

# makedtx v1.3: a Perl script to help create a DTX file from source code

Nicola Talbot

<http://theoal.cmp.uea.ac.uk/~nlct/>

February 19, 2018

## Contents

<b>1 Introduction</b>	<b>2</b>
<b>2 Installation</b>	<b>2</b>
<b>3 makedtx.pl</b>	<b>3</b>
3.1 Compulsory Arguments . . . . .	3
3.2 Options . . . . .	3
<b>4 The creatdtx Package</b>	<b>6</b>
<b>5 Examples</b>	<b>6</b>
<b>6 Troubleshooting</b>	<b>10</b>
6.1 Known Bugs . . . . .	11
6.2 Possible errors encountered using makedtx.pl . . . . .	11
<b>A Perl Regular Expressions</b>	<b>11</b>
<b>B The Code</b>	<b>12</b>
<b>References</b>	<b>12</b>
<b>Index</b>	<b>13</b>

## Abstract

The **makedtx** bundle is provided to help developers to write the code and documentation in separate files, and then combine them into a single DTX file for distribution. It automatically generates the character table, and also writes the associated installation (.ins) script.

# 1 Introduction

Authors of  $\text{\LaTeX}$  2 $\epsilon$  class files or packages are encouraged to bundle their source and documentation together into a single DTX file. This makes distribution much easier, as users need only download the DTX file and possibly a corresponding installation script (INS file) instead of a multitude of `.sty`, `.cls`, `.def` etc files. However, having the documentation and code bundled together can cause problems if a developer wants to, say, use `ispell` to spell check the documentation, or convert the documentation to a format other than DVI, PostScript or PDF (such as HTML).

Why should I want to convert my documentation to HTML when I can just use PDF $\text{\LaTeX}$ ? The more general purpose packages that I write (such as `datetime` and `glossary`) I upload to CTAN, however most of the packages I write are specific to the School of Computing Sciences at the University of East Anglia, so these I keep on my web site, and as some of the faculty either don't have a PDF plug in or prefer to view HTML rather than PDF documents, I have taken to writing both PDF and HTML versions of my package documentation. However,  $\text{\LaTeX}$ 2HTML doesn't work on a `.dtx` file so I used to convert them manually which is fine for one or two small documents, but becomes rather cumbersome as soon as I have large documents or a lot of packages. Therefore I decided to write the documentation separately, and use a Perl script to bundle everything together. It also has the added convenience in that I don't have to keep copying and pasting the character table every time I write a new package, and it saves the laborious task of writing the installation script<sup>1</sup>.

This document is structured as follows: Section 2 describes how to install the `makedtx` bundle, Section 3 gives an overview of the `makedtx.pl` Perl script, Section 4 describes the `creatdtx` package, Section 5 illustrates the use of the `makedtx` bundle with examples and Section 6 gives a list of possible errors and their solutions.

# 2 Installation

You need to download both `makedtx.dtx` and `makedtx.ins`, and run the installation script through  $\text{\LaTeX}$ :

```
latex makedtx.ins
```

The following files will be created:

`makedtx.pl` Perl script

`creatdtx.sty`  $\text{\LaTeX}$  package for use with `makedtx.pl`

`creatdtx.perl` Corresponding Perl script for use with `LaTeX2HTML`

If you are using UNIX/Linux etc you will need to make `makedtx.pl` executable using `chmod`:

```
chmod a+x makedtx.pl
```

and place it somewhere on your path. If `perl` is located somewhere other than `/usr/bin/` you will need to edit the first line of `makedtx.pl`. (If you don't

---

<sup>1</sup>or at least, it's laborious if there are rather a lot of files associated with a package

know where `perl` is located, you can use the command: `which perl`.) The package `creatdtx.sty` needs to be placed somewhere on the L<sup>A</sup>T<sub>E</sub>X path and `creatdtx.perl` should be placed in a directory searched by L<sup>A</sup>T<sub>E</sub>X2HTML. (See the L<sup>A</sup>T<sub>E</sub>X2HTML documentation for details.)

