The svn-prov package

Use SVN Id keywords for package, class and file header

Martin Scharrer
martin@scharrer-online.de

Version v3.1858 - April 14, 2010

1 Introduction

This package is directed to authors of LaTeX packages and classes which use the version control software Subversion[1] (SVN) for their source files. It introduces three macros which are Subversion variants of the standard LaTeX header macros `\ProvidesPackage`, `\ProvidesClass` and `\ProvidesFile` which are used to identify package, class and other files, respectively. Instead of providing the package/class/file name and date manually they are extracted from a Subversion Id keywords string which is updated automatically by every time the source file is committed to the repository.

A similar package exists for RCS, the pre-predecessor of Subversion, in the `pgf`[2] bundle which is called `pgfrcs`. For further support for Subversion keywords see the author's other package `svn-multi`[3].

2 Usage

The following macros need an Id keyword which can initially be written as '`$Id:$`' and will be expanded by Subversion into the following format at the next commit:

`$Id:` ⟨*filename*⟩ ⟨*revision*⟩ ⟨*date*⟩ ⟨*time*⟩ ⟨*author*⟩ `$`

e.g. for the source file of this document:

```
$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $
```

For this to work the Subversion *property* `svn:keywords` must be set to (at least) '`Id`' for the source file(s). e.g. using the command line:

`svn propset 'svn:keyword' 'Id'` ⟨*filename(s)*⟩

More information about using Subversion in the LaTeX workflow can be found in the PracTeX Journal issue 2007-3[4].

[1] WWW: `http://subversion.tigris.org/`
[2] CTAN: `http://tug.ctan.org/pkg/pgf`
[3] CTAN: `http://tug.ctan.org/pkg/svn-multi`
[4] URL: `http://www.tug.org/pracjourn/2007-3/{skiadas-svn|ziegenhagen|scharrer}`

```
\ProvidesPackageSVN[⟨file name⟩]{$Id: $}[⟨version and/or description⟩][⟨description⟩]
\ProvidesClassSVN[⟨file name⟩]{$Id: $}[⟨version and/or description⟩][⟨description⟩]
\ProvidesFileSVN[⟨file name⟩]{$Id: $}[⟨version and/or description⟩][⟨description⟩]
```

All of these macros await a valid Subversion Id keyword string as a mandatory argument. The file name and date is extracted from this string. For cases when the file source is not stored in the correct file but packed inside a different one, like a `.dtx` file, the correct file name can be provided by an optional argument. Because the file extension of package and class files is predefined and therefore ignored this is not needed for them when they are packed inside a corresponding `.dtx` file, i.e. one with the same base name.

As with the standard macros mentioned above additional information can be given optionally. Since v0.3 the SVN macro provide two optional arguments (before only one). If only one optional argument is given it is taken as a description text which may start with an potential version number. This version number must start with '`v`' and not include spaces and is extracted from the description. Alternatively the version number and the description can be provided using two separate optional arguments. If no optional argument is given the default string `\revinfo` (see below) is used instead.

All three optional arguments can include the following macros which are only valid inside them, but not afterwards[5]:

`\rev` File revision.

`\Rev` File revision followed by a space.

`\revinfo` The default information used: "`(SVN Rev: ⟨revision⟩)`".

`\filebase` File base name (file name without extension).

`\fileext` File extension (without leading dot).

`\filename` File name.

`\filedate` File date (in the format YYYY/MM/DD).

`\filerev` File revision, like `\rev`.

```
\GetFileInfoSVN{⟨name⟩}
\GetFileInfoSVN*
```

This macro sets the macros `\filebase`, `\fileext`, `\filename`, `\filedate`, `\fileversion`, `\filerev`, `\fileinfo` and `\filetoday` to the corresponding values of the file given by ⟨*name*⟩. The file must have been read/loaded before and use both a `\Provides...SVN` macro and `\DefineFileInfoSVN`, otherwise

Non-star version added in v3. 2010/04/11

[5] They can be set using `\GetFileInfoSVN`

the above macros will be set to \relax. The ⟨*name*⟩ can be either the real filename or the optional short name used with \DefineFileInfoSVN.

The star version of this macro provides the file information of the last file which called one of the \Provides...SVN macros.

The macros \fileversion and \fileinfo hold the file version and description taken from optional argument of the \Provide...SVN macro. The version is defined only if this argument starts with 'v' and is otherwise empty. It includes all text up to the first space. The \filetoday macro generates a text representation of the \filedate using the \today macro. The format can be adjusted to a different language with the \date⟨*language*⟩ macro from the babel[6] package. The other macros are described above.

