The comicsans package[*]

Scott Pakin
scott+csan@pakin.org

September 29, 2008

1 Introduction

The comicsans package makes Microsoft's Comic Sans font available to LaTeX 2ε. comicsans supports all of the following:

- Roman text, **boldface text**, SMALL-CAPS TEXT, and—with a little extra effort—*italic text*
- Кирилица (римский шрифт, **жирный шрифт**, *каллиграфический шрифт*)
- Mathematics using Comic Sans wherever possible:

$$y'(x) \approx 3 \times 10^{\log_3 2\hat{\varepsilon}} + \sum_{k=x}^{\infty} \frac{\xi_k}{p_{k-1}}$$

Comic Sans is a TrueType (TTF) font. As such, it works particularly well with pdfLaTeX, which natively supports TrueType fonts. Some TeX distributions also support dynamic conversion of TTF to PK (a bitmapped font format long used by TeX) so TeX backends other than pdfTeX can (indirectly) utilize TrueType fonts, as well.

2 Installation

The following is a brief summary of the comicsans installation procedure:

1. Acquire and install the Comic Sans TrueType (`.ttf`) files.
2. [Optional] Generate the italic and/or Cyrillic variants of Comic Sans
3. Install the comicsans font files and refresh the TeX filename database.

[*]This document corresponds to comicsans v1.0f, dated 2008/07/12.

4. Point the TEX backends to the comicsans files.

Details are presented in Sections 2.1–2.4.

2.1 Acquire and install the TrueType files

comicsans requires the Comic Sans and Comic Sans Bold TrueType files (**`comic.ttf`** and **`comicbd.ttf`**). You may already have these installed. (On Windows, look in **`C:\WINDOWS\Fonts`** for **`Comic Sans MS (TrueType)`** and **`Comic Sans MS Bold (TrueType)`**.) If not, see if a package called msttcorefonts is available for your operating system or operating-system distribution. If not, then download **`comic32.exe`** from **`http://corefonts.sourceforge.net/`** and use the freely available **`cabextract`** utility to extract **`comic.ttf`** and **`comicbd.ttf`** from **`comic32.exe`**.

Install **`comic.ttf`** and **`comicbd.ttf`** in an appropriate, TEX-accessible location such as **`/usr/local/share/texmf/fonts/ttf/microsoft/comicsans/`**. (TEX distributions for Microsoft Windows may automatically search the system font directory but I haven't yet tested this hypothesis.)

2.2 Generate the italic and/or Cyrillic variants (optional)

To use the T2A-encoded Cyrillic versions of Comic Sans you'll need to install the cyrfinst package, which is available from CTAN.[1]

Because Microsoft doesn't make a Comic Sans Italic, and because TTF fonts don't accept the **`SlantFont`** modification, we need some way of handling italicized text. The best alternative is to convert the TTF fonts to PostScript Type 1 format and use **`SlantFont`** to dynamically create oblique variants. It may be possible to use **`ttf2pt1`** to do the conversion but I don't know how to specify the various TEX font encodings. Instead, I use a (free) program called FontForge to convert TTF to Type 1:

TEX base 1 (8r) encoding Open **`comic.ttf`** in FontForge. Select Element→Font Info..., click on the Encoding tab, and select "TεX Base (8r)" for the encoding. Click OK. Go to File→Generate Fonts... and create **`rcomic8r.pfb`**. Follow an analogous procedure to generate **`rcomicbd8r.pfb`** from **`comicbd.ttf`**.

T2A Adobe encoding (Cyrillic) Follow the same steps as above, but for Encoding, click on Load, select the **`t2a.enc`** file, then choose T2AAdobeEncoding for the encoding. Generate **`rcomiccyr.pfb`** from **`comic.ttf`** and **`rcomiccyrbd.pfb`** from **`comicbd.ttf`**.

[1]In practice only `t2a.enc` need be installed.

If you're unable to run FontForge on your system and you can't find an alternate TTF→PFB converter, don't worry. Although you won't be able to typeset italics, Section 3 describes some comicsans package options that make Comic Sans utilize either underlined or boldfaced text for emphasis.

2.3 Install font files and refresh TEX's database

The comicsans package consists of a large number of font files. These are organized in a TDS-compliant subdirectory rooted at **`texmf`**. You should be able to copy comicsans's **`texmf`** tree directly onto your TEX tree (i.e., **`/usr/local/share/texmf`**, **`C:\localtexmf`**, or wherever you normally install TEX files). If you generated italic and/or Cyrillic Comic Sans fonts (Section 2.2), install the corresponding **`.pfb`** files as well, typically in **`texmf/fonts/type1/microsoft/comicsans`**. Don't forget to refresh the filename database if necessary. See **`http://www.tex.ac.uk/cgi-bin/texfaq2html?label=inst-wlcf`** for details specific to your TEX distribution.

2.4 Point the TEX backends to the comicsans files

Most TEX backends (pdfTEX, Dvips, YAP, Xdvi, etc.) need to incorporate the contents of **`comicsans.map`** into their private font-map files. The exact procedure varies from one TEX distribution to another. See **`http://www.tex.ac.uk/cgi-bin/texfaq2html?label=instt1font`** for distribution-specific instructions on how to automatically update all of the various font-map files at once.

Notes

1. The Comic Sans math fonts don't seem to work properly in older versions of pdfTEX ($\leq$ 13x). If you have problems you should upgrade to a newer version.

2. It *is* possible to get Dvips to use a vector (i.e., Type 1) version of Comic Sans. If you have the patience, the following is the procedure. First, for *each* non-**`SlantFont`**ed line of **`comicsans.map`**, you'll need a separate Type 1 (**`.pfb`**) file—eight altogether—each with a different encoding and PostScript font name. I used FontForge to produce these. For example, I created an **`rcomic7m.pfb`** file with the PostScript name "**`ComicSansMS-7m`**" and with **`texmital.enc`** as the encoding vector. Next, store all of these **`.pfb`** files in a directory that Dvips searches. Finally, create a modified **`comicsans.map`** that omits the encodings (as the **`.pfb`** files are already properly encoded at this point). It should look something like the following:

```
rcomic8r ComicSansMS <rcomic8r.pfb
rcomicbd8r ComicSansMS-Bold <rcomicbd8r.pfb
rcomiccyr ComicSansMS-t2a <rcomict2a.pfb
rcomiccyrbd ComicSansMS-Bold-t2a <rcomicbdt2a.pfb
rcomic7m ComicSansMS-7m <rcomic7m.pfb
rcomicbd7m ComicSansMS-Bold-7m <rcomicbd7m.pfb
rcomic7y ComicSansMS-7y <rcomic7y.pfb
rcomic9z ComicSansMS-9z <rcomic9z.pfb
rcomico8r ComicSansMS "0.167 SlantFont" <rcomic8r.pfb
rcomicbdo8r ComicSansMS "0.167 SlantFont" <rcomicbd8r.pfb
rcomiccyro ComicSansMS-t2a "0.167 SlantFont" <rcomict2a.pfb
rcomiccyrbdo ComicSansMS-Bold-t2a "0.167 SlantFont" <rcomicbdt2a.pfb
```

3 Usage

Load comicsans like any other LaTeX 2ε package, by putting "**\usepackage{comicsans}**" in your document's preamble. This sets the default roman, typewriter, and sans-serif typefaces as shown in Table 1. Courier Bold is typeset 10% larger than the requested point size. This provides a better visual match to Comic Sans.

