The svn-multi Package

formerly known as svnkw

Martin Scharrer
martin@scharrer-online.de
http://www.scharrer-online.de/latex/svn-multi
CTAN: http://tug.ctan.org/pkg/svn-multi

Version 2.2
October 19, 2009

1 Introduction

This package allows to typeset version control (VC) information provided by Subversion[1] keywords (e.g. `$Id: ... $`) in LaTeX documents which can contain of multiple `.tex` files included using `\include` or `\input`. Subversion is a modern version control system designed to replace its predecessor CVS and uses integers as revision numbers.

This package reads the keywords of all files and provides the VC information of of the most recent changed file of the document to the user through a set of macros. This information is written to an auxiliary `.svn` file during the first LaTeX run and read back at the next which introduces the same delay known from the table of contents. The standard LaTeX switch `\nofiles` can be used to suppress the file generation.

In addition to this basic functionality several more features are provided:

- Macros to typeset the VC information of the current source file.
- Access of all parts of the VC date.
- Formatting of author names or revision numbers.
- Definition of groups and subgroups.
- Including of the VC information of external files.
- Table of Revisions.

[1] Subversion homepage: http://subversion.tigris.org/

1.1 Scope of Keywords

This package provides the Subversion keyword data in several different scopes: document-global, file-local and, new with v2.0, by group.

Document Global

The document global macros, like `\svnrev`, return the latest version control information (keyword data) for the whole multi-file document, i.e. the information of the latest changed file of the document. To collect, sort and provide this information is the main functionality of this package.

Local to Current File

There are also file-local macros, e.g. `\svnfilerev`, which return the version control information of the current file, i.e. the file they are used in. It is assumed here that every file using this macros calls first either a `\svnid` or `\svnidlong` macro or both. See section 2.2 for more details about the id macros. Please note that the file-local macros technically actually return the *last registered* information from the last `\svnid` or `\svnidlong`. As long the **filehooks** option (new in v2.0) is not enabled (explicit or implicit) this keyword macros will leak from one file over to the next. This will cause wrong results if they are used in a file before or without any id macros. With this option the macros will be reset at the beginning of every source file of the document.

Groups

Version 2.0 introduces the concept of groups. Several files of a multi-file LaTeX document can be grouped together and the latest version control information of all files of a group is provided by macros. This works in the same way as the global macros mentioned above but only with the files in the group. It can also be seen from the other side: the macros are local like the file-local macros mentioned above but for all files of the group, not only the current one.

These groups could also be called *file groups*, *keyword groups* or, like in programming languages, *namespaces*. In this manual they will be reference as simple *groups* most the time. In places where they could be confused with TeX groups (`{ }`, `\begingroup \endgroup`), e.g. "in the current group" or "group local", they will be called *keyword groups*.

There is no limitation (besides internal LaTeX resource limits) for the number of different groups. The files of one group do not have to be included in a row but can be included everywhere in the document. The version control information of the current group can be typeset with macros like `\svncgrev` (`cg` for *current group*). Also, a general but less robust macro `\svng{`⟨*group name*⟩`}{`⟨*key*⟩`}` is provided to access others groups by name everywhere in the document. To avoid some macro robustness problems the current group can be changed locally for the output macros using `\svnsetcg{`⟨*group name*⟩`}`.

See section 2.3 for further details and usage instructions on group macros.

2 Usage

The version control information are provided by Subversion keywords which first need to be read in by dedicated macros and can then be typeset using different macros.

2.1 Package Options

Since v2.0 this package provides options to enable only a needed features, e.g. to avoid problems with other packages or save TEX memory. For backwards compatibility to pre-2.0 package versions all old features are enabled by default and all new features are disabled to save a little of TEX memory.

All options except the first two are boolean key=value options (so far) and await either '`true`' or '`false`' as value. A missing value means '`true`'. So e.g. `[groups=true,verbatim=false,external]`, enables the **external** and **groups** options but disables the **verbatim** option.

The available options are:

old

Only pre-v2.0 features are active. This enables **verbatim** and disables all other options below. This is the default for reasons mentioned above.

all

Activates all features of the package except of the experimental ones.

verbatim

Controls the verbatim mode of the keyword parser macros. Normally verbatim mode is very much wanted to support strange characters in URLs and file names, but this options gives the user a possibility to disable verbatim, e.g. for trouble shooting. Please note that verbatim mode is needed in order to make `svn-multi` work with some packages, like `babel`[2] with the `french` option.

external

Controls the support for keywords from external files described in section 2.10. This needs either the external script `svn-multi.pl` or the **autokw** option.

If the **groups** option is enabled the macro `\svnexternalgroup{`⟨*group name*⟩`}` can be used to declare a own group which is used for all the external files. Otherwise they are placed in the currently active group. This macro can be used several times during the document where an empty argument means *no group* and a '`*`' means *current group*.

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

groups

Controls the keyword groups feature described in section 2.3.

subgroups

Controls the automatic declaration of all input files as subgroups so that there keyword information can be typeset inside other files. The group name is the file path without the file extension (‘`subdir/filebase`’). It is possible to disable and re-enable this using `\svnsubgroupsfalse` and `\svnsubgroupstrue` during the document preamble or body to exclude certain files. See section 2.3.1 for additional information.

graphics

This option allows to automatically declare all images included using the macro `\includegraphics` from the `graphics`[3]/`graphicx`[4] package as external files (see section 2.10). The options **external** and **autoload** are activated by this option so that the produced `.svx` files are loaded automatically. An `autoload=false` option after `graphics` will deactivate this, but then an `\svnexternal` macro must be included in all LaTeX files which should take the image revisions into account.

The `graphics` package is loaded if this option is active. If this package is needed with some special options it should be loaded by the LaTeX document before `svn-multi`.

Please note that this feature needs to tie itself into the `graphics` package and might fail if the internal structure of this package changes in future versions.

If the **groups** option is enabled the macro `\svngraphicsgroup{`⟨*group name*⟩`}` can be used to declare a own group which is used for all the graphic files (also for pgf images, see below). Otherwise they are placed in the group specified by `\svnexternalgroup` which defaults to the currently active group. This macro can be used several times during the document where an empty argument means *no group* and a ‘`*`’ means *current group*.

Some graphics like logos can appear frequently in a document. Do not count them as part of each chapter they can be ignored using `\svnignoregraphic{`⟨*file path*⟩`}`. The macro `\svnconsidergraphic{`⟨*file path*⟩`}` disables this again. Such graphics can be then included manually using an explicit `\svnexternal` macro.

pgfimages

Identical to like the **graphics** option but for the `pgf`[5] package (implemented against the version from 2008/01/15) with the `\pgfuseimage` and `\pgfimage` macros. Please also see the notes about package loading and ties mentioned above.

autoload

Controls automatic loading of corresponding `.svx` files at the begin of files included using `\input` or `\import`. This avoids the need of putting an `\svnexternal` macro

[3] CTAN: `http://tug.ctan.org/pkg/graphics`

[4] CTAN: `http://tug.ctan.org/pkg/graphicx`

[5] CTAN: `http://tug.ctan.org/pkg/pgf`

in every file just to load the `.svx` files created automatically by the **graphics** option. The option **external** is activated by **autoload**.

table

Controls the generation of a table of revisions which can be included using the `\tableofrevisions` macro. This table shows the revisions of all files and groups. This needs **groups** to work which is activated with **table**. Enable **subgroups** to include a list of all files per group. See the section 2.9 for more information.

filehooks

This option loads the `fink`[6] package and installs at-begin-input-file and at-end-input-file hooks which are needed for many of the options above. While this option is enabled automatically if needed it can be also enabled manually to ensure that the file-local macros are reset to empty values at the begin of each input file. This prevents the keyword from leaking over from one file to the next. After every subfile the file-local keyword macros are also restored to the value of the parent file.

autokw

This experimental feature allows the automatically extraction of the keyword values from the hidden Subversion working copy database. The database (a text file called '`entries`') is located in the hidden Subversion directory '`.svn`' (or '`_svn`' on some Windows installations) inside every directory which is under VC.

This feature makes an external script like `svn-multi.pl` or even keyword macros redundant as long the files are inside a Subversion working directory. However, this feature does not work if the files are exported (e.g. with `svn export`) or manually copied and also depends on the used version of Subversion. Only versions starting with 1.4 (with working copy format version 7) are supported. Earlier version used a different format for the `entries` file. Newer versions should be compatible as long the basic format is not changed again.

The experimental status will be lifted after the feature was tested using different Subversion versions on different platforms. Please do not hesitate to send error reports to the package author. Minimal examples and information about the used Subversion version and platform are very much appreciated.

This option allows the following values:

`false` Feature is disabled (default).

No value or **`true`** or **`all`** The keywords of all files are automatically extracted. No `\svnid` or `\svnidlong` macros or external scripts are necessary as long the files are inside a Subversion working directory and not exported.

`ext` Only the keywords of external files are extracted. This avoids the need of the `svn-multi.pl` script. The option **external** must be still enabled manually.

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

2.2 Including Subversion Keywords

Subversion keywords are included using `\svnid` or `\svnidlong`. These macros should be written very early in each file, i.e. in the preamble of the main document soon after `\documentclass` and `\usepackage{svn-multi}` and as first in *every* subfile before an `\chapter` or similar macro. They do not create any output. See section 2.4 to learn how to typeset the keyword values.

`\svnid{$Id$}`

The macro is for the `Id` keyword and must be written like shown. A trailing colon with or without spaces after the '`Id`' is also valid but **everything else** except a valid Subversion string will cause a TEX parse error. The subversion property `svn:keywords` must be set on all source files and include '`Id`' so that Subversion will expand it at the next commit.

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

Macro for a "long Id". Saves similar values like in '`Id`' but from the above four keywords. 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. However, both can also be used together. In this case the `\svnid` macro should be come last. Because its revision is not higher (but identical) than the revision of the `\svnidlong` macro it does not override its values. This way both the full time zone from the long and the file name from the short id macro can be accessed. Please note that all features from the 2.0 version load the `fink` package which lets you typeset the current file name anyway using `\finkbase.\finkext`[7].

This macro must be written like seen above while the order of arguments is not meaningfull. The Subversion property `svn:keywords` must be set on all source files with an value which includes '`HeadURL LastChangedDate LastChangedRevision LastChangedBy`' or one of their alternative spellings (e.g. '`URL`', '`Rev`', '`Date`', etc.).

Please note that the arguments are read verbatim as long the **verbatim** option is not disabled explicitly. Special precaution are taken to allow spaces, newlines and comments direct after the `\svnidlong` and after each of the four arguments. In fact everything not inside braces `{ }` is ignored.

`\svn{$`⟨*keyword*⟩`$}`

`\svn*{$`⟨*keyword*⟩`$}`

This macro let you typeset svn keywords directly. The dollars will be stripped and the rest is typeset as normal text. The star version strips also the space before

[7] The file name in `$Id$` is always the original Subversion file name while the one given by the `fink` package is the current file name. Both could differ if the file got renamed.

the last dollar. This macro alone was the very first version of `svnkw` and is still included for fast and simple keyword typesetting.

`\svnkwsave{$⟨keyword⟩$}`

This macro lets you include and save any keyword you like. 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. Please note that the argument is read verbatim and that there should be no space between the macro and the argument's left brace.

2.3 Groups

Starting with v2.0 files can be grouped together and the keyword values of the latest revision of a group can be accessed. Use the **groups** option to activate these macros.

`\svngroup{⟨group name⟩}`

This macro declares all following files until the next `\svngroup` as part of the given keyword group. It can be placed inside the main file before some `\include`/`\input` macros or inside subfiles before the id macros, i.e. direct at the start of the file.

The changes done by this macro are TEX global, i.e. there can't be caught using TEX groups (`{ }`). However, in order to prevent subfiles to change the group of the rest of the parent file the group will be restored to the previous one at the end of each input file.

The latest VC information of a group can be typeset with the `\svnvgXXX` macros or the `\svng` macro shown in section 2.4.

`\thesvngroup`

Returns the name of the current keyword group.

`\svnsetcg{⟨group name⟩}`

Normally the `\svncgXXX` macros mentioned below use the last keyword group defined by `\svngroup` but this can be changed using the `\svnsetcg` macro. The idea behind it is that the currently selected group can be changed locally to the current TEX group for the keyword output macros `\svncgXXX` only while the group for the keyword input macros like `\svnid` is unaffected.

To reset the used group to the last one defined by `\svngroup` simply use `\svnsetcg` with an '`*`' as argument.

Example 1: `{\svnsetcg{abc}\svnFullAuthor{\svncgauthor}}`
would output the full author's name of group *abc*.

Example 2: To typeset the three keyword values of group *abc* somewhere outside this group use:

```
{\svnsetcg{abc}Rev: \svncgrev\\Date: \svncgdate\\
Author: \svncgauthor\\}
```

Example 3: To typeset the date of group *abc* outside of this group in the format of `\today` use: `{\svnsetcg{abc}\svncgtoday}`

`\thesvncg`

Returns the name of the current group selected by `\svnsetcg`.

2.3.1 Files as Subgroups

The group feature could be used to access the version control information of single files anywhere in the document when these are defined as own groups for themselves. Because a file can only be in one group this would not be compatible with the normal usage of the group feature. Therefore a special feature was introduced to automatically or manually define a file as subgroup for itself which does not influence its membership in a normal group.

Declaration: This feature is enabled by the **subgroups** option. All files of the document are then automatically declared as extra groups. This can be disabled for parts of the document using `\svnsubgroupsfalse` and re-enabled using `\svnsubgroupstrue` macros. The current file can be manually declared as extra group with the `\svnsubgroup` macro.

`\svnsubgroup`

This macro declares the current file as subgroup. It is used automatically for every subfile if **subgroups** and `\svnsubgroupstrue` are enabled.

Exclude/Consider files extensions: The above mentioned automatically group declaration uses an hook which is triggered every time another file is read by the document. This unfortunately includes other packages, some auxiliary files and font, config and other files read in by this packages. An internal filter is in place to ignore this files by their file extension. This filter can by modified by the two following macros.

`\svnignoreextensions{`⟨*comma separated list of extension without leading dot*⟩`}`

Tells `svn-multi` to ignore the following file extension and never declare files with them as extra groups.

`\svnconsiderextensions{`⟨*comma separated list of extension without leading dot*⟩`}`

Tells `svn-multi` to (re-)consider the following file extension and declare files with them as extra groups if read in.

Typesetting: The keyword information of the subgroups (subfile including any included external files or subsubfiles) can be typeset using the normal group typeset macros mentioned below where the group name is the file path without extension. The keyword information of the `.tex` file alone can be typeset with the full file path including extension.

Example: `\svnsetcg{subdir/some_file.tex}\svncgrev` would typeset the revision of the file `some_file.tex` while `\svnsetcg{subdir/some_file}\svncgrev` would typeset the latest revision of the same file or any subfile or declared external file included by it.

2.4 Typesetting the Keyword Values

The following macros can be used to typeset the keyword values anywhere in the document. Please note that not all LaTeX fonts have all special characters, e.g. '`_`' is not provided in the standard roman font. To proper typeset file names and URLs containing these letters you can use either teletype font (`\texttt`) or use `{\urlstyle{rm}\svnnolinkurl{...}}` which requires the `hyperref`[8] package.

Like already mentioned `svn-multi` knows three scopes of keywords. The first contains of the keywords for the complete document which hold the values of the most recent committed file and the second contains of the *current* or *file local* keywords, e.g. the keywords of the current file. Only this two are described here while the third scope is described in section 2.3.

