Linux 字体配置要略
2012-02-15

我并不清楚 fontconfig、 xfont、 libfreetype 或某某库等等之间的界限与联系，其实作为普通用户也不必对其一清二楚，所以文中都以“系统”代称。

本文不咬文嚼字地谈论字体、字形、字库等狭义的术语概念。看到文中任何一个词，请想得宽一点，不要把思维限制在一条狭缝之中。

为求排版，流程图中对表条件选择的菱形框进行了纵向压缩，从而变成了六边形。

文中多处引号不对称，甚至可说非常混乱，这由 OpenOffice.org 很多年前就存在直到现在的 LibreOffice 还未消除的 bug 所致。手动一个个地去设字体工作量太大，而我又不懂宏，请不要怪我。见很多人报过该 bug，我也报过一次，但都杳无音信。红旗的 RedOffice 很多年前就已消除该 bug，但似乎并未贡献其补丁。

1 一些基础

字体分类不同角度自有不同分法。西文常分衬线、非衬线、等宽三类，中文常分宋、黑、楷、行、草、隶、篆等多类。至于衬线等宽、非衬线等宽、书宋、报宋、中黑、粗黑之细分，通常并无多大必要。

传统认为衬线比非衬线更易识读，而非衬线比衬线更为饱满醒目。所以出版物中，正文常是西文衬线中文宋，标题则是西文非衬线中文黑。屏幕由于分辨率太低，小字体衬线表现并不理想，故常用非衬线。汉字都等宽，但西文为求美观通常都不等宽，等宽主要用在程序代码等力求准确之地。

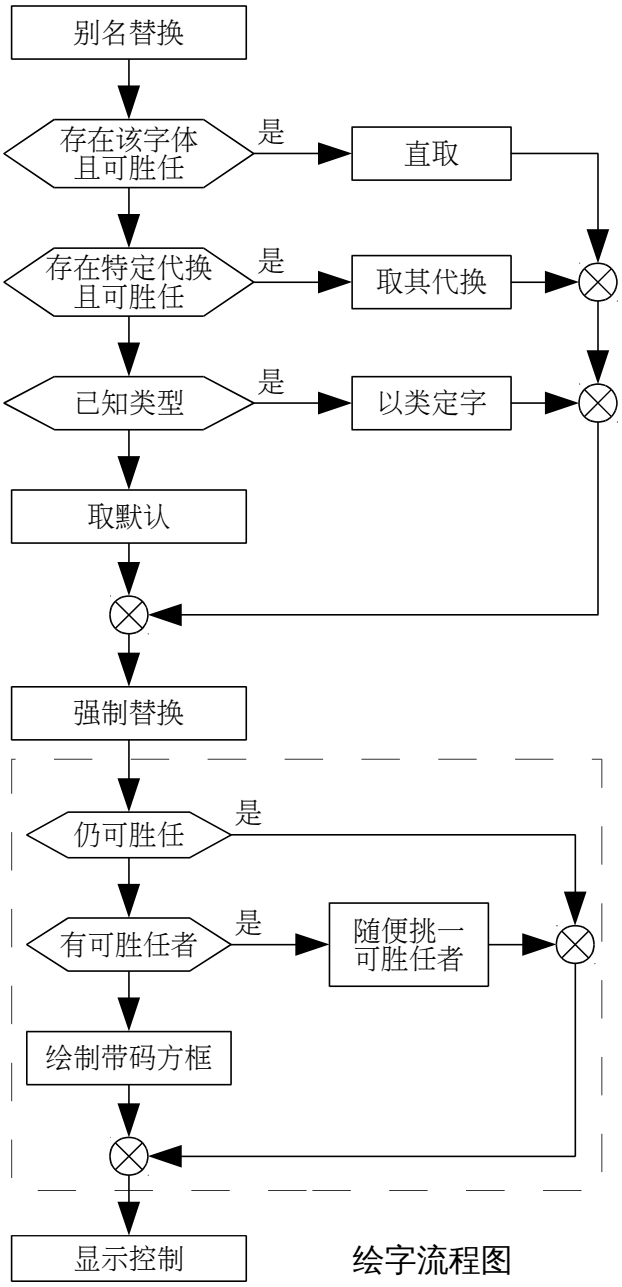
西方人名常分“first name”“last name”，生活中他们的各种表单也都特别突出这两个概念，通常都必须分开填写，而中国通常都是姓与名集中讨论不会要求分开填写。计算机系统中，字体名同样有“family（族）”“style（风格）”之别，是必须分开的硬指标，两者合起来才算是完整的字体名。西文字体通常两者都会用上，而中文字体通常只用族名。所以你能看到“Liberation Sans”这样的西文字体有“Regular”“Bold”，而“汉仪中黑”这样的中文字体却只有“Regular”。不要抱怨某某中文字体没有粗体云云，不是没有而是名字不同罢了，比如“汉仪粗黑”“汉仪超粗黑”，我国印刷业对字体命名素来如此。所以我们通常所说的字体名在计算机系统中仅指族名，而非包含两者之全名。当然，现在有些中文字体也用西方那套“两段式”命名，“微软雅黑”便是一例。

西文斜体常用于引文、注释等，对此中文则常用仿宋、楷体等。中文没有斜体，或许你可以钻着牛角尖说“仿宋不就是斜体吗，人家只是向上而不是向右斜而已”。

“mono”“宋体”“黑体”这样的类型名，既抽象又具体，有时又被称作虚拟字体。抽象性在于它可用来表示一类字体的总称，具体性在于它背后总有某个实际字体为其担当。文档或许仅指明要“mono”，但我们仍能看到我们所预期的结果，比如“文泉驿等宽正黑”。原因就在于我们总会为这些类型派出一个合适的代表为其担当，或者说填实这些虚拟字体。应用程序只要说个类型名，就能得到一个相应的实际字体，这就是以类定字。

2 需求逻辑

配置字体的基本原则是“尽量呈现文档原貌”而不是“用某种字体一统天下”。我想应该没多少人会反对这个观点，虽然人世间总不乏个性非凡者。据此原则我们不难得出一个如右图所示这样的逻辑。该图是我紧扣基本原则，根据系统原有诸配置文件整理而得的绘字流程，我的配置及本文解说也正是“以此为纲”而进行。只要理解了该逻辑，就能把握本文之重点，掌握配置之要诀。