Style	Default	With comicsans
Roman	Computer Modern	Comic Sans
Typewriter	Computer Modern Typewriter	**Courier Bold**
Sans-serif	Computer Modern Sans Serif	Helvetica

Table 1: comicsans font-family redefinitions

ulemph LaTeX's **\emph** is usually defined to produce italics. Unfortunately, Comic Sans doesn't include an italic variant. One alternative is to generate a slanted PostScript version of Comic Sans as described in Section 2. If this is too inconvenient or impossible an alternative is to use comicsans's **ulemph** package option. With **ulemph**, comicsans utilizes the soul package's underlining capabilities to typeset emphasized text like this. The drawback—apart from being ugly—is that underlining is limited to **\emph**; it doesn't work with **\em** or any of the italic macros (**\textit**, **\itshape**, **\it**, etc.), which are redefined as do-nothing commands. Also, underlined emphasis tends to fail when used in math mode.

boldemph The **boldemph** package option, like **ulemph**, alters the way that emphasized text is rendered in LaTeX. **boldemph** typesets **\emph** and **\em** in boldface **like this**. The various italic macros are redefined as do-nothing commands.

largesymbols Mathematical typesetting is clearly not a priority to Microsoft. As a result Comic Sans lacks most of the math characters that TeX requires. The comicsans package utilizes characters from the Computer Modern family to make up for this absense. While many of the characters are more-or-less compatible, the large symbols, with their thin strokes and serifed ends, particularly stand out to my eye:

$$y'(x) \approx 3 \times 10^{\log_3 2\hat{\varepsilon}} + \sum_{k=x}^{\infty} \frac{\xi_k}{p_{k-1}}$$

The **`largesymbols`** package option uses Comic Sans for a number of additional large symbols. The advantage of **`largesymbols`** is that more mathematical characters match the body font. The disadvantage—and the reason that **`largesymbols`** is off by default—is that the large symbols are merely scaled versions of their smaller counterparts, which unfortunately implies that their thickness scales as well:

$$y'(x) \approx 3 \times 10^{\log_3 2\hat{\varepsilon}} + \sum_{k=x}^{\infty} \frac{\xi_k}{p_{k-1}}$$

With the **`largesymbols`** package option comicsans gives you the ability to decide for yourself which is the lesser of the two evils.

`plusminus` LaTeX defines **`\pm`** as "$\pm$" and **`\mp`** as "$\mp$"—both taken from the Computer Modern Symbol font. Although Comic Sans provides a plus-or-minus glyph it lacks a corresponding minus-or-plus glyph. For consistency between the two glyphs comicsans draws both plus-or-minus and minus-or-plus from the Computer Modern Bold Symbol font: "$\pm$" and "$\mp$". The **`plusminus`** package option retains **`\mp`** as "$\mp$" but uses Comic Sans's "±" for **`\pm`**. This enables **`\pm`** to blend better with other Comic Sans characters at the expense of looking quite different from **`\mp`**.

4 Implementation: Core components

This section and the subsequent one contain the commented source code for the comicsans package. They are likely of little interest to the average user and can safely be ignored. Advanced users who want to customize or extend comicsans—please read the license agreement (Section 7) first—can use these sections to gain a detailed understanding of the code.

4.1 `comicsans.sty`

This is the comicsans package proper. It's primary purpose is to select Comic Sans as the default font for text and math.

<*package>

4.1.1 Option processing

`\if@ulemph` `\@ulemphtrue` `\@ulemphfalse`

The author can use underlining for emphasis (Section 4.1.3) using the **`ulemph`** option.

```
\newif\if@ulemph \DeclareOption{ulemph}{\@ulemphtrue\@boldemphfalse}
```

`\if@boldemph` `\@boldemphtrue` `\@boldemphfalse`

The author can use boldface for emphasis (Section 4.1.3) using the **`boldemph`** option.

```
\newif\if@boldemph
\DeclareOption{boldemph}{\@boldemphtrue\@ulemphfalse}
```

Using large, mathematical symbols in Comic Sans is still fairly experimental (read as: ugly). These symbols are disabled by default, but the author can enable them with the **`largesymbols`** option.

```
\DeclareOption{largesymbols}{%
  \DeclareSymbolFont{largesymbols}{OMX}{comic}{m}{n}%
}
```

`\if@csplusminus` `\@csplusminustrue` `\@csplusminusfalse`

Comic Sans defines a **`plusminus`** character ("±") but not a corresponding **`minusplus`** character. For consistency we normally draw both **`plusminus`** and **`minusplus`** from Computer Modern ("$\pm$" and "$\mp$"). However, the **`plusminus`** package option makes **`\pm`** match other Comic Sans symbols at the expense of not matching **`\mp`**.

```
\newif\if@csplusminus
\DeclareOption{plusminus}{\@csplusminustrue}
```

Finally, we process the package options.

```
\ProcessOptions\relax
```

4.1.2 Default font families

`\rmdefault` `\ttdefault` `\sfdefault`

We select Comic Sans as the default body font, Courier as the default fixed-width font, and Helvetica as the default sans-serif font. (Yes, this is a bit odd, given that Comic Sans is already sans-serif.)

```
\renewcommand{\rmdefault}{comic}
\renewcommand{\ttdefault}{pcr}
\renewcommand{\sfdefault}{phv}
```

We redefine Courier Medium as Courier Bold and Courier Italic as Courier Bold Oblique in the OT1 font encoding. We also increase the size by 10% to better match Comic Sans.

```
\DeclareFontFamily{OT1}{pcr}{}
\DeclareFontShape{OT1}{pcr}{b}{n}{
  <-> s * [1.1] pcrb7t
}{}
\DeclareFontShape{OT1}{pcr}{b}{it}{
  <-> s * [1.1] pcrbo7t
}{}
\DeclareFontShape{OT1}{pcr}{m}{n}{<->ssub * pcr/b/n}{}
\DeclareFontShape{OT1}{pcr}{bx}{n}{<->ssub * pcr/b/n}{}
\DeclareFontShape{OT1}{pcr}{m}{it}{<->ssub * pcr/b/it}{}
\DeclareFontShape{OT1}{pcr}{bx}{it}{<->ssub * pcr/b/it}{}
```

We now do the same for the T1 font encoding...

```
\DeclareFontFamily{T1}{pcr}{}
\DeclareFontShape{T1}{pcr}{b}{n}{
   <-> s * [1.1] pcrb8t
}{}
\DeclareFontShape{T1}{pcr}{b}{it}{
   <-> s * [1.1] pcrbo8t
}{}
\DeclareFontShape{T1}{pcr}{m}{n}{<->ssub * pcr/b/n}{}
\DeclareFontShape{T1}{pcr}{bx}{n}{<->ssub * pcr/b/n}{}
\DeclareFontShape{T1}{pcr}{m}{it}{<->ssub * pcr/b/it}{}
\DeclareFontShape{T1}{pcr}{bx}{it}{<->ssub * pcr/b/it}{}
```

...and the TS1 font encoding. We first ensure that the textcomp package is preloaded to avoid getting an "**`Encoding scheme 'TS1' unknown`**" error.

```
\RequirePackage{textcomp}
\DeclareFontFamily{TS1}{pcr}{}
\DeclareFontShape{TS1}{pcr}{b}{n}{
   <-> s * [1.1] pcrb8c
}{}
\DeclareFontShape{TS1}{pcr}{b}{it}{
   <-> s * [1.1] pcrbo8c
}{}
\DeclareFontShape{TS1}{pcr}{m}{n}{<->ssub * pcr/b/n}{}
\DeclareFontShape{TS1}{pcr}{bx}{n}{<->ssub * pcr/b/n}{}
\DeclareFontShape{TS1}{pcr}{m}{it}{<->ssub * pcr/b/it}{}
\DeclareFontShape{TS1}{pcr}{bx}{it}{<->ssub * pcr/b/it}{}
```

If the **`plusminus`** package option was specified we draw **`\textpm`** from **`\comic9z`**—the only Comic Sans font encoding that takes a **`plusminus`** character from Comic Sans instead of borrowing the one from Computer Modern Bold Symbol.

```
\if@csplusminus
  \DeclareTextSymbolDefault{\textpm}{U}
  \DeclareTextSymbol{\textpm}{U}{4}
\fi
```

4.1.3 Emphasis

Because Microsoft doesn't make a Comic Sans Italic and because TTF fonts don't accept the **`SlantFont`** modification we need some way of handling emphasized text. The best alternative is to use a program such as FontForge to convert the TTF fonts to PostScript Type 1 format (Section 2). Failing that, the author can specify with the **boldemph** package option that bold text should be used whenever emphasized text is requested. An alternative, with the **ulemph** package option, is to utilize the soul package to replace emphasis with underlining. Unfortunately, soul doesn't provide a way to enable underlining until the end of the current group (as is needed for LaTeX 2.09's `{\em ... }` construct). Furthermore, soul tends to choke on underlined mathematics.

