The PracTEX Journal, 2007, No. 3
Article revision 2007/08/07

Version Control of LaTeX Documents with `svn-multi`

Martin Scharrer

Email martin@scharrer-online.de
Website http://www.scharrer-online.de/latex/

Abstract This paper describes how to use the software Subversion to version control your LaTeX files while also placing the current revision information in your document using the package `svn-multi` (v1.3 or later). It covers all steps needed to setup and use Subversion, and to manage multi-file documents. Usage examples are provided to show the basic and advanced features to allow the reader to get the most out of the package.

1 Introduction to Version Control

Most people working with source code, whether it is a software programming language, markup languages like HTML/XML or in our case LaTeX, face the problem on how to organise their code when it spans over an increasing number of files. Backups should be created in order to roll back changes and to rescue lost data. Nontrivial changes should be logged and important revisions should be marked so they can be found at a later stage. Also when multiple people work on the same source code simultaneously their changes must be synchronized to ensure that their changes are not overwritten unintentionally.

This is the point where a *Version Control* software should be used. *Version Control* (VC), which is also called *Revision* or *Source Control*, manages your files, directories, and the changes made to them. For this all created revisions are stored in a repository which can be envisaged as a special kind of database. Every time the source code is revised it should be committed to the repository where it is saved in a compact differential form together with a log message describing the changes. The revision number is then incremented to identify the new revision.

Subversion [1] is a free modern general purpose VC system which was developed as a replacement to the widely used *Concurrent Versions System* (CVS).

Subversion can be easily used to manage LaTeX code of any size and any related files, e.g. image files, configuration files and the outputted PS or PDF files.

The author's LaTeX package `svn-multi` [3] can be used to place automatic revision keywords, like for example, the last changed revision number, author and date, into your LaTeX files and typeset this information anywhere in your document. If the document contains multiple files which are included (using `\include` or `\input`) in a master file then all files can and should contain these keywords. In addition to the keyword values of the current file, the most recent changed file is recognized and its values are also provided as macros using an auxiliary file (.svn extention).

2 Preparing Subversion

After Subversion has been installed, a repository must be created and existing source files must be placed in this repository for version control. To enable keyword expansion in Subversion a so-called "property" must be attached to these files.

2.1 Installation

The installation of subversion is simple, under MS Windows just download the installer (svn-⟨*version*⟩-setup.exe) from `http://subversion.tigris.org/project_packages.html#windows`, run this executable file and follow the installation instructions. Under Linux the normal software management tool of your distribution can be used to install the Subversion package. For example the author installed it on his Gentoo Linux OS using "`emerge -av subversion`".

It is also recommended to install the config file as outlined in Appendix A.

2.2 GUI Front-ends for Subversion

Subversion provides a command line interface which is good but can be difficult to use for some users who are primarily used to working with a graphical user interface under MS Windows.

Such users can use *TortoiseSVN*, which is a graphics front-end for MS Windows and integrates nicely with Windows Explorer, showing the current status of your

files as overlay icons. It can be freely downloaded from http://tortoisesvn.net/downloads and can be installed simply by running the installer executable file.

Under Linux, *KDESVN* (http://www.alwins-world.de/wiki/programs/kdesvn) can be used as a Konqueror plug-in or a stand-alone application for non-KDE users. It can be installed using the same software management tool used for the installation of Subversion.

Also other GUI front-ends exist, some of them can be used on both operating systems, e.g. *RapidSVN*, *SmartSVN*, *Subcommander* and more.

2.3 Creation of a local repository

A local Subversion repository is created in the command line using

`svnadmin create` ⟨*path of new repository*⟩

which works under Linux and Windows (as all Subversion command line tools do). Using TortoiseSVN just open the Windows Explorer, create an empty folder, right-click on it and select 'TortoiseSVN' → 'Create repository here'. Ensure a "FSFS" type repository is created as well.

2.4 Setting up a local working directory

While all revision of your code is saved in the repository the current version under development lies in a "working directory" which contains VC metadata in a hidden subdirectory.