绘字流程图

显示控制之前所有步骤可统称为“择字”。它处理应用程序的申请，解决究竟该拿什么字体来显示给用户看的问题，其目的就是显示正确的字体。对应用程序而言，这是个由“所望”产生“所得”的过程，字体配置的重点便在于此。

相应的显示控制则可称之为“控字”。它主管具体某字体的点阵、平滑、微调等渲染性工作，其目的就是把字体显示得漂漂亮亮。

西文，“mono”“monospace”二者无异，“sans”“sans serif”“sans-serif”三者同一；中文，“宋体”“宋體”也都互为别名并无二致。对此可先将诸别名替换为唯一“标准”名称供内部使用，化多为一，以便处理。该阶段主要针对类型，而不是具体的某个字体。

存在该字体即为已安装该字体。可胜任则指该字体中存在所需字符。直取即直接取用。我的系统安装有“文泉驿正黑”，当应用程序要用“文泉驿正黑”绘制“桂林山水”几个字时，自然当直接取用“文泉驿正黑”而不是别的字体。

特定代换指遇到未安装字体时尽可能用最适当的字体予以代换。比如，用“Liberation Serif”代替“Times New Roman”，用“文泉驿微米黑”顶替“微软雅黑”，显然这比随便用个字体作替身要更为妥当。

当某字体既未安装又无其特定代换信息时，就要看它属于哪种类型，然后再用那种类型的实体予以代换。例如，我既未安装“新細明體”，又无其特定代换信息，而它显然属于宋体一类。所以我就要用宋体类的实体去顶上，而不是盲目地使用默认字体。这便是以类定字。

倘若连那个字体的类型都不清楚，即既未安装又无登记任何信息的未知字体。此时也只好拿默认字体来用。假如某网页要“MyFont”或“无敌字体”，我哪知道这是什么字体啊，这时

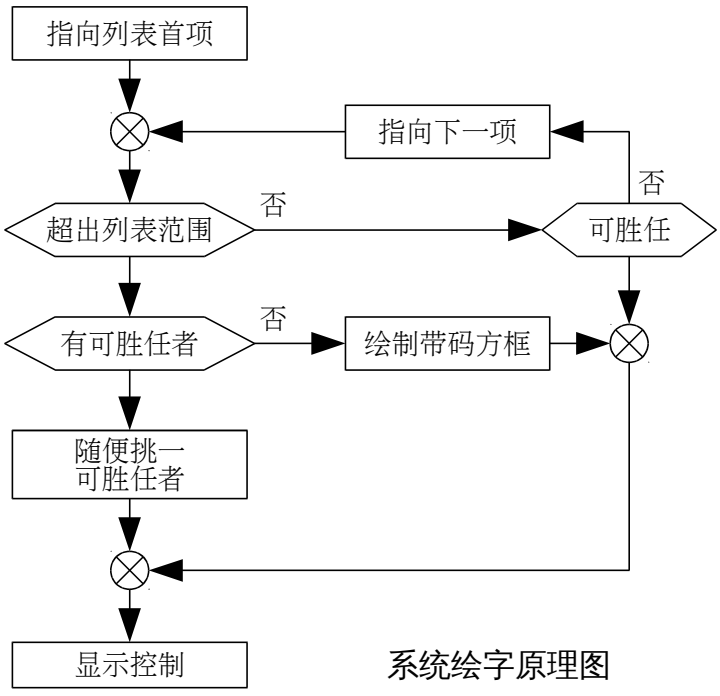
也就只好用默认字体顶上了。

强制替换在初步决定用什么实际的字体之后，通过一些规则进行。比如用专门的西文字体代替中文字体中的西文部分或者用黑体替换宋体的粗体风格等。

强制替换后或所设默认字体都有可能出现不可胜任的情况，所以系统要再次判断并做出最终决定，图中虚线框内部分。该部分完全由系统自动进行，不可配置，所以也不必太过关心。带码方框即中间写有字符编码的方框，而非空方框。

3 系统原理

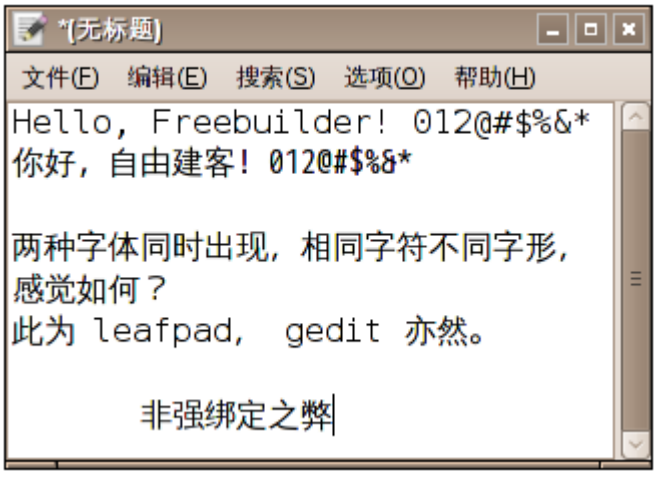
系统提供一个列表，其中每一项都记载着一个字体名。通常初始时列表只包含一项，也就是应用程序所申请的字体，所以该列表可称之为“所望列表”。择字过程就是对这个列表插插改改的过程。最后系统拿着这份字体清单进行绘字，如右图所示。从列表第一项开始，看其能否胜任，能则进入显示，不能则看下一项。若扫完整个列表仍没有可胜任者，再考虑自动挑选可胜任者或绘制带码方框的问题。



系统绘字原理图

有些程序会比较笨，只要前一项所指的字体已经安装，就绝不会看后面的。若该字体不可胜任的话就立马绘制一个空方框或空白。poppler、xterm 就属于此列。所以当字体没配置好时，用 evince、epdfview、okular 等后端为 poppler 的程序打开一些 pdf 文档，出现的是方框或空白而不是文字。