If **boldemph** was given as a package option we utilize bold text for emphasis. Because we lack a true italic—or even an oblique variant of Comic Sans—we replace all of the explicit italic commands with `\relax`.

```
\if@boldemph
  \let\emph=\textbf
  \let\em=\bf
  \let\itshape=\relax
  \let\it=\relax
\fi
```

If **ulemph** was given as a package option we utilize underlined text for emphasis. This requires the soul package. Because we lack a true italic—or even an oblique variant of Comic Sans—we replace all of the explicit italic commands with `\relax`.

```
\if@ulemph
  \RequirePackage{soul}
  \setul{1.5pt}{1pt}
  \let\emph=\ul
  \let\itshape=\relax
  \let\it=\relax
```

Out of necessity, we unfortunately also have to make `\em` a do-nothing command.

```
  \let\em=\relax
\fi
```

4.1.4 Mathematics

`operators` `letters` `symbols` For mathematical expressions, we draw operators, letters, and symbols from Comic Sans. Large symbols normally come from Computer Modern, but the **largesymbols** package option (Section 4.1.1) specifies that they should come from Comic Sans, as well.

```
\DeclareSymbolFont{operators}{OT1}{comic}{m}{n}
\DeclareSymbolFont{letters}{OML}{comic}{m}{n}
\DeclareSymbolFont{symbols}{OMS}{comic}{m}{n}
```

`\neq` `\pm` We define one additional symbol font, "**othercomics**", from which we define `\neq` as the glyph "≠" and—if the **plusminus** package option was specified—`\pm` as the glyph "±".

```
\let\neq=\undefined
```

```
\DeclareSymbolFont{othercomics}{U}{comic}{m}{n}
\DeclareMathSymbol{\neq}{\mathrel}{othercomics}{3}
\if@csplusminus
  \DeclareMathSymbol{\pm}{\mathbin}{othercomics}{4}
\fi
```

`\frac` TEX's default fraction bar is much too thin for Comic Sans. We therefore redefine **`\frac`** to use a fraction bar with a more compatible thickness.

```
\def\frac#1#2{{%
  \begingroup#1\endgroup\abovewithdelims..0.75pt#2}}
```

</package>

4.2 comicsans.map

This is a map file for pdfLATEX that provides the association between TFM names (*e.g.*, **rcomic8r**) and PostScript names (*e.g.*, **ComicSansMS**). It also specifies how fonts should be re-encoded so that characters appear at the expected offsets in each font.

<*comicsans.map>

```
rcomic8r ComicSansMS "TeXBase1Encoding ReEncodeFont" <8r.enc <comic.ttf
rcomicbd8r ComicSansMS-Bold "TeXBase1Encoding ReEncodeFont" <8r.enc <comicbd
rcomiccyr ComicSansMS "T2AAdobeEncoding ReEncodeFont" <t2a.enc <comic.ttf
rcomiccyrbd ComicSansMS-Bold "T2AAdobeEncoding ReEncodeFont" <t2a.enc <comic
rcomic7m ComicSansMS "TeXMathItalicEncoding ReEncodeFont" <texmital.enc <com
rcomicbd7m ComicSansMS-Bold "TeXMathItalicEncoding ReEncodeFont" <texmital.e
rcomic7y ComicSansMS "TeXMathSymbolEncoding ReEncodeFont" <texmsym.enc <comi
rcomic9z ComicSansMS "ComicSansExtraEncoding ReEncodeFont" <csextras.enc <co
```

The following four lines assume that you have PostScript Type 1 versions of the various Comic Sans fonts. Although Section 2 describes a technique for converting TrueType to Type 1, my understanding of copyright law is that I am not allowed to distribute **rcomico8r.pfb** or **rcomicbdo8r.pfb** myself as these are considered derivitive works from **comic.ttf** and **comicbd.ttf**.

```
rcomico8r ComicSansMS "0.167 SlantFont" <rcomic8r.pfb
rcomicbdo8r ComicSansMS-Bold "0.167 SlantFont" <rcomicbd8r.pfb
rcomiccyro ComicSansMS "0.167 SlantFont" <rcomiccyr.pfb
rcomiccyrbdo ComicSansMS-Bold "0.167 SlantFont" <rcomiccyrbd.pfb
```

</comicsans.map>

4.3 csextras.enc

csextras.enc is an encoding file that tells the pdfLATEX backend how to reorder the glyphs in **comic.ttf** to match the order expected by **rcomic9z.tfm**. **csextras.enc** specifies only those glyphs that **rcomic9z.tfm** uses (the comicsans "extra" glyphs).

<*csextras.enc>

ComicSansExtraEncoding
integral
Sigma
Pi
notequal
plusminus

This encoding defines **integral** ("$\int$"), **summation** ("$\sum$"), and **product** ("$\prod$"). **comic7v.vf** maps TEX's ⟨*symbol*⟩**text** and ⟨*symbol*⟩**display** symbols onto these. We also define **notequal** ("≠") because this looks better than the composite of **not** and **equal** ("$\not=$"); and we define **plusminus** ("±") because **comic7y** uses **cmbsy10**'s **plusminus** character ("$\pm$"), which better matches its **minusplus** ("$\mp$").

```
/ComicSansExtraEncoding [
  /integral
```

The following two symbols are *supposed* to be **/summation** and **/product**. For some reason that I don't yet understand, pdfLATEX is unable to find those symbols in **comic.ttf** even though FontForge can. As a workaround we use **/Sigma** and **/Pi**, which are sufficiently similar.

```
  /Sigma
  /Pi
  /notequal
  /plusminus
```

We pad the encoding to exactly 256 characters using **/.notdefs**, as some programs (e.g., **ttf2pk**) expect to see exactly 256 encoded characters.

```
  /.notdef /.notdef /.notdef /.notdef /.notdef
  /.notdef /.notdef /.notdef /.notdef /.notdef
  /.notdef /.notdef /.notdef /.notdef /.notdef
                          ⋮
  /.notdef /.notdef /.notdef /.notdef /.notdef
] def
```

</csextras.enc>

4.4 ttfonts.map

Dvips doesn't currently support TrueType fonts. However, the **ttf2pk** utility (included with the FreeType library) can convert a TrueType font file (**.ttf**) into a TEX packed-font file (**.pk**) for use with Dvips or similar tools. **ttf2pk** requires a mapping file, **ttfonts.map**, which specifies the mapping between TEX font names and the corresponding TrueType font file.

