The svn package[*]

Richard Lewis
`rpil2+svn.sty@rtf.org.uk`

25th September 2007

1 Introduction

Subversion is a replacement for CVS and RCS. It is similar to CVS but with some improvements (e.g., it understands renaming and deletion of version controlled files—see `http://subversion.tigris.org/` for more information). As with CVS and RCS, a file registered with Subversion may contain keywords (such as `$Date$` or `$Revision$`) that Subversion will replace with status information about the file (such as the date the file was last committed, or the revision at which it last changed).[1]

For typesetting the contents of RCS and CVS keywords there is the rcs package[2]; although highly recommended, that package does not cope with the format of Subversion's `$Date$` keyword, so I wrote the svn package to do just that.

2 Usage

2.1 Quick Example

The main use for this package is to get the date the file was last committed into the output of `\maketitle`. The solution is simple:

```
\documentclass{article}
\usepackage{svn}
\SVNdate $Date$
\title{Hope this works}

\begin{document}
\maketitle
\end{document}
```

---

[*] This document corresponds to svn r43, dated 2007/09/25.

[1] Unlike RCS and CVS, the expansion of such keywords is customisable, and not enabled by default: use '`svn propset svn:keywords "Date Id" myfile.tex`' to tell Subversion to expand the keywords `$Date$` and `$Id$` in '`myfile.tex`'.

[2] Written by Joachim Schrod with minor modification by Jeffrey Goldberg

2.2 More General Usage

As usual, load the svn package with `\usepackage{svn}`.

The main command is `\SVN $`⟨*Keyword*⟩`$` (which mimics '`\RCS $`⟨*Keyword*⟩`$`' from the rcs package). By default the following happens:

- If you say `\SVN $Keyword: stuff $` (i.e, `$Keyword$` has been expanded to '`stuff`') then:
  - If `$Keyword$` is `$Date$` or `$\LastChangedDate$`, then `stuff` is parsed and `\SVNDate` is defined to be the date, and `\SVNTime` the time, that the file was checked in. `\SVNRawDate` is defined to be the whole string '`stuff`'.
  - Otherwise a command `\SVNKeyword` is defined to be '`stuff`'.
- If you say `\SVN $Keyword$` (i.e., `$Keyword$` was not expanded—perhaps it doesn't appear in the `svn:keywords` property, or perhaps the file has not been checked in since the line was added), then:
  - If `$Keyword$` is `$Date$` (or `$\LastChangedDate$`), `\SVNDate` is defined to be `\today`, and `\SVNTime` and `\SVNRawDate` are set to `\SVNempty` (which is empty by default, and may be changed with `\renewcommand`).
  - Otherwise `\SVNKeyword` is defined to be `\SVNempty`.

In principle you may use `\SVN` anywhere, but you may find problems if some package has made characters appearing in keywords active (e.g., babel with the `french` option—`\SVN` still works in the preamble though).

2.3 `\SVNdate`

Since you probably want to have the date of check-in the output of `\maketitle`, we provide the construct '`\SVNdate $Date$`' to do just that (note the difference between this and `\SVNDate`: the latter expands to the check-in time (or `\today`)). This is exactly the same as saying '`\SVN $Date$ \date{\SVNDate}`', but saves some typing.

2.4 Advanced Usage and Customisation

The default behaviour described above can be modified to do all kinds of fancy things with all kinds of fancy keywords. When you say `\SVN $keYwoRd: stuff$`, if the command `\SVN@keYwoRd@expanded` exists[3] then it will be executed with two arguments: '`\SVN@keYwoRd@expanded{keYwoRd}{stuff : }`' (note the trailing '`␣:␣`'). If `\SVN@keYwoRd@expanded` does not exist then `\SVN@generic@expanded` is run (again with arguments '`{keYwoRd}{stuff : }`), which defines `\SVNkeYwoRD` to be `stuff`.

