The `cmolddig` package

Rowland McDonnell

23rd August 1999

Contents

1 Introduction

The `cmolddig` package sets things up so that numbers are printed using old style digits – 1234567890 – rather than the usual lining digits – 1234567890. It only works with the original Computer Modern founts. You can choose whether numbers are printed as old style or lining digits in text or in maths mode.

You can get old style digits without any extra help by using the command `\oldstylenums` in maths mode, but this package arranges things so that all numbers are printed using old style digits without you needing to do that.

To cut a very involved tale down to the barest essentials: old style digits look better and are easier to read in text and the like; while lining digits are more suitable for maths, in tables, and the like. Both these assertions are argued over by typographers.

If you have the T1 encoded European Modern founts installed, you might be interested in the `eco` package, which is the T1 equivalent of `cmolddig`. It avoids many of the limits of this package. The `cmolddig` package doesn't contain any new characters: you need to have the OT1 encoded Computer

Modern Roman founts installed before this package is useful. As far as I know, all TEX systems come with these founts installed by default. It doesn't matter if you're using Metafont, PostScript Type 1, or TrueType versions of the Computer Modern founts; but you do need a TEX system able to use virtual founts. Almost all modern (1999) TEX systems can use virtual founts; the only exception I know of is Y&Y TEX's version.

`Cmolddig` works by switching to a set of virtual founts that have the normal lining digits replaced with old style digits from the corresponding maths italic fount. These digits are conventional upright digits despite living in a nominally italic fount.

The original Computer Modern founts have old style digits available for the Computer Modern Roman family only, and even then only for the upright shapes: Computer Modern Roman upright, bold extended, bold, and small caps. Because of this, this package can't give you old style digits for Computer Modern Sanserif or Computer Modern Typewriter; nor does it give you old style digits if you're using any italic or slanted fount.

Options exist to use old style digits in maths or not, independently of the style of numbers in normal text. You can also choose to make use of the extra small caps sizes included with the AMS fount set if you have it installed.

2 Installation

You need to put three directories of files in the right places on your TEX system's directory tree: with that done, installation is complete.

All the files in the `texinputs` directory should go into a directory that TEX searches for input files. All the files in the `tfm` directory should go into a directory that TEX searches for `tfm` (fount) files. All the files in the `vf` directory should go into a directory that TEX searches for `vf` (virtual fount) files. If you don't know where these places are, have a read of your TEX system's documentation.

Moving these files to the right places is all you need to do. Once you've done this, try running LATEX on `odshrtst.tex`: you should see some old style digits in the output, showing that you've installed the package correctly.

You can ignore all the files in the `source` directory unless you're interested in how I created the virtual founts at the heart of this package. The `source` directory can be thrown away if you like.

3 Using the `cmolddig` package

Make sure you include the `cmolddig` package by saying:

```
\usepackage{cmolddig}
```

in the preamble of your document. This will switch you to Computer Modern Roman with old style digits in normal text only, not in maths mode.

3.1 Controlling the package

You can switch between Computer Modern Roman with old style digits and with normal lining digits inside your document using the new commands: `\cmoldstyledigits` and `\cmliningdigits`.

```
\cmoldstyledigits  35,000 cabbages with old style digits,
\cmliningdigits    or 15 turnips with lining digits.
```

By default, the `cmolddig` package doesn't affect numbers in maths mode. If you want old style digits in maths mode, you can give the package the `cmoldstyledigits` option:

```
\usepackage[cmoldstyledigits]{cmolddig}
```

the `cmliningmathdigit` option forces lining digits in maths mode; this is what `cmolddig` usually does, so there's probably no need to know about this option.

You can switch between Computer Modern Roman with old style digits in maths and with normal lining digits in maths inside your document using the commands: `\cmoldstyledigits` and `\cmliningdigits`.

```
\cmoldstylemathdigits $35 \pi = x$%       old style digits
\cmliningmathdigits   $17x^{2}-32x+17=y$% lining digits
```

You might like to note that it's *very* unusual to use old style digits in maths. I have come across one typographer who asserts that, despite this, maths is more readable when typeset with old style digits. Whatever the truth of the matter, it's probably best to keep to lining digits in maths if you are producing work that needs to meet any standard of conventionality.