\DefineFileInfoSVN[⟨*name*⟩]

Defined a set of macros which provide the information collected by a previous \Provides... macro. The macros have the form \⟨*name*⟩@⟨*data*⟩ where ⟨*name*⟩ is by default the filename either with the file extension (general files) or without (packages and classes). This default can be overwritten by the optional argument. The ⟨*data*⟩ stands for version, rev (revision), date and info (the information part without the version number) and, since v3, file name base and ext(ension). *New in v1. 2009/05/03* *Updated in v3. 2010/04/11*

Example: Applied to the .dtx file of this very package the following macros are defined:

Macro	Definition
\svn-prov.dtx@version	v3.1858
\svn-prov.dtx@rev	1858
\svn-prov.dtx@date	2010/04/14
\svn-prov.dtx@info	DTX for svn-prov.sty
\svn-prov.dtx@base	svn-prov
\svn-prov.dtx@ext	dtx

The style file however would get macros like \svn-prov@version. Because '-' is not a letter the macros can only be accessed using \csname. Therefore the optional argument [svnprov] is used to name the macros \svnprov@version etc..

3 Examples

The following examples illustrate the usage of the provided macros and how they call the equivalent standard macros internally. The example *results* are produced by expanding the corresponding example *code* while the standard provide macros are locally redefined to typeset their own name and arguments

[6] CTAN: http://tug.ctan.org/pkg/babel

in verbatim style. This does not only simplifies the generation of this document but makes this examples also test cases which allow the package author to test the result of the defined macros.

While mostly the package macro is used here the usage is identical to the class and file macros. Of course before this macros are used it must be made sure that the `svn-prov` package is loaded which is done by using the following code direct before them:

```
\RequirePackage{svn-prov}
```

Minimal usage

The following code:

```
\ProvidesPackageSVN
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
```

is equivalent to:

```
\ProvidesPackage{svn-prov}[2010/04/14 (SVN Rev: 1858)]
```

The following code:

```
\ProvidesClassSVN
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
```

is equivalent to:

```
\ProvidesClass{svn-prov}[2010/04/14 (SVN Rev: 1858)]
```

The following code:

```
\ProvidesFileSVN
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
```

is equivalent to:

```
\ProvidesFile{svn-prov.dtx}[2010/04/14 (SVN Rev: 1858)]
```

Normal Usage

The following code:

```
\ProvidesPackageSVN
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
  [v1.0 Example Description]
```

is equivalent to:

```
\ProvidesPackage{svn-prov}[2010/04/14 v1.0 Example Description]
```

The following code:

```
\ProvidesClassSVN
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
  [v1.0 Example Description]
```

is equivalent to:

```
\ProvidesClass{svn-prov}[2010/04/14 v1.0 Example Description]
```

The following code:

```
\ProvidesFileSVN
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
  [v1.0 Example Description]
```

is equivalent to:

```
\ProvidesFile{svn-prov.dtx}[2010/04/14 v1.0 Example Description]
```

Normal Usage with only Description

The following code:

```
\ProvidesFileSVN
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
  [Example Description]
```

is equivalent to:

```
\ProvidesFile{svn-prov.dtx}[2010/04/14 Example Description]
```

Normal Usage with separate Version and Description

The following code:

```
\ProvidesFileSVN
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
  [v1.0][Example Description]
```

is equivalent to:

```
\ProvidesFile{svn-prov.dtx}[2010/04/14 v1.0 Example Description]
```

Overwriting Name

The following code:

```
\ProvidesPackageSVN[othername]
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
  [v1.0 Example Description]
```

is equivalent to:

```
\ProvidesPackage{othername}[2010/04/14 v1.0 Example Description]
```

Overwriting Name including unneeded Extension

The following code:

```
\ProvidesPackageSVN[othername.sty]
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
  [v1.0 Example Description]
```

is equivalent to:

```
\ProvidesPackage{othername}[2010/04/14 v1.0 Example Description]
```

Overwriting Name using Macros

The following code:

```
\ProvidesFileSVN[\filebase.cfg]
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
  [v1.0 Example Description]
```

is equivalent to:

```
\ProvidesFile{svn-prov.cfg}[2010/04/14 v1.0 Example Description]
```

Using Macros in File Information String

The following code:

```
\ProvidesPackageSVN
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
  [v1.\Rev Example Description]
```

is equivalent to:

```
\ProvidesPackage{svn-prov}[2010/04/14 v1.1858 Example Description]
```

