The `dozenal` Package

Donald P. Goodman III

June 6, 2009

Abstract

The `dozenal` package provides some simple mechanisms for working with the dozenal (duodecimal or "base 12") numerical system. It redefines all basic LaTeX counters, provides a command for converting arbitrary decimal numbers into dozenal, and provides new, real Metafont characters for ten and eleven, though the commands for producing them can be redefined to produce any figure. This package uses the `\basexii` algorithm by David Kastrup.

Contents

1 Introduction

While most would probably call it at best overoptimistic and at worst foolish, some people (the author included) do still find themselves attracted to the dozenal (base-twelve) system. These people, however, have been pretty hard up[1] in the LaTeX world. There is no package file available which produces dozenal counters, like page and chapter numbers, nor were there *any* (I made a pretty diligent search) dozenal characters for ten and eleven, leaving dozenalists forced to use such makeshift ugliness as the "X/E" or "T/E" or "*/#" or whatever other standard they decided to use. While this sort of thing may be acceptable in ASCII, it's absolutely unacceptable in a beautiful, typeset document.

Enter the `dozenal` package. This package automates all the messiness of being a dozenalist and using LaTeX. It redefines all the counters (though you'll have to redefine them yourself if you're using your own), provides an algorithm (generously donated by the intrepid David Kastrup) for converting arbitrary positive whole

[1] This is an Americanism for "out of luck" or "in difficult circumstances," for those who do not know.

numbers into dozenal (this is eTEX, but all modern distributions will compile that), and finally, it includes original dozenal characters, specifically designed to blend in well with Knuth's Computer Modern fonts, though they should do fine with the more common body fonts, as well.

This document was typeset in accordance with the LATEX DOCSTRIP utility.

2 Usage

The `dozenal` package provides four new commands (though I can only take credit for two of them). The first, and by far the most important given the purpose and content of this package, is `\basexii`. This is a very simple command which takes the following structure:

`\basexii{`⟨*number*⟩`}{`⟨*ten symbol*⟩`}{`⟨*eleven symbol*⟩`}`

What the above means is that the command is `\basexii` and it takes three mandatory arguments: first, the symbol that should be used for ten; second, the symbol that should be used for eleven; and third, the number that should be converted into dozenal using those two symbols. This number should be positive and whole; that is, it should be zero or higher, and it should not contain a fractional part. TEX is a typesetting program, after all; if you want a robust decimal to dozenal converter, there are many options that any dozenalists caring enough to use this package will already know about.

This `\basexii` algorithm was produced by David Kastrup, well known and admired in the TEX world for his many useful packages and other contributions. He posted this algorithm on comp.text.tex; it is included here with his kind and generous permission.

That one would want to use the same ten and eleven symbols throughout a document seems a reasonable assumption; therefore, I have provided a simplified version of the `\basexii` command, `\dozens`. `\dozens` takes only a single argument, the number to be converted; the ten and eleven symbols used are those produced by the commands `\x` and `\e`, to which we'll get in a moment.

`\x` `\e` `\x` and `\e` are the commands used to quickly and easily access the symbols for ten and eleven without having to use active characters (the TEX gurus will know what that means; if you don't know, that's okay; you don't need to to use this package effectively). In any case, `\x` and `\e` default to using the special dozenal characters that are part of this package; they could be easily redefined if for some reason you don't like the Pitman characters (which this package, and the Dozenal Society of Great Britain, prefer), in the following manner:

```
\renewcommand\x{X}
```

Or whichever characters you like to use. If you prefer the Dozenal Society of America's proposed characters (a stylized X and E), then this package will disappoint you. May I suggest `$\chi$` (χ) and `$\xi$` (ξ) as a stopgap while you locate or produce real characters of your own? Sorry; I'm an American myself, but I much prefer the Pitman characters for a variety of reasons (feel free to email me if you

care), and creating fonts in METAFONT, even small and inconsequential ones like this, is too much work for characters that I don't even like.

The `dozenal` package also redefines all the standard LaTeX counters, such as `section` and `enumii`. If you've defined your own counters, you'll need to define them yourself; however, this is an easy matter:

```
\renewcommand\thecounter{\basexii{\arabic{counter}}{\x}{\e}}
```

For example. Of course, you can fill in the `\x` and `\e` with whatever you want (though it would make more sense to simply redefine `\x` and `\e`, so that all the counters would use the same characters), or you could use the `\dozens` command instead. Whatever your pleasure might be.

