

X₃TeX about:fonts

-X₃TeX 字体调用简介 -

KMC

2008 年 12 月 14 日

本文旨在帮助有一定 TeX 知识的爱好者了解和上手 $X_3\text{TeX}$ ，以方便地调用各种字体。水平有限，欢迎指正¹。

¹kmc.best@gmail.com

目录

1	X_gTeX的安装和使用	3
1.1	一点点背景	3
1.2	编译环境	3
1.3	安装和简易使用	3
1.4	编码问题	4
2	X_gTeX基本字体调用	5
2.1	fontconfig	5
2.2	字体调用	6
2.3	字体名称	7
2.4	外部字体调用	7
3	使用 fontspec 宏包	8
3.1	基本介绍	8
3.2	调用字体	8
3.3	外部字体调用	10
3.4	配合 xeCJK 宏包调用中文字体	10
	参考文献	11

1 X_YT_EX 的安装和使用

1.1 一点点背景

ƎT_EX 的字体调用曾经一直是 ƎT_EX 的一个难点。以往广泛使用的 Type 1 字体²（也叫 PostScript 字体），即便使用 fontinst 来安装，手续也相当繁琐，而且对字体使用 Berry 命名规则十分不直观；这之后出现了 TrueType 和 OpenType 字体后，为了在 ƎT_EX 中使用，更加需要相当的背景知识来拆包 ttf 或 otf 生成 tfm/pfb 等文件，在此过程中很有可能造成 OpenType 的专业属性丢失。

X_YT_EX 的出现大大简化了 ƎT_EX 的字体调用，它基于 e-_TE_X 引擎，默认采用 UTF8/UTF16 编码方式并且能直接用字体名来调用系统中已有的字体和尚未安装的字体（见 2.4 和 3.3）。故而 X_YT_EX 吸引了相当的注意，可它最初发布于 Mac OS X 下，经过各种努力，X_YT_EX 被拓展到其它主流平台下，这其中 Akira Kakuto 的 W₃2T_EX 便是 Windows 平台的扩展，在此之后，2007 年 9 月，X_YT_EX 0.997 终于出现在 MiKTeX 2.7 的 beta 版中。

1.2 编译环境

对于新上手的人来说，在查阅相关文档的时候如果能知道这个文档是如何生成的，或者有这个文档的源代码，对照着学会容易许多。由于本文将给出源代码，所以这里也介绍一下编译环境和一些设置。

本文的编译环境：MiKTeX 2.7，ctexart 文档类配合 xeCJK 宏包^[2]，编辑器采用 SciTE (Scintilla Text Editor)。正文中文字体为 Adobe Song Std，英文采用 Minion Pro (serif), Myriad Pro (sans serif) 与 Computer Modern Typewriter (monospace) 的组合，目录和标题是 SimHei。

1.3 安装和简易使用

安装 MiKTeX 2.7 的过程比较简单，X_YT_EX 也是默认的安装组件。安装好之后就可以用你喜欢的编辑器去编辑一个简单的 ƎT_EX 文档，对于基本的文档来说，能用 pdfƎT_EX 编译也就能用 X_YƎT_EX 编译，例如：

```
\documentclass{article}
\begin{document}
    The quick brown fox jumps over the lazy dog.
    \textbf{The quick brown fox jumps over the lazy dog.}
```

²参阅 http://partners.adobe.com/public/developer/en/font/T1_SPEC.PDF

```
\emph{The quick brown fox jumps over the lazy dog.}
\end{document}
```

将该文件保存成 `sample.tex` 文件后，命令行输入

```
xelatex sample.tex
```

即可得到 `sample.pdf`，要注意的是 `xelatex` 命令直接生成 `pdf` 文档而没有 `dvi` 文件，用

```
xelatex -no-pdf sample.tex
```

可以生成 `sample.xdv`，事实上这个 `xdv` 是生成 `pdf` 的中间文件，它是被 `xdvipdfmx` 命令转换成 `pdf` 后删除掉的。

1.4 编码问题

X_YT_EX 默认使用 UTF-8 编码，但为了照顾兼容性，可以用参数来控制输入文件的编码，而不用更改整个输入文件的编码。例如本文的 `tex` 源代码用的是 GB2312 编码方式，这时只需要在文档的开头加上 `\XeTeXinputencoding "GB2312"`，并在 `\begin{document}` 前面加上 `\XeTeXdefaultencoding "UTF8"`（似乎不加也可以）把输入改回 UTF-8，文档即可正常编译。这样主要是省去了用户调整源文码编码的麻烦，加上常用的 WinEdt 编辑器对 UTF-8 的支持并不好。不过，如果要混排中文和带特殊字符的西文（如法语），WinEdt 依然会显得力不从心，这时可以考虑使用像 SciTE 这样对 UTF-8 支持良好的编辑器。