### 3 makedtx.pl

The Perl script `makedtx.pl` has the following syntax:

```
makedtx.pl [options] -src "<expr1>=><expr2>" -doc <filename> <basename>
```

#### 3.1 Compulsory Arguments

The very last argument *<basename>* is the basename of the `.dtx` and `.ins` files you want to create. The `-doc <filename>` switch indicates the file containing the documentation and `-src "<expr1>=><expr2>"` indicates the original source file(s), given by *<expr1>*, and the corresponding file name when it has been extracted from the `.dtx` file, given by *<expr2>*. This switch is a little complicated, so it's best described using examples.

Suppose you have your documentation in the file `foodoc.tex`, and the original source code is in the file `foosrc.sty`. You want to create the files `foo.dtx` and `foo.ins`. When you L<sup>A</sup>T<sub>E</sub>X `foo.dtx` you want the documentation as specified in `foodoc.tex` and when you L<sup>A</sup>T<sub>E</sub>X `foo.ins` you want the file `foo.sty` to be created, using the code specified in `foosrc.sty`. You will need to do:

```
makedtx.pl -src "foosrc\sty=>foo.sty" -doc foodoc.tex foo
```

You may have multiple invocations of the `-src` switch. For example, suppose you also have the file `barsrc.sty` which you want to be extracted from the `.dtx` file as `bar.sty`, you can do:

```
makedtx.pl -src "foosrc\sty=>foo.sty" -src "barsrc\sty=>bar.sty" -doc foodoc.tex foo
```

Alternatively, you can use Perl-type regular expressions:

```
makedtx.pl -src "(.*)src\sty=>\1.sty" -doc foodoc.tex foo
```

(Note the use of double quotes to prevent shell expansion.) Appendix A gives a brief overview of Perl regular expressions for the uninitiated.

#### 3.2 Options

`-h` or `-help` Prints on-line help, and exits.

`-version` Prints version number, and exits.

`-v` Uses verbose mode.

`-dir <name>` Specifies directory containing source files, as specified by the `-src` switch. For example, suppose you have source files `foo.sty`, `bar.sty` in the subdirectory `sourcefiles` you can do:

```
makedtx.pl -dir sourcefiles -src "(.*)\sty=>\1.sty" -doc foodoc.tex foo
```

- op *<character>* sets the Perl pattern matching operator (the default is set to = symbol since the / character is used as the directory divider).
- askforoverwrite uses \askforoverwritetrue in the installation script.
- noaskforoverwrite uses \askforoverwritefalse in the installation script (default).
- noins Don't create the installation script (.ins file). This is useful if you want to tweak the file manually and you don't want your modifications overwritten.
- preamble *<text>* Set the preamble to text. The default preamble is:
 

```

      <basename>.dtx
      Copyright <date> <author>
      This work may be distributed and/or modified under the
      conditions of the LaTeX Project Public License, either version 1.3
      of this license or (at your option) any later version.
      The latest version of this license is in
      http://www.latex-project.org/lppl.txt
      and version 1.3 or later is part of all distributions of LaTeX
      version 2005/12/01 or later.
      This work has the LPPL maintenance status 'maintained'.
      The Current Maintainer of this work is <author>.
      This work consists of the files <basename>.dtx and
      <basename>.ins and the derived files <file list>.
      where <date> is the copyright date, and <author> is the author's name (see
      below).

      Note that this has been changed as of makedtx v0.94. Older versions of makedtx do not conform to any of the free licenses.
      
```
- license *<name>* This sets the preamble to the license text for the license *<name>*. Currently *<name>* may be either lppl (the default, which produces the default preamble detailed above), bsd or gpl. You may use either -license or -preamble, but not both. (If you have used the -preamble switch, the -license switch will be ignored.)
- postamble *<text>* Set the postamble to text. If this is omitted the \postamble command is omitted from the installation script.
- author *<name>* The author's name (as used in the default preamble). If omitted the user's name is used.
- date *<text>* The copyright date (as used in the default preamble). If omitted the current year is used.