<*ttfonts.map>

The first part of **ttfonts.map** contains analogous entries to those in **comicsans.map** (Section 4.2).

```
rcomic8r      comic.ttf    Encoding=8r.enc
rcomicbd8r    comicbd.ttf  Encoding=8r.enc
rcomiccyr     comic.ttf    Encoding=t2a.enc
rcomiccyrbd   comicbd.ttf  Encoding=t2a.enc
rcomic7m      comic.ttf    Encoding=texmital.enc
rcomicbd7m    comicbd.ttf  Encoding=texmital.enc
rcomic7y      comic.ttf    Encoding=texmsym.enc
```

```
rcomic9z      comic.ttf    Encoding=csextras.enc
```

Although pdfLaTeX can dynamically slant only PostScript files, not TrueType files, **ttf2pk** has no such limitation when producing **.pk** bitmaps.

```
rcomico8r     comic.ttf    Encoding=8r.enc  Slant=0.167
rcomicbdo8r   comicbd.ttf  Encoding=8r.enc  Slant=0.167
rcomiccyro    comic.ttf    Encoding=t2a.enc Slant=0.167
rcomiccyrbdo  comicbd.ttf  Encoding=t2a.enc Slant=0.167
```

</ttfonts.map>

5 Implementation: Extras

The files documented in this section are what I used to automate creation of the TeX/LaTeX bindings for Comic Sans. They are needed only if you want to modify or extend these bindings. Please read the license agreement (Section 7), however, before modifying any part of the comicsans package.

5.1 csextras.etx

csextras.etx is a fontinst encoding file that is used to create **rcomic9z.pl**. It specifies all of the characters that should appear in **rcomic9z.pl**.

We start with some boilerplate initialization.

<*csextras.etx>

```
\relax
\encoding
\needsfontinstversion{1.800}
```

Next, we specify the symbols that we're interested in. We begin with the large TeX symbols.

integral "∫"

```
\setslot{integral}
\endsetslot
```

summation "∑"

```
\setslot{summation}
\endsetslot
```

product "∏"

```
\setslot{product}
\endsetslot
```

The remaining large symbols are all scaled versions of ordinary symbols—parentheses, brackets, braces, etc.—and hence don't need to appear in this file. We therefore conclude with **notequal** (a nonstandard TeX character) and **plusminus** (which already exists in **comic7y** but uses the Computer Modern Bold Symbol version).

notequal "≠"

```
\setslot{notequal}
\endsetslot
```

plusminus "±"

```
\setslot{plusminus}
\endsetslot
\endencoding
```

</csextras.etx>

5.2 csextras.mtx

csextras.mtx is a fontinst metrics file that is used to help create **comic7v.vpl**. **csextras.mtx** maps TEX glyph names such as "**integraltext**" to Comic Sans font names such as "**integral**".

One problem is that TEX defines "text style" (small) and "display style" (large) versions of various symbols, while Comic Sans typically defines only the small size. We therefore do all that we can, which is to scale up the small version to a larger size. The unfortunate result is that display-style symbols tend to be excessively thick. *C'est la vie.*

We start with some boilerplate initialization.

<*csextras.mtx>

```
\relax
\metrics
```

\bigbiggerbiggest To save typing, we create a macro that defines **\big**, **\Big**, **\bigg**, and **\Bigg** versions of a given symbol.

```
\setcommand\bigbiggerbiggest#1{%
  \setglyph{#1big}
    \glyph{#1}{1000}
  \endsetglyph
  \setglyph{#1Big}
    \glyph{#1}{2500}
  \endsetglyph
  \setglyph{#1bigg}
    \glyph{#1}{4000}
  \endsetglyph
  \setglyph{#1Bigg}
    \glyph{#1}{5500}
  \endsetglyph
}
```

integraltext
integraldisplay

Define “$\int$” and “$\displaystyle\int$”.

```
\setglyph{integraltext}
  \glyph{integral}{1000}
\endsetglyph
\setglyph{integraldisplay}
  \glyph{integral}{3000}
\endsetglyph
```

summationtext
summationdisplay

Define “$\sum$” and “$\displaystyle\sum$”.

```
\setglyph{summationtext}
  \glyph{summation}{1000}
\endsetglyph
\setglyph{summationdisplay}
  \glyph{summation}{3000}
\endsetglyph
```

producttext
productdisplay

Define “$\prod$” and “$\displaystyle\prod$”.

```
\setglyph{producttext}
  \glyph{product}{1000}
\endsetglyph
\setglyph{productdisplay}
  \glyph{product}{3000}
\endsetglyph
```

parenleftbig
parenleftBig
parenleftbigg
parenleftBigg
parenrightbig
parenrightBig
parenrightbigg
parenrightBigg

Define a range of sizes for “(” and “)”.

```
\bigbiggerbiggest{parenleft}
\bigbiggerbiggest{parenright}
```

bracketleftbig
bracketleftBig
bracketleftbigg
bracketleftBigg
bracketrightbig
bracketrightBig
bracketrightbigg
bracketrightBigg

Define a range of sizes for “[” and “]”.

```
\bigbiggerbiggest{bracketleft}
\bigbiggerbiggest{bracketright}
```

braceleftbig
braceleftBig
braceleftbigg
braceleftBigg
bracerightbig
bracerightBig
bracerightbigg
bracerightBigg

Define a range of sizes for “{” and “}”.

```
\bigbiggerbiggest{braceleft}
\bigbiggerbiggest{braceright}
```

slashbig **slashBig** **slashbigg** **slashBigg** **backslashbig** **backslashBig** **backslashbigg** **backslashBigg**

Define a range of sizes for "/" and "\".

```
\bigbiggerbiggest{slash}
\bigbiggerbiggest{backslash}
```

angleleftbig **angleleftBig** **angleleftbigg** **angleleftBigg** **anglerightbig** **anglerightBig** **anglerightbigg** **anglerightBigg**

Define a range of sizes for "⟨" and "⟩" (really "<" and ">"). Because the naming is inconsistent between Comic Sans and TEX ("**angleleft**" vs. "**less**") we can't use our **\bigbiggerbiggest** macro.

```
\setglyph{angleleftbig}
  \glyph{less}{1000}
\endsetglyph
\setglyph{angleleftBig}
  \glyph{less}{2500}
\endsetglyph
\setglyph{angleleftbigg}
  \glyph{less}{4000}
\endsetglyph
\setglyph{angleleftBigg}
  \glyph{less}{5500}
\endsetglyph

\setglyph{anglerightbig}
  \glyph{greater}{1000}
\endsetglyph
\setglyph{anglerightBig}
  \glyph{greater}{2500}
\endsetglyph
\setglyph{anglerightbigg}
  \glyph{greater}{4000}
\endsetglyph
\setglyph{anglerightBigg}
  \glyph{greater}{5500}
\endsetglyph
```

That's all for **csextras.mtx**.

```
\endmetrics
```

</csextras.mtx>

5.3 nompbul.mtx

nompbul.mtx is used by **fontcomic.tex** when producing an OMS-encoded version of Comic Sans. Comic Sans's **plusminus** looks fine, but the font lacks a matching **minusplus**. For consistency we discard the

plusminus, too. The **plusminus** package option (Section 4.1.1) can re-enable it on a per-document basis. Comic Sans also has puny **bullet** and **openbullet** characters so we discard those too.

<*nompbul.mtx>

```
\relax
\metrics
\unsetglyph{plusminus}
\unsetglyph{bullet}
\unsetglyph{openbullet}
\endmetrics
```

</nompbul.mtx>

5.4 fontcomic.tex

fontcomic.tex is a fontinst file that specifies how to derive various PL and VPL fonts from the TTF sources. **fontcomic.tex** relies on the cyrfinst package to produce Cyrillic fonts. Due to a restriction of cyrfinst, **fontcomic.tex** must be run through **latex**, not **tex**.

