TEX 学习总结

Lanphon

2010 年

目录

插图目录

评论

Lanphon 童鞋从来不是一个扯淡的好手，过去不曾是，现在不是，将来也不会是。但他执着的努力却值得每一个人的敬仰。

— 匿名

如果你们的评价是，“好久没有人扯淡也能扯的这么清新脱俗了”，我一定会感激不尽的。

— 作者

出来混，迟早是要还的。

— 无间道

生是北邮人，死是北邮死人。

— BYRer

世界无限广阔，男儿切莫固步自封！

— 紫川秀

前言

首先声明，本篇并不是 TEX 的入门性质的文档，如果要入门的话请参考 lshort 和 lnotes。强烈推荐后者，lnotes 的第二版正在撰写中，相比于年达久远的 lshort,lnotes 毕竟有太多的优势了。我也是看着 lnotes 长大的[1]，的确挺好的一片入门的文档。

本篇更多的应该是我自己学习 TEX 的一些总结吧，包括了一些很有用的，但是可能也很偏僻的小技巧的总结。同时也有一个劝学篇，更多的还是回忆自己的历史吧。

本篇文档是在没有互联网的基础上完成的，其由来很简单：暑假在家闲的蛋疼，只好学习 TEX 来自娱自乐了。有一些小小的收获不敢独享，赶紧写篇文档来，独乐乐不如众乐乐嘛。幸好有 TEX 做伴，这个暑假也就不再无聊了。

有互联网的同学可以到清华 TEX 版或者 CTEX 版交流经验，byr 木有 TEX 版，所有的 TEX er 基本上散布在 Linux 和科研与论文版面，想要交流问题的话也可以去这两个版面。唉，有互联网的孩子，真幸福啊。

由于 listings 宏包对中文的支持不大好，所以在列源码的时候，中文可能会变得飘逸起来。这个问题目前我也没有解决，只好先委屈各位，运用自己的判断力来决定这些字符究竟在什么位置了。如果有机会一定将文档的代码奉上，到时候大家就不用看蹩脚的 listings 了。

本篇文档所有示例均于 Windows 7(32-bit) 环境下，使用 TexLive 2009 的完全安装版，木有安装其他非官方宏包的情况下完成的，并且所有文档的测试类型均为 XELATEX，如果使用不同的发行版本或者不同的编译选项可能会有一些出入，请自行判断。

btw: 我的邮箱是 lanphon@bupt.edu.cn，若有任何问题，请不吝赐教。如果我能上网的话一定会认真答复的。

[1] 文字游戏而已，不用太在意

Chapter 1

TEX 劝学篇

TEX 是什么？[1]TEX 是斯坦福大学的 Donald E.Knuth 先生所发明的一个排版系统，可用来排版专业的书籍，论文等。Knuth 是一个妖人，每一个学习计算机或者相关的没有不知道他的长篇巨作，The Art of Computer Programming。而 TEX 的诞生，据说也与此有关。据说 Knuth 写书的时候发现当时的计算机排版很不令人满意，就放下手头的工作，转而设计一个高质量的排版系统，这就是 TEX 。十年岁月，终于给世界贡献了一个重要的工具。然后丫接着回去完成他的巨作了。

1.1 忆苦思甜

我刚接触 TEX 是 09 年的春天，当时也只是听说过，自己也从来没有试过。后来 09 年的秋天选修了数据库课程，课堂讲的是 SQL Serve，但曾经有过 MySQL 的基础知识，就和老师商量了一下，实验和报告都以 MySQL 为基础。当时正痴迷于 Linux，写文档自然不能用 M$ 的 office 了，好歹之前听说过 LATEX ，也就准备真真正正体验一回了。

刚开始的时候我并不熟悉任何相关的东西，甚至于什么是 TEX, 什么是 LATEX 都不熟悉。在网上好容易搜罗了一篇文章，apt-get install texlive-full，OK，万事大吉。然后四处搜寻模板，找能编译通过的改，就这么凑合着过了一个学期。而此段时间唯一学会的，也就是知道 TEX 编译出来的 pdf 很漂亮，其他都不知道了。