**-stopeventually** *<text>* Insert *<text>* into the argument of `\StopEventually`. For example: `-stopeventually "\\PrintIndex"` will result in the line: `\StopEventually{\PrintIndex}`. If `makedtx.pl` encounters a `\StopEventually` command within the document, this will be used instead. If there is no `\StopEventually` command in the document and the `-stopeventually` switch is absent `\StopEventually{<>}` will be inserted in the DTX file.

**-prefinale** *<text>* Inset *<text>* immediately prior to `\Finale` in the dtx file.

**-setambles** "*<expr>=><text>*" Sets the pre- and postambles for files matching *<expr>* within the `\file` command in the installation script. To illustrate this, let's suppose you have source files `foo.sty`, `bar.sty` and `foobar.pl` in the subdirectory `sourcefiles`. Since `foo.sty` and `bar.sty` are L<sup>A</sup>T<sub>E</sub>X files, they should have pre- and postambles, but `foobar.pl` is a Perl file, and since the percent symbol (%) is not a comment character in Perl, there should be no pre- and postambles for this file. Therefore you would need to do something like:

```
makedtx.pl -dir sourcefiles -src "(.*)\.sty=>\1.sty" -src "foobar.pl=>foobar.pl"
-setambles "foobar\.pl=>\nopreamble\nopostamble" -doc foodoc.tex foo
```

(Note that the line is only broken to fit it onto the page, and there should be no line break when entering at the command prompt.)

If the argument to `-setambles` contains the string `\nopreamble`, the character table will be excluded from the corresponding files. So, in the above example, when you do: `latex foo.ins` the resulting files `foo.sty` and `bar.sty` will contain the character table, but `foobar.pl` won't. (If for some reason you don't want a preamble but you do want the character table included use `\usepreamble\empty` instead of `\nopreamble`. Conversely, if you want a preamble but don't want the character table do something like `\nopreamble\usepreamble\defaultpreamble`.)

Note that the `=><text>` part is optional. If it is omitted, *<text>* is assumed to be empty.

**-macrocode** "*<expr>*" If source file matches the Perl regular expression given by *<expr>*, the source code is inserted into a `macrocode` environment in the DTX file.

Please note that the source code is supposed to be text insensitive to end of line format, as a matter of fact `makedtx.pl` will convert all end of lines of this source code to LF's.

Please note also that only the source files given by the `-src` switch are considered, the `-macrocode` switch by itself does not cause the source file to be inserted into the DTX file, it only affects *how* it is inserted.

**-comment** "*<expr>*" If the source file matches the Perl regular expression given by *<expr>*, the source code will be inserted between `\iffalse` `\fi` commands. The contents of this file will be included in the DTX file, but will be excluded from the documentation. Since this is provided mainly for non-TeX files (such as Perl scripts) the `-comment` switch will typically need to be used in conjunction with `-macrocode`.

- `-codetitle "<title>"` This sets the title for the code section. The default is `The Code`.
- `-nodrivendoc` This places the documentation body inside the driver block, as is ie. without commenting each line with a leading `%`, instead of placing it after the driver block as a docstrippable `LATEX` code, ie. with inserting a leading `%` comment mark into each line. The default is driven documentation body, i.e. this switch not active by default.
- `-drivendoc` This is the opposite of the `-nodrivendoc` switch. This switch is active by default — ie. documentation body is read by `\DocInput` rather than being right in the driver block.

## 4 The creatdtx Package

The documentation source code, as specified using the `-doc` switch will typically be a standard `LATEX` document using the `ltxdoc` class file. Unlike the DTX file, there is no `\DocInput` command, and the lines do not begin with a percent symbol, which means that the document can be, say, passed to the `LATEX2HTML` converter, or some other application that would otherwise be confused by a DTX file. The `creatdtx` package can be used in this document using

```
\usepackage{creatdtx}
```

although this package will be not be included in the DTX file by `makedtx.pl`. There is only one command defined in this package: `\ifmakedtx{<dtx text>}{<non dtx text>}`. The first argument `<dtx text>` will be copied to the DTX file by `makedtx.pl`, but the second argument `<non dtx text>` won't. However, if you `LATEX` the document, the first argument will be ignored, and the second argument will be used.