2 X_YTeX 基本字体调用

如果仅仅是 UTF-8 的优势的话 X_YTeX 不会得到如此的重视，因为 pdf_YTeX 中载入 `utf8x` 参数的 `inputenc` 宏包同样不差³，重要的还是字体的调用。

2.1 fontconfig

要调用字体，首先 X_YTeX 要认识字体，它采用 `fontconfig` 来扫描系统目录和用户指定的目录，从而缓存字体信息，归类同一 `family` 的字体（主要是西文字体）。安装好 MiKTeX 后，`fontconfig` 相关的文件出现在下面的目录：

```
C:\Documents and Settings\All Users\Application Data\MiKTeX\2.7\fontconfig
```

其中在 `config` 子目录中有一个 `localfonts.conf` 文件指定了默认的扫描目录，不要修改它，在 `localfonts2.conf` 文件中你可以指定默认扫描目录之外的字体目录。然后在命令行下用

```
fc-cache -f
```

来扫描字体，可惜，`fontconfig` 的效率十分低下，每扫描一次都要等很久，不仅如此，MiKTeX 的宏包管理器安装宏包也非要自动运行 `fontconfig`，使得每装一个宏包都很漫长。所以如果你的 MiKTeX 采用的是飞行安装宏包的模式的话，最好提前看看源文件中有哪些宏包没有安装，否则初次编译某个源文件时会需要相当长的时间。

扫描完毕后，字体都被缓存在 `cache` 子目录，二进制文件是看不明白的，要用

```
fc-list >>c:\fonts.txt
```

这个命令把字体列表输出到某个文本文档中，这样做是因为 Windows 命令提示符下显示 UTF-8 是乱码。而且，输出成某个文件后可以对它进行排序（如 EmEditor）和查找字体名，来确认该字体是否已经安装。如果字体并未安装，那么编译源文件时又会运行一遍 `fc-cache`，最后给出无尽的错误代码后报错退出。所以，当发现编译过程中硬盘长时间读取，估计就是 `fontconfig` 在找字体，这时建议赶紧 `ctrl+break` 中止编译，去查询 `fonts.txt` 中是否真的有这个字体或者是不是源文件中字体名写错了（大小写和空格都要考虑到，下文会具体说）。

³在 X_YTeX 中无需载入 `inputenc` 和 `fontenc` 宏包

2.2 字体调用

一个基本的Xe_LTeX字体调用语句是这样的：

```
\font\rm="Arno Pro" at 14pt \rm This is Arno Pro
```

效果：This is Arno Pro⁴

中文字体的调用也类似：

```
{\font\rm="STZhongsong" at 14pt \rm 这是一个华丽的测试}
```

效果：

这是一个华丽的测试

注：如果使用了 *xeCJK* 宏包，中文字体的调用需要通过 `\CJKfamily` 命令来实现，上面的效果实际上用的代码是 `{\CJKfamily{华中中宋} 这是一个华丽的测试}`。请参阅3.4。

Xe_LTeX 事实上是扩展了T_EX的`\font`命令，定义出一个字体命令`\rm`来。在指定字体时还可以带上字体选项（font options）和字体属性（font features），前提是 `fontconfig` 已经扫描并归类了字体。下面是几个例子：

```
\font\arnobi="Arno Pro/BI" at 12pt
\arnobi This is Arno Pro Bold Italic\
\font\arnobisc="Arno Pro/BI:+smcp" at 12pt
\arnobisc This is Arno Pro Bold Italic Smallcaps
```

效果：

This is Arno Pro Bold Italic

THIS IS ARNO PRO BOLD ITALIC SMALLCAPS

其中 `/BI` 是字体选项（粗斜体），类似的还有 `/B`（粗体），`/I`（斜体）等；`:+smcp` 是字体属性，可以通过用逗号或者分号分隔来复合载入，例如：

```
\font\arnoscred="Arno Pro:+smcp,color=FF0000" at 12pt
{\arnoscred Arno Pro Small Caps in red}
```

