The snapshot package

American Mathematical Society
Michael Downes

Version 1.14, 2002/03/05

1 Introduction

The `snapshot` package helps the owner of a LaTeX document obtain a list of the external dependencies of the document, in a form that can be embedded at the top of the document. To put it another way, it provides a snapshot of the current processing context of the document, insofar as it can be determined from inside LaTeX.

If a document contains such a dependency list, then it becomes possible to arrange that the document be processed always with the same versions of everything, in order to ensure the same output. This could be useful for someone wanting to keep a LaTeX document on hand and consistently reproduce an identical DVI file from it, on the fly; or for someone wanting to shield a document during the final stages of its production cycle from unexpected side effects of routine upgrades to the TeX system.

Normal usage of the `snapshot` package involves the following steps:

1. Add a `\RequirePackage` statement at the top of the document:

```
\RequirePackage{snapshot}
\documentclass{article}
  ...
```

2. Run LaTeX on the document. This will produce a dependency list in a file `\jobname.dep`.
3. Insert the `.dep` file at the top of the document, before `\documentclass`. The following example shows what you would end up with for a document that used the article documentclass and the graphicx package:

```
\RequirePackage{snapshot}[1999/11/03]
\RequireVersions{
  *{application}{TeX}     {1990/03/25 v3.x}
  *{format} {LaTeX2e}     {1999/06/01 v2.e}
  *{package}{snapshot}    {1999/11/03 v1.03}
  *{class}  {article}     {1999/01/07 v1.4a}
  *{file}   {size10.clo}  {1999/01/07 v1.4a}
  *{package}{graphicx}    {1999/02/16 v1.0f}
  *{package}{keyval}      {1999/03/16 v1.13}
  *{package}{graphics}    {1999/02/16 v1.0l}
  *{package}{trig}        {1999/03/16 v1.09}
  *{file}   {graphics.cfg}{0000/00/00 v0.0}
  *{file}   {dvips.def}   {1999/02/16 v3.0i}
}
\documentclass{article}
\usepackage{graphicx}
...
```

The package option `log` will cause the dependency list to appear in the LaTeX log file instead of in a separate `.dep` file:

```
\RequirePackage[log]{snapshot}
```

Making the necessary arrangements to ensure that future LaTeX runs of the document actually call in the specified versions is a separate problem. The `snapshot` package only provides a way to generate the dependency list. However, the `\RequireVersions` statement does record the given information in a form that can be accessed from within LaTeX. (It is for this purpose that it is not simply a comment.) In principle a package could be set up so that a later version would automatically attempt to emulate an earlier version if an earlier version was specified—much as LaTeX currently switches to 2.09 compatibility mode if it sees `\documentstyle` instead of `\documentclass`.

For maximum reliability font checksums should also be reported in the dependency list, but standard TeX 3.x does not provide direct access to font checksums for macro programmers. This information could be added by a separate script that scans the DVI file.

When a graphics file is read in by a LaTeX document using the standard `\includegraphics` command, it gets a dummy version number string of

```
Graphics file (type foo)
```

where foo is typically `eps`. This is with the current version of the `graphics` package (at the time of this writing: 1999/02/16 v1.0l). What this means in practice is that all graphics files will have their snapshot date and version number recorded as

```
Graphics v0.0
```

and will always compare equal (the string “Graphics” will be used in place of a date, but since comparison is done with `\ifx` it doesn’t make any real difference).

It would be possible, for `.eps` files at least, to read the CreationDate comment that is normally included in the file header and use that as the basis of comparison. Recording the bounding box numbers instead of a dummy version number would also be a possibly useful stratagem. But I have left these possibilities untouched for the time being [mjd,2001-05-14].