For example, if your code (in `foodoc.tex`) contains the line:

```
\ifmakedtx{}{\usepackage{html}}
```

the `html` package will only be loaded if you `LATEX` `foodoc.tex`, but not when you `LATEX` `foo.dtx`.

The Perl script `creatdtx.perl` ignores the following commands (and any associated arguments): `\OnlyDescription`, `\RecordChanges`, `\MakeShortVerb`, `\DeleteShortVerb`, `\DoNotIndex`, `\EnableCrossrefs`, `\CodelineIndex`, `\GetFileInfo`, `\PrintChanges`, `\changes`, `\Checksum`, `\DescribeMacro` and `\DescribeEnvironment`. So even if you don't use the `\ifmakedtx` command, using the `creatdtx` package will help ensure that extraneous text does not appear in the HTML document when using `LATEX2HTML`.

As from version 0.93b, `creatdtx.perl` also defines the commands `\cs`, `\marg`, `\oarg` and `\parg`, since there is no `LATEX2HTML` implementation of the `ltxdoc` class file.

## 5 Examples

Let's first consider a very simple example. Suppose you want to create a package that redefines `\today` so that the date is displayed in the form: `yyyy-m-d`. Let's

call this package dashdate. The file dashdate.sty should look something like:

```
% First define package:
%   \begin{macrocode}
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{dashdate}
%   \end{macrocode}
% Redefine |\today| command:
%   \begin{macrocode}
\renewcommand{\today}{\the\year-\the\month-\the\day}
%   \end{macrocode}
```

Now let's make some (very brief) documentation. Let's call the file, say manual.tex<sup>2</sup>:

```
\documentclass{ltxdoc}
\usepackage{creatdtx}

\begin{document}
\title{A Sample Package}
\author{AN Other}
\maketitle
```

The \texttt{dashdate} package redefines |\today| to produce the current date in the form: yyyy-m-d.

```
\end{document}
```

Suppose you have saved dashdate.sty and manual.tex in the subdirectory source. You can now create the .dtx and .ins file using the command:

```
makedtx.pl -author "AN Other" -dir source -src "dashdate\sty=>dashdate.sty"
-doc source/manual.tex dashdate
```

The file dashdate.dtx is created, and contains the following code:

```
%\iffalse
% dashdate.dtx generated using makedtx.pl version 0.9b (c) Nicola Talbot
% Command line args:
%   -dir "source"
%   -src "dashdate\sty=>dashdate.sty"
%   -author "AN Other"
%   -doc "source/manual.tex"
%   dashdate
% Created on 2005/2/10 22:22
%\fi
%\iffalse
%<*package>
%% \CharacterTable
%% {Upper-case   \A\B\C\D\E\F\G\H\I\J\K\L\M\N\O\P\Q\R\S\T\U\V\W\X\Y\Z
%%   Lower-case   \a\b\c\d\e\f\g\h\i\j\k\l\m\n\o\p\q\r\s\t\u\v\w\x\y\z
%%   Digits        \0\1\2\3\4\5\6\7\8\9
%%   Exclamation   \!     Double quote  \"     Hash (number) \#
%%   Dollar         \$     Percent      \%     Ampersand     \&
```

---

<sup>2</sup>Note: if you want to use L<sup>A</sup>T<sub>E</sub>X2HTML on this document, you will need to use, e.g., \verb!\today! instead of |\today| since it doesn't recognise \MakeShortVerb.

```

%% Acute accent \' Left paren \(\ Right paren \)
%% Asterisk \* Plus \+ Comma \,
%% Minus \- Point \. Solidus \/
%% Colon \: Semicolon \; Less than \<
%% Equals \= Greater than \> Question mark \?
%% Commercial at \@ Left bracket \[ Backslash \backslash
%% Right bracket \] Circumflex \^ Underscore \_
%% Grave accent ` Left brace \{ Vertical bar \|
%% Right brace \} Tilde \~}
%</package>
%\fi
% \iffalse
% Doc-Source file to use with LaTeX2e
% Copyright (C) 2005 AN Other, all rights reserved.
% \fi
% \iffalse
%<*>driver>
\documentclass{ltxdoc}

\begin{document}
\DocInput{dashdate.dtx}
\end{document}
%</driver>
%\fi
%\title{A Sample Package}
%\author{AN Other}
%\maketitle
%
%The \texttt{dashdate} package redefines "\today"
%to produce the current date in the form: yyyy-m-d.
%\end{document}
%
%\StopEventually{}
%\section{The Code}
%\iffalse
% \begin{macrocode}
%<*>dashdate.sty>
% \end{macrocode}
%\fi
% First define package:
% \begin{macrocode}
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{dashdate}
% \end{macrocode}
% Redefine |\today| command:
% \begin{macrocode}
\renewcommand{\today}{\the\year-\the\month-\the\day}
% \end{macrocode}
%\iffalse
% \begin{macrocode}
%</dashdate.sty>
% \end{macrocode}
%\fi
%\Finale