该列表各项还有一个叫做 binding 的属性，指示绑定类型，取值 strong、weak、same。其中 strong 最为简单，意为对于任何字符只要这个字体能胜任，那就用它搞定，只有不能胜任时才看后面的。换言之就是非万不得已不求人。weak 则是看一个要绘制的字符串中所有字符属于哪个语言集，若该字体覆盖该语言集则用，否则看下一项。same 似乎还要看语境，不同绑定方式组合更是闹不明白。总之很多时候不得不用 strong，否则便会出现如右图那样的现象。早期的 Ubuntu 默认配置就



存在此问题，直到推出了使用强绑定的 69-language-selector-*.conf 系列文件。所以我的配置也只用 strong，一路强硬到底！前面的系统绘字原理图也正是基于强绑定而得。

4 配置说明

请注意，本节内容仅作为手册（`man 5 fonts.conf`）的补充而非替代。对于本节可快速浏览而过，待到后面具体应用时再适当地返回查阅。

4.1 <match target="pattern">

target 指示处理目标，取值 pattern、 font，缺省 pattern。 pattern 字面意思即模式、样板，网上多数文章都称其为“字体样板”。其含义为应用程序所提出的申请，或说所期望的样子，或可简称“所望”。 font 指系统最后指派给应用程序的字体，或对应地称之为“所得”。从需求逻辑不难看出“择字所论，皆为所望。控字所操，概为所得。”

<match> 包括用 <test> 表示的条件部分和用 <edit> 表示的动作部分。可顺序使用多个 <test> 表示多个串联（逻辑与）条件，多个 <edit> 表示多个先后动作。代码示例：

```
<match>
    <test name="family">
        <string>宋体</string>
        <string>宋體</string>
    </test>
    <edit name="family">
        <string>zh-SongTi</string>
    </edit>
</match>
```

各 <test> 之间不得穿插有 <edit>。错误代码示例：

```
<match>
    <test ... /test>
    <edit ... /edit>
    <test ... /test>
    <edit ... /edit>
</match>
```

4.2 <test qual="any" name="property" target="default" compare="eq">

target 同样表示处理目标。当然，此处说成“检测目标”要更为妥当些。其缺省值为外层 target 之值，而非确定的 pattern。即外层 <match target="font">，那它就是 font。

compare 指示比较方式，缺省 eq。手册上有，不必多说。只是有两个针对字符串类型的 contains 及 not_contains 手册上没提，但看字面意思便可一清二楚。

qual 针对列表类型的属性，取值 any、 all，缺省 any。目前我尚不清楚除 family 之外还有什么属性是列表类型的。姑且先把要处理的列表称为“目标列表”，代码中给定的值列表叫作“参数列表”。假设某目标列表为 ["str1", "str2", "str3"]，对于下面代码，参数列表为 ["str1", "str2"]。若 qual="any" 则整个 <test> 结果为真，若 qual="all" 则其结果为假。

```
<test qual=... compare="eq">
    <string>str1<string>
    <string>str2<string>
</test>
```

原因在于 qual 指示目标列表而非参数列表，参数列表永远以 any 指示。所以当 qual="any" 时，只要目标列表中任何一项与参数列表任何一项相等，整个结果就为真。若 qual="all" 则是说目标列表所有项都要等于参数列表中的任何一项，自然结果为假。很无赖，哪有既等于 str1 又等于 str2 的。但试想若参数列表只有一项或 compare 不是 eq 呢?

4.3 <edit name="property" mode="assign" binding="weak">

binding 如前面系统原理中所说指示绑定类型，此处其缺省值为 weak。

mode 主要针对列表类型的属性，手册上也都有说明，缺省为 assign。对非列表类型自然只能是 assign。

4.4 <alias binding="xxx">

binding 如前面所说指示绑定类型，在此缺省何值尚不清楚，总之不是 strong。

中间配合 <family>、 <prefer>、 <accept>、 <default> 实现所需功能。如下代码:

```
<alias binding="sgrong">
    <family>Font1</family>
    <family>Font2</family>
    <accept>
        <family>FontXXX</family>
        <family>FontYYY</family>
    </accept>
</alias>
```

其实功能上等效于:

```
<match>
    <test name="family">
        <string>Font1</string>
        <string>Font2</string>
    </test>
    <edit name="family" mode="append" binding="strong">
        <string>FontXXX</string>
        <string>FontYYY</string>
    </edit>
</match>
```

<prefer>、 <default> 也有相应的等效形式，区别只在 mode 之值， <accept> 是 append，<prefer> 是 prepend， <default> 则是 append_last。

<match> 与 <alias> 的区别主要在于适用范围的宽窄，看 <alias> 的等效 <match> 代码便知。其次， <alias> 更利于优化，无论当前系统是否已经经过优化。总之要尽量使用 <alias>。

5 以例说法

如果你看到本文之时并未见随同附上的 fonts.conf 文件，可到 http://freebuilder.ys168.com/ 下载此文章的最新版本的压缩包，内含对应版本的 fonts.conf 文件。该文件是我自己目前所用的配置文件。该文件替换系统原有的 /etc/fonts/fonts.conf，从而完全忽略 /etc/fonts/conf.d/*、/etc/fonts/local.conf、 ~/.fonts.conf*。其实我也并不喜欢这种替换系统文件的做法，但确实非常无奈。一是系统原有配置非常混乱，太过复杂，不便解说。二是我没想出较好的拆分文件的方法。

若要原封不动地使用该配置文件建议先安装如下字体：

“Liberation”“Libris ADF Std”“文泉驿正黑” “AR PL UKai”，其在 Debian / Ubuntu 中的安装命令为：

```
aptitude install -R ttf-liberation ttf-adf-libris ttf-wqy-zenhei ttf-arphic-ukai
```

“文鼎ＰＬ报宋二GBK”“HAN NOM”“花園明朝”三款可在 http://freebuilder.ys168.com/ 找到最新有效下载链接。

当然，如果只有一种中文字体，该配置也能工作，只是没有五彩斑斓的效果罢了。

该文件共分八大部分，其中头部部分简单且变动几率不大，查手册便可基本明了，主要取自原来的 /etc/fonts/fonts.conf。后七部分是配置中的重点，本节也着重介绍后七部分。fonts.conf 采用一种看似顺序无关的 xml 格式，实际各段落之间顺序密切相关，不得随意调换。所以我觉得还不如用命令式语言来得实在。