`\svnrev`

`\svndate`

`\svnauthor`

These macros hold the keyword values of the whole document, i.e. of the most recent revision. They can be used everywhere in every file of the LaTeX document, after `\usepackage{svn}` of course. Please see section 2.5 how to typeset parts of the date.

`\svnfilerev`

`\svnfiledate`

`\svnfileauthor`

These macros hold the keyword values of the current LaTeX file, but only if it contains a `\svnid` or `\svnidlong` macro. Otherwise the macros hold either zero values or the values of the last file dependent on whether an option is enabled which enabled the `fink` package. Please see section 2.5 how to typeset parts of the date. See `\svnkw` below for all other keywords.

[8] CTAN: `http://tug.ctan.org/pkg/hyperref`

`\svncgrev`

`\svncgauthor`

`\svncgdate`

These macros return keyword values of the currently selected keyword group. In order to hold them robust, which is important to use them in macros like `\svnFullAuthor`, they do not provide any arguments to select other groups than the current one. To access keyword values of other groups use the general macro `\svng` or change the locally selected keyword group using the macro `\svnsetcg`.

`\svng{`⟨*group name*⟩`}{`⟨*key*⟩`}`

This macro is a general form of the `\svncgXXX` macro mentioned above. The first argument is the requested keyword group, the second one the requested keyword in the form of `rev`, `date`, `author`, `year`, etc.. Please note that this macro can not be used inside macros like `\svnFullAuthor`.

`\svnmainurl`

`\svnmainfilename`

The macro `\svnmainurl` and `\svnmainfilename` hold the URL and the filename of the main LaTeXfile as long the keywords `HeadURL` or `Id` were used in it, respectively. These can be used to typeset this information anywhere in the document which might be more descriptive as the name of the current file (which can be typeset with `\svnkw{HeadURL}` or `\svnkw{Filename}` after `\svnid` or `\svnidlong`, respectively).

`\svnsetmainfile`

This will declare the current file as the main LaTeX file by defining the above macros. It will automatically be called at the end of the preamble so the user normally doesn't have to use it by him- or herself as long it isn't needed in the preamble.

Please note that this macro changes the definition of `\svnmainurl` and `\svnmainfilename` directly without going over the auxiliary file. Calling it in several files will make this two macros inconsistent.

`\svnkw{`⟨*keyword name*⟩`}`

All keywords saved with `\svnid`, `\svnidlong` or `\svnkwsave` can be typeset by this macro which is a holdover from a very early version of this package when multiple files where not supported. 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. `svn-multi` is also providing a macro \svnnolinkurl which works like \url but doesn't adds an hyperlink. See the description of this macro for more details.

If the given keyword doesn't exists a package warning is given to allow spelling errors to be tracked down. This doesn't work well when \svnkw is used inside \url. In this case the warning code will be typeset(!) verbatim into the document by \url.

\svnkwdef{⟨*keyword name*⟩}{⟨*value*⟩}

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 this macro has no influence on the calculation of the latest revision.

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}{123}` and then typeset it with `\svnkw{Revision}` or `\svnkw{LastChangedRev}` if you like.

New in version 2.2. Will change in future versions:

\ifsvnfilemodified{⟨*case: modified*⟩}{⟨*case: not modified*⟩}

\ifsvnmodified{⟨*case: modified*⟩}{⟨*case: not modified*⟩}

This two macro can be used to check if either the current or any file was modified after it was last checked into the repository. At the moment (v2.2) a file is marked 'modified' if there is either a '`*`' or '`M`' after the revision number, e.g. '`$Rev: 123M $`'. Such an marker is automatically added for exported files ('`svn export`') if there where locally modified, but can also be added manually.

Future versions of this package might mark files as modified by use of the **autokw** option which can read this information out of the working directory Subversion entries.

2.5 Accessing Date Values

\svnyear
\svnmonth
\svnday
\svnhour
\svnminute
\svnsecond
\svntimezone
\svntimezonehour
\svntimezoneminute

\svnfileyear
\svnfilemonth
\svnfileday
\svnfilehour
\svnfileminute
\svnfilesecond
\svnfiletimezone
\svnfiletimezonehour
\svnfiletimezoneminute

\svncgyear
\svncgmonth
\svncgday
\svncghour
\svncgminute
\svncgsecond
\svncgtimezone
\svncgtimezonehour
\svncgtimezoneminute

Whenever the date information is read, i.e. by \svnkwsave{LastChangedDate} \svnkwsave{Date}, \svnidlong or \svnid, the following macros are set to the appropriate date parts for the current file (the \svnfile... versions) and for the whole document.

Please note that the hour and timezone are dependent on the keyword which defines the date information. The hour will be in UTC aka Zulu-time, i.e. timezone +0000, when the date comes from the Id keyword. Otherwise the hour and timezone will be in local time. To avoid confusion the Id and Date/LastChangedDate keywords, e.g. \svnid and \svnidlong, should not be intermixed and/or the timezone should always be typeset together with the time.

Starting with v1.4 of svn-multi the timezone macros return the full timezone, i.e. sign, hour and minute part, e.g. +0100, not only the sign and hour. The new macros \svntimezonehour/\svnfiletimezonehour and \svntimezoneminute/ \svnfiletimezoneminute can be used to access only the hour including sign or the minute part, respectively.

Older versions of this manual assumed the minute part as always 00 and suggested to add it manually if needed: \svnfiletimezone00 or \svntimezone00. In order not to "break" documents which followed this suggestion this two macros now remove a trailing 00 if present. However, this can be a problem when they are used inside an argument of another macro. One solution for this is to redefine them without the 00 removal part:

```
\renewcommand{\svntimezone}{\svntimezonehour\svntimezoneminute}
\renewcommand{\svnfiletimezone}{\svnfiletimezonehour\svnfiletimezoneminute}
```

To revert to the old (pre-v1.4) definition use:

```
\renewcommand{\svntimezone}{\svntimezonehour}
\renewcommand{\svnfiletimezone}{\svnfiletimezonehour}
```

\svntime

\svnfiletime

\svncgtime

This macros return the time part of the date only and simply return the corresponding hour, minute and second macros with a colon as separator.

`\svnpdfdate`

Returns the last changed date of the whole document in a format needed for `\pdfinfo`. Can be used like this:

```
\pdfinfo{ /CreationDate (D:\svnpdfdate) }
```

to set the PDF creation date to the last changed date if you use `pdflatex` to compile your LaTeX document.

`\svntoday`

`\svnfiletoday`

`\svncgtoday`

These macros typeset the document-global, current-file or current-group date, respectively, using the format of `\today` which depends on the used language. To adjust the language of your document use the `babel` package.

2.6 Using Full Author Names

If you like to have the full author[9] names, not only the usernames, in your document you can use the following macros. First you have to register all authors of the document with `\svnRegisterAuthor` and then you can write e.g. `\svnFullAuthor{\svnauthor}` or `\svnFullAuthor{\svnfileauthor}`.

`\svnRegisterAuthor{`⟨*author*⟩`}{`⟨*full name*⟩`}`

This macro registers ⟨*full name*⟩ as full name for ⟨*author*⟩ (a subversion username) for later use with `\svnFullAuthor`.

`\svnFullAuthor{`⟨*author name or macro*⟩`}`

`\svnFullAuthor*{`⟨*author name or macro*⟩`}`

Takes the username as argument and returns the full name if it was registered first with `\svnRegisterAuthor`, otherwise it returns the given username. The star version returns the username in parentheses after the full name. This is normally used in one of the following forms:

```
\svnFullAuthor{\svnauthor}
\svnFullAuthor{\svnfileauthor}
\svnFullAuthor{\svncgauthor}
```

2.7 Using Full Revision Names

Like the author's also revision names/tags can be registered and used later. These macros were implemented on user request and have the drawback that you have to guess the next revision number of your document in order to get correct results when you like to tag the to-be-checked-in revision. Please note that this has nothing to do with the normal subversion tagging.

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

`\svnRegisterRevision{⟨revision number⟩}{⟨tag name⟩}`

This registers ⟨*tag name*⟩ as tag name for ⟨*revision number*⟩ for later use with `\svnFullRevision`.

`\svnFullRevision{⟨revision number or macro⟩}`

`\svnFullRevision*{⟨revision number or macro⟩}`

Takes a revision number coming from a macro like `\svnrev`, `\svnfilerev` or a number as argument and returns the full name if it was registered first with `\svnRegisterRevision`, otherwise it returns "Revision ⟨*revision number*⟩". The star version returns also the revision number leaded by 'r' in parentheses after the tag name, e.g. `Name (r123)`.

2.8 Verbatim URLs with and without Hyperlinks

`\svnnolinkurl{⟨macro with returns special text⟩}`

This macro allows you to write `\svnnolinkurl{\svnkw{HeadURL}}` and get the Head URL typeset verbatim. However `\url{\svnkw{HeadURL}}` (`hyperref` package) gives you the same result with a hyperlink. Both macros require the `hyperref` package which is not automatically loaded by `svn-multi`. Please load it manually when you like to use `\svnnolinkurl`.

Since v1.3 all keywords are read and typeset verbatim so this macro isn't this important anymore. However together with `hyperref`'s `\urlstyle` macro it can be used to have keyword values with special characters in roman font, which normally doesn't hold letters like '_'.

Please note that you can't use `hyperref`'s `\nolinkurl` because it won't expand `\svnkw`.

2.9 Table of Revisions

Version 2.0 introduces this new feature which allows a overview table to be typeset which holds the version control information of all files of the document.

`\tableofrevisions`

If the **table** is enabled a table of revision is written into a file called ⟨*main LaTeX file*⟩`.svt` (`t` for table) which can be included using the `\tableofrevisions` macro. The table contains the revision information of the complete document and of all defined groups. If the **subgroups** option is set the table will also include the single files sorted by group.

Table Format Macros

The `.svt` file is in an general format which uses a lot of macros to format the table and the different cells. This macros should not be used by the user directly but rather can be redefined to fit the personal taste. Figure 1 shows this macros in their position inside the table of revisions.

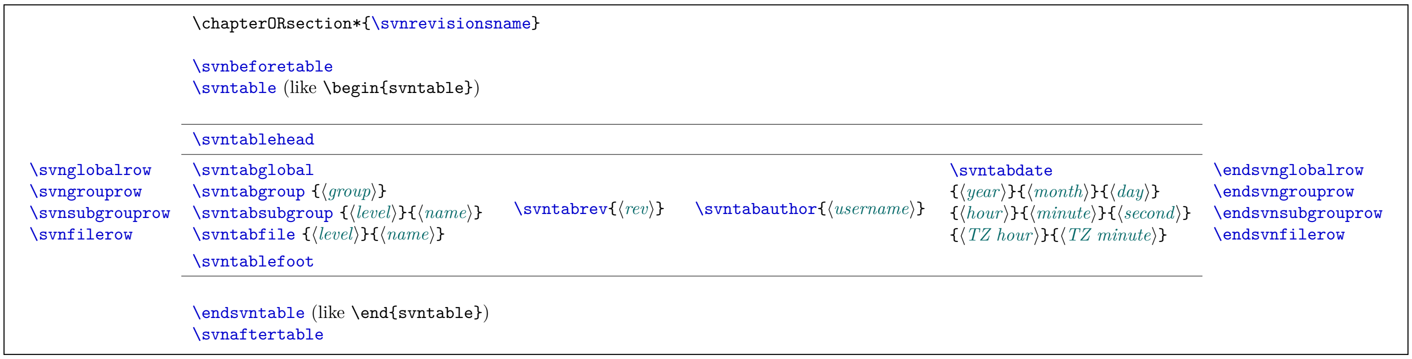

Figure 1: Table formatting macros and their position.

`\svnrevisionsname`

Holds the name of the table of revisions which is printed above it. The default is “Table of Revisions” which can be changed e.g. to another language.

`\svnbeforetable`

Can hold some content to be placed between the headline and the table.

`\svnaftertable`

Can hold some content to be placed after the table. By default it holds a macro to force a new page.

`\svntable`

`\endsvntable`

This macros build the table environment and hold a `\begin{tabular}{llll}` `\end{tabular}` group by default. They can be redefined using

`\renewcommand{svntable}{...}{...}`.

In order to support special table packages like `tabularx`[10] or `longtable`[11] the *content* of this macro is written verbatim into the `.svt` file. All other table macros are written as string and will be expanded when the `.svt` is read back.

`\svntablehead`

Holds the table head row “Name & Rev & Author & Date `\\\hline`”.

[10] CTAN: http://tug.ctan.org/pkg/tabularx

[11] CTAN: http://tug.ctan.org/pkg/longtable

`\svntablefoot`

Can hold the table foot row but is empty by default.

`\svnglobalrow`	`\endsvnglobalrow`
`\svngrouprow`	`\endsvngrouprow`
`\svnsubgrouprow`	`\endsvnsubgrouprow`
`\svnfilerow`	`\endsvnfilerow`

This macros are placed at the begin and end of every corresponding row and are empty be default. They can be used to insert horizontal lines before or after certain row types.

`\svntabglobal`

Simply typesets the name for the row holding the information of the whole document, e.g. “Document”.

`\svntabgroup{`⟨*group name*⟩`}`

Formats the group name.

`\svntabsubgroup{`⟨*nesting level (1,2,3,...)*⟩`}{`⟨*subgroup name*⟩`}`

Formats a subgroup name. The name is derived from a file name and therefore could contain characters like ‘`_`’ which should be taken into account. The macro `\svnnolinkurl` can be used for this. Because subgroups can be nested a number is provided to tell the nesting depth. This can be used to produce different indents for different levels: e.g.: `\addtolength{\leftskip}{#1\someindentamount}`.

`\svntabfile{`⟨*nesting level (1,2,3,...)*⟩`}{`⟨*file path/name*⟩`}`

Same as for the subgroup above but for file names.

`\svntabrev{`⟨*revision number*⟩`}`

Allows the formatting of the revision number. If empty the number will be typeset as normal. Conditional formatting is possible using the `ifthen`[12] package: e.g. the highest revision should be bold:

```
\renewcommand*{\svntabrev}[1]{\ifthenelse{#1=\svnrev}{\textbf{#1}}{#1}}
```

`\svntabauthor{`⟨*author username*⟩`}`

Formats the user names in the table. The macro `\svnFullAuthor` is a good candidate to be used here.

[12] CTAN: `http://tug.ctan.org/pkg/ifthen`

`\svntabdate{⟨year⟩}{⟨month⟩}{⟨day⟩}{⟨hour⟩}{⟨minute⟩}{⟨second⟩}{⟨TZ hour⟩}{⟨TZ minute⟩}`

Allows the formatting of the date. All values are given as numbers. The year has four digits and the timezone hour has a `+` or `-` sign, but all other values have only two digits.

2.10 Including Keywords of External Files

Version 2.0 now supports the inclusion of keywords of external files using the following macros and the Perl script `svn-multi.pl`.

`\svnexternal[⟨group name⟩]{{⟨filea⟩}{⟨fileb⟩}...{⟨filex⟩}}`

Subversion keywords of external files (e.g. non-LaTeX files like images or even directories) can be included using this macro which awaits a list of files, each with the full path relative to the main LaTeX file and each enclosed by `{ }`. The files must be under version control by Subversion, of course. Use `\svnexternalpath` to specify paths to be scanned for this files if they are not located relative to the main file. The requested filenames are written into the `.svn` auxiliary file and then processed by the external script `svn-multi.pl` which must be executed like described below. The appropriate keywords are then written in *⟨source file⟩*`.svx` files (`x` like eXternal) which are read in by the same `\svnexternal` macro at the next LaTeX run. If a group name of '`*`' for the external files is specified using the optional argument the keywords will be placed in the same keyword group as this macro. The file local macros like `\svnfilerev` which appear in a source file after `\svnexternal` are affected, i.e. updated if one of the external revision is higher than the one of the source file. This makes sense if e.g. the included graphics are taken as logical part of a source file.

