The svnkw package*

Martin Scharrer
martin@scharrer-online.de

May 29, 2006

1 Introduction

This package lets you typeset keywords of the version control system Subversion[1] (svn), which is the successor of the popular CVS, inside your LaTeX files anywhere you like. Unlike the very similar package `svn`[2] the usage of multiple files for one LaTeX document is well supported. The package acquires the keywords of the last changed file and provides them to the user through macros. The package has to read all keywords of all files first and writes the most recent values in an axiliary file with an '.svn' extention. This file is read back at the next LaTeX run which introduces a delay like by the table of contents.

The special date features of the `svn` package are not (yet?) supported. If you need them you can try to use both packages together which should work but is not tested by the author.

2 Usage

2.1 Including of the Subversion keywords

To include your Subversion Id keywords use `\svnid` or `\svnidlong`. This macros should be written very early in each file, i.e. in the preamble of the main document soon after `\documentclass` and `\usepackage{svnkw}` and as first in every `\included` subfile before the `\chapter` macro. They do not create any output.

`\svnid` Macro for the svn Id keyword. Write the macro as `\svnid{$Id$}` into your files and set the subversion property `svn:keywords` of them to at least '`Id`'. Subversion will expand it. A trailing colon with spaces after the `Id` is also valid but **everything else** will cause a TeX parse error.

The keyword content will be interpreted and saved in the variables (ah, macros) `\svnfilerev`, `\svnfiledate` and `\svnfileauthor`. The most recent values (i.e. with the highest revision) are written as `\svnrev`, `\svndate` and `\svnauthor` in a .svn-file which is read back at the next LaTeX run. The idea is that the first variables holding the local values for the current file and the otheris the values for the complete document. Nevertheless all keywords can also be typeset with

*This document corresponds to svnkw v1.0, dated 2006/05/27.
[1]Subversion homepage: http://subversion.tigris.org/
[2]CTAN: http://www.ctan.org/tex-archive/macros/latex/contrib/svn/

`\svnkw{`⟨*keyword*⟩`}` which returns the latest saved value. This macro is used internally.

`\svnidlong` Macro for a "long Id". Saves similar values like in '`Id`' but from the keywords `HeadURL`, `LastChangedDate`, `LastChangedRevision` and `LastChangedBy`. The usage of `\svnid` or `\svnidlong` is a matter of taste. The second is more readable inside the code and results in a nicer date and a full URL, not only the filename. Both can be used together.

Write this as (order of arguments not meaningfull)

```
\svnidlong
{$HeadURL$}
{$LastChangedDate$}
{$LastChangedRevision$}
{$LastChangedBy$}
```

in your files and set the subversion property `svn:keywords` of them to '`HeadURL LastChangedDate LastChangedRevision LastChangedBy`'.

`\svn` This macro lets you set svn keywords directly. The only argument is the svn keyword in the usual dollars. The dollars will be stripped by the macro and the rest is typeset as normal text. This macro alone was the very first version of `svnkw` and is still included for fast and simple keyword typesetting.

`\svnkwsave` This macro lets you include and save any keyword you like. The only argument is the keyword wrapped in dollars. The keyword can be already expanded or not (no value and only "`:`" or nothing after the key name). This macro is also used internally and does not create any output.

2.2 Access to the keyword values

`\svnkw` Keyword values can be typeset by this macro. It takes one argument which must be a subversion keyword name. It then returns the current value of this keyword or nothing (`\relax`) when the keyword was not set yet.

Examples:

```
\textsl{Revision: \svnkw{Revision}}
URL: \url{\svnkw{HeadURL}}
```

In the second example `\url` (`hyperref` package) is used to add a hyperlink and to avoid problems with underscores (_) inside the URL.

`\svnkwdef` This macro is used to define the keyword values. This is normally only called internally but could be used by the user to override single keywords. The values can then be typeset by `\svnkw`. Note that `\svnid` and `\svnidlong` both sets some keywords to new values using this macro.