5.1 别名替换：标准命名第一

第 49~107 行，先看第 91 行：

```
<match>
    <test name="family">
        <string>mono</string>
    </test>
    <edit name="family">
        <string>monospace</string>
    </edit>
</match>
```

把“mono”替换为“monospace”，后期只处理“monospace”，“mono”仅作其别名。其语法简单，看手册和上一节的配置说明便很容易弄懂。同样，后面的非衬线和前面的几种中文类型也都是如此。

西文的标准三类中，实际有别名的只有两类，“serif”没有其它别名。这点可在系统原先的 /etc/fonts/fonts.conf 中找到。

中文类型的汉字名称，我都替换成了拼音。这样可提供更高的灵活性，方便国际化，同时也避免繁体简体派系之间的争吵。另外，数类到底是几类，请看下面字体归类部分。

5.2 字体归类：有根可循第二

第 109~315 行，先看第 217 行：

```
<alias binding="strong">
    <family>Bitstream Charter</family>
    <family>Bitstream Vera Serif</family>
    <family>DejaVu Serif</family>
    ...
    <accept>
        <family>serif</family>
    </accept>
</alias>
```

此段上层语义就是说明“Bitstream Charter”“Bitstream Vera Serif”等字体属于“serif”类型。系统底层实现便是在所望列表中只要碰到这些字体，就在其后添加一项叫作“serif”的虚拟字体。该虚拟字体作为标志项为后面的以类定字做铺垫。

/etc/fonts/conf.d/45-latin.conf 等文件中的归类方法用的是 <default>，我之所以用 <accept> 原因在于 Mozilla Firefox 不买 <default> 的账。因为 Firefox 的所望列表一开始就已经包含两项，第一项是网页指定的字体，第二项就是其首选项中的“默认字体”。这会导致其无法以类定字。另外，这也正是为什么 Firefox 中的默认字体总是自己的一套而不是下面将要介绍的默认字体。有兴趣的朋友可将代码化为 <match> 形式予以分析。

此部分要点在于在 <accept> 之前列出的是已知的实际字体的名称，无论是否已安装该字体，只要是常见的都应该列出。<accept> 中只有一项，那就是字体类型名称。

西文除标准三类外还有“fantasy”“cursive”两个附加类，其实我也不懂这两类的相关知识，只是从 /etc/fonts/conf.d/60-latin.conf 中抄得。

中文，我只定义了常见的四类，没有为行书、草书等单独定义一类。一是用这种字体的情况较少，二是用这种字体时通常都是直接用“华文行楷”“方正黄草”这样的实体名，并不需要“行书”“草书”这样的虚拟字体名。

“新宋体”“細明體”这样的等宽字体，并未归入“monospace”而是归入“zh-SongTi”。因为网页中指定要这些名字的，多半是冲其中文字形，而不是冲其西文等宽，要等宽的网页多会指定“Courier New”“Lucida Console”之类。

默认字体

第 296 行：

```
<match>
    <test name="family" qual="all" compare="not_eq">
        <string>sans-serif</string>
    </test>
    <test name="family" qual="all" compare="not_eq">
        <string>serif</string>
    </test>
    <test name="family" qual="all" compare="not_eq">
```

```
        <string>monospace</string>
    </test>
    <edit name="family" mode="append_last" binding="strong">
        <string>WenQuanYi Zen Hei Mono</string>
        <string>HAN NOM A</string>
        <string>HAN NOM B</string>
        <!--
        <string>sans-serif</string>
        -->
    </edit>
</match>
```

这段代码源自 /etc/fonts/conf.d/49-sansserif.conf。在所望列表经过前两个阶段处理后，只要是三大标准类型的都会出现“serif”“sans-serif”或“monospace”的标志项。若在此阶段找不到这三个标记中的任何一个，系统就有理由相信该列表所列的通通都是“未知”字体，就应当为其指派个默认字体以作候补。

网上有很多文章说解决程序方框或空白无字问题，说是修改 49-sansserif.conf，甚至将其删除。但有多少人懂得这段代码究竟何意呢？原始代码的含义是三类之外者以非衬线论，看似完美的配置但很可能会出问题，原因就在于前面提过的有些程序比较笨。为了那些程序，我们不得不在“sans-serif”前面添加一个覆盖字符集尽可能大且没有参与强制替换的字体，比如说“WenQuanYi Zen Hei Mono”，并同时指定 binding="strong"。总之要保证最后的所望列表中第一项是覆盖字符集尽可能大的字体。此处去掉“sans-serif”而直接加上“HAN NOM A/B”，这可避免对“sans-serif”的展开操作所带来一点点无谓的效率损失。有些 Ubuntu 用户通过修改 69-language-selector-*.conf 来解决该问题，其方式不同，但原理相同。

另外，删除这段代码的后果将是系统自动决定使用何种字体，可能会出乎意料。

5.3 特定代换：务求最佳第三

第 317~475 行，先看第 443 行：

```
<alias binding="strong">
    <family>Times New Roman</family>
    <family>Thorndale</family>
    <family>Thorndale AMT</family>
    <accept>
        <family>Liberation Serif</family>
    </accept>
</alias>
```

“Times New Roman”等都是没有安装且有共同的合适替身的字体，不像字体归类中无论是否已安装都予以列出。“Liberation Serif”作为它们的替身，自然必须是确实已安装字体，否则将形同虚设。

/etc/fonts/conf.d/* 是特定代换在前字体归类在后，与本配置文件的字体归类在前特定代换在后此正好相反，而前面我又说过“各段落之间顺序密切相关，不得随意调换”，原因何在？因为本配置文件字体归类用的是 <accept> 而不是 <defalut>，不明白的将其化作 <match> 形式便可一清二楚。

另外，第 452 行。对于喜欢苹果 Lucida 风格字体的朋友可安装 Luxi 字体（Debian / Ubuntu 安装命令： `aptitude install -R ttf-xfree86-nonfree`），再将该部分注释去掉即可。我并不是很喜欢 Lucida 风格，所以没有安装。再有，安装了“文泉驿微米黑”的朋友也可考虑用其作“微软雅黑”的替身，这份配置就像模板，如法炮制便是。