`\svnexternalpath{{⟨patha⟩}{⟨pathb⟩}...{⟨pathx⟩}}`

This macro can be used in the document preamble to declare a set of paths to be scanned for files specified with `\svnexternal`. This avoids the need to provide the path again and again for every file. The paths need to be enclosed in `{ }` and must be in Unix style, i.e. with '`/`' as directory separator and should end with a '`/`'. Windows users should just replace all '`\`' with '`/`', e.g. '`C:\My dir`' gets '`{C:/My dir/}`'.

Script `svn-multi.pl`

The file `svn-multi.pl` which comes with the `svn-multi` package is an external Perl script which has to be run in the command line or by a LaTeX development environment/editor like other tools like `BibTeX` or `Makeindex`. A Perl interpreter and a Subversion command line client (`svn`) must be installed to execute this script. Both are available for free for all major operating systems.

The script should be run inside the document folder in the following order:

1. Compile LaTeX document, `.svn` file is generated.

2. Run `svn-multi.pl` script, `.svx` files are generated.

3. Compile LaTeX document, `.svx` files are read in.

The script can be used with three different sets of arguments and with any combination of them. Please note that the word `jobname` stands for the main LaTeX file name without the `.tex` extension.

`svn-multi.pl` **⟨*jobname*⟩**

As already mentioned in the `\svnexternal` description above this script reads the requested external filename from the ⟨*jobname*⟩`.svn` file. The Subversion command line client `svn` is then used to fetch the needed keywords which are placed in a `\svnidlong` macro inside a ⟨*source file*⟩`.svx` file. Every single source file which uses `\svnexternal` will become its own `.svx` file which allows to attach specific external files to one (or more) specific source files.

`svn-multi.pl` **⟨*jobname*⟩ [--group ⟨*group name*⟩] ⟨*file(s)*⟩ ...**

The second way to use `svn-multi.pl` is to call it with a list of external files. A keyword group can be specified using the `--group` ⟨*group name*⟩ option which can placed any number of times between the file names. The group is used for all external files listed after the option until the next group is specified. All keywords of these files are written in the ⟨*jobname*⟩`.svx` file and read in by the main LaTeX file if a, possible empty, `\svnexternal` macro is included. This allows for easy including of many external files without specifying them all inside the source file. For example `svn-multi.pl` `*/*.jpg` (under Linux/Unix) will include the keywords of all JPG files in all subdirectories.

It is also possible to do this with a sub-(LaTeX)-file by calling the script on it: `svn-multi.pl` ⟨*sub file*⟩ ⟨*external files for sub file*⟩, which will create/overwrite the ⟨*sub file*⟩`.svx` file. However the files given by `\svnexternal` in this sub-file will not be honoured in this case.

`svn-multi.pl` **⟨*jobname*⟩ [--group ⟨*group name*⟩] --fls**

Instead of providing a list of all non-LaTeX/external files the `--fls` option can be used to read this list from the ⟨*jobname*⟩`.fls` file. This file is produced by the LaTeX compiler when run with the `--recorder` option and contains a list of all input and output files. Only input files with a relative path are used. A corresponding keyword group can also be specified.

3 Compile Guide

A document which uses `svn-multi` needs to be compiled by LaTeX in the following ways depending on the used features.

Basic Features – Global and Group Keywords

1. Compile document with LaTeX. The `.svn` file is generated. Document global and group keyword macros are not valid yet.

2. Compile document again with LaTeX. The `.svn` file is read by `svn-multi`. Document global and group keyword macros are now valid.

Table of Revisions

1. Compile document with LaTeX. Document global and group keyword macros are calculated and written to the `.svt` file.

2. Compile document again with LaTeX. The `.svt` file is read and typeset by the `\tableofrevisions` macro.

External Files

1. Compile document with LaTeX. The `.svn` file is generated with references to the external files. Document global and group keyword macros are not valid yet. External files are not taken into account.

2. Run `svn-multi.pl` with the main base name (main file name without extension) as argument. This generates `.svx` files for each `.tex` file which used `\svnexternal`. This files contain the keywords of the external files.

3. Compile document again with LaTeX. The `.svx` files are read by `svn-multi` and the `.svn` file is updated to take the new keywords into account. Document global and group keywords macros only hold internal values.

4. Compile document again with LaTeX. The `.svn` file is read by `svn-multi`. Document global and group keyword macros are now fully valid.

4 Known Issues

This section lists some known issues of the `svn-multi` package and tries to provide some workaround. Please feel free to write `svn-multi` author if you detect any side effects or other issues causes by this package.

4.1 Packet `listings` uses `\input`

If a file ⟨*basename*⟩.⟨*extension*⟩ is typeset verbatim using `\lstinputlisting`, which uses `\input` to read the file, an existing ⟨*basename*⟩`.svx` file is also included as part of the listing. This can be avoided by code like this:

```
{\makeatletter\let\input\@input
\lstinputlisting[options]{filename}
}
```

5 Package Dependencies and Acknowledgements

This package uses some features from other packages and/or patches some macros of them to provide additional related features. This section is used to list this packages, their internal macro which got used and acknowledge the authors/maintainers of them. Please send error reports to the author of svn-multi and not to the people listed below.

All packages (including svn-multi) stand under the LaTeX Project Public Licence (LPPL) which can be found at http://www.latex-project.org/lppl/ and can be freely downloaded from the Comprehensive TeX Archive Network (CTAN) at http://www.ctan.org/.

hyperref

The macro \svnnolinkurl is resembling the hyperref macro \nolinkurl and uses some its internal macros from the \url macro definition.

Used internal macros: \hyper@normalise, \Hurl

Version used: 2008/11/18 v6.78m

Licence: LPPL, any version

Authors/Maintainers: Sebastian Rahtz, Heiko Oberdiek

Location: CTAN: http://tug.ctan.org/pkg/hyperref

fink

The FiNK (File Name Keeper) package is used to get the file name and path of the input files. Two macros are patched to install hooks to execute own macros before and after each input file. This is only done if needed for an enabled option and therefore can be disabled using the option **old**.

Used internal macros: \fink@file, \fink@nextdir, \fink@nextext, \fink@nextbase

Patched internal macros: \fink@prepare, \fink@restore

Version used: 2008/02/27 v2.1.1

Licence: LPPL, any version

Author/Maintainer: Didier Verna

Location: CTAN: http://tug.ctan.org/pkg/fink

graphics

If the **graphics** option is enabled the following macro is patched to record the file name and path of the included graphic.

Patched internal macros: \Gin@setfile

Version used: 2006/02/20 v1.0o

Licence: LPPL, any version

Author/Maintainer: David Carlisle, LaTeX3 Project

Location: CTAN: `http://tug.ctan.org/pkg/graphics`

`pgf`

Like the `graphics` package above a macro of this package is pathed to record the file names and paths of included images when the option **pgfimages** is enabled. Because this images pre-declares images for later use the internal declared 'image macros' are patched as well.

Patched internal macros: `\pgf@declareimage`, `\pgf@image@`⟨*image name*⟩`!`

Used internal macros: `\pgf@filename`, `\pgf@image`

Version used: 2008/01/15 v2.00

Licence: LPPL v1.3c and GPL v2

Author&Maintainer: Till Tantau

Location: CTAN: `http://tug.ctan.org/pkg/pgf`

`latex`

Parts of the macro definitions of the `\tableofcontents` macros from the `article` and `book` class of standard LaTeX were used to define a similar `\tableofrevisions` macro for both this classes and other similar classes.

Version used: 2005/09/16 v1.4f

Licence: LPPL v1.3c

Authors/Maintainers: LaTeX3 Project

Location: CTAN: `http://tug.ctan.org/pkg/latex`

6 Further Reading

The svn-multi package (in version 1.3) and its usage got discussed in the following articles:

[1] Martin Scharrer, "Version Control of LaTeX Documents with svn-multi", The PracTeX Journal, (3), 2007. URL: `http://www.tug.org/pracjourn/2007-3/scharrer/`

[2] Mark Eli Kalderon, "LaTeX and Subversion", The PracTeX Journal, (3), 2007. URL: `http://www.tug.org/pracjourn/2007-3/kalderon-svnmulti/`

[3] Uwe Ziegenhagen , "LaTeX Document Management with Subversion", The PracTeX Journal, (3), 2007. URL: `http://www.tug.org/pracjourn/2007-3/ziegenhagen/`

7 Implementation

7.1 Package Header

Package Identification

```
\NeedsTeXFormat{LaTeX2e}[1999/12/01]
\ProvidesPackageSVN
  {$Id: svn-multi.dtx 1448 2009-10-19 21:28:41Z martin $}
  [\svnmulti@version\space SVN Keywords for multi-file LaTeX documents]
```

Options

Declaration of options and internal switches.

```
\RequirePackage{kvoptions}

\SetupKeyvalOptions{%
  family = svn-multi,
  prefix = @svnmulti@
}
\newif\if@svnmulti@anygraphic
\newif\if@svnmulti@autoload
\newif\if@svnmulti@autokw
\newif\if@svnmulti@autokwall

\DeclareVoidOption{old}{%
  \@svnmulti@verbatimtrue
  \@svnmulti@groupsfalse
  \@svnmulti@externalfalse
  \@svnmulti@graphicsfalse
  \@svnmulti@pgfimagesfalse
  \@svnmulti@autoloadfalse
  \@svnmulti@tablefalse
  \@svnmulti@filehooksfalse
  \@svnmulti@subgroupsfalse
}
\DeclareVoidOption{all}{%
  \@svnmulti@verbatimtrue
  \@svnmulti@groupstrue
  \@svnmulti@externaltrue
  \@svnmulti@graphicstrue
  \@svnmulti@pgfimagestrue
  \@svnmulti@autoloadtrue
  \@svnmulti@tabletrue
  \@svnmulti@filehookstrue
  \@svnmulti@subgroupstrue
}
\DeclareBoolOption[true]{verbatim}
\DeclareBoolOption[false]{groups}
\DeclareBoolOption[false]{external}
\DeclareBoolOption[false]{subgroups}
```

```
\DeclareBoolOption[false]{graphics}
\DeclareBoolOption[false]{pgfimages}
\DeclareStringOption{autoload}[true]
\DeclareBoolOption[false]{table}
\DeclareBoolOption[false]{filehooks}
\DeclareStringOption[false]{autokw}[all]

\ExecuteOptions{old}
\ProcessKeyvalOptions{svn-multi}
```

Enable dependent options:

```
\def\svn@depoption#1{%
  \csname if@svnmulti@#1\endcsname\else
  \message{svn-multi: Required option ’#1’ enabled.}%
  \csname @svnmulti@#1true\endcsname
  \fi
}

\if@svnmulti@groups
  \svn@depoption{filehooks}
\fi
\if@svnmulti@external
  \svn@depoption{filehooks}
\fi
\if@svnmulti@subgroups
  \svn@depoption{groups}
  \svn@depoption{filehooks}
\fi
\if@svnmulti@graphics
  \svn@depoption{external}
  \svn@depoption{autoload}
  \svn@depoption{filehooks}
\fi
\if@svnmulti@pgfimages
  \svn@depoption{external}
  \svn@depoption{autoload}
  \svn@depoption{filehooks}
\fi
\if@svnmulti@autoload
  \svn@depoption{external}
  \svn@depoption{filehooks}
\fi
\if@svnmulti@table
  \svn@depoption{groups}
  \svn@depoption{filehooks}
\fi
```

Check if **autoload** was set explicitly and obey the value.

```
\ifx\@svnmulti@autoload\@undefined
\else
\ifx\@svnmulti@autoload\empty
```

```
\else
\def\svn@temp{true}
\ifx\@svnmulti@autoload\svn@temp
  \@svnmulti@autoloadtrue
  \svn@depoption{external}
  \svn@depoption{filehooks}
\else
\def\svn@temp{false}
\ifx\@svnmulti@autoload\svn@temp
  \if@svnmulti@autoload
  \PackageWarning{svn-multi}{Option 'autoload' disabled.}
  \fi
  \@svnmulti@autoloadfalse
\else
  \PackageError{svn-multi}%
    {Invalid value for 'autoload' option: '\@svnmulti@autoload'^^J%
     ! Only 'true','false' or empty (='true') are allowed!}
\fi\fi\fi\fi
```

Set **autokw** modes:

```
\def\svn@temp{true}
\ifx\@svnmulti@autokw\svn@temp
  \@svnmulti@autokwtrue
  \@svnmulti@autokwalltrue
  \svn@depoption{filehooks}
\fi
\def\svn@temp{all}
\ifx\@svnmulti@autokw\svn@temp
  \@svnmulti@autokwtrue
  \@svnmulti@autokwalltrue
  \svn@depoption{filehooks}
\fi
\def\svn@temp{ext}
\ifx\@svnmulti@autokw\svn@temp
  \@svnmulti@autokwtrue
  \@svnmulti@autokwallfalse
\fi
\def\svn@temp{false}
\ifx\@svnmulti@autokw\svn@temp
  \@svnmulti@autokwfalse
  \@svnmulti@autokwallfalse
\fi
```

General switch if any graphic option is enabled:

```
\if@svnmulti@graphics
  \@svnmulti@anygraphictrue
\fi
\if@svnmulti@pgfimages
  \@svnmulti@anygraphictrue
\fi
```

7.2 General Internal Macros

Some internal used macro which don't fit in any other section.

\svn@ifempty

#1: string

Tests if the given argument is empty. If so the first of the next two token will be expanded, the second one otherwise.

```
\def\svn@ifempty#1{%
  \begingroup
  \edef\svn@temp{#1}%
  \ifx\svn@temp\empty
    \endgroup
    \expandafter
    \@firstoftwo
  \else
    \endgroup
    \expandafter
    \@secondoftwo
  \fi
}
```

\svn@ifequal

#1: string a
#2: string b

Tests if the given arguments are identical, e.g. same strings. If so the first of the next two token will be expanded, the second one otherwise.

```
\def\svn@ifequal#1#2{%
  \begingroup
  \edef\svn@stringa{#1}%
  \edef\svn@stringb{#2}%
  \ifx\svn@stringa\svn@stringb
    \endgroup
    \expandafter
    \@firstoftwo
  \else
    \endgroup
    \expandafter
    \@secondoftwo
  \fi
}
```

\svn@ifvalidrev

#1: macro name

Checks if the given macro (by name) is a valid revision, i.e. defined and not equal to the init value.

```
\def\svn@ifvalidrev#1{%
  \begingroup
  \@ifundefined{#1}%
    {\let\svn@temp\svn@revinit}%
    {\expandafter\edef
     \expandafter\svn@temp\expandafter{\csname #1\endcsname}}%
  \ifnum\svn@temp=\svn@revinit\relax
    \endgroup
    \expandafter
    \@secondoftwo
  \else
    \endgroup
    \expandafter
    \@firstoftwo
  \fi
}
```

`\svn@ifeof`

`#1:` input file handle

Checks if the input file is at the end-of-file (or not open).

```
\def\svn@ifeof#1{%
  \ifeof#1%
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi
}
```

`\svn@ifonlyone`

`#1:` group name

Checks if there is only one element in the given group file list. It looks whether there is a comma in the list.

```
\def\svn@ifonlyone#1{%
  \expandafter\expandafter\expandafter
  \svn@@ifonlyone\csname @svng@#1@files\endcsname,\relax
}

\def\svn@@ifonlyone#1,#2\relax{%
  \svn@ifempty{#2}
}
```

`\svn@input`

`#1:` file name/path

Macro to load `.svx` and `.svt` files. The current keyword group is saved away and restored after the `.svx` file is loaded. The macros `\IfFileExists` with `\@@input` are used because `\InputIfFileExists` got redefined by the `fink` package and there is no need to use `fink` for this files.

```
\def\svn@input#1{%
  \begingroup
```

```
    \let\svn@rg\svn@g
    \IfFileExists{#1}{\@@input #1\relax}{}%
    \global\let\svn@g\svn@rg
  \endgroup
}
```

\svn@inputsvx

`#1`: file name/path without extension

Used to save and restore file keywords when reading `.svx` files.

```
\def\svn@inputsvx#1{%
  \svn@pushfilestack
  \svn@input{#1.svx}%
  \svn@popfilestack
}
```

7.3 Definition of init values

Initialisation of at least the revision to a numeric value is necessary to not break the `\ifnum` tests later in this package. The revision is initialised to -2, but will be set to 0 if an unexpanded `$Rev:$` keyword is read. This way it can be tested if a file had any keyword macros or not.