去年的时候并不了解有什么 TEX ，学的都是 latex *.tex 生成 *.dvi 文件，然后 dvi2pdf 再转成 pdf。至于字体，也只敢用 CJK 自带的那几个。也曾想过安装一些，但被网上找到的那些复杂的步骤都吓到了，也就碌碌无为了。

学术不精，浅尝辄止就会造成这样的后果啊，那个时候 TEX 给我的印象也就是结果很漂亮，过程很复杂，难以驾驭。后来痛下决心，洗心革面，认真学习，才知道当年的自己实在是幼稚的很。TEX 不只有 LATEX ，还有 pdfLATEX 。而类似的引擎也不只有 TEX ，还有 XETEX，以及与之对应的

[1]颇有点儿笑口里的那个“人生是什么”的意思，没听过这个笑话的人请去笑口挖坟，不过这个笑话其实并不是多么好笑，只是结合当前的环境，让人忍俊不禁

XeLaTeX 。发行版也不只有 TexLive，还有 MikTex,CTex 等。想到当年自己还困惑于为什么要装 LaTeX 却要 install texlive-full，就不觉羞愧不已。

1.2 选择你喜欢的

同学也曾询问过我这些东西之间的区别。鉴于他也有编程的经验，我就这么解释。TEX 就像是排版界的汇编语言，虽然功能强大，可毕竟也不是普通人能够熟悉的。Knuth 当年发布了 TEX 时候，真正用到排版中的也是扩展了宏的 TEX，也就是 plain TEX 。而 plain TEX 由于跟不上时代的潮流，现在已经很少有人用了。LaTeX 则属于 TEX 的增强版，类似于 C 语言。

XeTeX 是和 TEX 一个等级的引擎，就像汇编语言有 X86 和 Motor68K 一样的道理，其出现是为了解决当初 TEX 刚诞生的时候仅仅支持 7 位（后改为 8 位）编码的问题的。XeTeX 由于诞生的比较晚，因而可以充分总结以前设计的优点和缺点，扬长避短。一个显著的改进就是 XeTeX 支持 utf-8，大家都知道这意味着什么。另外再举一个例子，TEX 插入的图片只有 ps 格式，后来改进的 pdfLaTeX 也只是零敲碎打。但 XeTeX 支持直接插入 jpg,png 等现在流行格式的图片。此外更振奋人心的一点就是，XeTeX 可以直接调用系统中的字体，包括 ttf,otf 格式，而不用之前复杂的安装了。XeLaTeX 是基于 XeTeX 引擎的扩展。可以说 XeTeX 和 XeLaTeX 天生就具有无以伦比的优势，所以建议大家在选择的时候都用更加先进的 XeLaTeX，而不要再选择陈旧的东西了。当然，如果有考古癖或者折腾倾向的人可以尽情的折腾去。

OK, 其实说了半天，总而言之就是一句话，选择 XeLaTeX 是符合时代潮流的，是符合最广大人民利益的，是符合先进生产力和先进文化的前进方向的。我对于 TEX 各个之间的关系自然不如专业人士来的精辟，如果像认真了解这方面内容的童鞋建议参看 lnotes 第二版。

至于发行版本，就和 visual studio 差不多，既能编译 C++，也能搞定 C++++[1], Windows 下可以选择 CTeX，或者像我一样装个 TexLive 也行。Linux 环境下装 TexLive 就足够了，openware.byr.edu.cn 里有 TexLive 的镜像。

1.3 TEX 能做什么，不能做什么

和所有其他的软件一样，TEX 也有其使用的范围。

TEX 对数学公式的处理很方便，排版出来也十分漂亮。所以，如果你的文档中有很多的数学公式的话，选择 TEX 是没有错的。用 word 也可以，如果你有足够的耐心鼠标一个一个的选择那些希腊字母或者数学符号的话。

TEX 中插入图片不是很容易，虽然现在的 XeLaTeX 已经大大提高了图片的宽容度，但是总归还是要用命令的。所以，如果你的文档中可能有大量的图片展示，那么最好还是别用 TEX 来写，word 可能更加擅长。

