makedtx : a Perl script to help create a DTX file from source code

Nicola Talbot

11th February 2005

Contents

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 LaTeX 2_ε 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 PDFLaTeX? 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 the PDF documents, I have taken to writing both PDF and HTML versions of my package documentation. However, LaTeX2HTML 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[1].

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 LaTeX:

```
latex makedtx.ins
```

The following files will be created:

`makedtx.pl` Perl script

`creatdtx.sty` 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 know where `perl` is located, you can use the command: `which perl`.) The package `creatdtx.sty` needs to be placed somewhere on the LaTeX path and `creatdtx.perl` should be placed in a directory searched by LaTeX2HTML. (See the LaTeX2HTML documentation for details.)

3 makedtx.pl

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

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

[1] or at least, it's laborious if there are rather a lot of files associated with a package

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 LaTeX `foo.dtx` you want the documentation as specified in `foodoc.tex` and when you LaTeX `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.

-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:

```
Copyright (C) ⟨date⟩ ⟨author⟩, all rights reserved.  If you modify
this file, you must change its name first.  You are NOT ALLOWED
to distribute this file alone.  You are NOT ALLOWED to take money
for the distribution or use of either this file or a changed version,
except for a nominal charge for copying etc.
```

where ⟨*date*⟩ is the copyright date, and ⟨*author*⟩ is the author's name (see below).

-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.

-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 LaTeX 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`, although I can't think of a good reason for wanting either of these situations.)

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.

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.

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`[2]:

```
\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     \&
%%   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  \[     Backslash     \\
%%   Right bracket \]     Circumflex    \^     Underscore    \_
%%   Grave accent  \`     Left brace    \{     Vertical bar  \|
%%   Right brace   \}     Tilde         \~}
%</package>
%\fi
% \iffalse
```

[2]Note: if you want to use LaTeX2HTML on this document, you will need to use, e.g., `\verb!\today!` instead of `|\today|` since it doesn't recognise `\MakeShortVerb`.

```
% 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}
%    \begin{macrocode}
%<*dashdate.sty>
%    \end{macrocode}
% 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}
%    \begin{macrocode}
%</dashdate.sty>
%    \end{macrocode}
%\Finale
\endinput
```

The installation file `dashdate.ins` looks like:

```
% dashdate.ins generated using makedtx.pl version 0.9b 2005/2/10 22:22
\input docstrip

\preamble
Copyright (C) 2005 AN Other, all rights reserved.
If you modify this file, you must change its name first.
You are NOT ALLOWED to distribute this file alone. You are NOT
ALLOWED to take money for the distribution or use of either this
file or a changed version, except for a nominal charge for copying
etc.
\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 LaTeX2HTML. This will need to be called `dashdate.perl` and will look something like:

```
package main;

sub do_cmd_today{
   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 LaTeX* [2, Appendix D] or *The LaTeX companion* [1, Chapter 14])

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.)

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

References

[1] The LaTeX companion. Michel Goossens, Frank Mittelbach and Alexander Samarin. Addison-Wesley 1993.

[2] A guide to LaTeX. Helmut Kopka and Patrick W. Daly. Addison-Wesley 1998.

Contact Details

Dr Nicola Talbot
School of Computing Sciences
University of East Anglia
Norwich. NR4 7TJ
United Kingdom
http://theoval.cmp.uea.ac.uk/~nlct/