Note that the fonts produced by **fontcomic.tex** do not follow the Berry naming scheme except for appending the encoding scheme onto the end of the name. Personally, I find "**comicbd8r**" more readable than "**jcsb8r**" for Comic Sans Bold in the **8r** encoding.

We start by inputting **fontinst.sty** and the various **.tex** files provided by cyrfinst for creating Cyrillic fonts.

<*fontcomic.tex>

```
\input fontinst.sty
\input fnstcorr
\input cyralias
```

I have tested **fontcomic.tex** only with fontinst version 1.800 so we should require that explicitly.

```
\needsfontinstversion{1.800}
\installfonts
```

rcomic8r.pl
rcomic8r.mtx
rcomicbd8r.pl
rcomicbd8r.mtx
rcomic7m.pl
rcomic7m.mtx
rcomicbd7m.pl
rcomicbd7m.mtx
rcomic7y.pl
rcomic7y.mtx
rcomic9z.pl
rcomic9z.mtx
rcomiccyr.pl
rcomiccyr.mtx
rcomiccyrbd.pl
rcomiccyrbd.mtx

First, we create some "raw" fonts, from which everything else is derived. These are the only fonts that are referenced by **comicsans.map** (Section 4.2); all other fonts produced by **fontcomic.tex** are defined in terms of the following.

```
  \transformfont{rcomic8r}%
    {\reencodefont{8r}{\fromafm{rcomic}}}
  \transformfont{rcomicbd8r}%
    {\reencodefont{8r}{\fromafm{rcomicbd}}}
  \transformfont{rcomic7m}%
    {\reencodefont{oml}{\fromafm{rcomic}}}
  \transformfont{rcomicbd7m}%
    {\reencodefont{oml}{\fromafm{rcomicbd}}}
  \transformfont{rcomic7y}%
  {\reencodefont{oms}{\fromafm{rcomic}}}
\transformfont{rcomic9z}%
  {\reencodefont{csextras}{\fromafm{rcomic}}}
\transformfont{rcomiccyr}%
  {\reencodefont{t2a}{\fromafm{rcomic}}}
\transformfont{rcomiccyrbd}%
  {\reencodefont{t2a}{\fromafm{rcomicbd}}}
```

`rcomico8r.pl` `rcomico8r.mtx` `rcomicbdo8r.pl` `rcomicbdo8r.mtx` `rcomiccyro.pl` `rcomiccyro.mtx` `rcomiccyrbdo.pl` `rcomiccyrbdo.mtx`

Next, we create "raw" oblique versions of Comic Sans and Comic Sans Bold as Microsoft doesn't provide a true italic.

```
\transformfont{rcomico8r}%
  {\slantfont{167}{%
    \reencodefont{8r}{\fromafm{rcomic}}}}
\transformfont{rcomicbdo8r}%
  {\slantfont{167}{%
    \reencodefont{8r}{\fromafm{rcomicbd}}}}
\transformfont{rcomiccyro}%
  {\slantfont{167}{%
    \reencodefont{t2a}{\fromafm{rcomic}}}}
\transformfont{rcomiccyrbdo}%
  {\slantfont{167}{%
    \reencodefont{t2a}{\fromafm{rcomicbd}}}}
```

`ot1comic.fd` `comic7t.vpl` `comicbd7t.vpl` `comico7t.vpl` `comicbdo7t.vpl` `comicsc7t.vpl`

We create versions of Comic Sans and Comic Sans Bold that are encoded with the OT1 encoding (Knuth's original 7-bit encoding scheme).

```
\installfamily{OT1}{comic}{}
\installfont{comic7t}
  {rcomic8r,rcomic7m,latin}
  {OT1}{OT1}{comic}{m}{n}{}
\installfont{comicbd7t}
  {rcomicbd8r,rcomicbd7m,latin}
  {OT1}{OT1}{comic}{b}{n}{}
\installfont{comico7t}
  {rcomico8r,rcomic7m,latin}
  {OT1}{OT1}{comic}{m}{sl}{}
\installfont{comicbdo7t}
  {rcomicbdo8r,rcomicbd7m,latin}
  {OT1}{OT1}{comic}{b}{sl}{}
\installfont{comicsc7t}
  {rcomic8r,rcomic7m,latin}
  {OT1C}{OT1}{comic}{m}{sc}{}
```

`t1comic.fd` `comic8t.vpl` `comicbd8t.vpl` `comico8t.vpl` `comicbdo8t.vpl` `comicsc8t.vpl`

We now do the same thing for the T1 (Cork) 8-bit encoding.

```
\installfamily{T1}{comic}{}
\installfont{comic8t}
  {rcomic8r,latin}
  {T1}{T1}{comic}{m}{n}{}
\installfont{comicbd8t}
  {rcomicbd8r,latin}
  {T1}{T1}{comic}{b}{n}{}
\installfont{comico8t}
  {rcomico8r,latin}
  {T1}{T1}{comic}{m}{sl}{}
\installfont{comicbdo8t}
  {rcomicbdo8r,latin}
  {T1}{T1}{comic}{b}{sl}{}
\installfont{comicsc8t}
  {rcomic8r,latin}
  {T1C}{T1}{comic}{m}{sc}{}
```

ts1comic.fd
comic8c.vpl
comicbd8c.vpl
comico8c.vpl
comicbdo8c.vpl

Comic Sans provides many of the textcomp symbols, so we encode some fonts for those. Note that we take the **bullet** and **openbullet** characters from Computer Modern Bold Symbol instead of Comic Sans. The Comic Sans versions are too small, in my opinion.

```
\installfamily{TS1}{comic}{}
\installfont{comic8c}
  {rcomic8r,nompbul,cmbsy10,textcomp}
  {TS1}{TS1}{comic}{m}{n}{}
\installfont{comicbd8c}
  {rcomicbd8r,nompbul,cmbsy10,textcomp}
  {TS1}{TS1}{comic}{b}{n}{}
\installfont{comico8c}
  {rcomico8r,nompbul,cmbsy10,textcomp}
  {TS1}{TS1}{comic}{m}{sl}{}
\installfont{comicbdo8c}
  {rcomicbdo8r,nompbul,cmbsy10,textcomp}
  {TS1}{TS1}{comic}{b}{sl}{}
```

t2acomic.fd
comiccyr.vpl
comiccyrbd.vpl
comiccyro.vpl
comiccyrbdo.vpl

Thanks to the cyrfinst package, it's fairly straightforward to extract the Comic Sans Cyrillic characters into a LaTeX-accessible font.

```
\installfamily{T2A}{comic}{}
\installfont{comiccyr}
  {rcomiccyr}
  {T2A}{T2A}{comic}{m}{n}{}
\installfont{comiccyrbd}
  {rcomiccyrbd}
  {T2A}{T2A}{comic}{b}{n}{}
\installfont{comiccyro}
  {rcomiccyro}
  {T2A}{T2A}{comic}{m}{sl}{}
\installfont{comiccyrbdo}
  {rcomiccyrbdo}
  {T2A}{T2A}{comic}{b}{sl}{}
```

omlcomic.fd
comic7m.vpl
comicbd7m.vpl

The remaining fonts produced by **fontcomic.tex** are math fonts. We start with math italic (the OML 7-bit encoding), although we use roman Comic Sans characters. Missing math italic characters are taken from Computer Modern 10 pt. Math Italic Bold (**cmmib10**).

```
\installfamily{OML}{comic}{\skewchar\font=127}
\installfont{comic7m}
  {rcomic7m,kernoff,cmmib10,kernon,mathit}
  {OML}{OML}{comic}{m}{n}{}
\installfont{comicbd7m}
  {rcomicbd7m,kernoff,cmmib10,kernon,mathit}
  {OML}{OML}{comic}{b}{n}{}
```

