Das dateiliste-Package – Liste der verwendeten Dateien im Dokument*

Paul Ebermann†‡

3. April 2006

Zusammenfassung

Dieses Paket implementiert drei unabhängig verwendbare, aber dennoch zusammen besonders nützliche Befehle, um den Überblick über Versionen und Änderungsdaten von LaTeX-Quelltexten zu behalten.

Inhaltsverzeichnis

*Dieses Dokument gehört zu dateiliste v0.1, vom 2006/04/03.

†E-Mail: Paul-Ebermann@gmx.de

‡Rolf Niepraschk (Rolf.Niepraschk@ptb.de) hat das Package printfilelist geschrieben und mir geschickt, dessen Code bildete die Basis für \printFileList. Für den jetztigen Code (insbesondere dessen Fehler) bin ich (Paul) aber selbst verantwortlich.

1 Benutzerdokumentation der Befehle

1.1 Automatische Versionsinfos

`\ProvideFileInfos` `{`⟨*id-string*⟩`}{`⟨*kurzbeschreibung*⟩`}`

Ändert die Informationen für die Datei, in der es aufgerufen wurde. ⟨*id-string*⟩ sollte ein String sein, der wie

```
$Id: dateiliste.dtx,v 1.4 2006/04/03 16:46:52 ebermann Exp $
```

(Beispiel für diese Datei) aussieht. Diesen lässt man am besten von seinem CVS produzieren, schreibt also etwas wie

```
\ProvideFileInfos{$Id:$}{Algebraische Geometrie I}
```

in seine Datei und macht dann ein `cvs commit` – dadurch werden von CVS die passenden Informationen eingefüllt.

⟨*kurzbeschreibung*⟩ sollte eine kurze Beschreibung der Funktion/des Inhaltes der Datei sein, optimalerweise nur ASCII-Zeichen.

Das Makro sorgt dann dafür, dass in der durch `\listfiles` sowie auch der durch `\printFileList` (siehe Abschnitt 1.3) erzeugten Liste die richtigen Daten (d.h. das Commit-Datum und die RCS/CVS-Versionsnummer) stehen.

1.2 Hauptdatei in der Dateiliste, Package-Optionen

`\mainFileToList` Fügt die Haupt-Datei (⟨*jobname*⟩`.tex`) am Anfang der Dateiliste ein, falls sie existiert und dort noch nicht vorhanden ist (oder letzteres nicht festgestellt werden kann, wofür `\scantokens` aus ε-TEX notwendig ist[1]).

`noaddmain` `addmain` Dieses Makro wird automatisch am Ende des Dokumentes ausgeführt, falls nicht die Package-Option `noaddmain` gesetzt wurde. Mit `addmain` kann das Vorgabe-Verhalten erzwungen werden.

1.3 Dateilisten-Ausgabe

`\printFileList` `[`⟨*gliederung*⟩`]`

Fügt an der aktuellen Stelle eine Liste der im aktuellen Dokument verwendeten Dateien (ohne die Haupt-Datei) ein. ⟨*gliederung*⟩ ist ein Gliederungsbefehl (wie `\section`, `\chapter` etc., Vorgabewert ist `\section*` (für einen unnummerierten Abschnitt).

Damit eine Liste ausgegeben wird, muss in der Präambel des Dokumentes ein `\listfiles` auftauchen. (Andernfalls gibt `\printFileList` nur eine Warnung auf der Konsole aus und tut sonst nichts.)

Die Liste selbst ist erst ab dem zweiten LaTeX-Lauf im Dokument zu sehen (und enthält ab dem dritten Lauf dann auch die Dateilisten-Datei).

Die Liste wird als 4-spaltige Tabelle gesetzt.

[1] Das Package kann auch ohne ε-TEX verwendet werden, nur kann es dann vorkommen, dass eventuell die Hauptdatei doppelt in der Liste auftaucht.

Falls das Paket babel vor oder nach diesem Paket geladen wird, sind die Spaltenüberschriften, die Gliederung sowie die Präambel auch übersetzbar – zur Zeit werden von diesem Paket die Sprachoptionen `english`, `german`, `ngerman` und `esperanto` unterstützt.