Note that for `\svnkw` and `\svnkwdef` all different names for one keyword are valid and result in the access of the same variable. So e.g. subversion treats `Rev`, `Revision` and `LastChangedRev` the same way and so does this macros. You can e.g. say `\svnkwdef{Rev}` and then typeset it with `\svnkw{Revision}` or `\svnkw{LastChangedRev}` if you like.

2.3 Using full author names

If you like to have the full author[3] names, not only the usernames, in your document you can use the following macros. First you have to register all au-

[3] This means subversion authors, e.g. the persons who commit changes into the svn reprository.

thors of the document with `\svnRegisterAuthor` and then you can write e.g. `\svnFullAuthor{\svnauthor}` or `\svnFullAuthor{\svnfileauthor}`.

`\svnRegisterAuthor` The usage is `\svnRegisterAuthor{`⟨*username*⟩`}{`⟨*full name*⟩`}` which registers ⟨*full name*⟩ as full name for ⟨*username*⟩.

`\svnFullAuthor` Takes the username as argument and returns the full name if it was registered first with `\svnRegisterAuthor` or nothing (`\relax`) else. There is also a star version `\svnFullAuthor*` which also returns the username in parentheses after the full name.

3 Implementation

`\svn` Just calls TeX macro `\@svn` to strip the dollars.

```
\newcommand{\svn}[1]{\@svn#1}
\def\@svn$#1$#1}
```

`\svnkwdef` First we check if there is a 'setter'-macro for the keyword called `\svnkwdef@`⟨*keyword*⟩.

```
\newcommand{\svnkwdef}[2]{%
\@ifundefined{svnkwdef@#1}
```

If not we call the general macro `\svnkwdef@`.

```
{\svnkwdef@{#1}{#2}}
```

If yes we just call it with the value as argument.

```
{\csname svnkwdef@#1\endcsname{#2}}
}
```

`\svnkwdef@` This macro defines the second argument under `\svnkw`⟨*2nd argument*⟩. The `\xdef` is used to expand the content first (needed for internal use) and make the definition globally.

```
\newcommand{\svnkwdef@}[2]
{\expandafter\xdef\csname svnkw#1\endcsname{#2}}
```

Example: `\svnkwdef{Revision}{23}` will define `\svnkwRevision` as 23.

`\svnkwdef@...` 'Setter'-macros for single keywords, used by `\svnkwdef`.
The keywords `Rev`, `Author` and `Date` are just calling `\svnkwdef@` with a fixed first argument.

```
\def\svnkwdef@Rev#1{\svnkwdef@{Rev}{#1}}
\def\svnkwdef@Author#1{\svnkwdef@{Author}{#1}}
\def\svnkwdef@Date#1{\svnkwdef@{Date}{#1}}
```

The long keywords are defined then as aliases of the short,
first for writting

```
\let\svnkwdef@Revision=\svnkwdef@Rev
\let\svnkwdef@LastChangedRevision=\svnkwdef@Rev
\let\svnkwdef@LastChangedBy=\svnkwdef@Author
\let\svnkwdef@LastChangedAt=\svnkwdef@Date
```

and then for reading.

```
\def\svnkwRevision{\svnkwRev}
\def\svnkwLastChangedRevision{\svnkwRev}
\def\svnkwLastChangedBy{\svnkwAuthor}
\def\svnkwLastChangedAt{\svnkwDate}
```

So e.g. `\svnkw{LastChangedRevision}` is always be the same as `\svnkw{Rev}`.

We define default values for normal keywords. Keyword `Filename` is the name given by `Id` and not a real keyword.

```
\svnkwdef{Rev}{0} % must always be numerical
\svnkwdef{Date}{}
\svnkwdef{Author}{}
\svnkwdef{Filename}{}
\svnkwdef{HeadURL}{}
```

`\svnkw` Macro to get keyword value. Just calls `\svnkwARGUMENT` where `ARGUMENT` is the argument interpreted as text. So e.g. `\svnkw{Date}` is the same as `\svnkwDate` but this could be changed later so always use this interface to get the keyword values.

```
\newcommand{\svnkw}[1]{\csname svnkw#1\endcsname}
```