The `\RequireVersions` command scans its argument for file names and associated version number information. The syntax of a version line for a particular file is `\RequireVersio`

`*{` *file type* `}{` *file name* `}{` *version info* `}`

In other words, the `*` character in this context is like a command that takes three arguments. The extension part of the file name should be omitted in the second argument, except when the file type is `file` (following the conventions of LaTeX's `\ProvidesPackage` and `\ProvidesFile` commands). The most commonly used file types are as follows.

class A LaTeX documentclass file.

package A LaTeX package file.

tfm A TeX font metric file. In this case the "version number" is the checksum, and unless you are using an extended version of TeX this information is not accessible from inside LaTeX, so it must be filled in by an outside process. By default, font metric files are not listed in the dependency list since the checksum info is not available. There is a package option `tfm` to turn on the logging of metric files. (Not yet implemented [mjd,1999/09/23])

format This is almost always `LaTeX2e`. Other possibilities would be `elatex` or `lambda`. The information comes from `\fmtname`.

application This is usually `TeX` with version number `3.x`. From inside LaTeX there is no reliable way to get the exact version number (one could check to see if version 2.x is in use, but this is unlikely to be relevant, nowadays, so I have not bothered).

file None of the above: some other file of miscellaneous type, e.g., `.clo`, `.cfg`, `.tex`, or `.def`.

The `\RequireVersions` command can be given an optional "ident" argument, similar to the argument of a `\label` command. This is not used internally but it could be used to assign a label to particular groups of files in case that helps with external processing.

2 Warning and error options

If the `snapshot` package is invoked with the `error` option *and also* the document contains a `\RequireVersions` statement, then each subsequent `\ProvidesFile`, `\ProvidesPackage`, and `\ProvidesClass` statement will compare date and version number information with the corresponding information from the `\RequireVersions` statement and give an error message if a mismatch is detected. With the `warning` option you get warnings instead of errors. By default both the date and the version number are compared; this behavior can be modified, however, by giving additional options:

date compare only dates,

version compare only version numbers,

major-version use only the major version number when comparing.

Note: A file that doesn't have any sort of `\ProvidesFile` or `\ProvidesPackage` statement in it will show up in the dependency list, with a dummy date and version number of 0000/00/00 v0.0, but there is no way, of course, to give any meaningful warning or error message about version mismatches for such a file.

3 Implementation

Standard declaration of package name and date.

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
\ProvidesPackage{snapshot}[2002/03/05 v1.14]
```

```
\let\@xp\expandafter \let\@nx\noexpand
```

Calling the `snapshot` package in a document causes LaTeX to list the file names and versions in the TeX log or in a `.dep` file, so that the information may be easily copied into the document file. The list so generated is nothing more than a slight adaptation of the output from LaTeX's `\listfiles` command; it puts essentially the same information into a slightly more structured form so that it will be easier to use.

For the standard mechanisms that are already built into LaTeX (e.g., the handling of the second optional argument of `\LoadClass`), the de facto "version number" is the *date* given in the optional argument of a `\ProvidesClass` or similar command. Even though most `\ProvidesWhatever` commands also give something that follows the usual form of version numbers—a string of the form `v2.3`—this is only a convention, not used internally by LaTeX, and the identification string of a random loaded file is not guaranteed to include it. The `snapshot` package copies both pieces of information if available; if the second piece is not present, a dummy number `0.0` is supplied. Similarly, files that don't include any `\ProvidesWhatever` statement will get a dummy date of `0000/00/00`; TeX system administrators who want to ensure maximal accuracy of the snapshot information should therefore make it a practice to use `\ProvidesFile` in `.cfg` files and other local files that might have an impact on the output fidelity of their documents.

`\RequireVersions` Optional argument of `\RequireVersions` allows assigning a name to a particular collection of files. This might be useful for setting a TeX inputs path.

```
\newcommand{\RequireVersions}[2][]{%
  \let\snap@check\snap@compare@versions
  \let\snap@selfcheck\snap@selfcheck@a
  \@ifnextchar *\snap@store@version\snap@store@error#2*{end}{}{}%
}
\@onlypreamble\RequireVersions
```

```
\def\snap@store@error#1{%
  \PackageError{snapshot}{Expected '*' here, not '#1'}\@ehc
}
\@onlypreamble\snap@store@error
```

```
\def\snap@store@version #1#2#3#4{%
  \@xp\snap@store@b\csname snapx@#2\endcsname{#2}{#3}{#4}%
}
\@onlypreamble\snap@store@version
```

```
\def\@fmtextension{fmt}
\def\@tfmextension{tfm}
\edef\snapx@package{.\@pkgextension}
```

```
\edef\snapx@class{.\@clsextension}
\edef\snapx@format{.\@fmtextension}
\edef\snapx@tfm{.\@tfmextension}
\long\def\snapx@application{}
\let\snap@file=\@empty
\let\snapx@end\@@end
```

For a package named `foo.sty`, this function defines `\rqv@foo.sty` to hold the date and version information.