The simplest way to get a working directory is to check-out your fresh repository using

`svn checkout file://`⟨*path of your repository*⟩ ⟨*working dir name*⟩
or using TortoiseSVN:
Right click on an empty directory → 'SVN Checkout ...'

and then copy all your existing document files to the working dir. To enable version control on these files, perfom the following inside the working directory

`svn add *`
or
Mark all files (CTRL-A), right-click them → 'TortoiseSVN' → 'Add' then click the 'Ok' bottom

The Subversion team suggests to create three directories 'trunk', 'tags' and 'branches' off the repository root. All files should then put in this 'trunk' while the other directories are used to tag revisions and to create development branches. For LaTeX sources this doesn't have to be done as long you don't need tags and branches (in an extra directory).

The working directory must to be commited using

`svn commit`
or
Right-click on working directory → 'SVN Commit'

The log message can be "Initial commit" or similar.

2.5 Basic Workcycle

When using Subversion for version control a basic workcycle should be followed to reduce the risk of problems, e.g. with concurrent changes made by other authors. The cycle starts when you need to modify your LaTeX sources. Here it is assumed that a working directory is already present (as described in the last section).

Tables 1 and 2 list all the main Subversion subcommands and their most important options.

2.5.1 Update your working directory

To ensure that you are working with the latest revision of the repository, the working directory should be updated before every change is performed. If this step is omitted collisions between changes from different authors can easily occur. Even single author sources should be updated, especially before manipulating version controlled directories which must be at the latest revision in order to be changed.

The update process is straightforward:

`svn update [⟨file or dir or nothing (current dir)⟩]`
or
Right-click on working directory (or file or subdir) → 'SVN Update'

When updating the root of the working directory all contained files and subdirectories are also updated. Please note that the displayed brackets `[ ]` symbolise a optional argument which, when used, should be written without the brackets.

2.5.2 Make changes, add new or delete files

The files under version control can be changed as normal with our prefered editor or any other software. Please note that the files and directory are linked to the working directories by name, so if you wish to rename or move files you have to do this using the Subversion client tools and not using methods provided by the operating system. One important concept is that all changes described here only get taken over with the next commit and can be reverted to the previous revision at any time.

New created files can be added, i.e. put under version control, using

`svn add ⟨file(s) and/or dir(s)⟩`
or
Right-click on the new file or dir → 'TortoiseSVN' → 'Add' then click the 'Ok' bottom

To remove files from version control, which also deletes the working copy in the working directory, use

`svn del ⟨file(s) and/or dir(s)⟩`
or
Right-click on the file or dir → 'TortoiseSVN' → 'Delete'

The deleted files are no longer part of future revisions but of course still exist in older revisions.

Subversion, unlike CVS, also supports the copying and moving/renaming of files and whole directories under version control. The copies are created very efficiently in the repository by internally linking them to the revision of the original file or dir. This action attaches the revision history of the original file or dir to the new one and allows the later tracking of the relationship between them. Such

copies are very cheap in terms of repository space and are also used to create branches and tags. Copying is done using

> `svn copy` ⟨*existing file/dir*⟩ ⟨*new file/dir*⟩
> or
> Right-click on and hold existing file or dir, drag-n-drop it to the new location, release the right-click and select 'SVN Copy versioned files here' in the appearing menue.

The moving of version controlled items in Subversion is implemented by copying the original file or dir to the new location and then deleting it afterwards. This can performed using

> `svn move` ⟨*existing file/dir*⟩ ⟨*new file/dir*⟩
> or
> Right-click on and hold existing file or dir, drag-n-drop it to the new location, release the right-click and select 'SVN Move versioned files here' in the appearing menue.

2.5.3 Commit our changes to the repository

Finally all local changes made to the working directory are committed to the repository to create a new revision. A suitable log message should be provided to be able to understand and track changes. The commit, also called check-in, is done using

> `svn ci [`⟨*file or dir or nothing (current dir)*⟩`]`
> or
> Right-click on working directory (or file or other dir) → 'SVN Commit'

If any change is undesired, it can be reverted back to the previous revision prior to the commit using

> `svn revert [-R]` ⟨*file or dir or nothing (current dir)*⟩
> or
> Right-click on working directory (or file or other dir) → 'TortoiseSVN' → 'Revert' then click the 'Ok' button

The optional `-R` option used with directories ensures the recursive descent through the subdirectories, reversing changes to all included files.