`\svn@scanId` Scans svn Id (after it got parsed by `\svn@readkw`). Awaits only Id value without leading '`Id:`' and a trailing dollar as end marker. It calls `\@svn@updateid` to update global Id values and also sets the approbriate keywords.

```
\def\svn@scanId#1 #2 #3 #4 #5${%
% #1 is filename, #2 is revision, #3 is date (JJJJ-MM-DD),
% #4 is time (HH:MM:SST), #5 is author (username)
\@svn@updateid{#2}{#3 #4}{#5}%
\svnkwdef{Filename}{#1}%
\svnkwdef{Date}{#3 #4}%
\svnkwdef{Revision}{#2}%
\svnkwdef{Author}{#5}%
}
```

`\svn@readkw` General macro to read keywords. Saves key in `\svn@key` and the value in `\svn@value`. We have to analyse the argument but need it in full later. So we use the submacro `\svn@readkw@sub` to do the scan job. Sets `\svn@case` to one of three cases, where the first two cases result in identical code.

```
\def\svn@readkw$#1${%
\svn@readkw@sub$#1: $ % call submacro to get case
\ifnum\svn@case<3
```

If there is no value set `\svn@value` to empty and define `\svn@key` to the full argument.

```
\def\svn@key{#1}
\let\svn@value=\svn@empty
\svn@checkkeyforcolon$#1:$ % needed for case '$kw:$'
\relax
\else
```

If there is a value split the argument again by calling `\svn@readkw@def`.

```
\svn@readkw@def$#1$ % needed to remove leading space
\fi
}

\newcount\svn@case
\let\svn@empty=\empty
```

The submacro `\svn@readkw@sub` checks if the argument ends with ': '. In order to mach always we add a ': ' at the macro call and then test here if #2 is empty. This is the case if there was no ': ' before. We return 1 if there was no colon which means the keyword is not yet expanded by Subversion.

```
\def\svn@readkw@sub$#1: #2${%
\def\svn@temp{#2}
\ifx\svn@temp\empty
\svn@case=1
\else
```

If there was an tailing ': ' we check if #2 is now just ': ' which would be the ': ' we added at the macro call.

```
\svn@checkcolon$#2$
\ifx\svn@temp\empty
```

If we have nothing (empty value) in #2 return 2.

```
\svn@case=2
\else
```

If we have something in #2 return 3.

```
\svn@case=3
\fi\fi
}
```

`\svn@readkw@def` Submacro for `\svn@readkw` to split argument and to save the parts.

```
\def\svn@readkw@def$#1: #2 ${%
\def\svn@key{#1}
\def\svn@value{#2}
}
```

`\svn@checkcolon` Checks whether value is just ': ', if yes `\svn@temp` is empty.

```
\def\svn@checkcolon$#1: ${%
\def\svn@temp{#1}
}
```

`\svn@checkkeyforcolon` Checks whether key includes a trailing ':' and removes it. Must be called with an trailing ':' to always mach! So its check actually for tow trailing ':'.

```
\def\svn@checkkeyforcolon$#1:#2${%
\def\svn@temp{#2}
\ifx\svn@temp\empty
\else
\def\svn@key{#1}
\fi
}
```

Definition of init values.

```
% Init values
\def\svnrev{0}  % \
\def\svndate{}  %  > Values for whole project
\def\svnauthor{}  % /
\def\svnfilerev{0} % \
\def\svnfiledate{} %  > Values for current file
\def\svnfileauthor{} % /
\def\@svn@rev{0} % \
```

```
\def\@svn@date{} %  > Values for packet internal use
\def\@svn@author{} % /
```

`\svnid` We read the argument with `\svn@readkw` and provide the value to `\svn@scanId`.

```
\newcommand{\svnid}[1]{%
\svn@readkw#1 % Read keyword
\ifx\svn@value\empty % Check if value is empty
\else
\expandafter\svn@scanId\svn@value$ % Scan Id
\fi
}
```