---

[3] As ever, 'exists' means "defined and not equal to '`\relax`"'

If instead we had an unexpanded keyword (e.g., ‘`\SVN $keYwoRd$`’) then svn will try and run `\SVNkeYwoRd@unexpanded{keYwoRd}{}`, falling back to `\SVN@generic@unexpanded{keYwoRd}{}` if `\SVN@keYwoRd@unexpanded` does not exist. `\SVN@generic@unexpanded{keYwoRd}{}` will define `\SVNkeYwoRd` to be `\SVNempty`, which is initially just `\relax`, but may be redefined (just use `\renewcommand`).

So if you want some fancy behaviour for some fancy new keyword, you just need to define `\SVN@`⟨*Keyword*⟩`@expanded` and `\SVN@`⟨*Keyword*⟩`@unexpanded` to do what you want. Both variants should take two arguments which are `{`⟨*KeywordName*⟩`}{`⟨*expansion*⟩`}`. `\SVN@`⟨*Keyword*⟩`@unexpanded` will be called with ⟨*expansion*⟩ empty, and `\SVN@`⟨*Keyword*⟩`@expanded` will be called with ⟨*expansion*⟩ as the keyword expansion text plus a trailing ‘`␣:␣`’ (which can be removed using the predefined `\svn@set` command—see the following example).

As a simple example, `\SVN $Rev$` will define a `\SVNRev` command. Subversion treats `$LastChangedRevision$` as an alias for `$Rev$`, so if you wanted both `\SVN $Rev$` and `\SVN $LastChangedRevision$` to define both `\SVNLastChangedRevision` and `\SVNRev` then you could put the following in your preamble:

```
\makeatletter
 %%These first two are run when \SVN sees a ‘Rev’ keyword.
 \def\SVN@Rev@unexpanded#1#2{%
   \let\SVNRev\SVNempty
   \let\SVNLastChangedRevision\SVNRev
 }
 %%The ‘@expanded’ receives the keyword name as #1 and the
 %%keyword expansion (with trailing ‘ : ’) as #2.
 \def\SVN@Rev@expanded#1#2{%
   \svn@set\SVNRev$#2$%
   \let\SVNLastChangedRevision\SVNRev
 }
 %%These next two lines make \SVN treat ‘LastChangedRevision’
 %%exactly the same as ‘Rev’
 \let\SVN@LastChangedRevision@unexpanded\SVN@Rev@unexpanded
 \let\SVN@LastChangedRevision@expanded\SVN@Rev@expanded
\makeatother
```

2.5 Known Issues

If you use babel you will get the date produced by the `\SVNDate` command in the correct style for the current language, and if you change the language the text produced by `\SVNDate` may change. This may be undesirable, and the naïve solution is to say `\edef\SVNDateText{\SVNDate}` before the language change. However, with the code stolen from the rcs, inside an `\edef`, `\SVNDate` expands to `\today` whatever the check-in date. To work around this, `\SVNDate` has been designed to generate an error inside an `\edef`.

One way to store the check-in date in a language-independent way is the following, which defines `\fixatedSVNDate` to be the german version of the check-in date, but note that `\edef\foo{\fixatedSVNDate}\foo` will still give `\today`’s date (and no error).

```
\def\fixateSVNDate{%
  \def\foo{\today}
  \ifx\SVNDate\foo
    \let\fixatedSVNDate\today
  \else
    \expandafter\fixateSVNDateExpanded\SVNDate
  \fi
}

\def\fixateSVNDateExpanded\begingroup#1\day#2\today\endgroup{%
  \let\fixedtoday\today
  \def\fixatedSVNDate{\begingroup\day#2\fixedtoday\endgroup}%
}

%% To fix the Date format, use \fixateSVNDate:
\SVN $Date: 3999-07-30 14:58:54 +0100 (Thu, 30 Jul 3999) $
german: \selectlanguage{german}\fixateSVNDate\SVNDate\\
english : \selectlanguage{english} \SVNDate\\
We still have access to german format: \fixatedSVNDate
```

2.6 Avoiding Unwanted Keyword Expansion

Although nothing to do with this package, the following may be useful.