```
\def\snap@store@b#1#2#3#4{%
  \ifx#1\snapx@end
    \@xp\@gobblefour
  \else
    \ifx#1\relax \let#1\@empty\fi
    \def\@tempa##1 ##2 ##3\@nil{##1 ##2}%

    \ifx#1\snapx@application
    \else
      \xdef\rqv@list{\rqv@list
        \ifx\@empty\rqv@list\else,\fi
        #3#1%
      }%
    \fi
    \@xp\xdef\csname rqv@#3#1\endcsname{\@tempa#4 v?.? ? \relax\@nil}%
    \ifx#1\snapx@format \snap@check{#3.fmt}%
    \else \snap@selfcheck{#3.sty}%
    \fi
  \fi
  \@ifnextchar *\snap@store@version\snap@store@error
}
\@onlypreamble\snap@store@b
```

Default setup is geared to write the dependency list to a `.dep` file. The option `log` means write it to the TeX log instead.

```
\def\snap@write{\immediate\write\snap@out}
\let\snap@out\sixt@@n % fallback, probably never used

\DeclareOption{dep}{%
  \def\snap@write{\immediate\write\snap@out}%
}

\DeclareOption{log}{%
  \let\snap@write\typeout
}
```

The purpose of the ‘test’ option is to support a separate testing procedure that resolves file names. If the `\RequireVersions` data is extracted to a separate file, and

```
\RequirePackage[test]{snapshot}
```

is added at the top, then the file can be run as a small separate LaTeX job whose sole purpose is to produce in the log file a nice list of fully resolved file names. A limited, but system-independent variant of the `kpsewhich` idea.

```
\let\snap@fake@b\relax
\DeclareOption{test}{%
  \def\snap@fake@b{\endinput \futurelet\@let@token\snap@ignoline}%
}
```

For each font used by a document, we would like to list the `.tfm` file name and checksum. If TeX provided a `\fontchecksum` primitive similar to `\fontname` that could be used to get the checksum of any font, it would just about be feasible to do this entirely from within LaTeX. As a partial solution we could at least generate the list of font file names, to make it easier for an external utility to add the checksums.

In practice, extracting font names and checksums from the `.dvi` file will probably work well enough, leaving no work to be done by the `snapshot` package in this area. But theoretically speaking the output of a document could be affected by font metric files that are loaded during LaTeX processing but that do not show up in the `.dvi` file.

```
\DeclareOption{tfm}{%
  \typeout{Option 'tfm' not implemented yet [1999/09/23]}%
}
```

Warnings and errors.

```
\def\snap@mismatch@warning#1#2#3{\PackageWarningNoLine{#1}{#2}}
\def\snap@mismatch{\snap@mismatch@warning}

\DeclareOption{error}{%
  \def\snap@mismatch{\PackageError}%
  \ifx\snap@select\@empty \let\snap@select\snap@select@all \fi
}

\DeclareOption{warning}{%
  \def\snap@mismatch{\snap@mismatch@warning}%
  \ifx\snap@select\@empty \let\snap@select\snap@select@all \fi
}
```

Because the exact form of the version number is not mandated by LaTeX, just take the first two “words” delimited by spaces. And take a little extra care to properly handle multiple spaces between the words.

```
\def\snap@select@all#1#2 #3#4 #5\@nil{#1#2 #3#4}
\let\snap@select\@empty

\DeclareOption{date}{%
  \def\snap@select#1#2 #3\@nil{#1#2}%
}

\def\snap@select@version#1{%
  \ifodd 0#11 \@xp\snap@sva\@xp#1\else\@xp\snap@select@version\fi
}
\def\snap@sva#1.#2 #3\@nil{#1.#2}
```

```
\def\snap@select@major#1{%
  \ifodd 0#11 \@xp\snap@svm\@xp#1\else\@xp\snap@select@major\fi
}
\def\snap@svm#1.#2\@nil{#1}
```

```
\DeclareOption{version}{%
  \def\snap@select#1#2 #3{\snap@select@version #3}%
}
```

```
\DeclareOption{major-version}{%
  \def\snap@select#1#2 #3{\snap@select@major #3}%
}
```

```
\ProcessOptions\par
```

We need the following patch to make up for the fact that `\@pkgextension` and `\@clsextension` are marked in the LaTeX kernel as “only preamble”.