`\svnidlong` We clear the keyword value first to reduce the risk though bad user input.

```
\newcommand{\svnidlong}[4]{
\svnkwdef{HeadURL}{}%
\svnkwdef{LastChangedDate}{}%
\svnkwdef{LastChangedRevision}{0}%
\svnkwdef{LastChangedBy}{}%
```

Then we save the four keywords/arguments using `\svnkwsave`.

```
\svnkwsave{#1}\svnkwsave{#2}
\svnkwsave{#3}\svnkwsave{#4}
```

And update the latest values.

```
\@svn@updateid{\svnkw{LastChangedRevision}}{\svnkw{LastChangedDate}}%
{\svnkw{LastChangedBy}}
}
```

`\svnkwsave` Save macro. Takes a dollar wrapped keyword string, reads it though `\svn@readkw` and saves it using `\svnkwdef`.

```
\newcommand{\svnkwsave}[1]{%
\def\svn@temp{#1}
\ifx\svn@temp\empty
%% skip at empty argument
\else
\svn@readkw#1% read keyword
\svnkwdef{\svn@key}{\svn@value}
\fi
}
```

`\@svn@updateid` We first define the expanded arguments to variables for the user. The expansion is needed because the arguments content is mostly generic like `\svn@key` and `\svn@value` which can change very soon after this macro.

```
\def\@svn@updateid#1#2#3{% #1 = rev, #2 date, #3 author (username)
\edef\svnfilerev{#1}
\edef\svnfiledate{#2}
\edef\svnfileauthor{#3}
```

Then we check if the revision is non-empty (not yet expanded by subversion?) and larger then the current maximum value `\@svn@rev`. If yes we save all value to save them in the .svn-file later.

```
\ifx\svnfilerev\empty\else % skip rest if rev is empty
\ifnum\@svn@rev<\svnfilerev%
\edef\@svn@rev{#1}
```

```
\edef\@svn@date{#2}
\edef\@svn@author{#3}
\else\fi
\fi
}
```

`\svnRegisterAuthor` Saves the author's name by defining `\svn@author@`⟨*username*⟩ to it.

```
\newcommand{\svnRegisterAuthor}[2]{%
\expandafter\def\csname svn@author@#1\endcsname{#2}
}
```

`\svnFullAuthor` We test if the starred or the normal version is used and call the appropriate submacro `\svnFullAuthor@star` or `\svnFullAuthor@normal`.

```
\newcommand{\svnFullAuthor}{%
\@ifnextchar{*}
{\svnFullAuthor@star}
{\svnFullAuthor@normal}
}
```

Both submacros are calling `\svnFullAuthor@` but with different arguments. The star macro also removes the star of course.

```
\def\svnFullAuthor@star*#1{\svnFullAuthor@{#1}{\ (#1)}}
\def\svnFullAuthor@normal#1{\svnFullAuthor@{#1}{}}
```

`\svnFullAuthor@` now sets the author's full name. Note that `#2` is empty when the normal version is called.

```
\def\svnFullAuthor@#1#2{%
\csname svn@author@#1\endcsname #2
}
```

At the end of document we write the values to an auxiliary file.

```
\AtEndDocument{
```

We first check if we have somthing to save. Revision, date and author must be non-empty. This suppresses the auxiliary file if the user doesn't use the appropriate macros but other provided by this package.

```
 \ifx\@svn@rev\empty\else
 \ifnum\@svn@rev=0\else
 \ifx\@svn@date\empty\else
 \ifx\@svn@author\empty\else
```

Open outfile to write project keywords.

```
 \newwrite\svnwrite
 \immediate\openout\svnwrite=\jobname.svn
 \immediate\write\svnwrite{\@percentchar\space SVN cache}
 \immediate\write\svnwrite{\noexpand\def\noexpand\svnrev{\@svn@rev}}
 \immediate\write\svnwrite{\noexpand\def\noexpand\svndate{\@svn@date}}
 \immediate\write\svnwrite{\noexpand\def\noexpand\svnauthor{\@svn@author}}
 \immediate\closeout\svnwrite
 \fi\fi\fi\fi
}

```

Reread output from last compile run if it exists.

```
\InputIfFileExists{\jobname.svn}{}{}
```