关联字体对

第 336 行：

```
<!-- HAN NOM -->
<alias binding="strong">
    <family>HAN NOM B</family>
    <prefer>
        <family>HAN NOM A</family>
    </prefer>
</alias>
<alias binding="strong">
    <family>HAN NOM A</family>
    <accept>
        <family>HAN NOM B</family>
    </accept>
</alias>
```

“HAN NOM A”及“HAN NOM B”都是已经安装的字体，两字体区别在于前者只包含 CJK 和 Ext-A 字符，而后者只包含 Ext-B 字符，两者呈互补关系。如果没有此部分，当应用程序指定要“HAN NOM A”时，又出现 Ext-B 字符，那绘制 Ext-B 字符的未必就是“HAN NOM B”，可能会是“花園明朝B”，这样将造成风格不一致。反之若应用程序指定“HAN NOM B”亦然。这两个要么不出现，要么就成对出现。另外的“花園明朝”情况也是如此。

5.4 以类定字：通用匹配第四

该部分非常关键，因为多数时候应用程序都只说要“serif”“宋体”这样的虚拟字体。该部分的责任就是填实所有字体归类部分定义过的虚拟字体。

第 477~555 行，先看第 530 行：

```
<alias binding="strong">
    <family>serif</family>
    <prefer>
        <family>Liberation Serif</family>
        <family>AR PLBaosong2GBK Light</family>
        <family>HAN NOM A</family>
        <family>HAN NOM B</family>
    </prefer>
</alias>
```

意思是对“serif”类型用“Liberation Serif”等顺次填实。这几个字体都是确已安装的字体。很多配置文件该列表都非常长，你看到这个会不会有种异常短的感觉？够了，不短了，多多列举也不过是毫无意义。

对于“monospace”，很多朋友可能想在前面加上“DejaVu Sans Mono”，这将会因英文宽度

超过半个中文而造成中英混排时整体上的不齐。如果你对此无所谓的话，做也无妨。话说回来，即便是纯粹只用某一种字体，比如“WenQuanYi Zen Hei Mono”在 13px 的时候也还是对不齐的，因为 13 是奇数。

至于仿宋体，因为目前自由字体中唯一的仿宋体“cwTeX 仿宋體”只有繁体部分且尺寸总比一般的要小一号，所以只好先用“AR PL UKai”顶着。

5.5 强制替换：偷梁换柱第五

第 557 行：

```
<match>
    <test name="family">
        <string>文鼎ＰＬ报宋二 GBK</string>
        <string>AR PLBaosong2GBK Light</string>
    </test>
    <test name="pixelsize" compare="less">
        <double>16</double>
    </test>
    <edit name="family" binding="strong">
        <string>WenQuanYi Zen Hei</string>
    </edit>
</match>
```

“文鼎ＰＬ报宋二GBK”的屏显效果已经相当好了，当年的“AR PL UMing”哪有这么好效果。即便如此，该字体小于 16px 的仍是不尽如人意。别说是它，就是很多商业宋体，小到这尺度也都没法看。最后觉悟，还是把 16px 以下的宋体换成黑体算了，毕竟多数时候看的是网页，多数网页字体都小于 16px。当然，你也可以不用宋小黑代，而用内嵌点阵的“AR PL UMing”整个“点阵与矢量齐行，清晰共平滑一色”。

注意，如果 LibreOffice 新建文档时的默认字体小于 16px（96DPI 时 12 磅小四号），此举将可能导致 LibreOffice 无法使用该字体而只能用被替换后的字体，甚至更严重的看似莫名其妙的问题。所以必须保证“工具 -> 选项 -> LibreOffice Writer -> 标准字体（中日韩） -> 默认 -> 大小”起码是小四号。 OOo 估计有同样问题，但我没有试过。

纳闷问题

有两个令人纳闷的问题，其一是测试条件中必须把“AR PLBaosong2GBK Light”放在“文鼎ＰＬ报宋二GBK”之下，对于像“文泉驿正黑”那样有三个名字的字体也是。总之必须把英文名放在最下面，其上的几个顺序倒是无所谓。所以把“文泉驿正黑”的英文部分替换为“Liberation Sans”的正确代码应是：

```
<match>
    <test name="family">
        <string>文泉驿正黑</string>
        <string>文泉驛正黑</string>
        <string>WenQuanYi Zen Hei</string>
    </test>
    <edit name="family" mode="prepend" binding="strong">
```

```
        <string>Liberation Sans</string>
    </edit>
</match>
```

其二便是测试条件中只能是一个字体的多个名字，不能有第二个字体，否则必然只有一个有效。要对多个字体进行替换操作，就必须单独多写几遍类似的代码。例如如下代码，必然导致只有“文泉驿微米黑”替换成功而“文泉驿正黑”替换失败。

```
<match>
    <test name="family">
        <string>文泉驿微米黑</string>
        <string>文泉驛微米黑</string>
        <string>WenQuanYi Micro Hei</string>
        <string>文泉驿正黑</string>
        <string>文泉驛正黑</string>
        <string>WenQuanYi Zen Hei</string>
    </test>
    <edit name="family" mode="prepend" binding="strong">
        <string>Liberation Sans</string>
    </edit>
</match>
```

其实如果条件部分只是像这里这么的简单，大可换成 <alias> 形式，那样各字体名则不存在顺序问题，也可同时罗列多种需要替换的字体。例：

```
<alias binding="strong">
    <family>WenQuanYi Zen Hei</family>
    <family>文泉驿正黑</family>
    <family>文泉驛正黑</family>
    <family>WenQuanYi Micro Hei</family>
    <family>文泉驿微米黑</family>
    <family>文泉驛微米黑</family>
    <family>Microsoft YaHei</family>
    <family>微软雅黑</family>
    <prefer>
        <family>Liberation Sans</family>
    </prefer>
</alias>
```