omscomic.fd
comic7y.vpl

Next up are the math symbol characters (OMS 7-bit encoded). These are taken from Comic Sans when possible, Computer Modern 10 pt. Bold Symbol (**cmbsy10**) when not. Note that we utilize **nompbul.mtx** (Section 5.3) to exclude the **plusminus** glyph.

```
\installfamily{OMS}{comic}{}
\installfont{comic7y}
  {rcomic7y,rcomic8r,unsetalf,nompbul,cmbsy10,mathsy}
  {OMS}{OMS}{comic}{m}{n}{}
```

omxcomic.fd
comic7v.vpl

As our final math font, we produce a 7-bit OMX-encoded (math extension) version of Comic Sans. Comic Sans includes *none* of the required characters by default. However, **csextras.mtx** (Section 5.2) can rename a few glyphs to improve the situation. Nevertheless, OMX-encoded Comic Sans is still not a particularly pleasing font. Authors may want to use a different OMX-encoded font in its place.

```
\installfamily{OMX}{comic}{}
\installfont{comic7v}
  {rcomic9z,rcomic8r,csextras,cmex10,mathex}
  {OMX}{OMX}{comic}{m}{n}{}
```

ucomic.fd
comic9z.vpl

Leftover characters are assigned to a LaTeX "U"-encoded font, **comic9z**.

```
\installfamily{U}{comic}{}
\installfont{comic9z}
  {rcomic9z}
  {CSEXTRAS}{U}{comic}{m}{n}{}
```

Those are all of the Comic Sans fonts I could think to create. We can finish up now.

```
\endinstallfonts
\bye
```

</fontcomic.tex>

5.5 Makefile

The **Makefile** included below automates the generation of the various Comic Sans LaTeX fonts. I tested this **Makefile** only with GNU make, only on Linux, and only with the teTeX TeX distribution.

Note that the various "**verbatim**" lines are present for DocStrip's sake and do not actually appear in the resulting file.[2] Also, many TEX distributions do not honor tab characters when outputting files, although most **make** implementations *require* tabs. As a result, **comicsans.ins** specifies that the following code be written to **Makefile.NOTABS** with space- instead of tab-based indentation. It is up to the user to convert spaces to tabs. (In GNU Emacs, the "**M-x tabify**" sequence automates this conversion; entering "**cat Makefile.NOTABS | perl -ne 's/^␣␣␣␣␣␣␣␣/\t/g; print' > Makefile**" at the Unix prompt is even more automatic.)

<*Makefile>

TFMTARGETS **VFTARGETS** Because we produce so many TFM and VF files, we define **TFMTARGETS** and **VFTARGETS** targets for these.

```
%<<verbatim>
TFMTARGETS = comic7m.tfm comic7t.tfm comic7v.tfm        \
             comic7y.tfm comic8c.tfm comic8t.tfm        \
             comicbd7t.tfm comicbd8c.tfm comicbd8t.tfm  \
             comiccyr.tfm comiccyrbd.tfm rcomic.tfm     \
             rcomic7m.tfm rcomic8r.tfm rcomicbd.tfm     \
             rcomicbd8r.tfm rcomiccyr.tfm rcomic7y.tfm  \
             rcomiccyrbd.tfm rcomic9z.tfm comic9z.tfm   \
             rcomicbd7m.tfm comicbd7m.tfm               \
             rcomico8r.tfm rcomicbdo8r.tfm              \
             comico7t.tfm comicbdo7t.tfm                \
             comico8t.tfm comicbdo8t.tfm                \
             comico8c.tfm comicbdo8c.tfm                \
             rcomiccyro.tfm rcomiccyrbdo.tfm            \
             comiccyro.tfm comiccyrbdo.tfm              \
             comicsc7t.tfm comicsc8t.tfm

VFTARGETS =  comic7m.vf comic7t.vf comic7v.vf       \
             comic7y.vf comic8c.vf comic8t.vf       \
             comicbd7t.vf comicbd8c.vf comicbd8t.vf \
             comiccyr.vf comiccyrbd.vf comic9z.vf   \
             comicbd7m.vf                           \
             comico7t.vf comicbdo7t.vf              \
             comico8t.vf comicbdo8t.vf              \
             comico8c.vf comicbdo8c.vf              \
             comiccyro.vf comiccyrbdo.vf            \
             comicsc7t.vf comicsc8t.vf

%verbatim>
```

PACKAGEFILES **all** The primary Makefile targets are the **.tfm**, **.vf**, and **.fd** files.

[2]Without the "**verbatim**" lines, DocStrip would choke on all of the end-of-line "\" characters.

```
PACKAGEFILES = $(TFMTARGETS) $(VFTARGETS) $(FDOUTPUTS)

all: $(PACKAGEFILES)
```

We define a rule for converting a VPL file into a VF plus a TFM file and a rule for converting a PL file into a TFM file.

```
%<<verbatim>

.SUFFIXES: .vf .vpl .tfm .pl .ttf .afm

%.vf %.tfm: %.vpl
        vptovf $<

%.tfm: %.pl
        pltotf $<

%verbatim>
```

We would ideally like to define a rule for building a **.**⟨*DPI*⟩**pk** file that depends upon a corresponding **.tfm** file. Unfortunately, Makefile semantics do not support such usage. We therefore parse out ⟨*DPI*⟩ and call **make** recursively to ensure that the requisite **.tfm** file exists.

```
%<<verbatim>

%pk: comicsans.map comic.ttf comicbd.ttf
        DPI=`echo $@ | \
          perl -ne '/(\d+)pk$$/ && print $$1'` ; \
        BASE=`echo $@ | \
          perl -ne '/^(.*)\.\d+pk$$/ && print $$1'` ; \
        gsftopk -q --mapfile=comicsans.map $$BASE $$DPI

%verbatim>
```

cmmib10.pl
cmex10.pl
cmbsy10.pl

Kpathsea should find standard .tfm files even if they're not in the current directory. Hence, the following three targets have no dependencies.

```
cmmib10.pl:
        tftopl cmmib10.tfm > cmmib10.pl

cmex10.pl:
        tftopl cmex10.tfm > cmex10.pl

cmbsy10.pl:
        tftopl cmbsy10.tfm > cmbsy10.pl
```

FDOUTPUTS
LOGOUTPUTS
PLOUTPUTS
VPLOUTPUTS
MTXOUTPUTS
FONTINSTOUTPUTS

fontinst outputs a large number of files. To make these more manageable we define macros to represent various subsets.

```
%<<verbatim>

FDOUTPUTS  = ts1comic.fd t1comic.fd ot1comic.fd  \
                t2acomic.fd omlcomic.fd omxcomic.fd \
                omscomic.fd ucomic.fd
LOGOUTPUTS = fontcomic.log
PLOUTPUTS =  rcomic.pl rcomicbd.pl rcomiccyrbd.pl      \
             rcomic7m.pl rcomic8r.pl rcomicbd8r.pl     \
             rcomiccyr.pl rcomic9z.pl rcomic7y.pl      \
             rcomicbd7m.pl rcomico8r.pl rcomicbdo8r.pl \
             rcomiccyro.pl rcomiccyrbdo.pl
VPLOUTPUTS = comic8c.vpl comicbd8c.vpl comiccyrbd.vpl \
             comic7m.vpl comiccyr.vpl comic7t.vpl     \
             comicbd7t.vpl comic8t.vpl comicbd8t.vpl  \
             comic7v.vpl comic9z.vpl comic7y.vpl      \
             comicbd7m.vpl                            \
             comico7t.vpl comicbdo7t.vpl              \
             comico8t.vpl comicbdo8t.vpl              \
             comico8c.vpl comicbdo8c.vpl              \
             comiccyro.vpl comiccyrbdo.vpl            \
             comicsc7t.vpl comicsc8t.vpl
MTXOUTPUTS = cmbsy10.mtx cmex10.mtx cmmib10.mtx       \
             rcomic.mtx rcomicbd.mtx rcomiccyrbd.mtx  \
             rcomic7m.mtx rcomic8r.mtx rcomicbd8r.mtx \
             rcomiccyr.mtx rcomic9z.mtx rcomic7y.mtx  \
             rcomicbd7m.mtx                           \
             rcomico8r.mtx rcomicbdo8r.mtx            \
             rcomiccyro.mtx rcomiccyrbdo.mtx

FONTINSTOUTPUTS = $(FDOUTPUTS) $(LOGOUTPUTS) \
                  $(PLOUTPUTS) $(VPLOUTPUTS) \
                  $(MTXOUTPUTS)

%verbatim>
```