If you have the AMS founts which come with Computer Modern Roman caps and small caps in two extra sizes – 8 pt and 9 pt – you can tell `cmolddig` about it with the `amsfonts` option:

```
\usepackage[amsfonts]{cmolddig}
```

will tell `cmolddig` to use these extra sizes if needed. The `cmolddig` package won't use these two extra founts unless you tell it to. The `extracsc` option is identical in effect to the `amsfonts` option.

The package also includes the `noamsfonts` and `noextracsc`: a pair of options which tell `cmolddig` to avoid using the extra AMS small caps founts. I can't see any need for anyone to use them, because the `cmolddig` package avoids using these extra founts by default.

4 How it works and stuff

There are three separate things that can be considered: the virtual fount mechanism in TEX, which makes this package possible; the creation of the particular virtual founts used by `cmolddig`; and what the package actually does when you use it.

4.1 Virtual founts

A warning: the subject of TEX and founts is complicated and never-ending. This short section is inevitably inadequate, but I hope it sheds a little light.

To begin with, all TEX knows about a fount is the `tfm` file. When you select a fount for typesetting, TEX reads the appropriate `tfm` file which tells it how much space each character takes up (height, depth, and width), how much space to add or subtract between certain pairs of letters (kerning information), which pairs of characters should be replaced by a different character (ligature information; replacing `ff` with an 'ff' ligature, for example), and similar things. There is no information about the shape of any letters in this file.

Every `tfm` file has up to 256 different characters listed in it. Each character lives in a numbered slot: 0–255. There are a number of standard *encodings* defined for LATEX which say which character should live in which numbered slot. In OT1 encoding (used for the original Computer Modern Roman founts), character number 48 is the number '0', character number 49 is the number '1', and so on.

In the normal run of things, TEX places information about the fount in the `dvi` file, and when you come to print or preview that `dvi` file, the driver program finds a real fount (bitmap, PS Type 1, TrueType, or whatever) and displays it. This approach has a few problems, particularly when you come to work with founts that (in their natural state) don't have the same characters in the same slots as the standard TEX encoding. For example, character number 1 in an OT1 encoded fount is an upper-case Delta: Δ. This is unique to that encoding: character number 1 in a normal PS type 1 fount is undefined. If you tried to use a normal PS Type 1 fount with LATEX's OT1 encoding and took no precautions, things just wouldn't work at all right and you'd get the wrong characters printed for lots of things.

The virtual fount mechanism is one method for getting round this (and other) problems. The way it works (in brief) is this. You create a pair of files: a `tfm` file and a `vf` file. The `tfm` file is a perfectly ordinary `tfm` file. The `vf` (virtual fount) file is something new: it contains information on how to print each character referred to in the `tfm` file. The instructions might be to get all the characters except the numbers from (say) cmr10, and to get the numbers from cmmi10. This is exactly what `cmrj10.vf` does. These `vf` files can contain fragments of `dvi` file so all sorts of trickery is

possible. A more common use for virtual founts is re-mapping: re-arranging the characters so that they're in the expected slots for the encoding you're using. In other words, you might be using (say) a PS Type 1 version of Times using the `times` package and OT1 encoding. If you were to use a dot accent, you'd be asking for character number 95. In the 'natural' PS text encoding, character number 95 is an underscore. The magic of virtual founts means that character number 95 in the fount (`tfm` file) that TEX sees 'really' is a dot accent; when the dvi drivercomes to display that fount, it sees that it's a virtual fount, looks at the appropriate `vf` file, and displays character number 199 from the 'real' Times fount every time the `dvi` file asks for character number 95. (For the sake of a comprehensible explanation, I've ignored the widespread use of 8r encoding as an intermediate step, and ignored re-encoding done by the dvi driver).