另外，还有其它的诸如黑体替换粗宋体等，都可在此部分进行。总之用 <match> 可配置出各种复杂的替换动作，但必须注意 LibreOffice，强制替换很可能导致其工作不正常，要尽可能地避免强制替换。

5.6 伪造字体：无中生有第六

第 574~607 行，该部分来自 /etc/fonts/conf.d/90-synthetic.conf，只删除了关闭点阵一行，以使点阵字被伪斜、伪粗后依然是看似点阵的样子。当然，可能有些朋友觉得斜体还是矢量的好看些，那就请加回去那一行。

QT 不支持伪斜、粗体。奇怪的 LibreOffice 却是部分支持，转成 pdf 之后更是混乱。另外， Adobe Reader 打开有伪斜体、伪粗体的文档时，很可能撕裂或是黑成一坨，打印出来更是惨不忍睹。所以，千万不要指望能在 pdf 中正确使用伪斜体、伪粗体。中文排版的正道是换字体而不是指望某个字体有不同的“风格”。

5.7 显示控制：花容月貌第七

第 609~686 行，先看第 613 行：

```
<match target="font">
    <edit name="antialias"><bool>true</bool></edit>
    <edit name="autohint"><bool>false</bool></edit>
    <edit name="hinting"><bool>true</bool></edit>
    <edit name="hintstyle"><const>hintslight</const></edit>
</match>
```

现在到了显示控制阶段，该阶段的处理目标为“所得”，所以要显式指明 target="font"。该部分涉及到的字体属性也比较多，请参阅手册说明。

这段代码没有条件语句，所以对任何字体都有效。它开启平滑消锯齿（antialias）及人工微调（hinting）而关闭自动微调（autohint）。 hintslight 微调等级在我的系统中适用于大多数字体，尤其是中文字体，或许你的未必。就几个值而已，都试试看吧，由实际体验决定。 GTK 很有可能并不听 hintstyle 选项的话，请看后面优化液晶部分。

其实说人工微调似乎并不是很恰当，称之为先天性微调或许更合适。它只不过是集成在字库内部的微调信息，该信息可能是在字库开发时用机器处理而未必是人工处理的。所谓自动微调则是在绘字时完全由系统进行处理，即相对的后天性微调。在大多数情况下开启先天性微调效果会更好，即便是中文。

第 665 行：

```
<match target="font">
    <test name="pixelsize" compare="less">
        <double>7.5</double>
    </test>
    <test name="family" compare="contains">
        <string>Liberation</string>
        <string>DejaVu</string>
        <string>Nimbus</string>
        <string>Bitstream Vera</string>
    </test>
    <edit name="autohint"><bool>true</bool></edit>
    <edit name="hinting"><bool>false</bool></edit>
</match>
```

这段表示当要显示很小个的字时，关闭先天性微调改用后天性微调。其实到这尺度的英文无论怎么微调都已经很难辨认，中文则更不必说了。

第 678 行：

```
<match target="font">
    <test name="family" compare="contains">
        <string>Liberation</string>
        <string>DejaVu</string>
        <string>Nimbus Mono L</string>
        <string>Bitstream Vera</string>
    </test>
    <edit name="hintstyle"><const>hintfull</const></edit>
</match>
```

这段指出单独对这几种字体启用 hintfull 微调等级，尤其是 DejaVu，在该等级能获得最佳效果。

很奇怪的是这里的测试条件中又可列举多种字体，而不像前面强制替换中的只能列举一个字体的多个别名。即便这里 compare="eq"，然后字体名写上“Liberation Serif”等的全名。例如以下代码是可行的：

```
<match target="font">
    <test name="family">
        <string>Liberation Serif</string>
        <string>Liberation Sans</string>
        <string>DejaVu Serif</string>
        <string>DejaVu Sans</string>
    </test>
    <edit name="hintstyle"><const>hintfull</const></edit>
</match>
```

优化液晶

第 619 行：

```
<match target="font">
    <test name="pixelsize" compare="less">
        <double>18</double>
    </test>
    <edit name="rgba"><const>rgb</const></edit>
    <edit name="lcdfilter"><const>lcdlight</const></edit>
</match>
```

这段代码指示当字体小于 18px 时，启用次像素功能，并设 LCD 过滤器为 lcdlight。同样，rgba 及 lcdfilter 之值也要由实际体验决定。 CRT 应当去掉这段。

GTK 除了可能不服从 hintstyle 选项还有可能不服从 lcdfilter 选项， Debian squeeze 便是如此， Debian wheezy 和 Ubuntu 倒是听 lcdfilter 的话。可以给 Debian squeeze 升级 libcairo，以求能服从 lcdfilter。我试过，这将导致 GTK 程序慢得像 Java，而所能提升的显示质量也并不明显，得不偿失。关键还是在 hintstyle 上。

给 Debian squeeze 重编译 cairo，以使 GTK 服从 hintstyle：

```
wget http://ftp.tw.debian.org/debian/pool/main/c/cairo/cairo_1.8.10-6.dsc
wget http://ftp.tw.debian.org/debian/pool/main/c/cairo/cairo_1.8.10.orig.tar.gz
wget http://ftp.tw.debian.org/debian/pool/main/c/cairo/cairo_1.8.10-6.debian.tar.gz
dpkg-source -x cairo_1.8.10-6.dsc
cd cairo-1.8.10/

# 修改源码，参见 http://moto.debian.tw/viewtopic.php?f=11&t=11951
vi src/cairo-ft-font.c #{
  /* 注释掉约第 1451 行， _cairo_ft_options_merge 函数中，内容如下 */
  /* if (options->base.hint_style == CAIRO_HINT_STYLE_DEFAULT) */
#}

# 编译，大概要十多分钟
dpkg-buildpackage -rfakeroot -us -uc
```

```
# 编译成功后有若干个包，但通常只安装下面一个即可
cd ..
dpkg -i libcairo2_1.8.10-6_amd64.deb

# 锁定该包版本，否则将在下次更新系统时换回原系统官方版本
aptitude hold libcairo2
```

若搞不懂上述各命令，可发贴请人帮忙编译。若是 AMD64 平台的，可直接用我编译好的包，可在 http://freebuilder.ys168.com/ 下载。但一定要注意版本，相同的才能用。