Note that there a two different macros for the document global keywords:

The user level `\svn`⟨*kw*⟩ macros hold the global value and are only valid after a LaTeX run. They are initialised here and defined in the `.svn` file which is read at the end of the package if it exists and written at the end of the document.

The internal macros `\@svn@`⟨*kw*⟩ store the oldest (i.e. highest revision) keywords read so far from the `\svnid` and `\svnidlong` macros. They change during the document and are used to produce the values of the `\svn`⟨*kw*⟩ macros when the `.svn` file is written.

Group wide macros are initialised when the group is first defined and have three different macros: `\svng@`⟨*group*⟩`@`⟨*kw*⟩ (defined in `.svn`), `\@svng@`⟨*group*⟩`@`⟨*kw*⟩ (accumulator) and also an access macro `\svncg`⟨*group*⟩ which uses `\svn@g`⟨*current group*⟩`@`⟨*kw*⟩.

```
% Init values
\def\svn@revinit{-2}
\let\svnrev\svn@revinit     \let\@svn@rev\svn@revinit
\let\ifsvnmodified\@secondoftwo
\def\@svn@modified{@secondoftwo}%
\def\svndate{}              \def\@svn@date{}
\def\svnauthor{}            \def\@svn@author{}
\def\svnyear{0000}          \def\@svn@year{0000}
\def\svnmonth{00}           \def\@svn@month{00}
\def\svnday{00}             \def\@svn@day{00}
\def\svnhour{00}            \def\@svn@hour{00}
\def\svnminute{00}          \def\@svn@minute{00}
\def\svnsecond{00}          \def\@svn@second{00}
\def\svntimezonehour{+00}   \def\@svn@timezonehour{+00}
```

```
\def\svntimezoneminute{00}  \def\@svn@timezoneminute{00}
\def\svnmainurl{NOT SET}    \def\svnmainfilename{NOT SET}
\def\svnurl{} \def\svnfname{}
\def\svn@temp{}

\def\svn@pg{} \def\svn@g{} \def\svn@cg{\svn@g} \def\svn@rg{\svn@pg}
\let\@svng@@files\relax

\def\svn@initfile{%
  \global\let\svnfilerev\svn@revinit
  \global\let\ifsvnfilemodified\@secondoftwo
  \gdef\svnfiledate{}%
  \gdef\svnfileauthor{}%
  \gdef\svnfileyear{0000}%
  \gdef\svnfilemonth{00}%
  \gdef\svnfileday{00}%
  \gdef\svnfilehour{00}%
  \gdef\svnfileminute{00}%
  \gdef\svnfilesecond{00}%
  \gdef\svnfiletimezonehour{+00}%
  \gdef\svnfiletimezoneminute{00}%
  \gdef\svnfileurl{}%
  \gdef\svnfilefname{}%
  \gdef\svnfiledir{}%
}
\svn@initfile

\newif\ifsvn@modified
```

7.4 Auto-Keywords

Special care must be taken for the line feed character, otherwise it causes an error if **autokw** is disabled.

```
\begingroup
\@makeother\^^L
\if@svnmulti@autokw
\gdef\svne@ff{^^L}
\fi
\endgroup
```

```
\if@svnmulti@autokw
```

```
\newread\svne@read
```

\svne@catcodes

Sets the catcodes for verbatim input reading. Also removes the end-of-line character.

```
\newcommand*{\svne@catcodes}{%
  \let\do\@makeother
  \endlinechar=-1
  \dospecials
  \do\- \do\: \do\. \do\^^L
}
```

\svne@readline

`#1:` macro

Reads the next line to the provided macro and handles the end-of-file case correctly.

```
\def\svne@readline#1{%
  \ifeof\svne@read
    \def#1{}%
  \else
    \read\svne@read to #1\relax
  \fi
}
```

\svne@gobblerest

Gobbles the rest of the current entry.

```
\def\svne@gobblerest{%
  \ifeof\svne@read
    \let\next\relax
  \else
    \read\svne@read to \svn@temp
    \ifx\svn@temp\svne@ff
      \let\next\relax
    \else
      \let\next\svne@gobblerest
    \fi
  \fi
  \next
}
```

\svne@endread

Stops the reading process of the entries file.

```
\def\svne@endread{%
  \closein\svne@read
}
```

\svne@parseentriesfile

`#1:` file path

```
\newcommand*{\svne@parseentriesfile}[1]{%
  \begingroup
    \let\next\relax
```

Open the format file to read the version number:

```
    \openin\svne@read=#1format\relax
    \ifeof\svne@read\else
      \svne@readline\svne@version
      \closein\svne@read
```

Check the format version:

```
   \ifnum\svne@version>7\relax
```

Now open the entries file and read the version number from there again.

```
    \openin\svne@read=#1entries\relax
    \ifeof\svne@read\else
     \svne@catcodes
     \svne@readline\svne@version
```

Check the version and call the parse macros if OK:

```
     \ifnum\svne@version>7\relax
      \def\next{\svne@parsedirentry%
           \svne@parseentries}
     \else
      \closein\svne@read
     \fi
    \fi
   \fi
  \fi
  \next
 \endgroup
}
```

\svne@parsedirentry

Reads the first entry which is the directory entry and sets its URL as base URL for all other entries.

```
\newcommand*{\svne@parsedirentry}{%
 \svne@readline\svne@name
 \svne@readline\svne@kind
 \svn@ifempty{\svne@name}%
  {\svn@ifequal{\svne@kind}{dir}%
   {%
    {\svne@readline\svn@temp}%
    \svne@readline\svne@baseurl
    \svne@gobblerest
   }{}%
  }{}%
}
```

\svne@scandate

\svne@scandate@

Parses the date from the svn entries file. Special care is taken to handle the case when the TeX parsing would fail. The catcode of the characters '-', ':', '.' used inside the date is set explicitly to ensure the correct value.

```
\begingroup

\@makeother\-
\@makeother\:
\@makeother\.
```

```

\gdef\svne@scandate#1{%
  \expandafter\svne@scandate@#1\relax
  0000-00-00T00:00:00.00000Z\relax\relax
}

\gdef\svne@scandate@#1-#2-#3T#4:#5:#6.#7\relax#8\relax{%
  \gdef\svnfileyear{#1}%
  \gdef\svnfilemonth{#2}%
  \gdef\svnfileday{#3}%
  \gdef\svnfilehour{#4}%
  \gdef\svnfileminute{#5}%
  \gdef\svnfilesecond{#6}%
  \gdef\svnfiletimezonehour{+00}%
  \gdef\svnfiletimezoneminute{00}%
  \gdef\svnfiledate{#1-#2-#3 #4:#5:#6Z}%
  \def\svne@date{#1-#2-#3 #4:#5:#6Z}%
}

\endgroup
```

\svne@parseentries

```
\newcommand*{\svne@parseentries}{%
  \svn@ifeof{\svne@read}%
  {}%
  {%
    \svne@readline\svne@name
    \@onelevel@sanitize\svne@name
    \svn@ifeof{\svne@read}%
    {}%
    {%
    \svne@readline\svne@kind
    \svn@ifequal{\svne@kind}{file}%
      {%
      \svne@readline\svn@temp
      \svne@readline\svn@temp
      \svne@readline\svn@temp
      \svne@readline\svn@temp
      \svne@readline\svn@temp
      \svne@readline\svn@temp
      \svne@readline\svne@date
      \svne@readline\svne@rev
      \svne@readline\svne@author
      %\@onelevel@sanitize\svne@date
      \svne@scandate{\svne@date}%
      \edef\svne@url{\svne@baseurl/\svne@name}%
      \svne@handleentry
      }{}%
    \svne@gobblerest
    \svne@parseentries
    }%
  }%
}
```

\svne@handleentry

This macro is called for every entry except the first one which stands for the directory. The VC data is located in the following macros: \svne@name, \svne@date, \svne@rev, \svne@author, \svne@url.

This implementation sets the correct keywords and calls the update macro to emulate the behaviour of \svnidlong. Then the \svne@endread macro is used to stop the file reading.

```
\def\svne@handleentry{%
  \svn@ifequal{\svne@name}{\svnfilefname}%
    {%
      \svnkwdef{Filename}{\svne@name}%
      \svnkwdef{Date}{\svne@date}%
      \svnkwdef{Revision}{\svne@rev}%
      \svnkwdef{Author}{\svne@author}%
      \svnkwdef{HeadURL}{\svne@url}%
      \@svn@updateid{\svne@rev}{\svne@date}{\svne@author}{\svne@url}%
      \svne@endread
    }{}%
}%
```

\svnegetfile

#1: file path

```
\def\svnegetfile#1{%
  \begingroup
    \svn@getfilename{#1}%
    \edef\svnfilefname{\svnfilefname}%
    \@onelevel@sanitize\svnfilefname
    \svne@parseentriesfile{\svnfiledir .svn/}%
    \svne@parseentriesfile{\svnfiledir _svn/}%
  \endgroup
}
```

Load keywords of main document at begin of the document body if option is set to 'all'.

```
\if@svnmulti@autokwall
\AtBeginDocument{%
    \svnegetfile{\jobname.\svn@mainext}%
}
\fi
```

```
\fi
```

7.5 Timezone macros

\svntimezone

\svnfiletimezone

\svncgtimezone

These macros return the global, file-local and current group time zones, respectively. Since v1.4 the minute part is returned as well and the macro removes manually added 00 after it to support older documents.

```
\def\svntimezone{\svntimezonehour\svntimezoneminute\svn@gobblezeros}
\def\svnfiletimezone{\svnfiletimezonehour\svnfiletimezoneminute\svn@gobblezeros}
\def\svncgtimezone{\svncgtimezonehour\svncgtimezoneminute}
```

\svn@gobblezeros

\svn@gobblezeros@

This two cascaded macros remove a trailing 00 and are used by \svnfiletimezone and \svntimezone.

```
\def\svn@gobblezeros{%
  \futurelet\svn@nextchar\svn@gobblezeros@
}
\def\svn@gobblezeros@{%
  \let\@tempa=\relax
  \def\@tempb{0}%
  \ifx0\svn@nextchar
    \let\@tempa=\@gobbletwo
  \fi
  \@tempa
}
```

\svntime

\svnfiletime

\svncgtime

This macros simple use the hour, minute and second macros.

```
\def\svntime{\svnhour:\svnminute:\svnsecond}
\def\svnfiletime{\svnfilehour:\svnfileminute:\svnfilesecond}
\def\svncgtime{\svncghour:\svncgminute:\svncgsecond}
```

7.6 *Today* macros

These macros use the \today macro to typeset the current date using the local language settings. Thanks and credit goes to Manuel Pégourié-Gonnard for suggesting this feature and for providing the code.

\svntoday

```
\newcommand*{\svntoday}{%
  \begingroup
    \year\svnyear \month\svnmonth \day\svnday
    \relax \today
  \endgroup
}
```

\svnfiletoday

```
\newcommand*{\svnfiletoday}{%
  \begingroup
    \year\svnfileyear \month\svnfilemonth \day\svnfileday
    \relax \today
  \endgroup
}
```

\svncgtoday

```
\newcommand*{\svncgtoday}{%
  \@ifundefined{svng@\svn@cg @year}{??}{%
    \begingroup
      \year\svncgyear \month\svncgmonth \day\svncgday
      \relax \today
    \endgroup
  }%
}%
```

7.7 Id macros

7.7.1 Normal Id

\svnid

Calls \svnkwsave with \@svnidswtrue so that the Id keyword will be parsed at the end of \svnkwsave.

```
\newcommand*{\svnid}{%
  \@svnidswtrue
  \svnkwsave
}
\newif\if@svnidsw
\@svnidswfalse
```

\svn@scanId

#1: file name
#2: revision
#3: date (YYYY-MM-DD)
#4: time (HH:MM:SSZ)
#5: author (username)

Scans svn Id (after it got parsed by \svnkwsave). Awaits only Id value without leading 'Id:' and a trailing \relax as end marker. It calls \@svn@scandate to extract the date information and \@svn@updateid to update global Id values and also sets the appropriate keywords.

```
\def\svn@scanId#1 #2 #3 #4 #5\relax{%
```

```
  \@svn@scandate{#3 #4}%
  \@svn@updateid{#2}{#3 #4}{#5}{#1}%
  \svnkwdef{Filename}{#1}%
  \svnkwdef{Date}{#3 #4}%
  \svnkwdef{Revision}{#2}%
  \svnkwdef{Author}{#5}%
}
```

\@svn@updateid

#1: rev
#2: date
#3: author (username)
#4: url

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

```
\def\@svn@updateid#1#2#3#4{%
  \xdef\svnfilerev{#1}%
  \ifsvn@modified
    \global\let\ifsvnfilemodified\@firstoftwo
  \else
    \global\let\ifsvnfilemodified\@secondoftwo
  \fi
  \xdef\svnfiledate{#2}%
  \xdef\svnfileauthor{#3}%
  \xdef\svnfileurl{#4}%
  \svn@getfilename\svnfileurl%
```

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
    \ifnum\@svn@rev<\svnfilerev
      \xdef\@svn@rev{\svnfilerev}%
      \xdef\@svn@modified{\ifsvnfilemodified{@firstoftwo}{@secondoftwo}}%
      \xdef\@svn@date{\svnfiledate}%
      \xdef\@svn@author{\svnfileauthor}%
      \xdef\@svn@year{\svnfileyear}%
      \xdef\@svn@month{\svnfilemonth}%
      \xdef\@svn@day{\svnfileday}%
      \xdef\@svn@hour{\svnfilehour}%
      \xdef\@svn@minute{\svnfileminute}%
      \xdef\@svn@second{\svnfilesecond}%
      \xdef\@svn@timezonehour{\svnfiletimezonehour}%
      \xdef\@svn@timezoneminute{\svnfiletimezoneminute}%
      \xdef\@svn@url{\svnfileurl}%
      \xdef\@svn@fname{\svnfilefname}%
    \fi

    \if@svnmulti@groups
      \ifx\svn@g\empty\else
        \svn@updategroup{\svn@g}%
      \fi
      \if@svnmulti@subgroups
        \ifsvnsubgroups
          \svn@updategroup{\svn@filedir\svn@filebase}%
        \fi
      \fi
    \fi
  \fi
}

\def\@svncg@save#1#2{%
  \expandafter\xdef\csname @svng@\svn@g @#1\endcsname{#2}%
}

```

7.7.2 Long Id

\svnidlong

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

```
\newcommand{\svnidlong}{%
  \svnkwdef{URL}{}%
  \svnkwdef{Date}{}%
  \svnkwdef{Revision}{0}%
  \svnkwdef{Author}{}%
```

The catcodes are changed by \svn@catcodes to allow TEX-special characters inside the keywords. The braces { } are changed to allow comments between the arguments. \svnidlong@readargsfull is called to read the arguments.