```



```
\endinput
```

The installation file `dashdate.ins` looks like:

```
% dashdate.ins generated using makedtx.pl version 0.94b 2007/8/19 22:22
\input docstrip
```

```
\preamble
dashdate.dtx
Copyright 2007 AN Other
```

```
This work may be distributed and/or modified under the
conditions of the LaTeX Project Public License, either version 1.3
of this license or (at your option) any later version.
The latest version of this license is in
  http://www.latex-project.org/lppl.txt
and version 1.3 or later is part of all distributions of LaTeX
version 2005/12/01 or later.
```

```
This work has the LPPL maintenance status 'maintained'.
```

```
The Current Maintainer of this work is AN Other.
```

```
This work consists of the files dashdate.dtx and
dashdate.ins and the derived file dashdate.sty.
\endpreamble
```

```
\askforoverwritefalse
```

```
\generate{\file{dashdate.sty}{\usepreamble\defaultpreamble
\usepostamble\defaultpostamble\from{dashdate.dtx}{dashdate.sty,package}}
}
```

```
\endbatchfile
```

Note that the command `\usepackage{creatdtx}` has not been transcribed to `dashdate.dtx` (although in this simple example it's not really needed).

Now let's extend the example: suppose you want to create an analogous Perl script for use with L<sup>A</sup>T<sub>E</sub>X2HTML. This will need to be called `dashdate.perl` and will look something like:

```
package main;

sub do_cmd_today{
    local($_) = @_;
    local($today) = &get_date();
    $today =~ s|(\d+)/(\d+)/(\d+)|$3-$1-$2|;
    "$today$_";
}

1;
```

You will now need to call `makedtx.pl` as follows:

```
makedtx.pl -author "AN Other" -dir source -src "dashdate\*.sty=>dashdate.sty"
```

```
-src "dashdate\perl=>dashdate.perl"
-setambles "dashdate\perl=>\nopreamble\nopostamble"
-macrocode "dashdate\perl" -doc source/manual.tex dashdate
```

(Note that the line is only broken to allow it to fit onto the page, there should be no line break when you enter it on the command line.) Alternatively, you could save typing and do:

```
makedtx.pl -author "AN Other" -dir source -src "dashdate\.*=>dashdate\1"
-setambles "dashdate\perl=>\nopreamble\nopostamble" -macrocode "dashdate\perl"
-doc source/manual.tex dashdate
```

Note the use of the `-setambles` switch which suppresses the insertion of text at the start and end of the Perl script which would only confuse Perl. Note also the use of the `-macrocode` switch. This is not needed for `dashdate.sty` since it has already been included in the source code, but since `%` is not a comment character in Perl, the `macrocode` environment is not included in the source code, and needs to be added. (If you are unfamiliar with DocStrip and the use of the `macrocode` environment, I suggest you read either *A guide to L<sup>A</sup>T<sub>E</sub>X* [2, Appendix D] or *The L<sup>A</sup>T<sub>E</sub>X companion* [1, Chapter 14].)