Adding Text to Default Information

The following code:

```
\ProvidesPackageSVN
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
  [v1.\Rev Extra Text \revinfo]
```

is equivalent to:

```
\ProvidesPackage{svn-prov}[2010/04/14 v1.1858 Extra Text (SVN Rev:
1858)]
```

Getting the File Information

The following code:

```
\ProvidesPackageSVN
  {$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}
  [v1.\Rev Extra Text \revinfo]
\GetFileInfoSVN*
% ...
\begin{tabular}{l@{\ :\ \ }l}
  File Name      & \filename    \\
  File Base Name & \filebase    \\
  File Extension & \fileext     \\
  File Date      & \filedate    \\
  File Revision  & \filerev     \\
  File Version   & \fileversion \\
  File Info      & \fileinfo    \\
\end{tabular}
```

results in:

File Name	:	svn-prov.dtx
File Base Name	:	svn-prov
File Extension	:	dtx
File Date	:	2010/04/14
File Revision	:	1858
File Version	:	v1.1858
File Info	:	Extra Text (SVN Rev: 1858)

The correct package file extension '`.sty`' for `\fileext` can be forced by using `[\filebase.sty]` as a first optional argument.

4 Implementation

```
\NeedsTeXFormat{LaTeX2e}[1999/12/01]
```

`\ProvidesClassSVN`

Calls the generic macro with the original LaTeX macro and the string to be used as filename.

```
\def\ProvidesClassSVN{%
  \svnprov@generic\ProvidesClass{\svnprov@filebase@}%
}
```

`\ProvidesFileSVN`

Calls the generic macro with the original LaTeX macro and the string to be used as filename.

```
\def\ProvidesFileSVN{%
  \svnprov@generic\ProvidesFile{\svnprov@filebase@.\
     svnprov@fileext@}%
}
```

`\ProvidesPackageSVN`

Calls the generic macro with the original LaTeX macro and the string to be used as filename.

```
\def\ProvidesPackageSVN{%
  \svnprov@generic\ProvidesPackage{\svnprov@filebase@
     }%
}
```

`\svnprov@generic`

Stores the arguments (1: original macro, 2: file mask (full filename if only base is used?)). Then tests if a explicit file name was given as optional argument. If not the file name from the SVN Id string is used.

```
\def\svnprov@generic#1#2{%
  \def\svnprov@ltxprov{#1}%
  \def\svnprov@filemask{#2}%
  \begingroup
  \svnprov@catcodes
  \@ifnextchar{[}%
    {\svnprov@getid}%
    {\svnprov@getid[\svnprov@svnfilename]}%
}
```

\svnprov@catcodes

Sets the normal catcodes for all characters required by the `getid` macro.

```
\def\svnprov@catcodes{%
  \catcode`\ =10%
  \catcode`\$=3%
  \@makeother\:%
  \@makeother\-%
}
```

Enforce normal catcodes for the definition of the `Id` scanning macros. This makes sure that all scan patterns have the same catcodes during definition and execution.

```
\begingroup
\svnprov@catcodes
```

\svnprov@getid

Saves first argument as filename and calls the scan macro with the second. A fall-back string is provided to avoid TEX parsing errors.

```
\gdef\svnprov@getid[#1]#2{%
  \endgroup
  \def\svnprov@filename{#1}%
  \svnprov@scanid #2\relax $%
    Id: unknown.xxx 0 0000-00-00 00:00:00Z user $\↙
       empty\svnprov@endmarker
}
```

\svnprov@scanid

Parses the Id string and tests if it is correct (#1=empty, #8=`\relax`). If correct the values are stored in macros and the next macro is called. Otherwise a warning message is printed. In both cases any remaining text of the parsing procedure is gobbled before the next step.

```
\gdef\svnprov@scanid#1$%
  Id: #2 #3 #4-#5-#6 #7 $#8{%
  \def\next{%
    \begingroup
    \PackageWarning{svn-prov}{Invalid SVN Id line ↙
       found! File name might be
    '#2' or '\expandafter\strip@prefix\meaning\↙
       @filef@und'. This occured}{}{}{}%
    \endgroup
    \svnprov@gobbleopt
```

```
  }%
  \ifx\relax#1\relax
    \ifx\relax#8\empty
      \def\svnprov@svnfilename{#2}%
      \svnprov@splitfilename{#2}%
      \def\svnprov@filerev@{#3}%
      \def\svnprov@filedate@{#4/#5/#6}%
      \def\next{\begingroup\svnprov@catcodes\
         svnprov@buildstring}%
    \fi
  \fi
  \expandafter\next\svnprov@gobblerest
}% $
```