显示过宽

第 626 行：

```
<match target="font">
    <test name="lang" compare="contains">
        <string>zh</string>
        <string>ja</string>
        <string>ko</string>
    </test>
    <test name="spacing">
        <const>dual</const>
    </test>
    <edit name="spacing"><const>proportional</const></edit>
    <edit name="globaladvance"><bool>false</bool></edit>
</match>
```

传说某些时候等宽字体的英文会变得中文那么宽，用这段代码便可解决。我保留了此代码但却将其注释不用。因为我从来没碰到过传说中显示过宽的情况。

点阵开关

第 660 行：

```
    <edit name="embeddedbitmap"><bool>false</bool></edit>
```

embeddedbitmap 在手册中并没有说明，但意思很明确，就是开闭内嵌点阵。这里显式指出关闭文泉驿正黑中的点阵。有意思的是，对于“AR PL UMing”，在 GTK 中默认关闭点阵，而在 QT 中则是默认开启。所以，无论是要关还是要开都要显式指定。或许“AR PL UMing”只是个案，或许还有其它字体也是如此。总之，对内嵌点阵指明其开关总是有益而无害。

6 增删字体

6.1 字体来源及目录

众所周知的字体，许可允许的话通常都会被包含在系统的软件源中，安装非常方便，更重要的是不必操心其更新问题。源中没有的多数也可在网上找到，将其 ttf、 ttc 等字库文件置于配置文件开头部分众 <dir> 所指定的目录或其子目录之一即可。仅供某账户单独使用自然是 ~/.fonts，供所有账户使用的建议 /usr/local/share/fonts，因为 /usr/share/fonts 是系统包管理器所维

护的目录， /usr/X11R6/lib/X11/fonts 则是历史遗留。一些字体可能还包含一些配置文件，可留作参考。通常并不需要执行 `fc-cache -f -v` 命令，除非碰到问题。倘若你不放心，想执行一下，那就请先以 root 帐户身份执行一次，再分别以各普通帐户身份执行一次。

6.2 字体归类与替换

新安装的字体属于哪种类型，就应当将其所有名称加入配置文件的相应段落。字体名可由 `fc-list | sort` 命令获得，比如可见文泉驿正黑一行冒号之前有逗号分隔的简、繁、英三个名称。如果是很出名的字体，或许早已被登记，比如“微软雅黑”。不想用“微软雅黑”时，删除对应字库文件后应当保留其登记项，因为它太出名。

若新装的字体以前有特定替身，比如新装了“Times New Roman”，在特定代换中可将其删除。也可保留，这并不影响功能，而且删除该字体时也不用再把它加回来。

若新装了个能作某些字体更好替身的字体，比如 Luxi 系列。应当为其在特定代换部分加入相应的段落。删除字体时也最好将该段落删掉。

若新装的字体要做“serif”“宋体”等虚字体的实体，则要在以类定字部分做相应修改。

另外，或许你还需要强制替换，那只能自己琢磨了。

6.3 显示控制或其它

若新装字体在全局配置的效果下并不理想，可加入相应段落为其单独设置。可参考新装字库的配套说明或其附带的配置文件。

7 一些字体

7.1 文泉驿正黑 / 文泉驿微米黑

这似乎是目前唯一能自由使用的黑体，正黑授权条款为 GNU General Public License(GPL) 以及其中文翻译，微米黑的授权为 Apache 2.0 和 GPL v3 双授权。

正黑打印效果不及方正等的商业字体，但一般还能接受。微米黑压根就不适合打印。

微米黑屏显要到 16px 才耐看， 13px 相当勉强，笔画多的就不行了， 12px 则是令人忍无可忍。正黑小字屏显效果则要清晰得多，但 12~13px 小部分字会有些参差。

两种字体的标点都不是很好，尤其是引号。

7.2 文鼎ＰＬ报宋二GBK / 文鼎ＰＬ明體U20-L

文鼎于 2010 年推出这两套字体，原因或许是 2001 年捐赠的“文鼎ＰＬ简报宋”“文鼎ＰＬ細上海宋”两套字体内部编码或 GBK 或 Big5。这两款字体实际效果已非常好，不仅取代那

两款老字体很好，更是大有取代“AR PL UMing”之势，虽然它们并无内嵌点阵。

两种字体笔画都比较细，是真正的报宋级别，比“SimSun”还要细。屏显不错，但超过 12pt 小四号的打印则会显得有些空洞，不够饱满。建议用于填实“宋体”以作屏显。

报宋面向大陆，用的都是新字形，标点也比以前好多了，虽然还不是很好。明体则是面向港台，有部分旧字形。

7.3 AR PL UMing

合并了“文鼎ＰＬ简报宋”“文鼎ＰＬ細上海宋”以及萤火飞的点阵，并采用 ISO10646 编码。但标点明显不符合大陆风格，且字形新旧参杂。另外，“陕”字中“入”写成了“人”，从而看上去成了“陕”。不建议使用，虽然它被加入了 CJKUnifonts 计划。

7.4 AR PL UKai

与“AR PL UMing”对应，由“文鼎ＰＬ简中楷”“文鼎ＰＬ中楷”合并而得。似乎是目前唯一能自由使用的楷体。

7.5 HAN NOM

越南的覆盖超大字符集的自由字体，包括“HAN NOM A”“HAN NOM B”。前者覆盖 CJK 及 Ext-A，后者仅覆盖 Ext-B，两者互补。该字体用的是中国大陆的新字形，标点也是大陆风，虽然句号左右居中破折号也有点短小，但仍可谓是当前大陆 Linux 爱好者不可或缺的字体。笔画粗细同“SimSun”，可作为一般文档打印。但屏显效果不如“文鼎ＰＬ报宋二GBK”。

7.6 花園明朝

又一个覆盖超大字符集的自由字体，同样分为互补的“花園明朝A”“花園明朝B”。喜欢日本字形的朋友不要错过。我最喜欢它的一点是不含英文字符。

7.7 Liberation 系列

RedHat 发的，包含“Liberation Serif”“Liberation Sans”“Liberation Mono”三种字体，目标非常明确，就是取代微软的“Times New Roman”“Arial”“Courier New”。字形尺寸与微软的完全一样，所以用这三种字体代替微软的那三种不会导致文档排版的偏差与错乱。但令人郁闷的是衬线与微软的并不相同而且还并不怎么好看。所以，对于打印我通常优先选择 Nimbus 系列。