```
  \begingroup
    \if@svnmulti@verbatim
      \svn@catcodes
      \catcode`\{=12
      \catcode`\}=12
    \else
      \def\svnidlong@readargsfull{\svnidlong@readargs}%
    \fi
    \svnidlong@readargsfull
}
```

\svn@catcodes

Changes all TEX-special character to category “other”. The newline aka return is changed to category “ignore” so line breaks are not taken as part of the verbatim arguments.

```
\if@svnmulti@verbatim
\def\svn@catcodes{%
```

```
  \let\do\@makeother
  \dospecials
  \catcode`\^^M9
  \catcode`\ 10
  \catcode`\{1
  \catcode`\}2
}
\else
  \def\svn@catcodes{}
\fi
```

\svnidlong@readargsfull

#1: some text, ignored
#2: Keyword 1
#3: some text, ignored
#4: Keyword 2
#5: some text, ignored
#6: Keyword 3
#7: some text, ignored
#8: Keyword 4

Reads all four arguments of \svnidlong and passes them to \svnidlong@readargs. The normal argument braces are changed to category "other" and put into the macros parameter text to remove all code between them. This is done to avoid problems with comments direct after one of the arguments. Because the braces are now non-special the parentheses are made to a local replacement.

```
\if@svnmulti@verbatim
\begingroup
\catcode`\{=12\catcode`\}=12
\catcode`\(=1\catcode`\)=2
\gdef\svnidlong@readargsfull#1{#2}#3{#4}#5{#6}#7{#8}(%
 \svnidlong@readargs(#2)(#4)(#6)(#8)%
)
\endgroup
\fi
```

\svnidlong@readargs

#1: Keyword 1
#2: Keyword 2
#3: Keyword 3
#4: Keyword 4

Calls sub macro for all four arguments and ends the catcode changes made by \svnidlong.

```
\def\svnidlong@readargs#1#2#3#4{%
    \svnkwsave@read #1\relax
    \svnkwsave@read #2\relax
    \svnkwsave@read #3\relax
    \svnkwsave@read #4\relax
```

```
  \endgroup
```

Now the update macros for date and id are called.

```
  \ifx\svnkwDate\empty\else
    \@svn@scanlongdate{\svnkwDate}%
  \fi
  \@svn@updateid{\svnkw{Revision}}{\svnkw{Date}}%
  {\svnkw{Author}}{\svnkw{URL}}%
  \ignorespaces
}%
```

7.8 KeyWord Macros

\svnkwsave

Enabled verbatim mode and uses a sub macro to read the arguments afterwards.

```
\def\svnkwsave{%
  \begingroup
    \svn@catcodes
    \svnkwsave@readargs
}
```

\svnkwsave@readargs

#1: $kw: value$

Reads full argument, calls parse submacro and ends catcode changes. If \svnkwsave was called by \svnid scans the id keyword by calling the scan macro.

```
\gdef\svnkwsave@readargs#1{%
    \svnkwsave@read#1\relax
  \endgroup
  \if@svnidsw
    \ifx\svnkwId\empty\else
      \expandafter
      \svn@scanId\svnkwId\relax
      \@svnidswfalse
    \fi
  \fi
  \ignorespaces
}
```

\svnkwsave@read

#1: keyword line without surrounding $ $

Reads the full keyword and strips the dollars.

```
\begingroup
\if@svnmulti@verbatim
\catcode‘\$=12
\fi
\gdef\svnkwsave@read $#1$\relax{%
  \svn@checkcolon#1:\relax
}
\endgroup
```

\svnkwsave@parse

#1: key
#2: value

Parse the keyword and save it away.

```
\begingroup
\catcode`\$=11
\gdef\svnkwsave@parse$#1:#2${%
  \expandafter\xdef\csname svnkw#1\endcsname{#2}%
}%
\endgroup
```

\svnkwdef

#1: key
#2: value

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@

#1: key
#2: value

This macro defines the second argument under \svnkw⟨*1st 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@Rev

\svnkwdef@Author

\svnkwdef@Date

\svnkwdef@URL

#1: value

‘Setter’-macros for single keywords, used by \svnkwdef.

These are needed to have have a common value for all alternative keyword names ala Rev, Revision, LastChangedRevision.

The keywords Author and Date are just calling \svnkwdef@ with a fixed first argument. For the revision the value is checked if empty and then a 0 is substituted. Also a temp counter is used to strip any trailing characters like 'M' which indicate an exported and modified file.

```
\def\svnkwdef@Rev#1{%
  \svn@ifempty{#1}%
    {\svnkwdef@{Rev}{0}}%
    {%
     \afterassignment\svnkwdef@Rev@
     \@tempcnta=#1\relax
    }%
}
\def\svnkwdef@Rev@#1\relax{%
  \svnkwdef@{Rev}{\the\@tempcnta}%
  \def\svn@temp{#1}%
  \if M\svn@temp\relax
    \global\svn@modifiedtrue
  \else
    \if *\svn@temp\relax
      \global\svn@modifiedtrue
    \else
      \global\svn@modifiedfalse
    \fi
  \fi
}
\def\svnkwdef@Author#1{\svnkwdef@{Author}{#1}}
\def\svnkwdef@Date#1{\svnkwdef@{Date}{#1}}
\def\svnkwdef@URL#1{\svnkwdef@{HeadURL}{#1}}
```

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

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

and then for reading.

```
\def\svnkwRevision{\svnkwRev}
\def\svnkwLastChangedRevision{\svnkwRev}
\def\svnkwLastChangedBy{\svnkwAuthor}
\def\svnkwLastChangedDate{\svnkwDate}
\def\svnkwURL{\svnkwHeadURL}
```

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}
\svnkwdef{Date}{}
```

```
\svnkwdef{Author}{}
\svnkwdef{Filename}{}
\svnkwdef{HeadURL}{}
```

\svnkw

#1: keyword name

Macro to get keyword value. Just calls \svnkw⟨*ARGUMENT*⟩ where 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]{%
  \@ifundefined{svnkw#1}%
    {\PackageWarning{svn-multi}{SVN keyword '#1' not defined (typo?)}}%
    {\csname svnkw#1\endcsname}%
}%
```

7.9 Keyword check and strip macros

The following macros are used to test whether the given keywords are fully expanded or not. Subversion supports unexpanded keywords as input with or without colon and with or without trailing space(s), i.e. a: `$KW$`, b: `$KW:$` or c: `$KW: $`. To avoid LaTeX syntax errors in this pre-commit state the keyword is checked by the following macros. Unexpanded keywords result in an empty value. Also leading and trailing spaces are removed.

\svn@checkcolon

#1: key

#2: potential value, might be empty

Checks if the keyword contains a colon. It is called by \svnkwsave@read with a trailing :\relax so that #2 will be empty if there is no earlier colon or will hold the value with this trailing colon otherwise. The first case means that the keyword is unexpanded without colon (case a) which leads to an empty value. In the second case \svn@stripcolon is called to strip the colon and surrounding spaces. The final value is returned by \svn@value.

```
\def\svn@checkcolon#1:#2\relax{%
  \svn@ifempty{#2}%
    {\svnkwdef{#1}{}}%
    {\svn@stripcolon#2\relax\svnkwdef{#1}{\svn@value}}%
}
```

\svn@stripcolon

#1: potential value

Strips the previous added colon (for \svn@checkcolon). The remaining argument is checked if it's empty (case b) or only a space (case c). Otherwise the keyword is expanded and \svn@stripspace is called to strip the spaces.

```
\def\svn@stripcolon#1:\relax{%
  \svn@ifempty{#1}%
```

```
    {\gdef\svn@value{}}%
    {\svn@ifequal{#1}{ }%
      {\gdef\svn@value{}}%
      {\svn@stripspace#1\relax\relax}%
    }%
}
```

\svn@stripspace

#1: first character

#2: rest of string

Strips leading space if present and calls \svn@striptrailingspace to strip the trailing space.

```
\def\svn@stripspace#1#2\relax{%
  \svn@ifequal{#1}{ }%
    {\gdef\svn@value{#2}}%
    {\svn@striptrailingspace#1#2\relax}%
}
```

\svn@striptrailingspace

#1: string

Strips trailing space using the macros parameter text. Must be called with \relax as end marker.

```
\def\svn@striptrailingspace#1 \relax{%
  \gdef\svn@value{#1}%
}
```

\svn@gdefverb

#1: macro

```
\def\svn@gdefverb#1{%
  \begingroup
    \def\svn@temp{#1}%
    \begingroup
      \if@svnmulti@verbatim
        \svn@catcodes
      \fi
      \svn@gdefverb@
}
```

\svn@defverb@

#1: verbatim stuff

```
\def\svn@gdefverb@#1{%
    \endgroup
    \expandafter\gdef\svn@temp{#1}%
  \endgroup
}
```

\svn@namegdefverb

#1: macro name

```
\def\svn@namegdefverb#1{%
  \begingroup
    \expandafter\def
    \expandafter\svn@temp
    \expandafter{\csname #1\endcsname}%
    \begingroup
      \if@svnmulti@verbatim
        \svn@catcodes
      \fi
      \svn@gdefverb@
}
```

7.10 Date Macros

\@svn@scandate

#1: date

Scans data information in Id keyword and saves them in macros.

```
\def\@svn@scandate#1{\@svn@scandate@#1\relax}

\def\@svn@scandate@#1-#2-#3 #4:#5:#6#7#8\relax{%
  \gdef\svnfileyear{#1}%
  \gdef\svnfilemonth{#2}%
  \gdef\svnfileday{#3}%
  \gdef\svnfilehour{#4}%
  \gdef\svnfileminute{#5}%
  \gdef\svnfilesecond{#6#7}%
  \gdef\svnfiletimezonehour{+00}%
  \gdef\svnfiletimezoneminute{00}% #8 always 'Z' for Zulu-time (UTC)
}
```

\@svn@scanlongdate

#1: Year
#2: Month
#3: Day
#4: Hour
#5: Minute
#6: Second
#7: Timezone
#8: Date description string (ignored)

Scans date information in Date keyword and saves them in macros.

```
\def\@svn@scanlongdate#1{\expandafter\@svn@scanlongdate@#1\relax}
%
\def\@svn@scanlongdate@#1-#2-#3 #4:#5:#6 #7 #8\relax{%
  \gdef\svnfileyear{#1}%
  \gdef\svnfilemonth{#2}%
  \gdef\svnfileday{#3}%
  \gdef\svnfilehour{#4}%
  \gdef\svnfileminute{#5}%
  \gdef\svnfilesecond{#6}%
  \@svn@parsetimezone#7\relax%
}
```

`\@svn@parsetimezone`

`#1:` sign (+/-)
`#2:` hour first digit
`#3:` hour second digit
`#4:` minute first digit
`#5:` minute second digit

Scans timezone and splits hour and minute part.

```
\def\@svn@parsetimezone#1#2#3#4#5\relax{%
  \gdef\svnfiletimezonehour{#1#2#3}%
  \gdef\svnfiletimezoneminute{#4#5}%
}
```

`\svnpdfdate`

Returns date in a format needed for `\pdfinfo`.

```
\def\svnpdfdate{%
  \svnyear\svnmonth\svnday
  \svnhour\svnminute\svnsecond\svntimezonehour'\svntimezoneminute'%
}
```

7.11 Mainfile Makros

`\svnsetmainfile`

Saves the current `HeadURL` and `Filename` keywords to macros. Will be called automatically in the preamble.

```
\newcommand{\svnsetmainfile}{%
  \xdef\svnmainurl{\svnfileurl}%
  \xdef\svnmainfilename{\svnfilefname}%
}
\AtBeginDocument{\svnsetmainfile}
```

7.12 Register and FullName Macros

`\svnRegisterAuthor`

`#1:` author username
`#2:` Full Name

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

```
\newcommand{\svnRegisterAuthor}[2]{%
```

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

\svnFullAuthor

\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}%
}%
```

\svnFullAuthor@star

`#1:` username

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

```
\def\svnFullAuthor@star*#1{%
  \edef\svn@temp{#1}%
  \svnFullAuthor@{\svn@temp}{~(\svn@temp)}%
}%
```

\svnFullAuthor@normal

`#1:` username

```
\def\svnFullAuthor@normal#1{%
  \edef\svn@temp{#1}%
  \svnFullAuthor@{\svn@temp}{}%
}%
```

\svnFullAuthor@

`#1:` username

`#2:` previous defined trailing string

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

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

\svnRegisterRevision

`#1:` revision number

`#2:` tag name

Saves the revision's name or tag by defining `svn@revision@`⟨*revisionnumber*⟩ to it.

```
\newcommand{\svnRegisterRevision}[2]{%
```

```
  \expandafter\def\csname svn@revision@#1\endcsname{#2}%
}
```

\svnFullRevision

\svnFullRevision*

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

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

\svnFullRevision@star

`#1:` revision number

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

```
\def\svnFullRevision@star*#1{%
  \edef\svn@temp{#1}%
  \svnFullRevision@{\svn@temp}{~(r\svn@temp)}%
}
```

\svnFullRevision@normal

`#1:` revision number

```
\def\svnFullRevision@normal#1{%
  \edef\svn@temp{#1}%
  \svnFullRevision@{\svn@temp}{}%
}
```

\svnFullRevision@

`#1:` revision number

`#2:` previous defined trailing string

`svnFullRevision@` now sets the revision name. Note that `#2` is empty when the normal version is called.

```
\def\svnFullRevision@#1#2{%
  \@ifundefined{svn@revision@#1}%
    {Revision #1}%
    {\csname svn@revision@#1\endcsname #2}%
}
```

7.13 Input File Name

The `fink` package is used to get the input file names. AtBegin/AtEnd hooks are installed which will be used later.

```
\if@svnmulti@filehooks
```

Load fink package and check if all needed macros are provided.

```
\RequirePackage{fink}[2008/02/27]
\begingroup
\def\svn@finkerror{%
\PackageError{svn-multi}{Your installed version of the 'fink' package does not
provide the needed macros. It is either too old or too new.
Try a different version, e.g. v2.1.1 from 2008/02/27}{}%
\let\svn@finkerror\relax
}
\@ifundefined{finkpath}{\svn@finkerror}{}%
\@ifundefined{finkdir}{\svn@finkerror}{}%
\@ifundefined{finkbase}{\svn@finkerror}{}%
\@ifundefined{fink@prepare}{\svn@finkerror}{}%
\@ifundefined{fink@restore}{\svn@finkerror}{}%
\@ifundefined{fnk@maindir}{\svn@finkerror}{}%
\@ifundefined{fnk@mainext}{\svn@finkerror}{}%
\endgroup
```

\svn@removedotslash

#1: string (e.g. file path) which might start with ./

Removes leading './' from given macro (holding a directory path). Awaits a macro as argument which is redefined inside the current group!

```
\def\svn@removedotslash#1{%
  \def\svn@removedotslash@##1##2##3\relax{%
    \svn@ifequal{./}{##1##2}%
      {\def\next{\svn@removedotslash@##3\empty\empty\empty\relax}}%
      {\xdef#1{##1##2##3}\let\next\relax}%
    \next
  }%
  \expandafter\svn@removedotslash@#1\empty\empty\empty\relax
}
```

Init values for file name macros.

```
\let\svn@mainext\fnk@mainext
\let\svn@maindir\fnk@maindir
\svn@removedotslash\svn@maindir
\edef\svn@filebase{\jobname}%
\edef\svn@fileext{\svn@mainext}%
\edef\svn@filedir{\svn@maindir}%
```