2.6 Install `svn-multi`

The installation of the `svn-multi` package can be done using the normal LaTeX package installer provided by your distribution (e.g. MikTeX), using the enclosed Makefile or manually as per the following procedure

1. Download the zipped package from CTAN
   http://theory.uwinnipeg.ca/scripts/CTAN/macros/latex/contrib/svn-multi.zip
2. Unpack ZIP file using your standard unzipper.
3. Run file `svn-multi.ins` through LaTeX, i.e.:
   `[pdf]latex svn-multi.ins`
4. The documentation can be produced with:
   ```
   [pdf]latex svn-multi.dtx
   [pdf]latex svn-multi.dtx
   makeindex -s gind.io svn-multi.ind svn-multi.idx
   makeindex -s gglo.-o svn-multi.gls svn-multi.glo
   [pdf]latex svn-mudtx
   ```
5. Copy the files `svn-multi.sty` and `svnkw.sty` to your TEXMF tree, e.g.:
   `cp svn-multi.sty svnkw.sty /usr/share/texmf/tex/latex/svn-multi/`
   or, for current user only
   `cp svn-multi.sty svnkw.sty ~/texmf/tex/latex/svn-multi/`
   or, for Windows
   `copy svn-multi.sty svnkw.sty C:\texmf\tex\latex\svn-multi\`
6. Optionally the documentation can be copied into the directory `/doc/latex/svn-multi/` in your TEXMF tree.

2.7 Subversion properties

Subversion utilises a concept called "properties" which allows you to attach arbitrary data to files using a key:value pair. The property name is the key and the value can be ASCII or binary data. Subversion itself uses properties to store file configurations. These properties always start with 'svn:', e.g. 'svn:keywords' or 'svn:mime-type'.

Properties can be set, edited, listed, returned and deleted using

`svn propset` ⟨*prop name*⟩ ⟨*prop value*⟩ ⟨*file(s) or dir(s)*⟩
`svn propedit` ⟨*prop name*⟩ ⟨*file(s) or dir(s)*⟩
`svn proplist` ⟨*file(s) or dir(s)*⟩
`svn propget` ⟨*prop name*⟩ ⟨*file(s) or dir(s)*⟩
`svn propdel` ⟨*prop name*⟩ ⟨*file(s) or dir(s)*⟩
or
Right-click on working directory (or file or other dir) → 'TortoiseSVN' → 'Properties' and use the graphical dialog

Table 1: Subversion Basic Subcommands with important options

Command	Alt.	Options	Arguments	Description
add		-N	⟨*files*⟩	Places ⟨*files*⟩ under VC
update	up	-N -r⟨*arg*⟩	[⟨*files*⟩]	Updates local copy
commit	ci	-N -r⟨*arg*⟩	[⟨*files*⟩]	Commits changes to repository
checkout	co	-N -r⟨*arg*⟩	⟨*URL*⟩ [⟨*dir*⟩]	Checks out ⟨*URL*⟩ to ⟨*dir*⟩ or to last dir element in ⟨*URL*⟩
log		-r⟨*arg*⟩	[⟨*path*⟩]	Show log messages of ⟨*path*⟩ or current dir
revert		-R	⟨*path*⟩	Reverts ⟨*path*⟩ to last committed state
propset	ps	-R -F⟨*file*⟩	⟨*name*⟩ ⟨*value*⟩ ⟨*path*⟩	Sets property ⟨*name*⟩ of ⟨*path*⟩ to ⟨*value*⟩
proplist	pl	-R -v	⟨*path*⟩	List properties of ⟨*path*⟩

Table 2: Subversion Command Options

Name	Option	Argument	Description
Non-recursive	-N		Operate on single directory only
Recursive	-R		Descend recursively
Revision	-r	⟨*revision*⟩	Specify revision (by number, date or name)
From File	-F	⟨*file*⟩	Read value from ⟨*file*⟩
Verbose	-v		Print extra information

For all commands and options see [2, Chapter 9] or type "svn help [⟨*subcommand*⟩]".