该系列字体屏显效果非常不错。大小与 DejaVu 系列相当，但间距要小些，所以看起来更舒服些，尤其是和中文字体配合时。

7.8 Nimbus 系列

包含“Nimbus Roman No9 L”“Nimbus Sans L”“Nimbus Mono L”三种。目标直指 Adobe 的

“Times”“Helvetica”“Courier”，主要供打印使用。与 Liberation 系列不同的是，其衬线看起来和微软的很接近，但尺寸不同。因为微软的那几款字体本来也就是针对 Adobe 的，字形和 Adobe 的几乎一样而尺寸却并没做到一样。

7.9 Luxi 系列

和苹果的 Lucida 系列字体很像，通常都用它代替苹果字体。尤其是很多网页的样式表都首选“Lucida Grande”，对此可用“Luxi Sans”代替，这样比用“Liberation Sans”之类的代替更能还原网页效果。

8 一些问题

8.1 部分程序方框或空白无字

具体原因与解决方法在前面的系统原理和默认字体中已经说过，但这并不意味着能彻底解决问题。典型的，在 xterm 中显示藏文及 Ext-B 字符相当困难。除非用一个单独的真正的同时包含藏文和 Ext-B 字符的超大字符集字体，而不是“HAN NOM A/B”那样的字体对。或是报 bug 然后等待应用程序支持字体组合。用超大字符集字体显然不是什么上策，但报 bug 指望其支持字体组合周期又明显太长。所以，如果在你的正常需求中经常碰到藏文和 Ext-B 字符，那也只好改用超大字符集字体或是干脆换用别的程序。

8.2 GTK 程序下划线太高

这是混搭字体造成的，几乎所有的中英文字体混搭都会造成该问题。比如“文泉驿正黑”和“Liberation Sans”， 12px 时或许并不明显，但再大点就会有明显的迹象。然而我偏偏不可能不用混搭，所以也只好默默忍受。

QT 在这方面做得很好，无论怎么混搭，下划线都会在一个非常恰当的位置。

8.3 GTK / QT 程序的字体选择紊乱

先说个例子，在我的配置中，当为 xchat 指定“HAN NOM B”时，关闭字体选择对话框后会看到自动变成了“HAN NOM A”，“花園明朝B”也是如此。或许你并不觉得惊讶，因为这例子用的是关联字体对嘛！但若对“文泉驿正黑”进行了强制替换，比如用“Liberation Sans”替换其西文部分。那么选择“文泉驿正黑”其结果也必然自动变成“Liberation Sans”。这回该有点诧异了吧！这也正是为什么我现在的配置中没有对“文泉驿正黑”做强制替换，而只是在以类定字中插入了“Liberation Sans”。

这是因为 GTK 的字体选择对话框对选中的字体项进行了一步“预处理”。这一步预处理会把你选择的字体名称给处理掉，除非你指定的是三大标准类名。也就是它会按系统字体配置文件展开所望列表，然后再取其首项可用字体的名称。这样便产生了很多问题，尤其是强制替

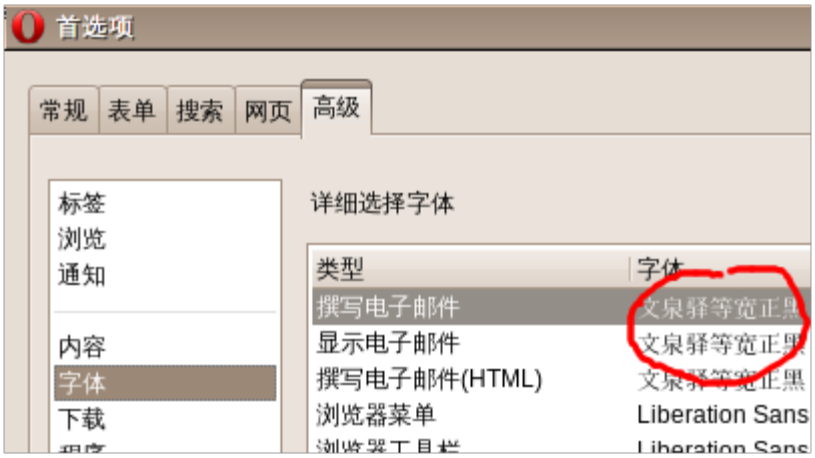
换和三类之外的虚字体。当参与强制替换或自己引入的虚字体的中英文字体呈多对一出现时，便会陷入极度混乱。

像 xchat 这样的程序还好，你可以直接输入字体名而不用字体选择对话框，从而绕开对话框的“多此一举”。然而多数程序都不能让你自己输入字体名。QT 程序更甚，“多此一举”发生在程序运行时，而非选择字体时。所以 QT 在这一点上可能会出现更混乱的情况。

所以我们要避免使用那些理想而复杂的配置，尤其是强制替换。对于设立虚字体另立类别（比如中文“宋体”）也应当适可而止，还要避免出现中英多对一的情况。

8.4 Opera / LiberOffice 字体混乱

Opera 自脱离 QT 之后，不再听从 fonts.conf 的配置，自行一套。且自身显示非常混乱，请看右图。网页内容可用 user.css 强制指定只用某一种字体，但这样将无法看到网页效果，况且这对于程序本身的界面这也并无作用。报 bug 然后等待吧。当然，你可以找回脱离 QT 之前的那个版本来用，不幸的是你可能会碰到更多其它方面的问题。如果你想要好的字体效果建议换用 Firefox。



LibreOffice 情况比 Opera 要好很多，只是界面字体可能发生混乱，而文档不会。它还是很听 fonts.conf 的话的，只要不乱用强制替换。

8.5 Java 程序字体难看

Java 字体配置并不在本文所讨论范围之内，因为它完完全全自行一套自成体系。要为 Java 程序换个界面主题尚属不易何况配置个漂亮的字体。将就着用吧。