Filename and -path are build using the other macros:

```
\def\svn@filename{\fink@file\svn@filebase\svn@fileext}%
\def\svn@filepath{\svn@filedir\svn@filename}%
```

\svnmulti@begininputfilehook

This hook is installed in the \fink@prepare macro from the fink package which will be executed at the begin of a input file. The file name and path are not yet in \finkpath etc. but in \fink@nextpath.

```
\def\svnmulti@begininputfilehook{}
```

```
\message{Package svn-multi: patching macro ’\string\fink@prepare’ from the
’fink’ package!}%
\let\svnmulti@fink@prepare\fink@prepare
\renewcommand*{\fink@prepare}[1]{%
  \svnmulti@fink@prepare{#1}%
  \svn@pushfilestack
  \if@svnmulti@groups
    \svn@ifequal{\svn@filepath}{\jobname.\svn@mainext}%
      {\xdef\svn@pg{\svn@g}}%
      {\xdef\svn@pg{\svn@filedir\svn@filebase}}%
  \fi
  \xdef\svn@filebase{\fink@nextbase}%
  \xdef\svn@fileext{\fink@nextext}%
  \xdef\svn@filedir{\fink@nextdir}%
  \svn@removedotslash\svn@filedir
  \svnmulti@begininputfilehook
}%
```

\svnmulti@endinputfilehook

This hook is installed in the `\fink@restore` macro from the `fink` package which will be executed at the end of a input file. The file path `\finkpath` etc. is still valid.

```
\def\svnmulti@endinputfilehook{}
\message{Package svn-multi: patching macro ’\string\fink@restore’ from the
’fink’ package!}%
\let\svnmulti@fink@restore\fink@restore
\def\fink@restore#1{%
  \svnmulti@endinputfilehook
  \svnmulti@fink@restore{#1}%
  \svn@popfilestack
  \xdef\svn@filebase{\finkbase}%
  \xdef\svn@fileext{\finkext}%
  \xdef\svn@filedir{\finkdir}%
  \svn@removedotslash\svn@filedir
}%
```

\svnmulti@atbegininputfile

This macro adds the argument to the end of the `\svnmulti@begininputfilehook`.

```
\def\svnmulti@atbegininputfile{%
  \g@addto@macro\svnmulti@begininputfilehook
}
```

\svnmulti@atendinputfile

This macro adds the argument to the *begin* of the `\svnmulti@endinputfilehook`. This ensures that code added first is more at the end than code added later. The code below was adapted from the definition of the LaTeX2e macro `\g@addto@macro` which was used above.

```
\long\def\svnmulti@atendinputfile#1{%
```

```
  \begingroup
    \@temptokena\expandafter{\svnmulti@endinputfilehook}%
    \toks@{#1}%
    \xdef\svnmulti@endinputfilehook{\the\toks@\the\@temptokena}%
  \endgroup
}
```

```
\def\svn@filestack{{}}

\def\svn@pushfilestack{%
  \xdef\svn@filestack{{%
    {\svnfilerev}%
    {\svnfiledate}%
    {\svnfileauthor}%
    {\svnfileyear}%
    {\svnfilemonth}%
    {\svnfileday}%
    {\svnfilehour}%
    {\svnfileminute}%
    {\svnfilesecond}%
    {\svnfiletimezonehour}%
    {\svnfiletimezoneminute}%
    {\svnfileurl}%
    {\svnfilefname}%
    {\svn@g}%
    {\svn@pg}%
    {\ifsvnfilemodified{@firstoftwo}{@secondoftwo}}%
  }\svn@filestack}%
}

\def\svn@restorefilekws#1#2\relax{%
  \svn@restorefilekws@#1\empty
  \empty \empty \empty \empty
  \empty \empty \empty \empty
  \empty \empty \empty \empty \empty
  \svn@ifempty{#2}%
    {\gdef\svn@filestack{{}}}%
    {\gdef\svn@filestack{#2}}%
}
\def\svn@restorefilekws@#1#2#3#4#5#6#7#8#9{%
  \gdef\svnfilerev{#1}%
  \gdef\svnfiledate{#2}%
  \gdef\svnfileauthor{#3}%
  \gdef\svnfileyear{#4}%
  \gdef\svnfilemonth{#5}%
  \gdef\svnfileday{#6}%
  \gdef\svnfilehour{#7}%
  \gdef\svnfileminute{#8}%
  \gdef\svnfilesecond{#9}%
  \svn@restorefilekws@@
}

\def\svn@restorefilekws@@#1#2#3#4#5#6#7{%
  \gdef\svnfiletimezonehour{#1}%
  \gdef\svnfiletimezoneminute{#2}%
  \gdef\svnfileurl{#3}%
  \gdef\svnfilefname{#4}%
  \gdef\svn@g{#5}%
  \gdef\svn@pg{#6}%
  \expandafter\global\expandafter\let
  \expandafter\ifsvnfilemodified\csname#7\endcsname%
}

\def\svn@popfilestack{%
  \ifx\svn@filestack\empty
    \PackageWarning{svn-multi}{Underflow of file keyword stack!}%
  \else
    \svn@ifequal{\svn@filestack}{{}}%
      {\PackageWarning{svn-multi}{Underflow of file keyword stack!}}%
      {\expandafter\svn@restorefilekws\svn@filestack\relax}%
  \fi
}

%
```

```
\fi
```

7.14 Keyword Group Macros

These macros implement the user interface for the keyword group functionality introduced with v2.0.

The list of keyword groups `\svn@glist` is initial set empty and will be filled by `\svngroup`.

```
\if@svnmulti@groups
\let\svn@glist=\empty
```

`\svngroup`

`#1:` group name

Saves the group to `\svn@g` and initiates `\svn@g@`⟨*group name*⟩`@rev` and `\@svn@g@`⟨*group name*⟩`@rev` if this is the first time the group got used.

The current group symbol '`*`' is invalid here because there is no way to change to a current group.

```
\def\svngroup#1{%
  \svn@ifequal{#1}{*}%
    {\PackageError{svn-multi}%
      {The group name '*' is invalid for '\string\svngroup'}{}{}%
    }{}%
  \xdef\svn@g{#1}%
  \let\svn@pg\svn@g
  \ifx\svn@g\empty\else%
```

Only initialise the group at first usage:

```
    \expandafter
    \ifx\csname @svng@\svn@g @rev\endcsname\relax%
      \svn@initgroup{\svn@g}%
```

Now save new group to list. The list is checked if its empty to avoid an unwanted leading comma.

```
      \ifx\svn@glist\empty
        \xdef\svn@glist{#1}%
      \else
        \xdef\svn@glist{\svn@glist,#1}%
      \fi
    \fi
  \fi
}
```

`\thesvngroup`

Returns the current group name to the user.

```
\def\thesvngroup{\svn@g}
```

`\svnsetcg`

`#1:` group name

Defines `\svn@cg` to the given argument or to `\svn@g` if the argument was ‘`*`’.

```
\def\svnsetcg#1{%
  \svn@ifequal{#1}{*}%
    {\def\svn@cg{\svn@g}}%
    {\def\svn@cg{#1}}%
}
```

`\svncg@def`

`#1:` key name, e.g. ‘rev’, ‘date’

Defines a `\svncgXXX` macro, e.g. `svncgrev`, which returns the requested keyword values of the current keyword group.

```
\def\svncg@def#1{%
  \expandafter
  \def\csname svncg#1\endcsname{%
    \@ifundefined{svng@\svn@cg @#1}{??}{%
    \csname svng@\svn@cg @#1\endcsname}%
  }%
}
```

\svncgrev

\svncgdate

\svncgauthor

\svncgyear

\svncgmonth

\svncgday

\svncghour

\svncgminute

\svncgsecond

\svncgtimezonehour

\svncgtimezoneminute

\svncgurl

\svncgfname

Define all `\svncgXXX` macros by calling `\svncg@def` in a for loop.

```
\@for\@tempa:=%
  rev,author,date,year,month,day,hour,minute,second,%
  timezonehour,timezoneminute,url,fname%
\do{%
  \expandafter\svncg@def\expandafter{\@tempa}%
}
```

\thesvncg

Simply return the internal macro.

```
\def\thesvncg{\svn@cg}
```

\svng

`#1:` group name
`#2:` keyword name

Simply returns `svng@#1@#2` if defined, '??' otherwise.

```
\def\svng#1#2{%
  \@ifundefined{svng@\svn@temp @#2}%
    {??}%
    {\csname svng@\svn@temp @#2\endcsname}%
}
```

\svn@addfiletogroup

`#1:` file name
`#2:` group name

Adds the given file to the given group. If the group list doesn't exist yet it is initialised. A extra macro for each file is used to remember that the file is already in the group. This could be avoided using a list search.

This is an internal macro so no '*' substitution for the group name.

```
\def\svn@addfiletogroup#1#2{%
  \expandafter
  \ifx\csname @svng@#2@files@#1\endcsname\relax%
    \expandafter\gdef\csname @svng@#2@files@#1\endcsname{1}%
    %
    \@ifundefined{@svng@#2@files}%
      {\expandafter\xdef\csname @svng@#2@files\endcsname{#1}}%
      {\expandafter\xdef\csname @svng@#2@files\endcsname{%
        \csname @svng@#2@files\endcsname,#1%
       }%
      }%
  \fi
}
```

The input files are added to the list of the current group at their begin to have them before the included graphics and other external files. Special care is taken to not re-initialise the main file which could happen in some special cases (e.g. `\lstinputlisting{\jobname .tex}`).

```
\svnmulti@atbegininputfile{%
  \svn@ifequal{\svn@filepath}{\svn@maindir\jobname.\svn@mainext}%
    {}%
    {\svn@initfile}%
  \svn@ifequal{\svn@fileext}{\svn@mainext}%
    {\svn@addfiletogroup{\svn@filedir\svn@filebase}{\svn@pg}}{}%
  \svn@ifequal{\svn@fileext}{sty}%
    {\svn@addfiletogroup{\svn@filedir\svn@filebase}{\svn@pg}}{}%
  \svn@ifequal{\svn@fileext}{cls}%
    {\svn@addfiletogroup{\svn@filedir\svn@filebase}{\svn@pg}}{}%
  \svn@addfiletogroup{\svn@filepath}{\svn@filedir\svn@filebase}%
}
```

\svn@writegroup

`#1:` group name

Writes group to `\svn@write` file.

```
\def\svn@writegroup#1{%
  \def\svn@writekw##1{%
   \immediate\write\svn@write{%
     \noexpand\@namedef{svng@#1@##1}{\csname @svng@#1@##1\endcsname}%
   }%
  }%
  \svn@writekw{rev}%
  \svn@writekw{date}%
  \svn@writekw{author}%
  \svn@writekw{year}%
  \svn@writekw{month}%
  \svn@writekw{day}%
  \svn@writekw{hour}%
  \svn@writekw{minute}%
  \svn@writekw{second}%
  \svn@writekw{timezonehour}%
  \svn@writekw{timezoneminute}%
  \@ifundefined{@svng@#1@files}{}{%
    \immediate\write\svn@write{%
      \noexpand
      \svn@namegdefverb{svng@#1@files}{\csname @svng@#1@files\endcsname}%
    }%
  }%
  \immediate\write\svn@write{%
    \noexpand
    \svn@namegdefverb{svng@#1@url}{\csname @svng@#1@url\endcsname}^^J%
    \noexpand
    \svn@namegdefverb{svng@#1@fname}{\csname @svng@#1@fname\endcsname}^^J%
  }%
}
```

\svn@writeallgroups

`#1:` macro holding a list of groups

```
\def\svn@writeallgroups#1{%
  \begingroup
    \ifx\relax#1\relax\else
      \@for\svn@temp:=#1\do{%
        \svn@ifvalidrev{@svng@\svn@temp @rev}%
          {%
            \expandafter
            \svn@cleanfilelist\csname @svng@\svn@temp @files\endcsname
            \svn@writegroup{\svn@temp}%
            \@ifundefined{@svng@\svn@temp @files}{}%
              {\expandafter\svn@writeallgroups
               \csname @svng@\svn@temp @files\endcsname
              }%
          }{}%
      }%
    \fi
  \endgroup
}
```

\svn@updategroup

`#1:` group name

Updates group with `\svnfile...` macro values.

```
\def\svn@updategroup#1{%
  \@ifundefined{@svng@#1@rev}%
    {\svn@initgroup{#1}}%
    {}%
  \expandafter
  \ifnum\csname @svng@#1@rev\endcsname<\svnfilerev
    \svn@gkwset{#1}{rev}{\svnfilerev}%
    \svn@gkwset{#1}{date}{\svnfiledate}%
    \svn@gkwset{#1}{author}{\svnfileauthor}%
    \svn@gkwset{#1}{year}{\svnfileyear}%
    \svn@gkwset{#1}{month}{\svnfilemonth}%
    \svn@gkwset{#1}{day}{\svnfileday}%
    \svn@gkwset{#1}{hour}{\svnfilehour}%
    \svn@gkwset{#1}{minute}{\svnfileminute}%
    \svn@gkwset{#1}{second}{\svnfilesecond}%
    \svn@gkwset{#1}{timezonehour}{\svnfiletimezonehour}%
    \svn@gkwset{#1}{timezoneminute}{\svnfiletimezoneminute}%
    \svn@gkwset{#1}{url}{\svnfileurl}%
    \svn@gkwset{#1}{fname}{\svnfilefname}%
  \fi
}
```

\svn@definegroup

`#1:` group name

Defines group value so that they are available for the user, e.g. instead of the internal `@svng@...` macros it sets the `svng@...` macros. This is done by calling `\svn@updategroup` with a modified version of `\svn@gkwset`.

```
\def\svn@definegroup#1{%
  \svn@gkwdef{#1}{rev}%
  \svn@gkwdef{#1}{date}%
  \svn@gkwdef{#1}{author}%
  \svn@gkwdef{#1}{year}%
  \svn@gkwdef{#1}{month}%
  \svn@gkwdef{#1}{day}%
  \svn@gkwdef{#1}{hour}%
  \svn@gkwdef{#1}{minute}%
  \svn@gkwdef{#1}{second}%
  \svn@gkwdef{#1}{timezonehour}%
  \svn@gkwdef{#1}{timezoneminute}%
  \svn@gkwdef{#1}{url}%
  \svn@gkwdef{#1}{fname}%
}
```

\svn@initgroup

`#1:` group name

Initialises group.

```
\def\svn@initgroup#1{%
  \svn@gkwset{#1}{rev}{\svn@revinit}%
  \svn@gkwset{#1}{date}{}%
  \svn@gkwset{#1}{author}{}%
  \svn@gkwset{#1}{year}{0000}%
  \svn@gkwset{#1}{month}{00}%
  \svn@gkwset{#1}{day}{00}%
  \svn@gkwset{#1}{hour}{00}%
  \svn@gkwset{#1}{minute}{00}%
  \svn@gkwset{#1}{second}{00}%
  \svn@gkwset{#1}{timezonehour}{+00}%
  \svn@gkwset{#1}{timezoneminute}{00}%
  \svn@gkwset{#1}{url}{}%
  \svn@gkwset{#1}{fname}{}%
}
```

\svn@gkwset

#1: group name
#2: keyword name
#3: value

Sets ⟨*value*⟩ for ⟨*keyword*⟩ in ⟨*group*⟩.

```
\def\svn@gkwset#1#2#3{%
  \expandafter
  \xdef\csname @svng@#1@#2\endcsname{#3}%
}
```

\svn@gkwdef

#1: group name
#2: keyword name

Defines `svng@...` macros used by the user macros to the value of the internal `@svng@...` macros.

```
\def\svn@gkwdef#1#2{%
  \expandafter
  \xdef\csname svng@#1@#2\endcsname{\csname @svng@#1@#2\endcsname}%
}
```

\svn@cleanfilelist

#1: macro holing a file list

Takes a macro which holds a file list and removes all files from the list which don't have a valid revision number.

```
\def\svn@cleanfilelist#1{
  \begingroup
    \def\svn@tmplist{}%
    \ifx\relax#1\relax\else
      \@for\svn@temp:=#1\do{%
        \expandafter\svn@ifvalidrev
        \expandafter{@svng@\svn@temp @rev}%
          {\edef\svn@tmplist{\svn@tmplist,\svn@temp}}%
          {}%
      }%
      \xdef#1{\expandafter\@gobble\svn@tmplist\empty}%
    \fi
  \endgroup
}
```

```
\fi
```

7.15 Files as extra groups

Macros which allow single files to be declared as extra groups so that their keywords can be accessed in the whole document like with normal groups. This special groups are not added to the list of groups.