```
\edef\snap@restore@extensions{%
  \def\@nx\@pkgextension{\@pkgextension}%
  \def\@nx\@clsextension{\@clsextension}%
}
```

Pad filename strings out to 8+3 length so that the list will look pretty.

```
\def\snap@pad#1#2#3#4#5#6#7#8#9{\snap@pad@a{#1#2#3#4#5#6#7#8#9}}
\def\snap@pad@a#1#2#3#4#5\@nil{\snap@pad@b#1#2#3#4\space\@nil}
\def\snap@pad@b#1\space#2\@nil#3{\def#3{#2}}
```

First stage: discard leading spaces before the first and second nonspace strings in the argument. Take the first nonspace string as the date. Since we only do equal/not-equal testing on dates, it does not seem essential to test if it is really a valid date string or not (yyyy/mm/dd).

```
\def\snap@trim@version#1#2 #3{#1#2 \snap@trim@b #3}
```

Second stage: Scan for a version number. In order to handle some idiosyncratic cases, such as url.sty version 1.4, we can’t simply take the second nonspace string as the version number but need to look for a leading digit.

```
\def\snap@trim@b#1{\ifodd 0#11 v#1\@xp\snap@trim@c\fi \snap@trim@b}
```

Arg 1 here is `\snap@trim@b`, which we just need to discard.

```
\def\snap@trim@c#1#2 #3\@nil{#2}
```

```
\let\rqv@list=\@empty
```

If `\fmtname.fmt` is not already in the file list, add it.

```
\edef\@tempc#1\fmtname{#1\fmtname}\@tempc
\def\@tempa#1,\fmtname.fmt,#2#3\@nil{#2}
\edef\@tempb{\@nx\@tempa,\@filelist,\fmtname.fmt,}
\if ?\@tempb?\@nil
  \edef\@filelist{\fmtname.fmt,\@filelist}%
  \def\@tempc{LaTeX2e}%
  \@xp\edef\csname ver@\fmtname.fmt\endcsname{%
    \fmtversion\space
    v\ifx\@tempc\fmtname 2.e\else ?.?\fi
    }%
  \fi
  \listfiles
  \def\@dofilelist{%
    \snap@restore@extensions
    \ifx\rqv@list\@empty
    \else \rqv@compare@lists
    \fi
    \ifx\snap@write\typeout
    \else
      \newwrite\snap@out
      \immediate\openout\snap@out=\jobname.dep \relax
    \fi
    \snap@write{\string\RequireVersions\@charlb}%
```

Since the exact version number of TEX is not normally accessible from inside LATEX, we use a nominal date of 1990/03/25, which is when version 3.0 of `tex.web` was released by Knuth.

```
    \snap@write{\space\space *{application}{TeX}%
               \space\space\space\space\space{1990/03/25 v3.x}}%
    \@for\@currname:=\@filelist\do{%
      \filename@parse\@currname
      \ifx\filename@ext\relax
        \def\@tempa{file}\def\@tempd{.tex}\def\filename@ext{tex}%
        \def\@tempb{~~~}%
      \else\ifx\filename@ext\@pkgextension
        \def\@tempa{package}\let\@tempd\@empty
        \def\@tempb{}%
      \else\ifx\filename@ext\@clsextension
        \def\@tempa{class}\let\@tempd\@empty
        \def\@tempb{~~}%
      \else\ifx\filename@ext\@fmtextension
        \def\@tempa{format}\let\@tempd\@empty
        \def\@tempb{~}%
      \else\ifx\filename@ext\@tfmextension
        \def\@tempa{tfm}\let\@tempd\@empty
        \def\@tempb{~~~~}%
      \else
        \def\@tempa{file}\edef\@tempd{.\filename@ext}%
        \def\@tempb{~~~}%
      \fi\fi\fi\fi\fi
      \@xp\let\@xp\@tempe
        \csname ver@\filename@base.\filename@ext\endcsname
```

If a file contains just `\ProvidesFoo{xyz}` without *any* optional argument, then `\ver@xyz` ends up empty. Resetting it to `\relax` is the easiest way to get the fallback version number in that case also.