End of area with enforced catcodes.

```
\endgroup
```

\svnprov@splitfilename

Expands the argument and initialises the file base macro before it calls the next macro with the expanded argument and a dot to protect for TEX parsing errors. The \relax is used as end marker.

```
\def\svnprov@splitfilename#1{%
  \edef\g@tempa{#1}%
  \let\svnprov@filebase@\@gobble
  \expandafter
  \svnprov@splitfilename@\g@tempa.\relax
}
```

\svnprov@splitfilename@

The second argument is tested if it is empty (end of file name reached). If not empty the first argument is concatenated to the file base macro and the macro calls itself on the second argument. This ensures correct handling of file name which contain multiple dots.

If the second argument was empty it is tested if the file base name is still in its initialised state which means that there is no file extension. Then the file base is defined to the first argument and the extension as empty. Otherwise the file extension is defined to the first argument and the file base macro is unchanged because it is already correct.

```
\def\svnprov@splitfilename@#1.#2\relax{%
  \if&#2&
    \ifx\svnprov@filebase@\@gobble
      \gdef\svnprov@filebase@{#1}%
```

```
      \gdef\svnprov@fileext@{}%
    \else
      \gdef\svnprov@fileext@{#1}%
    \fi
    \let\next\relax
  \else
    \xdef\svnprov@filebase@{\svnprov@filebase@.#1}%
    \def\next{\svnprov@splitfilename@#2\relax}%
  \fi
  \next
}
```

`\svnprov@gobblerest`

Simply gobbles everything up to the next endmarker.

```
\def\svnprov@gobblerest#1\svnprov@endmarker{}
```

`\svnprov@endmarker`

This is the end marker which should never be expanded. However it gets defined and set to an unique definition which will gobble itself if ever expanded.

```
\def\svnprov@endmarker{\@gobble{svn-prov endmarker}}
```

`\svnprov@gobbleopt`

Gobbles an optional argument if present.

```
\newcommand*\svnprov@gobbleopt[1][]{}
```

`\svnprov@defaultdesc`

Default description text to be used. Does not include the file date which is prepended later.

```
\def\svnprov@defaultdesc{%
  (SVN Rev:\space\svnprov@filerev@)%
}
```

`\svnprov@buildstring`

First aliases the internal macro to user-friendly names and then builds the info string. Finally the stored original LaTeX macro is called with the filename and information.

```
\newcommand*\svnprov@buildstring[1][\↙
    svnprov@defaultdesc]{%
  \@ifnextchar{[}%
    {\svnprov@buildstring@{#1}}%
    {\svnprov@buildstring@{#1}[\relax]}%
}
\def\svnprov@buildstring@#1[#2]{%
  \endgroup
  \begingroup
    \let\rev\svnprov@filerev@
    \let\filerev\svnprov@filerev@
    \def\Rev{\rev\space}%
    \let\revinfo\svnprov@defaultdesc
    \let\filebase\svnprov@filebase@
    \let\fileext\svnprov@fileext@
    \ifx\fileversion\@undefined
      \def\fileversion{v0.0}%
    \fi
    \edef\filename{\filebase.\fileext}%
    \xdef\svnprov@filename{\svnprov@filename}%
    \ifx\svnprov@filename\filename\else
      \svnprov@splitfilename{\svnprov@filename}%
    \fi
    \let\filename\svnprov@filename
    \ifx\relax#2\empty
      \xdef\svnprov@fileinfo@{#1}%
      \svnprov@getversion{#1}%
      \global\let\svnprov@filedesc@\svnprov@filedesc@
      \global\let\svnprov@fileinfo@\svnprov@fileinfo@
    \else
      \xdef\svnprov@fileversion@{#1}%
      \xdef\svnprov@filedesc@{#2}%
      \xdef\svnprov@fileinfo@{#1 #2}%
    \fi
  \endgroup
  \svnprov@ltxprov{\svnprov@filemask}%
    [\svnprov@filedate@
     \ifx\svnprov@fileinfo@\empty\else
      \space
      \svnprov@fileinfo@
     \fi
   ]%
}
```

`\GetFileInfoSVN`

The macro provides the file information of the given file, or (the star version) the last file which called one of the above \Provides... macros. For this the internal macros are simply copied to user-friendly names.