A user-level switch is declared to enable or disable the automatic declaration of every file as own group. This causes \svnsubgroup to be called for all input files. The if macro is defined outside the \if@svnmulti@subgroups because \newif inside \if is not a good idea.

```
\newif\ifsvnsubgroups
\svnsubgroupsfalse
```

```
\if@svnmulti@subgroups
\svnsubgroupstrue
```

\svnsubgroup

User level and internal macro to declare the current file as extra group. It produces the current file path and calls \svn@subgroup. Creates two groups one with and one without the file extension. The one without holds the latest revision of all files included in this file.

```
\def\svnsubgroup{%
  \begingroup
    \svn@removedotslash\svn@filedir
    \svn@subgroup{\svn@filedir\svn@filebase}%
    \svn@subgroup{\svn@filepath}%
  \endgroup
}
```

\svn@subgroup

#1: file name

Macro to write a file as group to .svn file. After checking if the filename was not already written, the .svn file is checked if it is open and then the file keyword information is written.

```
\def\svn@subgroup#1{%
  \ifnum\svnfilerev=\svn@revinit\else
    \expandafter\ifx\csname svn@g@#1\endcsname\relax%
      \expandafter\gdef\csname svn@g@#1\endcsname{1}%
      \svn@updategroup{#1}%
    \fi
  \fi
}
```

\svnignoreextensions

#1: A comma separated list of file name extensions (without leading dot) to ignore for automatic \svnsubgroup.

A special macro is defined for all extensions. The existents of this macro is then tested later to check if this extension should be ignored.

```
\def\svnignoreextensions#1{%
  \@for\svn@temp:=#1\do{%
    \expandafter\def\csname svn@ignore@ext@\svn@temp\endcsname{}%
  }%
}
```

\svnconsiderextensions

#1: A comma separated list of file name extensions (without leading dot) to consider for automatic \svnsubgroup.

The special macro defined by \svnignoreextentions is deleted, i.e. \let to \relax.

```
\def\svnconsiderextensions#1{%
  \@for\svn@temp:=#1\do{%
  \expandafter\let\csname svn@ignore@ext@\svn@temp\endcsname\relax%
  }%
}
```

The following extensions are ignored by default.

```
\svnignoreextensions{aux,bbl,fd,enc,fls,glo,idx,ilg,ind,ist,%
lof,log,lot,out,svn,svt,svx,toc}
```

Check at the end of every input file if files should be extra groups and declare this file as group if its extension is not configured to be ignored.

```
\svnmulti@atendinputfile{%
  \if@svnmulti@subgroups
    \ifsvnsubgroups
      \expandafter\ifx\csname svn@ignore@ext@\svn@fileext\endcsname\relax
      \svnsubgroup
      \fi
    \fi
  \fi
}
```

The main file is added to the main base name (\jobname) subgroup here. This subgroup is added as first element to the active group at begin of the document body.

```
\if@svnmulti@subgroups
  \ifsvnsubgroups
    \svn@addfiletogroup{\jobname .\svn@mainext}{\jobname}%
    \svnsubgroup
  \fi
\fi
\AtBeginDocument{%
  \if@svnmulti@subgroups
    \ifsvnsubgroups
    \@ifundefined{@svng@\svn@g @files@\jobname}%
      {%
      \@namedef{@svng@\svn@g @files@\jobname}{1}%
      \@ifundefined{@svng@\svn@g @files}%
        {%
```

```
          \expandafter
          \xdef\csname @svng@\svn@g @files\endcsname{\jobname}%
        }%
        {%
          \expandafter
          \xdef\csname @svng@\svn@g @files\endcsname
            {\jobname,\csname @svng@\svn@g @files\endcsname}%
        }%
      }{}%
      \svnsubgroup
    \fi
  \fi
}
```

```
\fi
```

7.16 External Files

Macros to declare external files and load the keywords from `.svx` files generated by `svn-multi.pl`.

```
\if@svnmulti@external
```

`\svnexternalgroup`

`#1:` group name

Defines the default group of external files. The default is to always use the current group. An empty argument puts the external files in no group. A '`*`' switches back to always use the current group.

```
\if@svnmulti@groups
\def\svnexternalgroup#1{%
  \svn@ifequal{#1}{*}%
    {\def\svn@externalgroup{\svn@pg}}%
    {\def\svn@externalgroup{#1}}%
}
\def\svn@externalgroup{\svn@pg}
\else
\def\svn@externalgroup{}
\fi
```

`\svnexternal`

`#1:` group name

`#2:` list of filenames in `{ }`

```
\if@svnmulti@autokw
```

```
\newcommand*\svnexternal[2][]{%
  \svn@pushfilestack
  \svn@ifequal{#1}{*}%
    {\edef\svn@eg{\svn@pg}}%
    {\svn@ifempty{#1}%
      {\edef\svn@eg{\svn@externalgroup}}%
      {\edef\svn@eg{#1}}%
    }%
  \svne@@external#2\relax
  \svn@popfilestack
}

\def\svne@@external#1{%
  \ifx\relax#1\empty\else
    \svnegetfile{#1}%
    \begingroup\svn@externalfile{\svn@eg}{#1}%
    \expandafter\svne@@external
  \fi
}
```

```
\else
```

Writes the current input file path and its argument as arguments of `\@svnexternal` into the `.svn` file.

```
\newcommand*\svnexternal[2][]{%
  \if@filesw
    \svn@checkwrite
    \begingroup
      \svn@ifequal{#1}{*}%
        {\def\svn@temp{\svn@pg}}%
        {\svn@ifempty{#1}%
          {\def\svn@temp{\svn@externalgroup}}%
          {\def\svn@temp{#1}}%
        }%
      \immediate\write\svn@write{%
        \noexpand\@svnexternal[\svn@temp]{\svn@filepath}{#2}%
      }%
    \endgroup
  \fi
  \svn@inputsvx{\svn@filedir\svn@filebase}%
}
```

```
\fi
```

`\svnexternalpath`

`#1`: list of paths in `{ }`

Writes its argument as argument of `\@svnexternalpath` into the `.svn` file.

```
\def\svnexternalpath#1{%
  \if@filesw
    \svn@checkwrite
    \immediate\write\svn@write{%
      \noexpand\@svnexternalpath{#1}%
    }%
  \fi
}
```

\@svnexternal

\@svnexternalpath

Discards the argument(s). These macros and their arguments are only used by the external svn-multi.pl script.

```
\newcommand*\@svnexternal[3][]{}
\def\@svnexternalpath#1{}
```

\svnexternalfile

This macro is generated by svn-multi.pl and should not be used by the user. If files-as-group is enabled some special characters are disabled and the \svn@externalfile is called to read the file name. Otherwise the argument is simply removed.

```
\newcommand*\svnexternalfile[1][\svn@filedir\svn@filebase]{%
  \begingroup % TODO: maybe use \svn@catcodes
    \catcode`\_=12
    \catcode`\&=12
    \catcode`\^=12
    \catcode`\$=12
    \catcode`\#=12
    \svn@externalfile{#1}%
}
```

\svn@externalfile

#1: group name

#2: file name

Ends group which began in \svnexternalfile and calls the appropriate macros.

```
\def\svn@externalfile#1#2{%
  \endgroup
  \if@svnmulti@subgroups
    \ifsvnsubgroups
      \svn@ifequal{#1}{\svn@rg}%
        {\svn@addfiletogroup{#2}{\svn@filedir\svn@filebase}}%
        {\svn@addfiletogroup{#2}{#1}}%
      \svn@subgroup{#2}%
    \fi
  \fi
}
```

If **external** option is not enabled a placeholder macro is defined which simply ignores its argument.

```
\else
  \def\svnexternalfile#1{}%
\fi
```

7.17 Auto loading of .svx files

Auto loading of .svx files at the begin of \input or \include files using the \svnmulti@atbegininputfile macro. The macros \svn@addfiletogroup and

\svnsubgroup are used to do the actual work.

```
\if@svnmulti@autoload

\svnmulti@atbegininputfile{%
  \svn@ifequal{\svn@fileext}{tex}%
    {\svn@inputsvx{\svn@filedir\svn@filebase}}%
    {}%
}
```

The main .svx is loaded at the end of the package.

```
%%\AtEndOfPackage{%
\AtBeginDocument{%
  \svn@inputsvx{\jobname}%
}
```

```
\fi
```

7.18 Support for Graphic Packages

7.18.1 Common Code

```
\if@svnmulti@anygraphic
```

\svngraphicsgroup

#1: graphic group name

Defines the default group of graphics files. The default is empty which means the current group.

```
\def\svngraphicsgroup#1{%
  \svn@ifequal{#1}{*}%
    {\def\svn@graphicsgroup{\svn@pg}}%
    {\def\svn@graphicsgroup{#1}}%
}
\def\svn@graphicsgroup{\svn@externalgroup}
```

\svnignoregraphic

#1: file name/path

Ignores the given graphic file by defining a special macro.

```
\def\svnignoregraphic#1{%
  \expandafter\def\csname svn@ignoregraphic@#1\endcsname{}%
}
```

\svnconsidergraphic

#1: file name/path

Deletes the special ignore macro to consider the graphic again.

```
\def\svnconsidergraphic#1{%
  \expandafter\let\csname svn@ignoregraphic@#1\endcsname\relax%
}
```

```
\fi
```

7.18.2 Package graphics

Automatic declaration of all images included by `\includegraphics` from the graphics package as external files. We use the `\Gin@setfile` macro from that package which receives the image file name as third argument.

```
\if@svnmulti@graphics
\RequirePackage{graphics}[2006/02/20]
```

`\Gin@setfile`

`#1:` ??, not used
`#2:` ??, not used
`#3:` graphic file name/path

```
\message{Package svn-multi: patching macro ’\string\Gin@setfile’ from the
’graphics’ package!}%
\let\svnmulti@Gin@setfile\Gin@setfile
\renewcommand*{\Gin@setfile}[3]{%
  \expandafter\ifx\csname svn@ignoregraphic@#3\endcsname\relax%
    \svnexternal[\svn@graphicsgroup]{{#3}}%
  \fi
  \svnmulti@Gin@setfile{#1}{#2}{#3}%
}
```

```
\fi
```

7.18.3 Package pgf

The pgf macro `\pgf@declareimage` which is called by the user macro `\pgfdeclareimage` is used.

```
\if@svnmulti@pgfimages
\RequirePackage{pgf}[2008/01/15]
```

`\pgf@declareimage`

`#1:` ??, not used
`#2:` image label
`#3:` ??, not used

```
\message{Package svn-multi: patching macro ’\string\pgf@declareimage’ and will
patch generated macros ’\string\pgf@image@<name>!’ from the ’pgf’ package!}%
\let\svnmulti@pgf@declareimage\pgf@declareimage
\renewcommand*{\pgf@declareimage}[3][]{%
  \svnmulti@pgf@declareimage[#1]{#2}{#3}%
```

At this point the used image filename is defined by `\pgf@filename` and the image itself is defined by `\pgf@image@#2!` which is a `\let` copy of temporary `\pgf@image`. An own copy of this is created and the old name `\pgf@image@#2!` is used to execute `\svnexternal` every time the image is included using `\pgfuseimage`.

```
  \ifx\pgf@filename\empty\else
    \expandafter\ifx\csname svn@ignoregraphic@\pgf@filename\endcsname\relax%
```

```
      \expandafter\global\expandafter%
      \let\csname svnmulti@pgf@image@#2!\endcsname=\pgf@image%
      \expandafter\xdef\csname pgf@image@#2!\endcsname{%
        \noexpand\svnexternal[\noexpand\svn@graphicsgroup]{{\pgf@filename}}%
        \csname svnmulti@pgf@image@#2!\endcsname
      }%
    \fi
  \fi
}
```

```
\fi
```

7.19 Table of Revisions

```
\if@svnmulti@table
\ifx\tableofcontents\relax\else
```

`\svnrevisionsname`

Simple definition for now. Language support over 'babel's `\languagename` possible.

```
\def\svnrevisionsname{Table of Revisions}%
```

`\svn@svt`

File ending for table of revision auxiliary file. A macro is used to allow redefinition by the user if another package is uses the same ending.

```
\def\svn@svt{svt}
```

`\tableofrevisions`

The `\tableofcontents` macro from standard LaTeX is adapted for this macro. Classes which provide chapters will get a different table then one which not.

The external (i.e. non-svn-multi) if-switches are masked using `\ifx` and `\csname` to avoid TeX if-parsing errors when they are not defined.

```
\AtBeginDocument{%
\ifx\chapter\@undefined

%% Adapted from the \tableofcontents macro, LaTeX 'article' class [2005/09/16]
\newcommand\tableofrevisions{%
  \section*{\svnrevisionsname
  \@mkboth{\MakeUppercase\svnrevisionsname}{\MakeUppercase\svnrevisionsname}}%
  \svn@input{\jobname .\svn@svt}%
}

\else

%% Adapted from the \tableofcontents macro, LaTeX 'book' class [2005/09/16]
\newcommand\tableofrevisions{%
  \expandafter\ifx
  \csname if@twocolumn\expandafter\endcsname
  \csname iftrue\endcsname
```

```
    \@restonecoltrue\onecolumn
  \else
    \@restonecolfalse
  \fi
  \chapter*{\svnrevisionsname
    \@mkboth{\MakeUppercase\svnrevisionsname}{\MakeUppercase\svnrevisionsname}}%
  \svn@input{\jobname .\svn@svt}%
  \expandafter\ifx
  \csname if@restonecol\expandafter\endcsname
  \csname iftrue\endcsname
    \twocolumn
  \fi
}

\fi
}
\fi % defined \tableofcontents
```

\svn@writerow

#1: row type ('group', 'file', 'global', . . .)