```
      \ifx\@tempe\@empty \let\@tempe\relax \fi
      \edef\@tempe{%
        \ifx\@tempe\relax 0000/00/00 v0.0%
```

```
        \else
          \@xp\@xp\@xp\snap@trim@version\@xp\@tempe\space v0.0 v0.0 \@nil
        \fi
      }%
      \edef\@tempc{\filename@area\filename@base\@tempd}% full file name
      \@xp\snap@pad\@tempc\space~~~~~~~~~~~~~~~~\@nil\@tempd
      \begingroup \let~\space
        \snap@write{\space\space *{\@tempa}\@tempb{\@tempc}\@tempd{\@tempe}}%
      \endgroup
    }%
    \snap@write{\@charrb}%
    \ifx\snap@write\typeout
    \else \immediate\closeout\snap@out
      \typeout{Dependency list written on \jobname.dep.}%
    \fi
  }%
```

The `\rqv@compare@lists` function checks to see if any files are found only in the RequireVersions list or only in `\@filelist`. To see which files are only in `\@filelist`, we map the `\rqv@condense` function across both lists, reinitializing `\L` (used here as a scratch variable) in between. As a side effect this leaves the desired file names in `\L`. Then the same process with the order of the lists reversed tells us which ones are only in `\rqv@list`.

```
  \def\rqv@condense#1,{%
    \if ,#1,%
    \else
      \@xp\ifx\csname ver@#1\endcsname\N
      \else
        \edef\L{\L,#1}%
        \@xp\let\csname ver@#1\endcsname=\N
      \fi
    \fi
    \rqv@condense
  }

  \def\rqv@compare@lists{%
    \begingroup
    \def\N{1}\let\L=\@gobble
    \@xp\rqv@condense \rqv@list,TeX,{,\relax\@xp\@gobbletwo\@xp},%
    \ifx\L\@gobble\let\L\@empty\fi
    \let\rqv@list=\L
    \let\L=\@gobble
    \@xp\rqv@condense \@filelist,{,\relax\@xp\@gobbletwo\@xp},%
    \ifx\L\@gobble\let\L\@empty\fi
    \@for\@currname:=\L\do{%
      \snap@mismatch{snapshot}{^^J%
        File \@currname\space loaded though not in
        \noexpand\RequireVersions list%
      }\@ehc
    }%
```

```
    \def\N{2}\let\L=\@gobble
    \@xp\rqv@condense\@filelist,TeX,{,\relax\@xp\@gobbletwo\@xp},%
    \let\L=\@gobble
    \@xp\rqv@condense\rqv@list,{,\relax\@xp\@gobbletwo\@xp},%
    \ifx\L\@gobble\let\L\@empty\fi
    \@for\@currname:=\L\do{%
      \snap@mismatch{snapshot}{^^J%
        File \@currname\space [\csname rqv@\@currname\endcsname]
        required but not loaded%
      }\@ehc
    }%
    \endgroup
  }
```

See the documentation above for the 'test' option.

```
  \begingroup \catcode\endlinechar=12\relax %
  \long\gdef\snap@ignoline#1
  {}\endgroup %
```

```
  \def\snap@fake@input#1#2#3#4{%
    \ifx#1\snapx@end
      \aftergroup\@@end \@xp\@gobblefour
    \else
      \ifx#1\snapx@format
      \else
        \message{^^J}%
        \@xp\snap@fake@b\@@input #3#1\relax
      \fi
    \fi
    \@ifnextchar *\snap@store@version\snap@store@error
  }
```

```
  \newcommand{\rqvTest}[2][]{%
    \begingroup \catcode\endlinechar=12
    \catcode`\%=12 \catcode`\{=12 \catcode`\}=12\relax
```

Since `\snap@fake@input` just compares `\snapx@foo` with `\ifx`, making `\snapx@application` and `\snapx@tfm` compare equal to `\snap@format` ensures that only one comparison is needed to tell if we shouldn't attempt to input the current file type.