It's likely that you may not want the Perl code to appear in the document, but you still want it included in the DTX file. In addition to the `-macrocode` switch, you would then also need to use the `-comment` switch:

```
makedtx.pl -author "AN Other" -dir source -src "dashdate\.*=>dashdate\1"
-setambles "dashdate\perl=>\nopreamble\nopostamble" -macrocode "dashdate\perl"
-comment "dashdate\perl" -doc source/manual.tex dashdate
```

As another example, consider the `datetime` package. Version 2.42 of this package has 2 `.sty` files and 42 `.def` files. The documentation is written in the file `manual.tex`, and the `.sty` and `.def` files are saved in a subdirectory called `source`. Since these are the only files in this directory, they can easily be merged into one `.dtx` file using:

```
makedtx.pl -author "Nicola Talbot" -dir source -src "(.+)\.(.+)=>\1.\2"
-doc manual.tex datetime
```

This creates the files `datetime.dtx` and `datetime.ins` which can then be distributed. The PDF version of the documentation is obtained by doing:

```
pdflatex datetime.dtx
```

and the HTML version (`manual.html`) is obtained by doing:

```
latex2html -split 0 -nonavigation -nofootnode -numbered_footnotes -noinfo manual
```

Any minor differences between the HTML and PDF versions are dealt by using `\ifmakedtx` in the original file `manual.tex`.

## 6 Troubleshooting

The `makedtx` bundle has only been tested under Linux using Perl v5.6.0. There are no guarantees whether or not it will work on other operating systems or on different versions (in fact, there are no guarantees or warranties at all).

## 6.1 Known Bugs

It's possible to confuse `makedtx.pl` by placing either the command `\end{<document>}` or the command `\ifmakedtx` in a `\verb` command, or by having the `\ifmakedtx` command on the same line as `\begin{<document>}`. You will also need to take care about lines beginning with a percent symbol (%) in the documentation, as this will get converted into a line beginning with %% in the `.dtx` file, which has a special meaning. Either place a space immediately prior the percent symbol, or do `\relax%` if you really don't want the extra space (or place your comment in an `\iffalse ... \fi` conditional).

## 6.2 Possible errors encountered using makedtx.pl

Note: be careful to use double quotes around arguments that contain characters that the shell might try interpreting, e.g. `*` or `>`.

Syntax error messages:

1. No document source specified (missing `-doc`)  
You must use the `-doc` switch.
2. No source code specified (missing `-src`)  
You must specify at least one `-src` switch.
3. No basename specified  
You must specify the basename of the `.dtx` and `.ins` files. This should be the last argument passed to `makedtx.pl`.
4. `-src ... argument invalid` (no output file specified)  
You have omitted the `=>` separator in the argument of the `-src` switch.
5. `-src argument ... invalid` (too many `=>` specified)  
You have used too many `=>` separators in the argument of the `-src` switch.  
(Similarly for the `-setambles` switch.)

## A Perl Regular Expressions

This section gives a very brief overview of Perl regular expressions. For more detail, look at the Perl documentation (use `man perlre` for the man page.)

<code>\</code>	Quote the next character
<code>.</code>	Match any character
<code> </code>	Alternation
<code>()</code>	Grouping
<code>[]</code>	Character class
<code>*</code>	Match 0 or more times
<code>+</code>	Match 1 or more times
<code>?</code>	Match 1 or 0 times
<code>{n}</code>	Match exactly <code>n</code> times
<code>{n,}</code>	Match at least <code>n</code> times
<code>{n,m}</code>	Match at least <code>n</code> but no more than <code>m</code> times.

In the replacement text, a backslash followed by a number  $\langle n \rangle$  indicates the text from the  $\langle n \rangle$ th group.

For example, suppose you have the following files:

```
abcsrc.sty
abcsrc.bst
abcsrc.perl
foosrc.sty
foobarsrc.sty
```

then if you pass the following switch to `makedtx.pl`:

- `-src "abcsrc\[styb]+\>abc.\1"` will be equivalent to:

```
-src "abcsrc.sty=>abc.sty" -src "abcsrc.bst=>abc.bst"
```

since `[styb]+` will match one or more of the letters `styb` (so it will match `sty` and `bst`). `\1` indicates the text found in the first group, which in this example will either be `sty` or `bst`.