The `vf` (virtual fount) file is a new type of file for TEX systems, but it's ignored by TEX itself: `vf` files are used only by `dvi` drivers. What happens is this: when a modern `dvi` driver looks at a `dvi` file, it checks to see if any of the `tfm` files in it have corresponding `vf` files. If so, the driver looks in the `vf` files and follows the instructions in them on how to print the characters referred to. A `vf` file can refer to any real fount or any virtual fount, so this business of looking things up in virtual founts can get complicated.

You can find out more about virtual founts by running `weave` and then TEX on these files from CTAN:

```
systems/knuth/etc/vftovp.web
systems/knuth/etc/vptovf.web
```

4.2 What the package does

A brief explanation to begin with: every fount in LATEX belongs to a *family*. Every family has a short internal name by which it is referred to inside LATEX. The internal name of the Computer Modern Roman family is `cmr`. The `cmolddig` package defines a new family, which has the internal name `cmrj`. The 'j' is a code that means 'with old style digits', so `cmrj` stands for 'Computer Modern Roman with old style digits'. If you're interested, the fontname documentation available from CTAN explains the full details of this code.

This package's default behaviour is to define the `\rmdefault` family to be `cmrj`. This command defines the fount family used for typesetting normal text. What this change means is that when LATEX comes across a request to typeset normal text, it sees that it needs to use the `cmrj` family, it loads the file `ot1cmrj.fd` (this is assuming that you're using OT1 encoding – if you're not, you'll get an error message instead). This file tells LATEX which `tfm` file to load when you ask for any given fount. In this case, LATEX is told to use the new virtual founts where they exist. You can learn more about `fd` files by reading `fntguide.tex`, which is part of the standard LATEX distribution.

The cmolddig package makes two new commands: \cmoldstyledigits, which changes \rmdefault to cmrj and switches to \rmfamily; and \cmliningdigits, which changes \rmdefault to cmr and also switches to \rmfamily

Maths is a bit more tricky. By default, the cmolddig package doesn't affect maths mode behaviour. The package defines two commands for switching between the two sorts of digits in maths mode: \cmoldstylemathdigits and \cmliningmathdigits. These commands are used by the package when you use the cmoldstylemathdigits and cmliningmathdigit options. What do they do? They change the mathcode of the numbers 0–9 to the appropriate values to select either normal lining digits from the usual place, or old style digits from the appropriate maths italic fount (which is where the normal upright roman old style digits are kept, for reasons best known to Donald Knuth). If that makes almost no sense to you and you'd like to know more, the best solution is to read Knuth's *The TEXbook*, the LaTeX source code, and the cmolddig package file. In case you're wondering, I don't understand maths fount selection at all well, and I found out how to perform this particular switching by asking on news://comp.text.tex.

The maths digit switching doesn't require the fount files that come with this package: it uses the normal 'real' Computer Modern founts without needing any virtual fount trickery.

4.3 Creating the cmolddig virtual founts

You don't need to do this job: all the useful files that this process creates are part of the cmolddig distribution. I've included this section and the corresponding source files to satisfy the curiosity of people who want to know how I did the deed.

The contents of the virtual founts used by cmolddig are pretty straightforward: rounding errors aside, they are identical to the Computer Modern originals with normal lining digits in the text founts replaced with old style digits from the appropriate maths italic fount. These old style digits are upright despite living in an italic fount; ask Dr Knuth if you want to know why. The job of creating them was non-trivial only because of certain eccentricities of fontinst and the equally eccentric encodings used by Donald Knuth in his original Computer Modern family of founts.

I used fontinst to help create the virtual founts used by this package; fontinst is a set of TeX macros available from CTAN: it'll run on any computer with a functioning TeX on it.

To create the virtual founts the way I did, you need to install fontinst, and use the files in cmolddig's source directory by following the instructions below. You also need tftopl and vptovf which should be part of your TeX system.

Some of the files used to create the virtual founts at the heart of this

package have names that break the '8+3' filename limit imposed by some archaic operating systems. Apologies for this; it's just the way they turned out, and I didn't notice until it was far too late.[1] Since there's no *need* to use the source files, I thought it'd be okay not to fix this.