效果：

ARNO PRO SMALL CAPS IN RED

`smcp` 是 OpenType 字体属性中的一个例子，并非所有的字体都有真 Smallcaps（一般专业字体才会拥有丰富的字体属性）。关于 OpenType 字体属性，可以参考[微软网站](#)或阅读 Michel

⁴Arno Pro 字体的 T 和 h 是相连的，与 fi 连字不同的是，这种连字并非每种字体都有

Goossens 的《The X_YT_EX Companion》[1]。查看 OpenType 字体的属性可以用 Microsoft 的 **Font properties extension** 或者 Eddie Kohler 的 **lcdf tools** 中的 **otfinfo**。理解了这些属性后，才能更好地使用后文提到的 **fontspec** 宏包 [3]。不过这些都基本是西文字体内含的，中文字体没有这么复杂（中文字体主要是 glyph 多）。

2.3 字体名称

上面说到，字体的名称正确拼写很重要，否则会出错，如果你已经导出了 **fonts.txt**，在其中看到的代码会是这样：

```
仿宋 _GB2312,FangSong_GB2312:style=Regular
```

```
黑体,SimHei:style=Regular
```

```
Garamond Premier Pro,Garamond Premr Pro Capt:style=Caption,Regular
```

这时你要选取的字体名应该是“仿宋_GB2312”，“黑体”，“Garamond Premier Pro”这样的根字体名。对中文来说选取逗号后面的字体名并无大碍，但对英文来说则容易出错，最好是由 X_YT_EX 来管理字体的变化而不是手动指定具体的变体。

2.4 外部字体调用

有时并不需要为了测试某种字体而将其装入系统，或者运行 **fontconfig**，只希望把字体放在与源文件同一目录时可以被 X_YT_EX 找到，这时可以用下面的命令（注意：中括号中写的是字体文件名而非字体名）：

```
\font\localfont="[geo703l.ttf]" at 14pt
{\localfont This is Geometric Slabserif 703 Light BT}
```

效果（把 **geo703l.ttf** 放到源文件目录下）：

This is Geometric Slabserif 703 Light BT

3 使用 fontspec 宏包

X_YTeX 默认的字体调用方式比较繁琐，主要是不太直观，为此，Will Robertson 写了 fontspec 宏包来简化它的操作。fontspec 宏包功能繁多，这里也只是结合上面的例子介绍一些快速上手的基本功能，更多请查阅 fontspec 宏包说明文档。

3.1 基本介绍

我们可以简单地载入 fontspec 宏包：

```
\usepackage{fontspec}
\usepackage{xunicode}
\usepackage{xltextra}
```

其中 xunicode 启用一些 L^AT_EX 中的功能如 \%, \\$, \textbullet, \"u, 等；xltextra 加入了一些针对 X_YTeX 的改进并且加入了 \XeTeX 命令来输入漂亮的 X_YTeX logo。

下面介绍一些 fontspec 宏包的常见参数：

- cm-default 默认载入 Computer Modern 字体而不是 Latin Modern 字体
- no-math 不改变数学字体
- no-config 不载入 fontspec.cfg 文件，该文件用来个性化 fontspec 的功能，跟 cls 文件类似，可以放在源文件同一目录下以取代系统默认设置
- quiet 命令行不输出复杂的信息，全部存入 .log 文件中

为了使用 T_EX 风格的连字符，可以加入：

```
\defaultfontfeatures{Mapping=tex-text}
```

中文的断行问题以前可以通过在文档开头加入

```
\XeTeXlinebreaklocale "zh"
\XeTeXlinebreakskip = 0pt plus 1pt
```

来解决，目前 xeCJK 和 zhspacing 宏包都很好地解决了这个问题。

3.2 调用字体

载入 fontspec 宏包后，就可以用 \fontspec 命令来调用字体：


```

\def\pangram{The five boxing wizards jump quickly.\\}
{\fontspec{Arno Pro} \pangram
  {\itshape \pangram}
  {\scshape \pangram}
  {\scshape \itshape \pangram}
\bfseries \pangram
  {\itshape \pangram}
  {\scshape \pangram}
  {\scshape \itshape \pangram}
}

```

效果:

The five boxing wizards jump quickly.

The five boxing wizards jump quickly.

THE FIVE BOXING WIZARDS JUMP QUICKLY.