3 Implementation

First, we name the package that we provide. This is not exactly the most difficult part of the code.

```
\ProvidesPackage{dozenal}
```

Now we need to make sure that we have fixltx2e loaded.

```
\RequirePackage{fixltx2e}
```

We then define the font that we're using for our METAFONT-produced Pitman characters. Incidentally, we also define the command `\doz`, though I can't foresee any decent use for it except in packages and preambles; it is then used to define `\x` and `\e`, which provide the ten and eleven symbols for all the counter redefinitions.

```
\DeclareFontFamily{OT1}{dozch}{}
\DeclareFontShape{OT1}{dozch}{m}{n}{<-7> dozchars6 <7> dozchars7 <8> dozchars8 <9> dozchars9 <1
\DeclareFontShape{OT1}{dozch}{b}{n}{<-> dozchb10 }{}
\DeclareFontShape{OT1}{dozch}{bx}{n}{<-6> dozchbx6 <7> dozchbx7 <8> dozchbx8 <9> dozchbx9 <10-1
\DeclareFontShape{OT1}{dozch}{m}{sl}{<-8> dozchsl8 <9> dozchsl9 <10-11> dozchsl10 <12-> dozchsl
\DeclareFontShape{OT1}{dozch}{bx}{sl}{<-> dozchbxsl10 }{}
\DeclareFontShape{OT1}{dozch}{m}{it}{<-7> dozchit7 <8> dozchit8 <9> dozchit9 <10-11> dozchit10
\DeclareFontShape{OT1}{dozch}{bx}{it}{<-> dozchbxi10 }{}
\newcommand\doz[1]{{\fontfamily{dozch}\selectfont #1}}
\newcommand\x{\TextOrMath{\protect\doz{{X}}}{\X}}%
\newcommand\e{\TextOrMath{\protect\doz{{E}}}{\E}}%
\DeclareSymbolFont{dozens}{OT1}{dozch}{m}{n}
\DeclareMathSymbol{\X}{\mathord}{dozens}{88}
\DeclareMathSymbol{\E}{\mathord}{dozens}{69}
```

Then we define our command which will produce the dozenal numbers from decimal sources. This algorithm was taken directly from the publicly available archives of comp.text.tex, where it was posted by the well-known and redoubtable David Kastrup. We also define the `\dozens` command, a simplified `\basexii` (which, in fact, depends utterly upon `\basexii`), just to make it easy for everyone.

```
\def\basexii#1#2#3{\ifcase\numexpr(#1)\relax
0\or1\or2\or3\or4\or5\or6\or7\or8\or9\or#2\or#3\else
\expandafter\basexii\expandafter{\number\numexpr((#1)-6)/12}{#2}{#3}\expandafter\basexii\expand
\newcommand\dozens[1]{\basexii{#1}{\x}{\e}}
```

Now, of course, we simply redefine all the counters. This covers only those counters included in the basic LaTeX document classes, however, so if you've written your own, you'll need to redefine them yourself.

```
\@ifundefined{c@page}{}{\renewcommand\thepage{\basexii{\arabic{page}}{\x}{\e}}}
\@ifundefined{c@chapter}{}{\renewcommand\thechapter{\basexii{\arabic{chapter}}{\x}{\e}}}
\@ifundefined{c@footnote}{}{\renewcommand\thefootnote{\basexii{\arabic{footnote}}{\x}{\e}}}
\@ifundefined{c@part}{}{\renewcommand\thepart{\basexii{\arabic{part}}{\x}{\e}}}
\@ifundefined{c@subparagraph}{}{\renewcommand\thesubparagraph{\basexii{\arabic{subparagraph}}{\
\@ifundefined{c@paragraph}{}{\renewcommand\theparagraph{\basexii{\arabic{paragraph}}{\x}{\e}}}
\@ifundefined{c@equation}{}{\renewcommand\theequation{\basexii{\arabic{equation}}{\x}{\e}}}
\@ifundefined{c@figure}{}{\renewcommand\thefigure{\basexii{\arabic{figure}}{\x}{\e}}}
\@ifundefined{c@table}{}{\renewcommand\thetable{\basexii{\arabic{table}}{\x}{\e}}}
\@ifundefined{c@table}{}{\renewcommand\thempfootnote{\basexii{\arabic{mpfootnote}}{\x}{\e}}}
\@ifundefined{c@enumi}{}{\renewcommand\theenumi{\basexii{\arabic{enumi}}{\x}{\e}}}
\@ifundefined{c@enumii}{}{\renewcommand\theenumii{\basexii{\arabic{enumii}}{\x}{\e}}}
\@ifundefined{c@enumiii}{}{\renewcommand\theenumiii{\basexii{\arabic{enumiii}}{\x}{\e}}}
\@ifundefined{c@enumiv}{}{\renewcommand\theenumiv{\basexii{\arabic{enumiv}}{\x}{\e}}}

\@ifundefined{chapter}{% if it's undefined
\renewcommand\thesection{\basexii{\arabic{section}}{\x}{\e}}
\renewcommand\thesubsection{\thesection.\basexii{\arabic{subsection}}{\x}{\e}}
\renewcommand\thesubsubsection{\thesubsection.\basexii{\arabic{subsubsection}}{\x}{\e}}
} % end undefined
{%if it's defined
\renewcommand\thesection{\thechapter.\basexii{\arabic{section}}{\x}{\e}}
\renewcommand\thesubsection{\thesection.\basexii{\arabic{subsection}}{\x}{\e}}
\renewcommand\thesubsubsection{\thesubsection.\basexii{\arabic{subsubsection}}{\x}{\e}}
} % end if it's defined
```

And that's the end. Thanks for reading, folks; please email me with any suggestions or improvements.