3 Version Control of a multi-file LaTeX Document

After Subversion has been installed with a repository and working directory set up the version control of your LaTeX document can begin.

3.1 Enable Subversion keyword substitution in your files

In order to have keyword substitution in your LaTeX files a special Subversion property `svn:keywords` must be assigned to them. The property value is a list of the wanted keywords. Normally it is good practise to enable all keywords even if only a subset are actually used. The property is set using

> `svn propset svn:keywords 'Id Author Date Rev URL' *.tex`
> or
> Right-click on LaTeX file(s) → 'TortoiseSVN' → 'Properties', press 'Add' and select 'svn:keywords' as name and 'Id Author Date Rev URL' as value, 'Ok'

3.2 Placing Subversion keywords in your files

All LaTeX files of the document must contain one or more Subversion keywords at the very start to allow the calculation of the last change. The shortest way is to use the compact `Id` keyword but the longer `Date`, `Rev`, `Author` and `Revision` keywords can also be used. The package `svn-multi` provides the macros `\svnid` for the `Id` keyword and `\svnidlong` for the other four keywords. Deciding which of the two macros is used depends on the document author. The first macro is shorter but only shows the file name not the whole URL. The second one is clearer arranged and easier to read in source code, while also providing the full file URL. It is also possible to use both together.

The recommendation is to put the following lines before any other code in every source file.

```
\svnidlong
{$HeadURL: $}
{$LastChangedDate: $}
{$LastChangedRevision: $}
{$LastChangedBy: $}
\svnid{$Id: $}
```

Please note that `\svnidlong` awaits four arguments which can be placed in a single line. No comments should be placed after this arguments because they are processed verbatim.

Subversion expands the keywords at the next commit, so the lines look like this afterwards:

```
\svnidlong
{$HeadURL: file://somewhere/file.tex $}
{$LastChangedDate: 2006-05-26 15:47:47 +0100 (Fri, 26 May 2006) $}
{$LastChangedRevision: 15 $}
{$LastChangedBy: maryd $}
\svnid{$Id: file.tex 15 2006-05-26 15:47:47Z maryd $}
```

3.3 Typesetting keyword values in your document

When each source file has all neccessary keyword macros, the information provided can be typeset anywhere in your document.

There are two groups of values: document-global and file-local ones, i.e. the VC information of the complete document and that of the current file. This allows to typeset the last changed revision number with date and author of the document, e.g. at the title page and also the same information of each single chapter. Please note that the global values are saved in an auxiliary file and have a one LaTeX run delay like the Table Of Contents.

The typeset macros are displayed in Table 3. The date is provided as a full string but also splitted in the single components. The macro `\svnkw` can be used to typeset arbitary keywords which is useful for non-Subversion keywords which are explained in Section 3.4.6.

3.4 Usage Examples

With the concepts outlined, some procatical examples are now given.

Table 3: `svn-multi` typeset macros

Keyword	Global	Local (current file)
Revision	`\svnrevision`	`\svnfilerevision`
Author	`\svnauthor`	`\svnfileauthor`
Date	`\svndate`	`\svnfiledate`
Year	`\svnyear`	`\svnfileyear`
Month	`\svnmonth`	`\svnfilemonth`
Day	`\svnday`	`\svnfileday`
Hour	`\svnhour`	`\svnfilehour`
Minute	`\svnminute`	`\svnfileminute`
Second	`\svnsecond`	`\svnfilesecond`
Timezone	`\svntimezone`	`\svnfiletimezone`
URL	`\svnmainurl`	`\svnkw{URL}`
Filename	`\svnmainfilename`	`\svnkw{Filename}`
Full Id		`\svnkw{Id}`
Any ⟨*keyword*⟩		`\svnkw{`⟨*keyword*⟩`}`