`\fileListName` `\fileListPreamble` `\fileNameName` `\dateName` `\verName` `\descriptionName`

Ansonsten kann durch Neudefinition der Befehle `\fileListName` (Überschrift), `\fileListPreamble` (einleitender Text, normalerweise mit Fußnote), sowie `\fileNameName`, `\dateName`, `\verName` und `\descriptionName` (Tabellenkopf) der Inhalt der statischen Texte verändert werden. (Auch das wirkt sich erst im folgenden LaTeX-Lauf aus.)

Im folgenden mal ein Beispiel aus diesem Dokument, erzeugt mit:

```
\printFileList[\subsubsection]
```

1.3.1 Liste der Dateinamen

Hier die Liste aller Dateien, die während des LaTeX-Laufes, welcher dieses Dokument erzeugte, verwendet wurden. [2]

Dateiname	**Datum**	**Ver.**	**Beschreibung**
ltxdoc.cls	1999/08/08	v2.0u	Standard LaTeX documentation class
article.cls	2004/02/16	v1.4f	Standard LaTeX document class
size10.clo	2004/02/16	v1.4f	Standard LaTeX file (size option)
doc.sty	2006/02/02	v2.1d	Standard LaTeX documentation package (FMi)
multicol.sty	2004/02/14	v1.6e	multicolumn formatting (FMi)
pauldoc.sty	2006/04/03	v0.4	Pauls Anpassungen fuer doc (PE)
inputenc.sty	2004/02/05	v1.0d	Input encoding file
latin1.def	2004/02/05	v1.0d	Input encoding file
babel.sty	2005/05/21	v3.8g	The Babel package
ngermanb.ldf	2004/02/20	v2.6m	new German support from the babel system
fontenc.sty	—		
t1enc.def	2004/02/22	v1.99f	Standard LaTeX file
dateiliste.sty	2006/04/03	v0.1	Ausgabe der Dateiliste (PE)
rcsinfo.sty	2005/02/20	v1.10	
rcsinfo.cfg	—		
ltxtable.sty	1995/12/11	v0.2	longtable/tabularx merge (DPC)
tabularx.sty	1999/01/07	v2.07	'tabularx' package (DPC)
array.sty	2003/12/17	v2.4a	Tabular extension package (FMi)
longtable.sty	2004/02/01	v4.11	Multi-page Table package (DPC)
dateiliste.dtx	—		
t1cmss.fd	1999/05/25	v2.5h	Standard LaTeX font definitions
t1cmtt.fd	1999/05/25	v2.5h	Standard LaTeX font definitions

[2] genauer: Es ist die Liste aller Dokumente, die einen LaTeX-Lauf früher verwendet wurden. Aber nach einigen Läufen sollte sich die Liste stabilisieren.

Dateiname	Datum	Ver.	Beschreibung
dateiliste.filelist	2006/04/03	—	automatically generated filelist
dateiliste.gls	—		
dateiliste.ind	—		

1.4 Abhängigkeiten

Für die Funktion des Paketes sind die Pakete rcsinfo (Jürgen Vollmer) und ltxtable – damit auch tabularx, longtable (alle drei von David Carlisle) und array (Frank Mittelbach) – notwendig. babel (Johannes Braams) wird, falls ebenfalls geladen, auch genutzt.

Falls das Paket pauldoc (von mir) ebenfalls geladen wird, werden einige spezielle Anpassungen getroffen.

Für die korrekte Erkennung, dass der Name der Hauptdatei schon in der Liste der geladenen Dateien auftaucht, ist der primitive ε-TEX-Befehl `\scantokens` notwendig – falls kein ε-TEX verwendet wird, kann eine nicht-Erkennung (und damit am Ende das doppelte Auftauchen der Haupt-Datei) vorkommen.