#2: row type specific argument

#3: row type specific argument

Writes a table row by using \svn@tabcell and \svn@tabcellarg defined by the \svn@writeXXXrow macro below.

```
\def\svn@writerow#1#2#3{%
  \immediate\write\svn@svtwrite{%
    \expandafter\noexpand\csname svn#1row\endcsname
    \expandafter\noexpand\csname svntab#1\endcsname{#2}{#3}\space
    \@ampersamchar\space
    \svn@tabcell{rev}\space\@ampersamchar\space
    \svn@tabcell{author}\space\@ampersamchar\space
    \noexpand\svntabdate%
    \svn@tabcellarg{year}%
    \svn@tabcellarg{month}%
    \svn@tabcellarg{day}%
    \svn@tabcellarg{hour}%
    \svn@tabcellarg{minute}%
    \svn@tabcellarg{second}%
    \svn@tabcellarg{timezonehour}%
    \svn@tabcellarg{timezoneminute}%
    \space\@backslashchar\@backslashchar
    \expandafter\noexpand\csname endsvn#1row\endcsname
  }%
}
```

\svn@writegrouprow

#1: current group

```
\def\svn@writegrouprow#1{%
```

```
  \begingroup
    \def\svn@tabcellarg##1{{\csname @svng@#1@##1\endcsname}}%
    \def\svn@tabcell##1{\expandafter\noexpand\csname svntab##1\endcsname%
      \svn@tabcellarg{##1}%
    }%
    \svn@writerow{group}{#1}{}%
  \endgroup
}
```

\svn@writesubgrouprow

`#1:` grouping level
`#2:` subgroup name

```
\def\svn@writesubgrouprow#1#2{%
  \begingroup
    \def\svn@tabcellarg##1{{\csname @svng@#2@##1\endcsname}}%
    \def\svn@tabcell##1{\expandafter\noexpand\csname svntab##1\endcsname%
      \svn@tabcellarg{##1}%
    }%
    \svn@writerow{subgroup}{#1}{#2}%
  \endgroup
}
```

\svn@writefilerow

`#1:` grouping level
`#2:` file name

```
\def\svn@writefilerow#1#2{%
  \begingroup
    \def\svn@tabcellarg##1{{\csname @svng@#2@##1\endcsname}}%
    \def\svn@tabcell##1{\expandafter\noexpand\csname svntab##1\endcsname%
      \svn@tabcellarg{##1}%
    }%
    \svn@writerow{file}{#1}{#2}%
  \endgroup
}
```

\svn@writeglobalrow

```
\def\svn@writeglobalrow{%
  \begingroup
  \def\svn@tabcellarg##1{{\csname @svn@##1\endcsname}}%
  \def\svn@tabcell##1{\expandafter\noexpand\csname svntab##1\endcsname%
    \svn@tabcellarg{##1}%
  }%
  \svn@writerow{global}{}{}%
  \endgroup
}
```

7.19.1 Table Format Macros

Generic format macro used in the `.svt` file. Can be redefined by the user to change table format. TODO: More documentation needed!

\svntable

```
\def\svntable{%
  \begin{tabular}{p{0.425\textwidth}rll}%
    \hline
}
```

\endsvntable

```
\def\endsvntable{\hline\end{tabular}}
```

\svntablehead

```
\def\svntablehead{%
    Name & Rev & Last Author & Last Changed At \\\hline
}
```

\svntablefoot

```
\def\svntablefoot{}
```

\svnbeforetable

```
\def\svnbeforetable{}
```

\svnaftertable

```
\def\svnaftertable{\clearpage}
```

\svnglobalrow

```
\def\svnglobalrow{}
```

\endsvnglobalrow

```
\def\endsvnglobalrow{}
```

\svngrouprow

```
\def\svngrouprow{}
```

\endsvngrouprow

```
\def\endsvngrouprow{}
```

\svnsubgrouprow

```
\def\svnsubgrouprow{}
```

\endsvnsubgrouprow

```
\def\endsvnsubgrouprow{}
```

\svnsubgrouprow

```
\def\svnfilerow{}
```

\endsvnfilerow

```
\def\endsvnfilerow{}
```

\svntabglobal

```
\def\svntabglobal{Document}
```

\svntabgroup

```
\def\svntabgroup#1{Group ‘#1’}
```

\svntabfile

```
\def\svntabsubgroup#1{%
  \raggedright
  \addtolength{\leftskip}{#1\medskipamount}%
  \begingroup
  \catcode‘\_=12
  \catcode‘\&=12
  \catcode‘\^=12
  \catcode‘\$=12
  \catcode‘\#=12
  \svn@tabsubgroup
}
\def\svn@tabsubgroup#1{\endgroup Subgroup ‘\texttt{\small #1}’}
```

\svntabfile

```
\def\svntabfile#1{%
  \raggedright
  \addtolength{\leftskip}{#1\medskipamount}%
  \begingroup
  \catcode‘\_=12
  \catcode‘\&=12
  \catcode‘\^=12
  \catcode‘\$=12
  \catcode‘\#=12
  \svn@tabfile
}
\def\svn@tabfile#1{\endgroup File ‘\texttt{\small #1}’}
```

\svntabrev

```
\def\svntabrev{}
```

\svntabauthor

```
\def\svntabauthor#1{\svnFullAuthor{#1}}
```

\svntabdate

```
\def\svntabdate#1#2#3#4#5#6#7#8{%
    #1-#2-#3 #4:#5:#6% #7#8%
}
```

```
\fi
```

7.20 Other macros

This section contains macros which don't fit in any other section.

\svn

\svn*

After *-testing, the intermediate macros \svn@s and \svn@n are called to strip the { } from \svn[*]{$...$} and to remove the *. Then the actual macros are called to strip the dollars with or without the space before the last dollar.

```
\newcommand{\svn}{\@ifnextchar{*}{\svn@s}{\svn@n}}
\def\svn@n#1{\@svn@n#1}
\def\svn@s*#1{\@svn@s#1}
\def\@svn@n$#1${#1}
\def\@svn@s$#1 ${#1}
```

\svnnolinkurl

#1: URL

This code is taken from the hyperref package and is the definition of \url just without the part which creates the actual hyperlink. This needs of course the hyperref package. A warning is given if it isn't loaded.

```
%% Adapted from the \url macro of the ‘hyperref‘ package.
\DeclareRobustCommand*{\svnnolinkurl}{%
  \@ifundefined{hyper@normalise}%
    {\PackageWarning{svn-multi}{Package hyperref is needed for \noexpand
     \svnnolinkurl.}}%
    {\hyper@normalise\svnnolinkurl@}%
}%
\def\svnnolinkurl@#1{\Hurl{#1}}%
```

\svn@getfilename

#1: URL

This macro expands the content using the temporary macro and sets it in front of the \svn@getfilename sub-macro together with /{} to make sure the macro does not break at values without directories. A \relax is used as end marker.

```
\def\svn@getfilename#1{%
  \begingroup
    \gdef\svnfiledir{}%
    \edef\svn@temp{#1}%
    \expandafter\@svn@getfilename\svn@temp/{}\relax
}%
```

\@svn@getfilename

#1: URL part before first slash
#2: URL part after first slash

Splits the content at the first slash (/) and checks if the remainder is empty. If so the end marker got reached and the part before the slash is the filename which is returned. Otherwise the macro recursively calls itself to split the remainder.

```
\def\@svn@getfilename#1/#2\relax{%
  \svn@ifempty{#2}%
    {%
      \endgroup
      \gdef\svnfilefname{#1}%
    }%
    {%
      \xdef\svnfiledir{\svnfiledir#1/}%
      \@svn@getfilename#2\relax
    }%
}%
```

7.21 Auxiliary file generation and read-back

Reread output from last compile run if it exists.

```
\@input{\jobname .svn}
```

`\svn@checkwrite`

Checks if .svn file is already open and if not opens it. This makes sure that the file is only created if really needed. The macro is only needed once, so it's redefines itself to `\relax` at the end.

```
\def\svn@checkwrite{%
 \@ifundefined{svn@write}{%
    \newwrite\svn@write
   \immediate\openout\svn@write=\jobname.svn\relax%
   \immediate\write\svn@write{\@percentchar\space SVN Keyword cache}%
   %\immediate\write\svn@write{\noexpand\makeatletter}%
 }{}%
 \let\svn@checkwrite=\relax
}
```

`\svn@writesvn`

This macro writes the `.svn` auxiliary file and is called from a `\AtEndDocument` macro later on.

```
{\catcode`\&=12
\gdef\@ampersamchar{&}
}
\def\svn@writesvn{%
```

Remove all files which do not have a revision number from list:

```
    \if@svnmulti@groups
    \fi
```

Write document global values:

```
    \svn@checkwrite
    \immediate\write\svn@write{^^J%
      \@percentchar\space Global values:^^J%
      \noexpand\def\noexpand\svnrev{\@svn@rev}^^J%
      \noexpand\let\noexpand\ifsvnmodified\@backslashchar\@svn@modified^^J%
      \noexpand\def\noexpand\svndate{\@svn@date}^^J%
      \noexpand\def\noexpand\svnauthor{\@svn@author}^^J%
      \noexpand\def\noexpand\svnyear{\@svn@year}^^J%
      \noexpand\def\noexpand\svnmonth{\@svn@month}^^J%
      \noexpand\def\noexpand\svnday{\@svn@day}^^J%
      \noexpand\def\noexpand\svnhour{\@svn@hour}^^J%
      \noexpand\def\noexpand\svnminute{\@svn@minute}^^J%
      \noexpand\def\noexpand\svnsecond{\@svn@second}^^J%
      \noexpand\def\noexpand\svntimezonehour{\@svn@timezonehour}^^J%
      \noexpand\def\noexpand\svntimezoneminute{\@svn@timezoneminute}^^J%
      \noexpand\svn@gdefverb\noexpand\svnurl{\@svn@url}^^J%
      \noexpand\svn@gdefverb\noexpand\svnfname{\@svn@fname}^^J%
    }%
```

Write group keyword macro definitions. Remove all files which do not have a revision number from list:

```
    \if@svnmulti@groups
      \svn@cleanfilelist\@svng@@files
      \immediate\write\svn@write{%
        \noexpand\def\noexpand\svng@@files{\@svng@@files}^^J%
      }%
      \svn@writeallgroups\@svng@@files
```

Write keyword group values if groups were specified:

```
      \ifx\svn@glist\empty\else
        \begingroup
          \immediate\write\svn@write{^^J%
            \@percentchar\space SVN File Groups: \svn@glist
          }%
          \svn@writeallgroups\svn@glist
        \endgroup
      \fi
    \else
      \immediate\write\svn@write{^^J}%
    \fi
```

Finally close output file:

```
    \immediate\closeout\svn@write%
}
```

\svn@writegroupfiles

#1: group name

Writes all files of a group to the .svt file. If the file is actually a subgroup it calls itself recursively.

```
\def\svn@writegroupfiles#1{%
  \begingroup
    \advance\svn@grouplevel by 1\relax
    \expandafter\let
    \expandafter\svn@files\csname @svng@#1@files\endcsname
```

Stop if no files in list.

```
    \ifx\svn@files\relax\else
      \svn@cleanfilelist\svn@files
      \@for\svn@file:=\svn@files\do{%
```

Check if VC data is set and then if it is a (sub)group or not by looking at the file list.

```
        \svn@ifvalidrev{@svng@\svn@file @rev}%
          {%
            \@ifundefined{@svng@\svn@file @files}%
              {\svn@writefilerow{\the\svn@grouplevel}{\svn@file}}%
```

If subgroup only contains out of a TeX file with the same name print it as file and only as subgroup otherwise. If the subgroup file list is empty the subgroup was generated in error and is not printed at all.

```
              {\svn@ifonlyone{\svn@file}%
                {\svn@writefilerow{\the\svn@grouplevel}%
                  {\csname @svng@\svn@file @files\endcsname}}%
                {\svn@ifempty{\csname @svng@\svn@file @files\endcsname}%
                  {}%
                  {%
                    \svn@writesubgrouprow{\the\svn@grouplevel}{\svn@file}%
                    \svn@writegroupfiles{\svn@file}%
                  }%
                }%
              }%
          }{}%
      }%
    \fi
  \endgroup
}%
```

\svn@writesvt

This macro writes the .svt auxiliary file and is called from a \AtEndDocument macro later on.

```
\def\svn@writesvt{%
```

Write table of revisions if enabled.

```
  \if@svnmulti@table
```

Open .svt file and write table header:

```
    \newwrite\svn@svtwrite
    \immediate\openout\svn@svtwrite=\jobname.\svn@svt\relax
    \@onelevel@sanitize\svntable%
    \immediate\write\svn@svtwrite{%
      \noexpand\svnbeforetable^^J%
      \svntable
      \noexpand\svntablehead^^J%
    }%
```

Group rows:

```
    \let\svn@grouplevel\@tempcnta
    \svn@grouplevel=0\relax
    \svn@writeglobalrow{}%
    \svn@writegroupfiles{}%
%
    \@for\svn@g:=\svn@glist\do{%
      \@ifundefined{@svng@\svn@g @rev}{}%
      {%
        \expandafter
        \ifnum\csname @svng@\svn@g @rev\endcsname>0\relax
          \svn@writegrouprow{\svn@g}%
          \svn@writegroupfiles{\svn@g}%
        \fi
      }%
    }%

```

Write table footer and close file:

```
    \@onelevel@sanitize\endsvntable%
    \immediate\write\svn@svtwrite{%
      \noexpand\svntablefoot^^J%
      \endsvntable^^J%
      \noexpand\svnaftertable
    }%
    \immediate\closeout\svn@svtwrite%
  \fi
}
```

Load the keywords of every subfile if the **autokw** option is enabled and the extension is not on the ignore list.

This code is placed here because it has to come after the at-begin-input-file code.

```
\if@svnmulti@autokwall

\svnmulti@atbegininputfile{%
  \expandafter
  \ifx\csname svn@ignore@ext@\svn@fileext\endcsname\relax
    \svnegetfile{\svn@filepath}%
  \fi
}

\fi
```

At the end of document the values are written to the auxiliary file.

```
\AtEndDocument{%
  \if@filesw
    \ifx\@svn@rev\empty\else
      \ifnum\@svn@rev<1\else
        \svn@writesvn
        \svn@writesvt
      \fi
    \fi
  \fi
}
```

7.22 Backward compatibility wrapper svnkw.sty

For backward compatibility a wrapper file with the old package name svnkw is provided. Newer documents should use the name svn-multi.

```
\NeedsTeXFormat{LaTeX2e}[1999/12/01]
\ProvidesPackage{svnkw}
 [2009/03/27 v2.1 Backward compatibility wrapper for svn-multi]
\PackageWarning{svnkw}{The package 'svnkw' got renamed to 'svn-multi' and is now
only a backward compatibility wrapper which loads 'svn-multi'.  Please adjust
your document preamble to use the new name.}
\RequirePackage{svn-multi}
```

Change History

v1.0

General: Initial version 1

v1.1

General: Added macros to extract and typeset date/time information. Added macros to set and typeset main URL or filename. 1

v1.2

General: Added star version. Normal version was not changed to not break existing documents with user defined keywords without leading space. . 69

Added warning when a wrong, maybe misspelled, keyword is given. 41

Macro now returns the username if the full name was not registered. 45

New macro 44–46, 69

Renamed package from svnkw to svn-multi to match CTAN directory. Wrapper file svnkw.sty is provided for backward compatibility. 1

v1.3

General: Added verbatim support. Keywords can now contain special character like _ ^ $ % & \. Rewrote keyword check macros to work with verbatim code. \nofiles is now obeyed. 1

v1.3a

General: Fixed issue with unwanted spaces generated by \svnid, \svnidlong and \svnkwsave, e.g. when used in a file which is included with \input 1

v1.3b

General: Changed the way catcodes are modified to be compatible with the french option of the babel package or other packages which modify the list of special characters. 1

v1.4

General: Added support for timezones with non-zero minute part, e.g. +0530. 1

Return now full timezone (hour + minute part). Manually added 00 minutes are removed. 33

v1.5

General: Added \today-style

macros \svntoday and \svnfiletoday. 1

v2.0

General: New features: keyword groups, external file support, auto-include of images, files as subgroups and table of revisions. 1

v2.1

General: Added automatic extraction of keywords from Subversion 'entries' file: 'autokw' option 1

v2.2

General: Added \ifsvnmodified and \ifsvnfilemodified macros. Some bug fixes. 1

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.