Sometimes your document contains strings of the form ‘`$...$`’ which, although looking like keywords, should not be expanded by Subversion. There are several ways to stop this expansion.

Firstly, Subversion only expands the keywords you tell it to, so if you say ‘`svn propset svn:keywords "Id" myfile.tex`’ (and then commit), `$Date$` will not be expanded anywhere. This leaves the case where you want to use something like `\SVNdate $Date$` at the top, but also use `$Date$` somewhere else.

In-line maths: If you are using `$Date$` because it is the product of the variables D, a, t and e, then you could use `\(Date\)` or replace the dollars with `^^24`: ‘`^^24Date^^24`’.

Verbatim: If you want the string `$Date$` to appear verbatim in your `dvi`, then you could use `\texttt{\string$Date\string$}` (or use `\verb` around the `$`, but that will break in footnotes)

3 Implementation

3.1 General Admin Stuff

`\svn@date`
`\svn@revision`

First we do the usual `\ProvidesPackage` stuff. Of course, `svn.dtx` is itself under Subversion, and we want to get the package date and version from the `$Id$`

keyword.

```
\NeedsTeXFormat{LaTeX2e}
\def\next $Id: #1 #2 #3-#4-#5 #6${%
  \def\svn@date{#3/#4/#5}%
  \def\svn@revision{#2}%
}
\next $Id: svn.dtx 43 2007-09-25 19:20:04Z repos $
\edef\next{%
  \noexpand\ProvidesPackage{svn}[\svn@date\space r\svn@revision\space
                                 Typeset Subversion keywords.]%
}
\next
```

3.2 The generic `\SVN` command

`\SVN`

`\SVN` is the main construct (see above for usage). The single argument should be of the form `$`⟨*Keyword*⟩`$` or `$`⟨*Keyword*⟩`:`⟨*space*⟩⟨*value*⟩⟨*space*⟩`$`, where ⟨*Keyword*⟩ and ⟨*value*⟩ must be non-empty as well as brace- and `\if`–`\fi`- balanced. ⟨*space*⟩ is a single space (if more are present they will be subsumed into ⟨*value*⟩). If `$empty$`, `$generic$`, `$RawDate$`, or `$Time$` ever become keywords, or if keywords containing `@` ever exist then we may have problems.

```
\def\SVN $#1${\svn@$#1: $}
```

`\SVNempty`

If ⟨*Keyword*⟩ is unexpanded then `\SVNKeyword` is `\let` to `\SVNempty`, which is initially empty.

```
\newcommand{\SVNempty}{}
```

`\svn@`
`\svn@tmp`

`\snv@` does the work for `\SVN`. It takes two arguments, the first is the ⟨*Keyword*⟩'s name, the second is empty (in which case ⟨*Keyword*⟩ was unexpanded) or ⟨*value*⟩, the expansion of ⟨*keyword*⟩.

```
\def\svn@$#1: #2${%
  \def\svn@tmp{#2}%
```

`\svn@suffix`

If `#2` is empty, then the keyword was unexpanded and `\svn@suffix` is set to `@unexpanded`, otherwise we had an expanded keyword so `\svn@suffix` is set to `@expanded`.

```
  \ifx\svn@tmp\@empty
    \def\svn@suffix{@unexpanded}%
  \else
    \def\svn@suffix{@expanded}%
  \fi
```

If `\SVN@#1`⟨*suffix*⟩ is defined then run it with arguments '`#1#2`', else run `\SVN@generic@`⟨*suffix*⟩ (again with argument `#1#2`—by default this defines '`\SVN`⟨*#1*⟩' to be `#2`, or `\SVNempty` in the unexpanded case).

```
  \@ifundefined{SVN@#1\svn@suffix}%
      {\@nameuse{SVN@generic\svn@suffix}{#1}{#2}}%
    {\@nameuse{SVN@#1\svn@suffix}{#1}{#2}}%
}
```