2 Implementation

```
⟨*package⟩
```

2.1 Optionen

Wir merken uns die ausgewählte Option in einem `\if`.

```
\newif\if@dateiliste@addMain
\DeclareOption{addmain} {%
   \@dateiliste@addMaintrue
}
\DeclareOption{noaddmain} {%
   \@dateiliste@addMainfalse
}
```

Die Standard-Option ist `addmain`.

```
\ExecuteOptions{addmain}
\ProcessOptions
```

2.2 Aktuelle Versionsnummern in der Dateiliste

Zunächst wollen wir in der durch `\listfiles` provozierten Ausgabe automatisch sinnvolle Infos haben. Dafür laden wir das Paket rcsinfo. Wir verwenden die Parameter `nofancy`, weil sonst die Fußzeile umgestellt wird, und `notoday`, weil sonst das aktuelle Datum umgestellt wird.

```
\RequirePackage[nofancy, notoday]{rcsinfo}
```

`\ProvideFileInfos` `{`⟨*id-string*⟩`}{`⟨*kurzbeschreibung*⟩`}`

```
\newcommand*{\ProvideFileInfos}[2] {%
```

Zunächst lassen wir `\rcsInfo` den ⟨*id-string*⟩ analysieren. Dies definiert (unter anderem) die Makros `\rcsInfoFile` (der Dateiname), `\rcsInfoDate` (Datum, im `YYYY/MM/DD`-Format) und `\rcsInfoRevision` (die Versionsnummer).

Das Leerzeichen nach dem `#1` ist notwendig, damit `\rcsInfo` erkennt, wo der ⟨*id-string*⟩ aufhört - in der Definition steht da nämlich ein Leerzeichen am Ende der Parameterliste.

```
13    \rcsInfo #1 %
```

Dann rufen wir `\ProvidesFile` aus dem LaTeX-Kernel auf.

```
14    \ProvidesFile%
```

Als erster Parameter wird der Dateiname übergeben, der von `\rcsInfo` ermittelt wurde. Mittels `\expandafter\@firstofone` entfernen wir dabei noch das von `\rcsInfo` (zumindest in meiner Version) eingebaute Leerzeichen am Anfang (welches ja einen anderen Namen ergibt und damit verhindern würde, dass die Information der richtigen Datei zugeschrieben wird).

```
15    {\expandafter\@firstofone\rcsInfoFile}%
```

Dann das, wofür wir das ganze eigentlich machen: Das Datum, ein Leerzeichen, dann die Versionsnummer (mit einem `v` davor). Schließlich hängen wir noch ⟨*kurzbeschreibung*⟩ an.

```
16    [\rcsInfoDate\space v\rcsInfoRevision\space #2]%
```

`\ProvidesFile` definiert jetzt ein Makro (`\ver@`⟨*dateiname*⟩) mit diesem Text als Inhalt, welches später von `\@dofilelist` (und unserem `\@writefilelist`) verwendet wird.

```
17 }
```

2.3 Dateiliste

Da die Liste ziemlich lang (länger als eine Seite) werden kann, verwende ich longtable statt der eingebauten (oder der von array verbesserten) `tabular`-Umgebung. Und damit ich in der letzten Spalte nicht die Breite fest einstellen muss, sondern einfach die restliche Breite (abhängig von Seitenbreite und der Breite der anderen Spalten, welche ja abhängig vom Inhalt ist) nehmen kann, lade ich ltxtable, welches longtable mit tabularx kreuzt (und beide Pakete auch lädt).

```
18 \RequirePackage{ltxtable}
```

`\dateiliste@preInclude`
`\dateiliste@postInclude`

Diese beiden Macros werden vor bzw. nach dem Laden (und setzen) der Dateiliste aufgerufen. Sie sorgen dafür, dass `'` innerhalb der Liste nicht mehr in den Verbatim-Mode schaltet, wie das von pauldoc eingestellt wird. Deswegen werden sie auch nur dann so definiert, wenn pauldoc geladen wurde. (Und weil `\@ifpackageloaded` nur in der Präambel erlaubt ist, müssen wir die beiden Befehle schon zu Beginn des Dokumentes definieren, anstatt einfach die Abfrage dann zu machen, wenn es gebraucht wird.)

```
19 \AtBeginDocument{%
20    \@ifpackageloaded{pauldoc}{%
21       \newcommand*{\dateiliste@preInclude}{\DeleteShortVerb{\'}}%
```

```
22      \newcommand*{\dateiliste@postInclude}{\MakeShortVerb{\'}}%
23    }{%
24      \newcommand*{\dateiliste@preInclude}{\relax}%
25      \newcommand*{\dateiliste@postInclude}{\relax}%
26    }%
27 }%
```