THE FIVE BOXING WIZARDS JUMP QUICKLY.

The five boxing wizards jump quickly.

The five boxing wizards jump quickly.

THE FIVE BOXING WIZARDS JUMP QUICKLY.

THE FIVE BOXING WIZARDS JUMP QUICKLY.

从这里可以看出XeTeX调用字体的优势: 只需要输入字体名, 后面的变体命令 (`\itshape`, `\bfseries` 等) 都能自动地找到相应的字体。

若要为整个文档指定默认字体:

```

\setmainfont{Adobe Garamond Pro}
\setsansfont{Myriad Pro}
\setmonofont{Consolas}

```

2.2中提到的字体属性, 在 `fontspec` 中也可以方便地调用:

```

{\fontspec[Numbers=OldStyle, Colour=8000FF]{Brioso Pro}
Back to 1945!}\\
{\fontspec[Numbers=SlashedZero]{Myriad Pro}
Hello TUG2008! 0123456789}

```

效果:

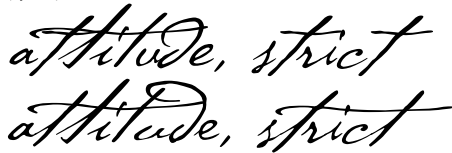
[Back to 1945!](#)

Hello TUG2008! 0123456789

不过依然有一些专业的字体属性没有默认出现在 `fontspec` 中，Contextual Alternates 就是一例，这时还是需要用到基本的字体调用功能，请对比下面两种手写的不同效果：

```
{\font\x="P22 Cezanne Pro:+salt" at 32pt \x attitude, strict}\\
{\font\y="P22 Cezanne Pro:+calt" at 32pt \y attitude, strict}
```

效果:



后者的效果显然要更优一些。

更新 (2008.4.25)：在 `fontspec` 宏包说明文档中看到，其实作者在 `v1.13` 版本开始考虑到了这样的情况，提供了一个 `RawFeature` 的选项：

```
{\fontspec[RawFeature=+calt]{P22 Cezanne Pro}attitude}
```

注：关于 *OpenType* 专业字体的专业属性还可以用 $X_{\text{L}}\text{TeX}$ 举出很多例子，由于使用范围不广，在此暂时不一一展示。

3.3 外部字体调用

相应地，外部字体调用的语法也稍作修改：

```
{\fontspec[ExternalLocation]{geo7031.ttf}
This is Still Geometric Slabserif 703 Light BT}
```

效果:

This is Still Geometric Slabserif 703 Light BT

3.4 配合 xeCJK 宏包调用中文字体

$X_{\text{L}}\text{TeX}$ 默认对中文的支持并不好，对此不同的作者给出了不同的解决方案，其中 `xeCJK` 宏包是跟 `ctex` 文档类配合很好的一个。在这个宏包中，对中文的调用不能直接使用 `\fontspec` 命令，而应该如下设置：

```

\setCJKmainfont[<font features>]{<font name>}
\setCJKsansfont[<font features>]{<font name>}
\setCJKmonofont[<font features>]{<font name>}
\setCJKfamilyfont{<family name>}[<font features>]{<font name>}

```

在最后一语句定义新的 CJKfamily 后，可以用 \CJKfamily 来改变当前的中文字体。可以看出，这些命令与 fontspec 的 \setmainfont 等命令可以同时使用，分别管理中文和西文字体互不冲突。而且 fontspec 中的一个字体链接的功能对没有粗体的中文字体尤其有用：

```

\setCJKmainfont[BoldFont={Adobe Heiti Std}, ItalicFont={Adobe Kaiti Std}]
{Adobe Song Std}

```

{默认字体} \\

{\bfseries 粗体效果} \\

{\itshape 斜体效果} \\

将得到：

默认字体

粗体效果

斜体效果

这是本文的字体设置，也就是说，正文字体使用 Adobe Song Std，在需要粗体的地方使用黑体，而不是使用伪粗体；中文根本没有斜体，倾斜的话很不好看，不如直接用楷体来代替。

参考文献

- [1] Michel Goossens, **The X_gTeX Companion**, 2008 年 1 月 21 日
- [2] 孙文昌, **xeCJK 宏包说明文档**, 2008 年 3 月 20 日
- [3] Will Robertson, **fontspec 宏包说明文档**, 2007 年 9 月 1 日