

# The `svn` package\*

Richard Lewis  
rpil+svn@rtf.org.uk

21st September 2003

## 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).<sup>1</sup>

For typesetting the contents of RCS and CVS keywords there is the `rct` package<sup>2</sup>; 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 $Date$
\begin{verbatim}
\title{Hope this works}

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

---

\*This document corresponds to `svn r32`, dated 2003/09/21.

<sup>1</sup>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'`.

<sup>2</sup>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  $\langle Keyword \rangle$`  (which mimics ‘`\RCS  $\langle Keyword \rangle$` ’ from the `rsc` package). By default the following happens:

- If you say `\SVN  $\langle Keyword \rangle$ : stuff  $\langle \rangle$`  (i.e.,  $\langle Keyword \rangle$  has been expanded to ‘`stuff`’) then:
  - If  $\langle Keyword \rangle$  is  $\langle Date \rangle$  or `\LastChangedDate`, then `\SVNDate` is defined to be the date the file was checked in, `\SVNTime` is the time of check-in, and `\SVNRawDate` is defined to be ‘`stuff`’.
  - Otherwise a command `\SVNKeyword` is defined to be ‘`stuff`’.
- If you say `\SVN  $\langle Keyword \rangle$`  (i.e.,  $\langle Keyword \rangle$  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  $\langle Keyword \rangle$  is  $\langle Date \rangle$  (or `\LastChangedDate`), then `\SVNDate` is defined to be `\today`, `\SVNTime` and `\SVNRawDate` are set to `\SVNEmpty`, (which is empty by default and can be changed with `\renewcommand`).
  - Otherwise `\SVNKeyword` is defined to be `\SVNEmpty`.

In principal you may use `\SVN` anywhere, but you may find problems if some package (e.g., `babel` with the `french` option) has made characters appearing in keywords active. `\SVN` should always work in the preamble.

## 2.3 `\SVNdate`

Since you probably want to have the date of check-in the output of `\maketitle`, we provide the construct ‘`\SVNdate  $\langle Date \rangle$` ’ 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  $\langle Date \rangle$  \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  $\langle keyWord \rangle$ : stuff`, if the command `\SVN@keyWord@expanded` exists<sup>3</sup> 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`.

If instead we had an unexpanded keyword (e.g., ‘`\SVN  $\langle keyWord \rangle$` ’) 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`).

---

<sup>3</sup>As ever, ‘exists’ means “defined and not equal to ‘`\relax`’”

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 `\SVNRevision` 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 `rct`, 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\beginngroup#1\day#2\today\endgroup{%
  \let\fixedtoday\today
  \def\fixatedSVNDate{\beginngroup\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

```

The second known issue is that the `ps` and `pdf` formatted documentation is different from the `dvi` format (look at page 6). This happens even when converting `dvi` to `ps` via ‘`dvips`’, so may be a bug in ‘`ggv`’?

## 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 `\newcommand{\somemaths}[1]{\$#1$}` and then `\somemaths{Date}`. Alternatively, 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 First we do the usual \ProvidesPackage stuff. Of course, svn.dtx is itself under
\svn@revision Subversion, and we want to get the package date and version from the $Id$
keyword.

```

```

1 \NeedsTeXFormat{LaTeX2e}
2 \def\next $Id: #1 #2 #3-#4-#5 #6${%
3   \def\svn@date{#3/#4/#5}%
4   \def\svn@revision{#2}%
5 }
6 \next $Id: svn.dtx 32 2003-09-21 19:08:50Z rpil2 $
7 \edef\next{%

```

```

8 \noexpand\ProvidesPackage{svn}[\svn@date\space r\svn@revision\space
9                               Typeset Subversion keywords.]%
10 }
11 \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  $\$(Keyword):\langle space\rangle\langle value\rangle\langle space\rangle\$,$  where  $\langle Keyword\rangle$  and  $\langle value\rangle$  must be non-empty as well as brace- and `\if-\fi`-balanced.  $\langle space\rangle$  is a single space (if more are present they will be subsumed into  $\langle value\rangle$ ). If ‘`\empty`’, ‘`\generic`’, ‘`\RawDate`’, ‘`\Time`’ or ever become keywords, or if keywords containing `@` ever exist then we may have problems.

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

`\SVNempty` If  $\langle Keyword\rangle$  is unexpanded then `\SVNkeyword` is `\let` to `\SVNempty`, which is initially empty.

```
13 \let\SVNempty\relax
```

`\svn@` `\snv@` does the work for `\SVN`. It takes two arguments, the first is the  $\langle Keyword\rangle$ ’s name, the second is empty (in which case  $\langle Keyword\rangle$  was unexpanded) or  $\langle value\rangle$ , the expansion of  $\langle keyword\rangle$ .

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

```

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

```
16   \ifx\svn@tmp@empty
17     \def\svn@suffi{@unexpanded}%
18   \else
19     \def\svn@suffi{@expanded}%
20   \fi

```

If `\SVN@#1\langle suffix\rangle` 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).

```
21   \@ifundefined{SVN@#1\svn@suffi}{%
22     {\@nameuse{SVN@generic\svn@suffi}{#1}{#2}}%
23     {\@nameuse{SVN@#1\svn@suffi}{#1}{#2}}%
24 }

```

### 3.3 Dealing with general \$Keyword\$

`\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>`.

```
25 \def\SVN@generic@expanded#1#2{%
26   \expandafter\svn@set\csname SVN#1\endcsname$#2$%
27 }

```

`\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`.

```
28 \def\SVN@generic@unexpanded#1#2{%
29   \expandafter\global\expandafter\let\csname SVN#1\endcsname\SVNempty
30 }
```

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

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

### 3.4 Dealing with the `$Date$` keyword

`\SVN@Date@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`.

`\SVN@LastChangedDate@unexpanded`

```
32 \def\SVN@Date@unexpanded#1#2{%
33   \gdef\SVNDate{\today}%
34   \global\let\SVNTime\SVNempty
35   \global\let\SVNRawDate\SVNempty
36 }
37 \let\SVN@LastChangedDate@unexpanded\SVN@Date@unexpanded
```

`\SVN@Date@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`.

`\SVN@LastChangedDate@expanded`

```
38 \def\SVN@Date@expanded#1#2{%
39   \svn@set\SVNRawDate$#2$%
40   \svn@parse@date$#2$%
41 }
42 \let\SVN@LastChangedDate@expanded\SVN@Date@expanded
```

`\svn@parse@date` `\svn@parse@date` is what actually puts the date of check-in (or `\today`) into `\SVNDate` and `\SVNTime`. 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}}
```