Die beiden Makros kann man sich auch selbst umdefinieren, falls andere Pakete Inkompatibilitäten ergeben.

2.3.1 Ausgabe der Liste

`\printFileList` `[`⟨*gliederung*⟩`]`

Der Vorgabewert für ⟨*gliederung*⟩ ist `\section*`, also ein unnummerierter Abschnitt.

```
28 \newcommand*{\printFileList}[1][\section*] {% \printFileList
```

Zunächst überprüfen wir, ob `\listfiles` in der Präambel gegeben wurde. Dies zeigt sich darin, dass das Kommando `\dofilelist` definiert ist. Andernfalls gibt es eine Warnung, und wir machen nichts.

```
29    \@ifundefined{@dofilelist}
30    {%
31      \PackageWarning{dateiliste}
32      {
33        \protect\printFileList\space works only if
34        \protect\listfiles\space is given in the preamble.
35      }
36    }%
37    {%                      else (\@ifundefined{@dofilelist})
```

Andernfalls beginnen wir einen neuen Abschnitt (oder ein Kapitel oder was auch immer mit ⟨*gliederung*⟩ festgelegt wurde), mit Namen `\fileListName` und einem Label, falls man mal von wo anders darauf verweisen möchte. Danach kommt etwas beschreibender Text in `\fileListPreamble`.

```
38      #1{\fileListName}\label{sec:filelist}%
39      \fileListPreamble
```

In der Datei ⟨*jobname*⟩`.filelist` befindet sich nach dem ersten LaTeX-Lauf der Inhalt der Tabelle (siehe unten). Wir überprüfen zunächst, ob die Datei schon existiert.

```
40      \IfFileExists{\jobname.filelist}{%
```

`\dateiliste@addtofilelist` `\@addtofilelist`

Falls ja, dann definieren wir zunächst `\@addtofilelist` um, da `\LTXtable` die Datei ⟨*jobname*⟩`.filelist` mehrfach einliest, wir aber nur einen Eintrag in der Dateiliste haben wollen. Wir verwenden nicht einfach `\@gobble`, um in dem Fall, dass durch das Setzen der Datei weitere Dateien (Schriften etc.) geladen werden, diese doch aufzunehmen. (Wir vergleichen also den Dateinamen mit dem unserer Dateinamens-Datei, und rufen im Fall der Nichtübereinstimmung das Original-`\@addtofilelist` auf.)

```
41        \let \dateiliste@addtofilelist = \@addtofilelist
```

```
        \def\@addtofilelist####1{%
           \edef\dateiliste@tempa{####1}%
           \edef\dateiliste@tempb{\jobname.filelist}\relax%
           \ifx\dateiliste@tempa\dateiliste@tempb
             \relax
           \else
             \dateiliste@addtofilelist{####1}
           \fi
        }%
```

`\dateiliste@preInclude` schaltet `'` als verbatim-Char ab (und das Makro `\dateiliste@postInclude` schaltet es nachher wieder an), falls pauldoc geladen wurde (ansonsten tun sie nichts, falls nicht von jemand anders neudefiniert). Die Datei selbst wird mittels `\LTXtable` geladen.

```
        \dateiliste@preInclude
        \LTXtable{\linewidth}{\jobname.filelist}%
        \dateiliste@postInclude
```

Danach stellen wir `\@addtofilelist` wieder her und fügen unsere Dateilisten-Datei auch hinzu.

```
        \let \@addtofilelist = \dateiliste@addtofilelist
        \@addtofilelist{\jobname.filelist}%
    }
```

Falls ⟨*jobname*⟩`.filelist` nicht vorhanden war, geben wir einen Hinweistext aus, dass man LaTeX noch einmal laufen lassen soll.

```
    {%
        \PackageWarning{dateiliste}{
           Run LaTeX again to include the File list.
        }%
    }%
```

2.3.2 Erstellen der Liste

Jetzt noch ein paar Befehle, um die Listen-Datei zu generieren ... (Wir sind immer noch innerhalb von `\printFileList`, das alles passiert also nur, wenn dieser Befehl aufgerufen wird.