AFMINPUTS **PLINPUTS** **CSEXTRAS** We now define macros for all of fontinst's input files, excluding those that need not exist in the current directory.

```
AFMINPUTS = rcomic.afm rcomicbd.afm
PLINPUTS  = cmbsy10.pl cmmib10.pl cmex10.pl
CSEXTRAS  = csextras.etx csextras.mtx
```

The most important part of the Makefile is to run the **fontcomic.tex** fontinst file through LaTeX. Normally fontinst files are run through TeXbut the cyrfinst package, which **fontcomic.tex** uses, requires LaTeX.

```
%<<verbatim>

$(FONTINSTOUTPUTS): fontcomic.tex \
                    $(AFMINPUTS) $(PLINPUTS) $(CSEXTRAS)
        latex fontcomic.tex

%verbatim>
```

doc
DOCOUTPUTS

To automate building the comicsans documentation, we define a **doc** target, which uses pdfLaTeX and MakeIndex to build a nicely formatted PDF document. For some reason "**\DoNotIndex{\␣}**" doesn't seem to work. We therefore explicitly **grep** away all of the "\␣" entries.

```
%<<verbatim>

doc: comicsans.pdf

DOCOUTPUTS = comicsans.pdf comicsans.aux comicsans.glo \
             comicsans.out comicsans.log comicsans.idx \
             comicsans.ind comicsans.ilg comicsans.gls

$(DOCOUTPUTS): comicsans.dtx $(PACKAGEFILES) comicsans.sty
        pdflatex '\pdfmapfile{pdftex.map}\pdfmapfile{comicsans.map}\input co
  sans.dtx'
        grep -v 'indexentry{! =' comicsans.idx | \
          makeindex -s gind.ist -o comicsans.ind
        makeindex -s gglo.ist comicsans.glo -o comicsans.gls
        pdflatex '\pdfmapfile{pdftex.map}\pdfmapfile{comicsans.map}\input co
  sans.dtx'
        pdflatex '\pdfmapfile{pdftex.map}\pdfmapfile{comicsans.map}\input co
  sans.dtx'
        pdfopt comicsans.pdf cs.pdf
        mv cs.pdf comicsans.pdf

%verbatim>
```

CSTEXMFDIR
CSVFDIR
CSTFMDIR
CSLTXDIR
CSDVIPSMAPDIR
CSDVIPSENCDIR
install
uninstall

Because comicsans consists of so many files, we provide an **install** target to automate installation. We assume a TeX Directory Standard (TDS) distribution although the user can override the various directory locations by assigning one or more of **CSTEXMFDIR**, **CSVFDIR**, **CSTFMDIR**, **CSLTXDIR**, **CSDVIPSMAPDIR**, or **CSDVIPSENDDIR** on the **make** command line. Although we also provide an **uninstall** target, this is not guaranteed to remove all of the directories created. Specifically, if **install** creates both a directory and a subdirectory (e.g., **microsoft/comicsans**), only the subdirectory (**comicsans**) will be deleted.

```
%<<verbatim>

CSTEXMFDIR    = /usr/local/share/texmf
CSVFDIR       = $(CSTEXMFDIR)/fonts/vf/microsoft/comicsans
CSTFMDIR      = $(CSTEXMFDIR)/fonts/tfm/microsoft/comicsans
CSLTXDIR      = $(CSTEXMFDIR)/tex/latex/comicsans
CSDVIPSMAPDIR = $(CSTEXMFDIR)/fonts/map/dvips/comicsans
CSDVIPSENCDIR = $(CSTEXMFDIR)/fonts/enc/dvips/comicsans
CSDOCDIR      = $(CSTEXMFDIR)/doc/latex/comicsans

install: $(CSTEXMFDIR) $(PACKAGEFILES) comicsans.sty comic-
  sans.pdf
        install -d $(CSVFDIR) $(CSTFMDIR) $(CSLTXDIR) \
          $(CSDVIPSMAPDIR) $(CSDVIPSENCDIR) $(CSDOCDIR)
        install -m 664 $(VFTARGETS) $(CSVFDIR)
        install -m 664 $(TFMTARGETS) $(CSTFMDIR)
        install -m 664 $(FDOUTPUTS) comicsans.sty $(CSLTXDIR)
        install -m 664 comicsans.map $(CSDVIPSMAPDIR)
        install -m 664 csextras.enc $(CSDVIPSENCDIR)
        install -m 664 comicsans.pdf $(CSDOCDIR)

uninstall:
        $(RM) -rf $(CSVFDIR) $(CSTFMDIR) $(CSLTXDIR) $(CS-
DOCDIR)
        $(RM) -rf $(CSDVIPSMAPDIR) $(CSDVIPSENCDIR)

%verbatim>
```

TARGZFILE **dist** We make it easy to create a **.tar.gz** file containing **comicsans.ins**, **comicsans.dtx**, and all of the prebuilt comicsans font files.

```
TARGZFILE = comicsans.tar.gz

dist: $(TARGZFILE)

$(TARGZFILE): $(PACKAGEFILES) doc
        install -d comicsans/comicsans
        install -m 664 README comicsans.pdf comicsans/comicsans
        install -m 664 comicsans.dtx comicsans.ins comicsans/comicsans
        install -d comicsans/texmf
        $(MAKE) CSTEXMFDIR=comicsans/texmf install
        cp -r comicsans/texmf/fonts/tfm/microsoft/comicsans comic-
sans/comicsans/tfm
        cp -r comicsans/texmf/fonts/vf/microsoft/comicsans comic-
sans/comicsans/vf
        mkdir comicsans/comicsans/dvips
        cp comicsans/texmf/fonts/map/dvips/comicsans/* comic-
sans/comicsans/dvips
        cp comicsans/texmf/fonts/enc/dvips/comicsans/* comic-
sans/comicsans/dvips
        cd comicsans/texmf ; \
          zip -r -9 -m ../comicsans.tds.zip *
        $(RM) -r comicsans/texmf
        tar -cf - comicsans | gzip --best > $(TARGZFILE)
        $(RM) -r comicsans
```