3.4.1 Putting 'Last Changed' values in header/footer

The header and/or footer line is an ideal location to place some of the 'Last Changed' values of the document. The current revision, last change author and date are normally the main information most people would need. The global/local scheme of `svn-multi` can be used to display the revision number of the document together with the revision number of the current chapter:

```
\usepackage{fancyhdr}

\pagestyle{fancy}

\fancyhead{}
\fancyhead[ol]{\slshape\leftmark}
\fancyfoot[ol]{Rev: \svnrev\ (\svnfilerev)}
\fancyfoot[or]{\svnyear-\svnmonth-\svnday\
\svnhour:\svnminute} %Date
```

```
% If the information should be also placed
% on the chapter page use:
\fancypagestyle{plain}{%
\fancyhead{}
\fancyhead[ol]{\slshape\leftmark}
\fancyfoot[ol]{Rev: \svnrev\ (\svnfilerev)}
\fancyfoot[or]{\svnyear-\svnmonth-\svnday\
\svnhour:\svnminute} %Date
}
```

This gives the following footer line:

Rev: 185 (154) 1 2006-11-10 09:54

where 185 is the last change document revision and 154 the last change chapter revision. The shortened date applies to the whole document.

3.4.2 Placing VC Information on title and chapter page

A good application of `svn-multi` is a longer document, e.g. a technical report, which is written by multiple authors responsible fro writing different chapters. These chapters are sometimes unrelated and therefore change independently. To track this changes, the VC information for the whole document can be typeset on the titlepage while the local values can be typeset for each chapter. Every chapter should lie in a own LaTeX-file which is `\included` in a main file.

The following code showns one way to achieve this. Remember to put a `\svnid` and/or `\svnidlong` on top of every chapter file and also in the main file somewhere after `\usepackage{svn-multi}`.

Titlepage

```
\begin{titlepage}
 \vspace*{\stretch{1}}
 {\Huge Title}
 \vspace*{\stretch{2}}

 \noindent
```

```
 Version control information:\\
 Last changed by: \svnFullAuthor{\svnauthor}\\
 Last changed date: \svndate\\
 Last changed revision: \svnrev\\
 Document URL: \svnmainurl\\
 %Document filename \svnmainfilename\\
\end{titlepage}
```

Chapter page

```
\svnidlong{...}{...}{...}{...}
\svnid{$Id: $}
\ chapter{Introduction}
 Version control information:\\
 Last changed by: \svnFullAuthor{\svnfileauthor}\\
 Last changed date: \svnfiledate\\
 Last changed revision: \svnfilerev\\
 Chapter URL: \svnkw{URL}\\
 %Chapter filename \svnkw{Filename}\\
```

3.4.3 Using full Author and Revision names

The `\svnauthor` and `\svnfileauthor` macros return the username of the author. However in most cases the full author name would be prefered, more legible, and therefore more useful. For this purpose `svn-multi` provides the following macros:

`\svnRegisterAuthor{`⟨*username*⟩`}{`⟨*full name*⟩`}`
`\svnFullAuthor`⟨*⟩`{`⟨*username*⟩`}`

The first macro should be used in the preamble to register all authors of the document. This is done by simply stating the full name of every username. The second macro can then be used to typeset the full name of a username which is normaly given by `\svnauthor` or `\svnfileauthor`. The star version also prints the username in parentheses.
For example:

```
\svnRegisterAuthor{martin}{Martin Scharrer}
\svnRegisterAuthor{other}{Some O. Author}
...
Last changed by: \svnFullAuthor*{\svnauthor}
```

gives

> Last changed by: Martin Scharrer (martin)

assumed that the author "martin" made the last change.

If the macro is used on a non-registered username the (empty) full name is suppressed and only the username is printed.

The same can be repeated for the revision number to assign a name. This has only limited use because the next revision number has to be guessed (it's normally the current revision plus one, but this is not always guaranteed). The macros for this are `\svnRegisterRevision` and `\svnFullRevision`.

3.4.4 Include the VC Values as PDF Metadata

The VC values can be used to generate PDF metadata like the author and the creation date. A trick can be used to avoid problems with the fragile `\svnFullAuthor` macro, the internally called macros `\svnFullAuthor@star` or `\svnFullAuthor@normal` are used directly.