Am Ende des Dokumentes – d.h., wenn die Dateiliste vollständig gesammelt wurde – schreiben wir sie – mit den passenden Formatierungsanweisungen – in eine Datei. (Das ganze in einer Gruppe, damit nichts kaputtgeht, und temporäre Makros nachher wieder freigegeben werden.)

```
    \AtEndDocument{%
        \begingroup
        \@writefilelist
        \endgroup
    }%
```

`\@writefilelist` Eine Variante von `\@dofilelist`, die den Inhalt – als Tabellenzeilen – in die Datei ⟨*jobname*⟩`.filelist` schreibt.

```
67      \newcommand*{\@writefilelist}{%  \@writefilelist
68        \newwrite\dateiliste@file
69        \immediate\openout\dateiliste@file = \jobname.filelist
```

Zunächst schreiben wir eine `\ProvidesFile`-Anweisung mit dem aktuellen Datum in die `.filelist`-Datei. (Das hat den Effekt, dass diese Datei selbst auch in der Liste erscheint.)

```
70        \edef\dateiliste@today{%
71          \the\year/\two@digits{\the\month}/\two@digits{\the\day}}%
72        \immediate\write\dateiliste@file{%
73          \string\ProvidesFile{\jobname.filelist}%
74          [\dateiliste@today\space --- automatically %
75          generated filelist]%
76        }%
```

Die eigentliche Liste wird in einer `longtable` gesetzt. Diese soll drei linksbündig gesetzte Spalten (`l`) und dann eine mit Blocksatz (`X` – mittels ltxtable aus tabularx importiert), welche den restlichen Platz ausfüllt, enthalten. (Linksbündig wäre mir zwar auch für die letzte Spalte lieber, aber das geht mit ltxtable nicht, dafür bräuchte man tabulary.)

```
77        \immediate\write\dateiliste@file{%
78          \string\LTleft=0pt%
79          \string\LTright=0pt%
80          \string\begin{longtable}{lllX}%
```

Die Überschrift – aus übersetzbaren Textteilen, siehe unten, bestehend – wiederholt sich auf jeder Seite (deswegen `\endhead` anstatt `\\`).

```
81            \string\textbf{\fileNameName} &
82            \string\textbf{\dateName} &
83            \string\textbf{\verName} &
84            \string\textbf{\descriptionName}
85            \string\endhead%
86          }%
```

Jetzt kommt die Schleife mit den einzelnen Dateien. Das ist zum Großteil abgekupfert von `\@dofilelist` aus dem LaTeX-Kernel (`ltfiles.dtx`), welches die Liste zum Terminal ausgibt.

```
87        \@for\@currname:=\@filelist\do{%  \@for
```

Zunächst bestimmen wir den genauen Dateinamen – d.h. wir hängen, falls nötig, ein `.tex` an. Außerdem finden wir den zugehörigen Versions-String heraus.

```
88          \filename@parse\@currname
89          \edef\dateiliste@filename{%
90            \filename@base.%
91            \ifx\filename@ext\relax tex\else\filename@ext\fi}%
92          \expandafter\let\expandafter\dateiliste@fileversion
93          \csname ver@\dateiliste@filename\endcsname
```

Jetzt schreiben wir, durch `&` getrennt, die einzelnen Felder raus. Zunächst der Dateiname, ...

```
94          \immediate\write\dateiliste@file{%
95            \dateiliste@filename\space& %
```

... dann entweder ein „—“ (falls kein Versions-String gegeben wurde), ...

```
                \ifx\dateiliste@fileversion\relax
                ---
                \else
```