如果文档结构性不是很强，比如报纸，那么 TEX 绝对是不合适的。因为 TEX 适合于那些有层次性，结构性的文档，鲜明的例子就是各种科学论文

[1] 当然你知道我说的是哪个语言 :)

了。

或许你只想随时记录一下自己的感悟，写下自己的心情，TEX 和 word 都足以满足你的要求。但我推荐选择 TEX，有一些控制结构实现起来也很简单，比如说目录，比如说超级链接，比如说图片编号，比如说章节设置。

当然，常常有一些极端分子争论到底 TEX 好还是 word 好，这本身就是一个很扯淡的问题，两者有着不同的适用范围，拿来比较也没什么意义，往往是你说服不了我，我也奈何不了你，到最后还是便宜了围观的群众们。在这方面国人要比国外的好多了，或许因为国人没有什么信仰吧。总之，黑猫白猫，能逮住老鼠就是好猫。何必为了一堆二进制组成东西的好坏而弄个肝火旺盛呢？

Chapter 2

TEX 应用篇

2.1 记录文档

这是我刚开始学习 TEX 的初衷，因为本身自己就是一个意识流派，平常偶尔就有那么一两个想法需要记录下来。用 word 吧，软件大不说[1]，用起来也那么烦。有一次要写一篇论文，要求一点五倍的行距。同组的人写完后给我看，我惊奇的发现行距不对。调整吧，全选，然后右键就没了段落这个选项，据说因为包含图片了。搞得老子辛辛苦苦一段一段的改了过来。自此对 WORD 彻底失去了信心。这东西，每个人都能用，但并不是每个人都能用好的。用 TXT 记录？老大，您开玩笑呢吧？TXT 看个电子书还可以，记录文档就免了，没任何结构可言，而且如果一时兴起，想要插入一幅图片，就变成 Mission Impossible 了。琢磨半天最后还是学 TEX 吧，刚开始可能的确会痛苦那么一阵子，但过后一切就方便多了。而且 TEX 对数学的支持也是挺不错的，熟练的话纯粹的键盘操作就可以完成公式的输入。要想在 WORD 下输入一个数学公式能把人难死，至于 TXT 就彻底不用想了，一个积分号就能累趴下。

选择 TEX 做记录文档，尤其是对于理工科学生来说，是必须的。我并没有见多少人用电脑来记录文档，毕竟在笔记本上画一个积分号要比在笔记本上画一个积分号要简单的多[2]。但纸质的不容易整理，也不容易交流，所以我想还是用电子版的比较方便吧。理工科中有大量的公式与图表，公式的排版显然是 TEX 的强项了。图可以直接插入，毕竟现在的 XeLaTeX 已经很方便了。如果有一些结构的图，比如流程图，你也可以选择在线绘制的方式来插入。表的话 TEX 中也是支持的，不过这一切都需要你认真的学习，仔细体会。

在这里我提一个建议，其实是我自己的做法，应该也有很多人也有同样的做法吧。TEX 中绝大多数的功能都是由宏包来实现的，这就和 C 语言绝大多数功能都是由库函数实现一个道理。所以在 TEX 中宏包的使用是一个

[1] TEX 的发行版虽然很大，如 TeXLive2009 足有两个 G 多接近三个 G，但精简版的都是很小的

[2] 猜猜哪个笔记本是哪个？

重要的方面。一个人说他要学习 TEX，更多意义上还是学习宏包的用法[1]。常常见到有人用了好多的宏包，然后导言区变得 N 长。其实可以将所有导言区的内容写入一个.sty 文件，然后在文档的导言区 usepackage 一下的。这样也方便了整理和移植，所有宏包的引用和设置都放置在 sty 文件中，相当于建立了自己的模板。这个用法是我从 caspar 的 ya-buptthesisbachlor 中偷师来的，至于 caspar 童鞋师承何处，那就不得而知，待好事者考究之了。

记录文档中有一些比较常见的需要分开的段落，一般需要空出一行。遗憾的是 tex 认为源文件中的空行表示段落的结束，一个如此，一万个空行亦如此。你可以用 \vspace{3ex}的做法达到空行的目的。vspace 在垂直方向上空出距离，有大括号内制定。