DPI **PKFILES** **pkfiles** My understanding of copyright law is that I am not allowed to distribute **.pk** files as these are considered derivitive works from **comic.ttf** and **comicbd.ttf**. However, I believe you *are* allowed to generate these files yourself for your own personal use. "**make pkfiles**" generates PK files for 600 DPI printers at the various standard LaTeX point sizes (taken from **ot1cmr.fd**). For printers with a different number of dots per inch, "**make DPI=**⟨*resolution*⟩ **pkfiles**" should override the 600-DPI default. If you need fonts at additional resolutions you can produce them individually with "**make** ⟨*font name*⟩**.**⟨*DPI*⟩**pk**".

```
%<<verbatim>

DPI = 600

PKFILES = $(shell perl -ane '                    \
  $$F[0] =~ /^\w/ || next;                       \
  foreach $$size (5..10, 10.95, 12, 14.4,        \
                  17.28, 20.74, 24.88) {         \
    printf "$$F[0].%dpk\n", $(DPI)*$$size/10     \
  }                                              \
' < comicsans.map)

pkfiles: $(TFMTARGETS) $(PKFILES)

%verbatim>
```

clean
cleaner

Finally, we define `clean` and `cleaner` target so that "`make clean`" will delete the myriad generated files. "`make cleaner`" additionally deletes the files that `comicsans.ins` had extracted from `comicsans.dtx`.

```
clean:
        $(RM) $(PKFILES)
        $(RM) $(TARGZFILE)
        $(RM) $(DOCOUTPUTS)
        $(RM) $(FONTINSTOUTPUTS)
        $(RM) $(PLINPUTS)
        $(RM) $(PACKAGEFILES)

cleaner: clean
        $(RM) comicsans.sty csextras.etx csextras.mtx
        $(RM) nompbul.mtx fontcomic.tex comicsans.map
        $(RM) csextras.enc ttfonts.map
        $(RM) rcomic.afm rcomicbd.afm Makefile.NOTABS
        $(RM) fonttopfb.ff alt-comicsans.map

.PHONY: doc install uninstall dist pkfiles clean cleaner
```

</Makefile>

5.6 rcomic.afm and rcomicbd.afm

`fontcomic.tex` (Section 5.4) depends upon `rcomic.afm` and `rcomicbd.afm`—the Adobe font metric files that specify the widths, heights, and depths of all of the characters in `comic.ttf` and `comicbd.ttf`. Although these can be produced automatically by the `ttf2afm` utility, `ttf2afm` misses a few characters, most notably `\summation` and `\product`. We therefore include versions of **rcomic.afm** and **rcomicbd.afm** that were generated by PfaEdit (FontForge's predecessor), which does a better job of finding glyphs than **ttf2afm**. Because these AFM files are long (∼12 pages apiece) we omit them from the comicsans documentation.

<*rcomic.afm>

⋮
599 lines of code omitted
⋮

</rcomic.afm>

<*rcomicbd.afm>

⋮
598 lines of code omitted
⋮

</rcomicbd.afm>

6 Implementation: Vietnamese typesetting support

In October 2006, Hàn Thế Thành requested a few changes to the comicsans package to support Vietnamese typesetting. Unfortunately, these changes require converting the Comic Sans fonts from TTF to Type 1 format using FontForge, which doesn't run natively under Windows. (Also, there is always some quality loss when converting font formats.) Furthermore, Microsoft's license prohibits distributing the generated Type 1 files directly.

This section presents Thành's instructions (reformatted but otherwise verbatim from his e-mail) and supplemental files needed to use the Comic Sans fonts in a Vietnamese-language context.

Hi,

I am working vietnamese support for the math font survey and encounter a problem with the comicsans package. The explanation is rather lengthy and dry, however the solution consists of 2 changes:

1. *replace the pfb's for each encoding by a single pfb, ie replace* ***rcomic8r.pfb*** *and* ***rcomiccyr.pfb*** *by* ***ComicSansMS.pfb****.* ***ComicSansMS.pfb*** *is just a pfb converted by fontforge from* ***comic.ttf*** *by running*

```
fontforge fonttopfb.ff comic.ttf comicbd.ttf
```

`fonttopfb.ff` is a script to convert ttf to pfb using fontforge, attached with this mail.

*<*fonttopfb.ff>*

```
#! /usr/bin/env fontforge

i = 1;
while (i < $argc)
  Print("converting ", $argv[i], "...");
  Open($argv[i]);

  SetFontOrder(3); # convert from quadratic to cubic curves
  ScaleToEm(1000);  # to standard Postscript sizes, also scales underli

  # clear TT hints and generate T1 hints
  SelectAll();
  ClearInstrs();
  ClearHints();
  AutoHint();

  Generate($fontname+".pfb", "", -1);
  i++;
endloop
```

</fonttopfb.ff>

2. *reencode the fonts explicitly by changing the map file **`comic-sans.map`** so that the following lines:*

```
  rcomico8r ComicSansMS "0.167 SlantFont" <rcomic8r.pfb
  rcomicbdo8r ComicSansMS "0.167 Slant-
Font" <rcomicbd8r.pfb
  rcomiccyro ComicSansMS "0.167 Slant-
Font" <rcomiccyr.pfb
  rcomiccyrbdo ComicSansMS "0.167 Slant-
Font" <rcomiccyrbd.pfb
```

become

```
  rcomico8r ComicSansMS "0.167 Slant-
Font TeXBase1Encoding ReEncodeFont" <ComicSansMS.pfb <8r.enc
  rcomicbdo8r ComicSansMS-Bold "0.167 Slant-
Font" <ComicSansMS-Bold.pfb <8r.enc
  rcomiccyro ComicSansMS "0.167 Slant-
Font T2AAdobeEncoding ReEncodeFont" <ComicSansMS.pfb <t2a.enc
  rcomiccyrbdo ComicSansMS-Bold "0.167 Slant-
Font T2AAdobeEncoding ReEncodeFont" <ComicSansMS-
Bold.pfb <t2a.enc
```

*<*alt-comicsans.map>*

```
rcomic8r ComicSansMS "TeXBase1Encoding ReEncodeFont" <8r.enc <comic.ttf
rcomicbd8r ComicSansMS-Bold "TeXBase1Encoding ReEncodeFont" <8r.enc <com
rcomiccyr ComicSansMS "T2AAdobeEncoding ReEncodeFont" <t2a.enc <comic.tt
rcomiccyrbd ComicSansMS-Bold "T2AAdobeEncoding ReEncodeFont" <t2a.enc <c
rcomic7m ComicSansMS "TeXMathItalicEncoding ReEncodeFont" <texmital.enc
rcomicbd7m ComicSansMS-Bold "TeXMathItalicEncoding ReEncodeFont" <texmit
rcomic7y ComicSansMS "TeXMathSymbolEncoding ReEncodeFont" <texmsym.enc <
rcomic9z ComicSansMS "ComicSansExtraEncoding ReEncodeFont" <csextras.enc

rcomico8r ComicSansMS "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <C
rcomicbdo8r ComicSansMS-Bold "0.167 SlantFont" <ComicSansMS-
Bold.pfb <8r.enc
rcomiccyro ComicSansMS "0.167 SlantFont T2AAdobeEncoding ReEncodeFont" <
rcomiccyrbdo ComicSansMS-Bold "0.167 SlantFont T2AAdobeEncoding ReEncode
Bold.pfb <t2a.enc
```

</alt-comicsans.map>

Do you think it is possible to adapt these changes to your package? It would simplify my life a lot ☺

Thanks for your consideration,
Thành

7 Copyright and license agreement

Change History

v1.0

General: Initial version 1

v1.0a

General: Included Hàn Thế Thành's modifications for Vietnamese typesetting 25

v1.0b

`dist`: Restructured the distribution tree according to Jim Hefferon's suggestions 23

v1.0d

`DOCOUTPUTS`: Modified to use **`\pdfmapfile`** to point pdflatex to comic-sans.map 22

v1.0e

dist: Moved the contents of the texmf directory to the top level of comic-sans.tds.zip as suggested by Dan Luecking 23

DOCOUTPUTS: Modified to run pdfopt on the generated PDF documentation 22

v1.0f

dist: Restructured the distribution tree according to Jim Hefferon's latest suggestions 23

install: Specified that the Makefile install comic-sans.pdf beneath the doc directory 22

Index

Numbers written in bold refer to the page where the corresponding entry is described or referenced, the ones underlined to the code line of the definition, the rest to the code lines where the entry is used.