... oder der Versionsstring selbst, an den ersten beiden Leerzeichen durch `&` getrennt. Dafür verfüttern wir das expandierte `\dateiliste@fileversion` an `\dateiliste@parse@ver`. (Für den Fall, dass da nicht genug Leerzeichen drin sind, sind am Ende noch ein paar `{}` mit Leerzeichen dazwischen – die werden am Ende ja nicht ausgegeben.)

```
                \expandafter\dateiliste@parse@ver
                \dateiliste@fileversion{} {} {} \relax
                \fi
```

Und jetzt noch ein \\, um die Tabellenzeile zu beenden.

```
                \string\\}%
          }%  \@for
```

Nach der Schleife beenden wir die Tabelle und schließen dann die Datei wieder.

```
       \immediate\write\dateiliste@file{\string\end{longtable}}
          \immediate\closeout\dateiliste@file
       }%
```

`\dateiliste@parse@ver` Dieses Makro nimmt zwei durch Leerzeichen getrennte Parameter, und gibt sie, mit zusätzlichen `&`, wieder zurück.

```
       \def\dateiliste@parse@ver##1 ##2 {##1 & ##2 & }%
```

Damit ist der else-Teil und auch das ganze Makro `\printFileList` zu Ende.

```
    }%
}%
```

2.3.3 Anpassbare Texte und Übersetzungen

`\fileListPreamble` `\fileListName` `\fileNameName` `\dateName` `\verName` `\descriptionName` Einige Namen für übersetzbare Texte – standardmäßig auf Englisch.

```
\newcommand*\fileListPreamble{
   Here is the list of all files used during the run of \LaTeX{}
   which produced this document. \footnote{More precisely, it is
      the list of files used one \LaTeX-run before the one which
      produced this document, but after some runs the list should
      stabilize.}
}
\newcommand*\fileListName{List of Files}
\newcommand*\fileNameName{file name}
\newcommand*\dateName{release date}
\newcommand*\verName{version}
\newcommand*\descriptionName{description}
```

`\dateiliste@babel` Hier noch gleich ein paar Übersetzungen. Wir definieren hier ein einmal-Makro, welches für mehrere Sprachen[3] zum jeweiligen Initialisierungsmakro Neudefinitionen dieser fünf Befehle hinzufügt.

```
\newcommand*{\dateiliste@babel}{
```

Zunächst Englisch - das sollte das gleiche wie die Standard-Einstellungen sein.

```
   \addto{\extrasenglish}{%
      \renewcommand*\fileListPreamble{%
         Here is the list of all files used during the run of \LaTeX{}
         which produced this document.\footnote{More precisely, it is
            the list of files used one \LaTeX-run before the one which
            produced this document, but after some runs the list
            should stabilize.}
      }
      \renewcommand*\fileListName{List of Files}%
      \renewcommand*\fileNameName{file name}
      \renewcommand*\dateName{release date}
      \renewcommand*\verName{ver.}
      \renewcommand*\descriptionName{description}
   }%
```

Deutsch mit neuer Rechtschreibung.

```
   \addto{\extrasgerman}{%
      \renewcommand*\fileListPreamble{%
         Hier die Liste aller Dateien, die während des \LaTeX-Laufes,
         welcher dieses Dokument erzeugte, verwendet wurden.
         \footnote{genauer: Es ist die Liste aller Dokumente, die
            einen \LaTeX-Lauf früher verwendet wurden. Aber nach
            einigen Läufen sollte sich die Liste stabilisieren.}
      }
      \renewcommand*\fileListName{Liste der Dateinamen}%
      \renewcommand*\fileNameName{Dateiname}
      \renewcommand*\dateName{Datum}
      \renewcommand*\verName{Ver.}
      \renewcommand*\descriptionName{Beschreibung}
   }%
```