You'll need to use `tftopl` to create the following `pl` files from the corresponding Computer Modern `tfm` files:

```
cmb10.pl
cmbx10.pl, cmbx12.pl, cmbx5.pl, cmbx6.pl, cmbx7.pl, cmbx8.pl, cmbx9.pl
cmcsc10.pl
cmmi10.pl, cmmi12.pl, cmmi5.pl, cmmi6.pl, cmmi7.pl, cmmi8.pl, cmmi9.pl
cmmib10.pl, cmmib5.pl, cmmib6.pl, cmmib7.pl, cmmib8.pl, cmmib9.pl
cmr10.pl, cmr12.pl, cmr17.pl, cmr5.pl, cmr6.pl, cmr7.pl, cmr8.pl, cmr9.pl
```

If you have access to the AMS extra founts, you'll also need:

```
cmcsc8.pl, cmcsc9.pl
```

The `source` directory contains these five files:

```
dostretch.mtx
instcmrj.tex
ot1noflig.etx, ot1nofligcj.etx, ot1nofligj.etx
```

Put these five source files in a directory with the `pl` files listed above. If you don't have the AMS extra founts, edit `instcmrj.tex` and comment out the lines referring to cmcscj9 and cmcscj8.

Then run Plain TEX on `instcmrj.tex`. The result of this run will be a huge pile of files: run `vptovf` on the files ending in `vpl` (virtual property list; human-readable precursors to a machine-readable `tfm`/`vf` pair). The resulting `tfm` and `vf` files are the files that do the trick. The `fd` file created by `fontinst` isn't particularly useful to begin with: compare it to the one supplied with `cmolddig` to see what modifications I felt were needed.

You can throw away the debris now if you like: all the files (`mtx`, `pl`, and `fd`) that have been worked with except the five files from `cmolddig`'s `source` directory are machine-generated so you don't need to keep them.

4.3.1 So how does it work?

You really ought to understand LATEX founts and `fontinst` to an extent before reading this section.

The `fontinst` file that does the work contains lines like this:

```
\installfont{cmrj10}{cmr10,kernoff,cmmi10,kernon,dostretch}{OT1j}
            {OT1}{cmrj}{m}{n}{<10> <10.95>}
```

[1] The computer I use now has a file system that dates back to 1984; it has a 31 character filename limit

The last 5 arguments (on the second line) are just the NFSS parameters used in the `fd` file that's created by `fontinst` for this installation. The interesting bit is the first line: cmrj10 is the (virtual) fount about to be created. In this case, the glyphs are taken from cmr10 and cmmi10 (`fontinst` converts the `pl` files to `mtx` files before it does anything with them).

The file `kernoff.mtx` tells `fontinst` to ignore kerning information from subsequent founts: this ensures that the newly created fount has kerning from cmr10 only. This is important, since maths fount kerning is entirely unsuitable for text. The file `kernon.mtx` tells `fontinst` to take notice of kerning again. It's redundant, since no kerning information is held in `dostretch.mtx`, but kept there just for the sake of neatness. So: kerning information is taken from cmr10, not from cmmi10.

The file `dostretch.mtx` tells `fontinst` to take the new virtual fount's spacing parameters from those that have been set by the first fount read – in other words, to take spacing from cmr10 in this case. The third argument is the encoding file used to build the fount: in this case, it's OT1j; that's OT1 encoding with old style digits. Rather than using the glyphs called 'one', 'two', 'three', and so on, this `mtx` file ensures that the glyphs called 'oneoldstyle', 'twooldstyle', 'threeoldstyle' are used instead. The 'oldstyle' digit glyphs are kept in the cmmi founts.

Some of these installation lines ask for the final encoding OT1NOFLIGj. This calls `ot1nofligj.etx`, which in turn calls `ot1noflig.etx`; this last file is a new encoding file I wrote for use with `fontinst v1.8`. It defines the TEX TEXT WITHOUT F-LIGATURES encoding, used in cmr5 and the caps and small caps founts.