The creation date can be set using `\pdfinfo` and `\svnpdfdate`, which returns the commit date in PDF format, e.g. `20060526154747+00'00'`.

```
% In preamble
\usepackage{svn-multi,hyperref}
\svnid{$Id: file.tex 15 2006-05-26 15:47:47Z maryd $}
\pdfinfo{ /CreationDate (D:\svnpdfdate) }
\makeatletter
\hypersetup{%
  pdfauthor={\svnFullAuthor@star*{\svnauthor}},
  %pdfauthor={\svnFullAuthor@normal{\svnauthor}},
  pdfkeywords={Revision \svnrev}
}
\makeatother
```

3.4.5 Keywords in moving arguments

All `svn-multi` keywords shown in Table 3 are robust and so can be used in moving arguments like chapter or section commands. But the `\svnFullAuthor` and `\svnFullRevision` macros are fragile and therefore must be preceded by `\protect` when used in such places.

3.4.6 Handling Non-Subversion Keywords

Non-Subversion keywords, e.g. user-defined[1] keywords or keywords from other version control software, can be placed and typeset with the `\svnkwsave` and `\svnkw` macros, respectively.

One example, taken from the Usenet group de.comp.text.tex, is:

```
\documentclass{article}

% Place, i.e. save, the keyword
\svnkwsave{$RepoUrl: file:///Server/repos/trunk $}

\ begin{document}
% Typeset keyword
Repository URL is ``\svnkw{RepoUrl}''.
\ end{document}
```

4 Future Features

While `svn-multi` is already "stable", i.e. non-beta, it is still under active development to enrich it with more useful features. User requests and suggestions are always appreciated.

One main feature in planning is the support of namespaces, i.e. to be able to group the keyword values of selected files together. This would expand the global/local scheme by providing values which are "global" only in the current namespace/group. The best example of this is a book or large report which is

1. User-defined keywords are not yet supported by the current Subversion version, i.e. 1.4.4, but might come in a future version.

separated in \parts. By creating a namespace for every part, the latest revision of each part will be available along with the file-local and document-global values. These can then be used to typeset a VC cover on the part page as shown for the titlepage in Section 3.4.2.

References

[1] Subversion project website http://subversion.tigris.org/.

[2] Collins-Sussmann, B. and Fitzpatrick, B.W. and Pilato, C.M., *Version Control with Subversion*, 2004, O'Reilly, ISBN: 0596004486. Available online under http://svnbook.red-bean.com/.

[3] Scharrer M., *svn-multi*, LaTeX package, current version 1.3a, 11 July 2007, CTAN: Package information: http://tug.ctan.org/pkg/svn-multi/, User manual: http://tug.ctan.org/tex-archive/macros/latex/contrib/svn-multi/svn-multi.pdf.

A Suggested Subversion Config File

The following listing shows the recommended Subversion user config file, which add the required properties to all newly added .tex files. The line endings are set to "native" so that the files always have the right format for the used operation system, even when multiple people working on the same document under different systems.

The location of this file is:

Linux/Unix: ~/.subversion/config

Windows: %APPDATA%\Subversion\config
where %APPDATA% is normaly
C:\Documents and Settings\⟨*username*⟩\Application Data

Subversion User Config File

```
[miscellany]
### Ignore LaTeX auxilary files (can also be set with svn:ignore to single
### direcories if not wanted global)
global-ignores = *.aux *.glo *.idx *.ind *.log *.out *.svn *.toc *.ilg *.gls

# Non-englisch people could require the next line
# log-encoding = latin1

# Automatically set the below properties to newly added files
enable-auto-props = yes

### Section for configuring automatic properties.
[auto-props]
*.tex = svn:mime-type=text/x-tex;svn:eol-style=native;svn:keywords=Id Date Author ↙
    Rev URL
*.ltx = svn:mime-type=text/x-tex;svn:eol-style=native
*.sty = svn:mime-type=text/x-tex;svn:eol-style=native
*.cls = svn:mime-type=text/x-tex;svn:eol-style=native
*.cfg = svn:mime-type=text/x-tex;svn:eol-style=native
Makefile = svn:mime-type=text/x-makefile;svn:eol-style=native
*.ps = svn:mime-type=application/postscript
*.eps = svn:mime-type=application/postscript
*.jpeg = svn:mime-type=image/jpeg
*.jpg = svn:mime-type=image/jpeg
*.png = svn:mime-type=image/png
*.dvi = svn:mime-type=application/x-dvi
*.pdf = svn:mime-type=application/pdf
```