- `-src "abcsrc\[.]+\>abc.\1"` will be equivalent to:

```
-src "abcsrc.sty=>abc.sty" -src "abcsrc.bst=>abc.bst" -src "abcsrc.perl=>abc.perl"
```

Note that a full stop represents any character so `.+` means any string of length 1 or more, whereas `\.` means an actual full stop character.

- `-src "foo(.*)src\[.].sty=>foo\1.sty"` will be equivalent to:

```
-src "foosrc.sty=>foo.sty" -src "foobarsrc.sty=>foobar.sty"
```

- `-src "(.+)src\[.].(.)=>\1.\2"` will be equivalent to

```
-src "abcsrc.sty=>abc.sty"
-src "abcsrc.bst=>abc.bst"
-src "abcsrc.perl=>abc.perl"
-src "foosrc.sty=>foo.sty"
-src "foobarsrc.sty=>foobar.sty"
```

## B The Code

This is the code for the `creatdtx` package. This package should not be used in a `.dtx` file. First define the package:

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{creatdtx}[2005/02/11 v0.9b (N.L.C. Talbot)]
```

Define `\fmakedtx`. The first argument will always be ignored by  $\text{\LaTeX}$ :

```
3 \newcommand{\fmakedtx}[2]{#2}
```

Redefine `\StopEventually` so that it simply prints its argument

```
4 \providecommand{\StopEventually}{}
5 \renewcommand{\StopEventually}[1]{#1}
```

## References

- [1] The L<sup>A</sup>T<sub>E</sub>X companion. Michel Goossens, Frank Mittelbach and Alexander Samarin. Addison-Wesley 1993.
- [2] A guide to L<sup>A</sup>T<sub>E</sub>X. Helmut Kopka and Patrick W. Daly. Addison-Wesley 1998.

## Change History

0.9		0.94	
General: Initial beta release . . . . .	2	General: default preamble changed to one conforming with the LPPL . . . . .	4
0.91		license switch added . . . . .	4
General: prefinale switch added . . . . .	5		
0.93		1.3	
General: Added codetitle switch . . . . .	6	General: Added drivendoc switch . . . . .	6
comment switch added . . . . .	5	Added nodrivendoc switch . . . . .	6
version switch added . . . . .	3		

## Index

<b>A</b>		<b>I</b>	
\askforoverwritefalse . . . . .	4	\ifmakedtx . . . . .	6, 11
\askforoverwritetrue . . . . .	4		
<b>C</b>		<b>M</b>	
\changes . . . . .	6	makedtx switches	
\Checksum . . . . .	6	-askforoverwrite . . . . .	4
\CodelineIndex . . . . .	6	-author . . . . .	4
\cs . . . . .	6	-codetitle . . . . .	6
		-comment . . . . .	5, 10
<b>D</b>		-date . . . . .	4
\DeleteShortVerb . . . . .	6	-dir . . . . .	3
\DescribeEnvironment . . . . .	6	-doc . . . . .	3, 6, 11
\DescribeMacro . . . . .	6	-drivendoc . . . . .	6
\DocInput . . . . .	6	-h . . . . .	3
\DoNotIndex . . . . .	6	-help . . . . .	3
		-license . . . . .	4
<b>E</b>		-macrocode . . . . .	5, 10
\EnableCrossrefs . . . . .	6	-noaskforoverwrite . . . . .	4
		-nodrivendoc . . . . .	6
<b>G</b>		-noins . . . . .	4
\GetFileInfo . . . . .	6	-op . . . . .	4
		-postamble . . . . .	4
		-preamble . . . . .	4
		-prefinale . . . . .	5
		-setambles . . . . .	5, 10, 11
		-src . . . . .	3, 5, 11
		-stopeventually . . . . .	5
		-v . . . . .	3
		-version . . . . .	3
		\MakeShortVerb . . . . .	6, 7
		\marg . . . . .	6
		<b>O</b>	
		\oarg . . . . .	6
		\OnlyDescription . . . . .	6
		<b>P</b>	
		\parg . . . . .	6
		\postamble . . . . .	4
		\PrintChanges . . . . .	6
		<b>R</b>	
		\RecordChanges . . . . .	6
		<b>S</b>	
		\StopEventually . . . . .	5