```
    \let\snapx@application=\snapx@format \let\snapx@tfm=\snap@format
    \@ifnextchar *\snap@store@version\snap@store@error#2*{end}{}{}%
    \endgroup
  }
```

```
  \@ifundefined{snap@fake@b}{}{%
    \let\snap@store@b\snap@fake@input
    \let\RequireVersions\rqvTest
  }
```

Compensate for a bug in old versions of `amsgen.sty`. This is a little tricky.
Old version: `\ver@amsgen`=1996/10/29 v1.2b
New version: `\ver@amsgen.sty`=1999/11/30 v2.0

```
%\@namedef{ver@amsgen.sty}{1996/10/29 v1.2b}
\AtBeginDocument{%
  \@ifundefined{ver@amsgen}{}{%
    \@xp\let\csname ver@amsgen.sty\@xp\endcsname
                    \csname ver@amsgen\endcsname
  }%
}
```

Terminate here without touching LaTeX internals, unless one of the relevant snapshot options was chosen.

```
\let\snap@compare@versions\@gobble \let\snap@check\@gobble
\let\snap@selfcheck\@gobble \let\snap@selfcheck@a\@gobble
\ifx\snap@select\@empty \endinput \fi
```

```
\begingroup \catcode`\.=11\relax
\gdef\snap@selfcheck@b#1\rqv@snapshot.sty#2#3\@nil{T#2}
\gdef\snap@selfcheck@a#1{%
  \if\@xp\snap@selfcheck@b\csname rqv@#1\endcsname T%
        \rqv@snapshot.sty F\@nil
    \snap@check{#1}%
  \fi
}
\endgroup
```

```
\def\@nofmt#1.fmt.#2 {#1 }
```

```
\def\snap@mismatch@a#1#2#3{%
  \snap@mismatch{snapshot}{^^J%
    \space\space Required version #2 of \@nofmt#1.fmt. and^^J%
    \space\space provided version #3 do not match%
  }\@ehc
}
```

When comparing `\rqv@foo.sty` (information from a previous LaTeX run) with `\ver@foo.sty` (information from current run), we first call `\snap@trim@version` on the latter to clear away any idiosyncrasies in the contents.

```
\def\snap@compare@versions#1{%
  \begingroup
    \@ifundefined{rqv@#1}{}{%
      \edef\0{\csname rqv@#1\endcsname}%
      \edef\1{\csname ver@#1\endcsname}%
      \edef\1{\@xp\snap@trim@version\1 v0.0 v0.0 \@nil}%
      \edef\@tempa{\@xp\snap@select\0 v0.0 v0.0 \@nil}%
      \edef\@tempb{\@xp\snap@select\1 v0.0 v0.0 \@nil}%
      \ifx\@tempa\@tempb
      \else
        \edef\@tempd{\@nx\snap@mismatch@a{#1}{\@tempa}{\@tempb}}%
        \@xp\@tempd
      \fi
    }%
  \endgroup
}
```

Because `\ProvidesFile` is used in `.fd` files which are normally read with special catcodes, there tend to be problems with whitespace characters being erroneously lost from the second argument. Since we have to put in a `\snap@check` call anyway, while we're at it let's fix a bug of this type that affected some older versions of LaTeX.

```
\def\ProvidesFile#1{%
  \def\snap@checker{\snap@check{#1}}%
  \begingroup
    \aftergroup\snap@checker
    \catcode`\ 10\catcode\endlinechar 10 %
    \@makeother\/%
    \@makeother\&%
    \@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}%
}
```

```
\def\@pr@videpackage[#1]{%
  \expandafter\xdef\csname ver@\@currname.\@currext\endcsname{#1}%
  \ifx\@currext\@clsextension
    \typeout{Document Class: \@gtempa\space#1}%
  \else
    \wlog{Package: \@gtempa\space#1}%
  \fi
  \snap@check{\@currname.\@currext}%
}
```

The usual `\endinput` to ensure that random garbage at the end of the file doesn't get copied by `docstrip`.

```
\endinput
```

To Do

- Provide a test to distinguish between shared-use files and document-specific files? Document-specific files would be things like graphics files loaded via `\includegraphics` or book chapters loaded via `\include`. But there might also be something like `\input{my-book-macros}` in the preamble. Maybe authors should be encouraged to put this after `\begin{document}`? Or, even, after `\begin{abstract}` so that the abstract will contain only lowest-common-denominator LaTeX commands?

  The non-document-specific files most commonly loaded after `\begin{document}` would be `.fd` files, but there is also the possibility of autoloaded stuff (`alatex` format).