This macro is inspired by the macro \GetFileInfo{⟨*file name*⟩} from the doc package.

```
\def\GetFileInfoSVN#1{%
  \ifx*#1\relax
    \let\filebase\svnprov@filebase@
    \let\fileext\svnprov@fileext@
    \let\filename\svnprov@filename
    \let\filedate\svnprov@filedate@
    \let\filerev\svnprov@filerev@
    \let\fileversion\svnprov@fileversion@
    \let\fileinfo\svnprov@filedesc@
    \expandafter\svnprov@settoday@\svnprov@filedate@\↙
       relax
  \else
```

Given argument could be filename or short name. If a short name exists for the argument it was a filename is is defined as such, otherwise the filename is read from the \⟨*short name*⟩@long macro.

```
    \expandafter\let\expandafter\@gtempa\csname#1↙
       @short\endcsname%
    \ifx\@gtempa\relax
      \def\@gtempa{#1}%
      \expandafter\let\expandafter\filename\csname#1↙
         @long\endcsname
    \else
      \edef\filename{#1}%
    \fi
    \expandafter\let\expandafter\filebase\csname\↙
       @gtempa @base\endcsname
    \expandafter\let\expandafter\fileext \csname\↙
       @gtempa @ext\endcsname
    \expandafter\let\expandafter\filedate\csname\↙
       @gtempa @date\endcsname
    \expandafter\let\expandafter\filerev \csname\↙
       @gtempa @rev\endcsname
    \expandafter\let\expandafter\fileversion\csname\↙
       @gtempa @version\endcsname
    \expandafter\let\expandafter\fileinfo\csname\↙
       @gtempa @info\endcsname
    \@ifundefined{\@gtempa @date}%
      {\def\filetoday{?}}%
      {\expandafter\expandafter\expandafter\↙
```

```
            svnprov@settoday@\csname\@gtempa @date\↙
            endcsname\relax}%
183   \fi
184 }
```

`\svnprov@settoday@`

Sets `\filetoday` by expanding `\today` after setting the year/month/day locally.

```
186 \def\svnprov@settoday@#1/#2/#3\relax{%
187   \def\filetoday{{%
188     \year#1\month#2\day#3\relax
189     \today
190   }}%
191 }
```

`\DefineFileInfoSVN`

Defines macros in the form `\⟨`*filename*`⟩@⟨`*xxx*`⟩`, where ⟨*xxx*⟩ is `date`, `version`, `rev`(ision), `info`, (file name)`base` and `ext`(ension).

```
193 \newcommand*\DefineFileInfoSVN[1][\svnprov@filemask]{↙
    %
194   \expandafter
195   \edef\csname\svnprov@filemask @short\endcsname{#1}%
196   \expandafter
197   \edef\csname#1@long\endcsname{\svnprov@filemask}%
198   \expandafter
199   \let\csname#1@base\endcsname\svnprov@filebase@
200   \expandafter
201   \let\csname#1@ext\endcsname\svnprov@fileext@
202   \expandafter
203   \let\csname#1@date\endcsname\svnprov@filedate@
204   \expandafter
205   \let\csname#1@version\endcsname\↙
        svnprov@fileversion@
206   \expandafter
207   \let\csname#1@rev\endcsname\svnprov@filerev@
208   \expandafter
209   \let\csname#1@info\endcsname\svnprov@filedesc@
210 }
```

`\svnprov@getversion`

Checks if the argument (a file description) starts with 'v'. If so everything until the first space is taken as version number. Otherwise the whole text is taken

as description without version. Special care is taken to avoid a parser error if there is no space included.

```
\def\svnprov@getversion#1{%
  \edef\@tempa{#1\space}%
  \expandafter\svnprov@@getversion\@tempa\svnprov@endmarker
}
\def\svnprov@@getversion{%
  \@ifnextchar{v}%
    {\svnprov@getversion@}%
    {\svnprov@getversion@@}%
}
\def\svnprov@getversion@#1 #2\svnprov@endmarker{%
  \gdef\svnprov@fileversion@{#1}%
  \ifx&#2&%
    \gdef\svnprov@filedesc@{}%
  \else
    \xdef\svnprov@filedesc@{\svnprov@zapspace#2\svnprov@endmarker}%
  \fi
}
\def\svnprov@getversion@@#1 \svnprov@endmarker{%
  \gdef\svnprov@fileversion@{}%
  \gdef\svnprov@filedesc@{#1}%
}
\def\svnprov@zapspace#1 \svnprov@endmarker{#1}
```

Finally, call the macros for this package itself.

```
\ProvidesPackageSVN{$Id: svn-prov.dtx 1858 2010-04-14 18:30:31Z martin $}%
  [\svnprov@version\space Package Date/Version from SVN Keywords]
\DefineFileInfoSVN[svnprov]
```