Deutsch mit alter Rechtschreibung: ist das gleiche (hier tauchen keine Fälle mit Änderungen auf.)

```
   \addto{\extrasngerman}{%
      \renewcommand*\fileListPreamble{%
         Hier die Liste aller Dateien, die während des \LaTeX-Laufes,
         welcher dieses Dokument erzeugte, verwendet wurden.
         \footnote{genauer: Es ist die Liste aller Dokumente, die
            einen \LaTeX-Lauf früher verwendet wurden. Aber nach
            einigen Läufen sollte sich die Liste stabilisieren.}
      }
      \renewcommand*\fileListName{Liste der Dateinamen}%
```

[3] Genauer: genau für die Sprachen, welche ich soweit beherrsche, dass ich diese Texte übersetzen konnte.

```
      \renewcommand*\fileNameName{Dateiname}
      \renewcommand*\dateName{Datum}
      \renewcommand*\verName{Ver.}
      \renewcommand*\descriptionName{Beschreibung}
   }%
```

Für die Verwender der Internationalen Sprache (siehe `www.esperanto.de`):

```
   \addto{\extrasesperanto}{%
      \renewcommand*\fileListPreamble{%
         Jen listo de \^ciuj dosieroj, kiuj estis uzitaj dum
         la \LaTeX-rulo, kiu produktis tiun \^ci dokumenton.
         \footnote{Pli precize: estas la listo de dosieroj uzitaj
            unu rulon anta\u{u} tiu, kiu produktis tiun \^ci
            dokumenton. Sed kutime post kelkaj ruloj la listo
            devus stabili\^gi.}
      }
      \renewcommand*\fileListName{Listo de dosieroj}%
      \renewcommand*\fileNameName{dosiernomo}
      \renewcommand*\dateName{dato}
      \renewcommand*\verName{versio}
      \renewcommand*\descriptionName{priskribo}
   }%
```

Am Ende der Ausführung von `\dateiliste@babel` vernichtet der Befehl sich selbst. Das spart etwas Speicher, und sorgt dafür, dass er nicht versehentlich mehrfach ausgeführt wird (auch wenn das wohl nicht schädlich wäre).

```
   \let \dateiliste@babel = \relax%
}%
```

Wir untersuchen jetzt, ob babel schon geladen wurde. Diese Fallunterscheidung ist notwendig, weil der Code von `\dateiliste@babel` zwar das Paket benötigt (also nach ihm ausgeführt werden sollte), aber nicht einfach direkt mit `\AtBeginDocument` ans Ende geschoben werden sollte, da er (falls babel schon vor diesem Paket geladen wurde) dort nach dem babel-Code (der die Sprache auswählt) kommen würde, und damit mehr nichts bewirkt.

Falls babel jetzt schon geladen wurde, ...

```
\@ifpackageloaded{babel}
{%
```

... informieren wir es sofort über die neuen Namen, die beim Sprachwechsel bitte angepasst werden sollten.

```
   \dateiliste@babel%
}%
```

Ansonsten verschieben wir das zum Beginn des Dokumentes (und machen das auch dann nur, wenn babel inzwischen geladen wurde – ansonsten ist das ganze ja überflüssig, und `\addto` gibt es auch nicht, also können wir dann `\dateiliste@babel` vernichten).

```
{%
   \AtBeginDocument{%
```

```
      \@ifpackageloaded{babel}{%
         \dateiliste@babel%
      }{%
         \let \dateiliste@babel = \relax
      }%
   }%
}%
```

2.4 Hauptdatei in die Dateiliste

\mainFileToList Zunächst sehen wir nach, ob es eine Datei mit Namen ⟨*jobname*⟩`.tex` gibt.

```
\newcommand*{\mainFileToList}{%  \mainFileToList
   \IfFileExists{\jobname.tex} {%
```

Falls ja, dann ist das höchstwahrscheinlich die Haupt-Datei des Dokumentes, und taucht wahrscheinlich – nämlich, wenn sie auf der Kommandozeile oder mit `\input` ⟨*jobname*⟩`.tex` anstatt `\input{`⟨*jobname*⟩`.tex}` geladen wurde – nicht in der Dateiliste auf. Um den letzten Fall – soweit möglich – abzufangen, der folgende Code.

