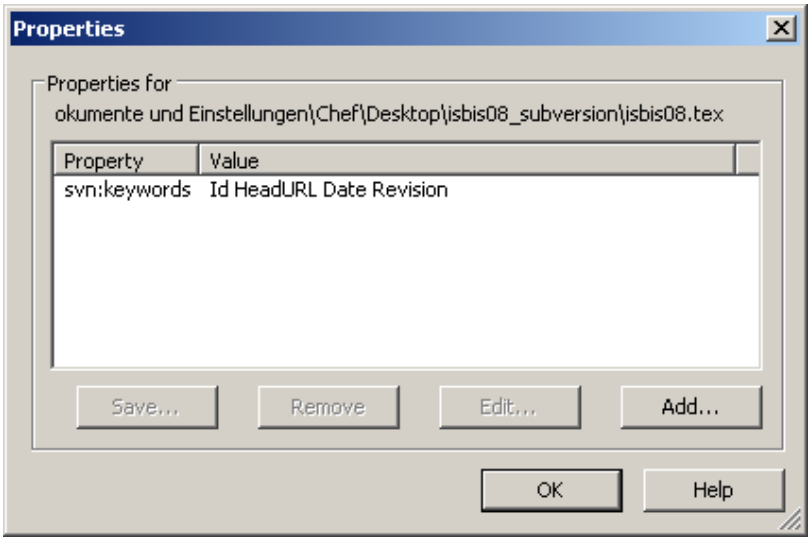

Figure 4: Set subversion keywords via TortoiseSVN context menu

After running `svn propset svn:keywords "Date HeadURL Revision Id" article_template.tex` from the commandline or using the TortoiseSVN context menu (see Figure 4) Subversion will expand those keywords (enclosed in $) in our file when we include them in the LaTeX code. Subversion will expand the keywords as following:

```
% $Revision: 25 $
% $HeadURL: http://tools.assembla.com/svn/svnArticle/svnArticle.tex $
% $Date: 2008-04-09 09:27:54 +0200 (Mi, 09 Apr 2008) $
% $Author$
% $Id: svnArticle.tex 25 2008-04-09 07:27:54Z uweziegenhagen $

\documentclass{article}
\begin{document}
 Hello World!
\end{document}
```

Listing 9: A sample file with expanded Subversion keywords

All LaTeXpackages introduced in the following are based on the evaluation of these keywords.

3.1 svn

The `svn` package allows access the metainformation by evaluating the Subversion information using a `\SVN $Keyword: <metadata>$` syntax. If the keywords are correctly expanded, then the `svn` package defines:

- `\SVNDate` for the date of the checkin, `\SVNTime` as the check-in time and `\SVNRawDate` as raw date and time if `Keyword` was $Date$.
- `\SVNKeyword` otherwise (Examples: \SVNId, \SVNHeadURL)

July 15, 2007
2007-07-15 17:33:30 +0200 (So, 15 Jul 2007)
17:33:30
article-template.tex 12 2007-07-15 15:33:30Z
http://localhost/svn/firstSample/article-template.tex

Figure 5: Output of article-template.tex (see Listing 10 with `svn` package

```
\documentclass{article}
\usepackage{svn}

\SVN $Id: svnArticle.tex 25 2008-04-09 07:27:54Z uweziegenhagen $
\SVN $Date: 2008-04-09 09:27:54 +0200 (Mi, 09 Apr 2008) $
\SVN $Id: svnArticle.tex 25 2008-04-09 07:27:54Z uweziegenhagen $
\SVN $HeadURL: http://tools.assembla.com/svn/svnArticle/svnArticle.tex $

\begin{document}

\SVNDate \\
\SVNRawDate \\
\SVNTime \\
\SVNId \\
\SVNHeadURL
\end{document}
```

Listing 10: A sample file using the svn package

3.2 svninfo

The svninfo package needs information from the Id keyword only which need to follow the \svnInfo command: \svnInfo *Id* : *article* − *template* − *svn* − *info.tex*182007 − 07 − 1516 : 11 : 21*Z*

To use the meta information the package defines the following commands:

- \svnInfoFile the name of the file
- \svnInfoRevision the revision number
- \svnInfoDate the date of the last check-in
- \svnInfoTime the time of the last check-in
- \svnInfoYear the year of \svnInfoDate
- \svnInfoMonth the month of \svnInfoDate
- \svnInfoDay the day of \svnInfoDate
- \svnInfoOwner the owner of the file (if specified at check-in)
- \svnToday date of last check-in in the \today format
- \svnInfoMinRevision the minimum revision of the document
- \svnInfoMaxRevision the maximum revision of the document

\svnInfoMinRevision and \svnInfoMaxRevision are useful for multi-file documents. Furthermore the packages allows a few optional parameters such as fancyhdr, eso-foot, scrpage2 to typeset Subversion information in the margin or the footer of the document. For details please see the manual.

3.3 svn-multi

The svn-multi package provides two commands, \svnid and \svnidlong, to capture the input from Subversion. To use the variables, the package provides the following commands:

- \svnrev the revision
- \svndate the date of the last check-in
- \svnauthor the author
- \svnfilerev the revision of the current file if it contains a \svnid or \svnidlong or the values of the last file if it does not contain one of these commands
- \svnmainurl and \svnmainfilename typeset the URL respectively name of the main file, as it was defined by the internal command \svnmainfile at the end of the preamble

Furthermore svn-multi uses \svn{*keyword*} and \svnkw{*keyword*} to print Subversion keywords directly. To access date information the package provides some more commands, explanations can be seen directly from each respective name: \svnfileyear, \svnfilemonth, \svnfileday, \svnfilehour, \svnfileminute, \svnfilesecond, \svnfiletimezone, \svnyear, \svnmonth, \svnday, \svnhour, \svnminute, \svnsecond and \svntimezone.

4 Conclusion

This article described the basic usage of Subversion from LaTeX and the most common features of the three LaTeX packages svn, svninfo and svn-multi. More information about integration with LaTeX or Subversion itself can be found in the documentation of the packages or in books on Subversion ([6, 8]. Feedback on this article is welcome, if you find any mistakes or have comments please send me an email. Updates of the article can later be found online at http://www.uweziegenhagen.de/latex/

References

[1] Apache 2 web server. URL http://httpd.apache.org.

[2] Subversion. URL http://subversion.tigris.org/.

[3] Svnserve based server. URL http://tortoisesvn.net/docs/nightly/TortoiseSVN_en/tsvn-serversetup-svnserve.html.

[4] TortoiseSVN. URL http://tortoisesvn.tigris.org.

[5] apachefriends.org. Xampp. URL http://www.apachefriends.org/en/.

[6] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. M. Pilato. *Version Control with Subversion. Next Generation Open Source Version Control*. O'Reilly, 2004. URL http://svnbook.red-bean.com/.

[7] Apache Foundation. Apache HTTP server version 2.2 documentation. URL http://httpd.apache.org/docs/2.0/en/.

[8] Mike Mason. *Pragmatic Version Control Using Subversion*. Pragmatic Programmers LLC., 2006.