3.3 Dealing with general `$Keyword$`s

`\SVN@generic@expanded`

When we see `\SVN $KeyWord: <stuff> $`, and no `\SVN@KeyWord@expanded` command exists, we use `\SVN@generic@expanded{KeyWord}{<stuff>}` to define `\SVNKeyWord` to be `<stuff>`.

```
\def\SVN@generic@expanded#1#2{%
  \expandafter\svn@set\csname SVN#1\endcsname$#2$%
}
```

`\SVN@generic@unexpanded`

When we see `\SVN $KeyWord$` and no `\SVN@KeyWord@unexpanded` command exists, we use `\SVN@generic@unexpanded{KeyWord}` to define `\SVNKeyWord` to be `\SVNempty`.

```
\def\SVN@generic@unexpanded#1#2{%
  \expandafter\global\expandafter\let\csname SVN#1\endcsname\SVNempty
}
```

`\svn@set`

`\svn@set#1$#2$` defines the command in `#1` to be `#2` without the trailing '`␣:␣`' that the call to `\svn@` added.

```
\def\svn@set#1$#2 : ${\gdef#1{#2}}
```

3.4 Dealing with the `$Date$` keyword

`\SVN@Date@unexpanded`
`\SVN@LastChangedDate@unexpanded`

When we see a `\SVN $Date$` (or `\SVN $LastChangedDate$`), we define `\SVNDate` and `\SVNTime` to be the current date and time. The argument `#1` will be the name of the keyword actually used (i.e., `Date` or `LastChangedDate`), and `#2` will be empty since `#1` was not expanded. Note that we don't say `\let\SVNDate\today` as we want babel to be able to influence the formatting of `\SVNDate`.

```
\def\SVN@Date@unexpanded#1#2{%
  \gdef\SVNDate{\today}%
  \global\let\SVNTime\SVNempty
  \global\let\SVNRawDate\SVNempty
}
\let\SVN@LastChangedDate@unexpanded\SVN@Date@unexpanded
```

`\SVN@Date@expanded`
`\SVN@LastChangedDate@expanded`

When we see `\SVN $Date: <date> <time> ... $`, we set `\SVNRawDate` to the whole '`<date> <time> ...`' string, and put the date and time of check-in into `\SVNDate` and `\SVNTime`.

```
\def\SVN@Date@expanded#1#2{%
  \svn@set\SVNRawDate$#2$%
  \svn@parse@date$#2$%
}
\let\SVN@LastChangedDate@expanded\SVN@Date@expanded
```

`\svn@parse@date`
`\SVNDate`
`\SVNTime`

`\svn@parse@date` is what actually puts the date of check-in (or `\today`) into `\SVNDate`. The idea for this is copied from the rcs package.

We use the `$`'s to remove the leading space and then, inside a group, we change the current date and then call `\today`—this way if babel is used, we'll get `\SVNdate` in the correct language format. Since the `\day` commands are not expandable but

`\today` is, we add a `\def` to give an error inside an `\edef` (see also the "Known Issues" section).

```
43 \def\svn@parse@date$#1-#2-#3 #4:#5:#6 #7${%
44   \gdef\SVNDate{%
45     \begingroup
46       \def\svn@tmp{\PackageError{svn}{\SVNDate should not
47       be used in an \protect\edef}{See the svn.sty documentation for a
48       work-around.}}%
49       \day#3 \month#2 \year#1
50       \today
51     \endgroup}%
```

We could add 'GMT' to `\SVNTime`. Or not bother.

```
52   \gdef\SVNTime{#4:#5:#6}%
53 }
```

`\SVNdate` `\SVNdate $Date$` puts the check-in date into the output of `\maketitle`.

```
54 \def\SVNdate $#1${\SVN $#1$\date{\SVNDate}}
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.