Wir testen, ob `\scantokens` definiert ist. Falls nicht, können wir nichts mehr machen, und merken uns den Dateinamen in einem neuen Makro. Dabei liefert `\jobname` leider die Zeichen (auch die Buchstaben) in Kategorie 12 (other) anstatt 11 (letter) wie die in `\@filelist`, wodurch wir das dortige Vorkommen nicht mittels `\@removeelement` entfernen können.

```
      \begingroup
      \@ifundefined{scantokens}
      {%
         \edef\dateiliste@mainfile{\jobname.tex}%
      }
      {%  ( = \scantokens defined)
```

In ε-TEX gibt es dagegen den `\scantokens`-Befehl, welcher es ermöglicht, im Speicher von TEX vorliegende Token neu aus einer Pseudo-Datei einzulesen. Wenn er definiert ist (das haben wir gerade getestet), rufen wir es hier auf – mit einigen `\expandafter`, um nur den `\jobname` vor der `\scantokens`-Ausführung zu expandieren, und nicht die Token `\edef\dateiliste@mainfile{` und `.tex}` drumherum. Das `\edef` wird dann also mit Catcode-11-Buchstaben (also „richtigen“) im Dateinamen ausgeführt. (Das `\makeatletter` und `\makeatother` ist notwendig, um das `@` auch als Buchstabe zuzulassen und somit `\dateiliste@mainfile` als einzelnen Makronamen anzusehen. Zu dem Zeitpunkt, zu dem `\scantokens` ausgeführt wird, sind ja die Dokumenten-Catcodes in Kraft, nicht die einer Package-Datei.)

```
         \scantokens
         \expandafter{%
            \expandafter\makeatletter
            \expandafter\edef
            \expandafter\dateiliste@mainfile
            \expandafter{%
```

```
                \jobname
                .tex}%
            \makeatother
        }%
```

Jetzt können wir erfolgreich mit `\@removeelement` (aus dem LaTeX-Kernel) das Vorkommen von ⟨*jobname*⟩`.tex` entfernen (falls der Name dort vorhanden ist – wahrscheinlich nicht). [4]

```
        \@expandtwoargs\@removeelement{\dateiliste@mainfile}%
        \@filelist\@filelist
    }
```

Anschließend fügen wir den Dateinamen an den *Anfang* der Liste an. (In dem seltenen (?) Fall, dass es schon da war und wir nicht ε-TeXverwenden, haben wir Pech und der Name taucht zweimal auf.)

```
    \xdef\@filelist{%
        \dateiliste@mainfile,\@filelist
    }%
    \endgroup
```

Falls ⟨*jobname*⟩`.tex` nicht existiert, ist dies sicher nicht die Hauptdatei. Dann haben wir es entweder mit einer `.dtx`-Datei zu tun (die sowieso durch das doppelte Einlesen noch einmal auftaucht), oder irgendeinen anderen Fall, den ich nicht vorhersehen kann. Also machen wir dann nichts.

```
  }{%
    \relax
  }%
}%
```

Am Ende des Dokumentes (aber noch vor dem Aufruf von `\@writefilelist`, der von `\printFileList` hinzugefügt wird) rufen wir, sofern die passende Option gesetzt war, das eben definierte Makro auf.

```
\if@dateiliste@addMain
  \AtEndDocument{\mainFileToList}
\fi
```

2.5 Schluss

```
\endinput
```
⟨/package⟩

[4] Ein alternativer Ansatz wäre es, `\@filelist` mittels `\detokenize` (auch aus ε-TeX) komplett in Cat-12-Zeichen umzuwandeln – auch dann arbeitet `\@removeelement` problemlos, allerdings sind die Nebenwirkungen auf die Liste (und eventuelle weitere Verwender) wohl größer ...

3 Liste der Änderungen

4 Index

Schräggedruckte Nummern verweisen auf die Seite, auf der der Eintrag beschrieben ist, unterstrichene Nummern zeigen auf die Zeilennummer der Definition, sonstige Zahlen auf die Zeilennummer einer Verwendung.