另外一个就是类似于论坛上常用的那种分割线了。如下这种的效果：

* * * * 我是华丽丽的分割线 * * * *

* * * * 看什么看，没见过分割线? * * * *

源码很简单，如下：

```
% 分割线命令
\newcommand{\fengexian}[1]{
{\centering \makebox[\textwidth][s]{
\textcolor{red}{* * * * \mbox{#1} * * * *}}}}
% 应该有空格，以便 s 选项正确断开相应的结构
```

你可以很清楚地知道那个 red 是什么意思，不是么？你当然可以改成你所喜欢的任何一种颜色，只要你愿意。

使用的时候如\fengexian{我是分割线}这样的用法，可以人为对文章的逻辑结构进行分割。我在写日记的时候经常用这个来分行，让我有一种在论坛灌水的感觉，棒极了。

2.2 幻灯片

制作幻灯片是一项必不可少的要求，office 套件里包含 PowerPoint 也就是这个含义。结合强大的数学功能，TEX 在幻灯片制作上也是独树一帜。你并不用过多的关心形式，只需要选择一个模板，TEX 就可以自动生成包括链接，目录，公式在内的幻灯片，不过是 PDF 格式的。显然，还能有什么格式可以选择呢？

2.2.1 幻灯片导言区的设置

XELATEX 中有个类 beamer，可以用来制作幻灯片。使用方法就和

```
\documentclass{beamer}
```

一样，只不过从本质上来讲，beamer 和 article，report 或者 book 是不同的，它不是一个标准的样式，而只是一个环境。当然，我们不用关心这个，能用就行。

[1] 恕我等浅薄，如果有想认真研究 TEX 的童鞋请笑过此说法

设定完 documentclass 以后，你可以使用预定好的模板来做幻灯片。beamer 中有很多优秀的模板可供使用，它们都订好了样式，链接和其他的一些元素，你所需要的只是往里边塞东西。

使用模板之后，导言区的东西就可以按照普通的 XeLaTeX 文件设置了。注意 beamer 使用的默认字体是无衬线字体，也就是 sansfont。如果你没有设置这个字体的话，英文还好说，中文则不会显示任何东西。

关于 beamer 导言区还有两个小玩意儿。一个是 logo 设定。如下：

```
\logo{\includegraphics[height=.25\textwidth]{pic/bupt_logo.png}
```

你可以试一下效果，确实很漂亮。另一个则是打开即全屏，可以用

```
\hypersetup{pdfpagemode=FullScreen}
```

这样生成的 pdf 文档打开的时候就是全屏播放，而不用手工设置全屏了。可能第一次打开的时候会提示你是否允许自动全屏，选择允许，之后就再也不会出现这样的询问了。

导言区的东西基本上就这么多。我的导言区设置如下，你也可以参考:

```
\documentclass{beamer}
\usetheme{Warsaw}

\logo{\includegraphics[height=0.25\textwidth]{pic/bupt_logo.png}}
\usepackage{pgf}
\usepackage{fontspec}
\usepackage{xeCJK}%中文处理相关，包括自动换行等
\setsansfont黑体{}

\usepackage{amsthm,amsmath,fancyhdr,color,graphicx}
%设置用打开就会全屏显示acrobat
\hypersetup{pdfpagemode=FullScreen}
```

2.2.2 正文部分

在 beamer 的正文部分，你可以任意使用任何 TEX 允许的语法，比如 author，title，date，section,chapter 结构等，但这些都不会出现在最终生成的幻灯片里。beamer 定义了一个环境，或者成为命令，来生成幻灯片。如下:

```
\frame{}
```

或者是如下:

```
\begin{frame}
\end{frame}
```

每一个 frame，中文称为帧，都将生成一组幻灯片。为什么不是一个呢？想象一下幻灯片常用的处理方法：N 个项目一个接一个跳出来。我们的幻灯片没有动画效果，但是可以用多个幻灯片连续播放的方式达到这个要求。比如 4 个项目，那么这个 frame 就会生成 4 个幻灯片，每个比前一个多一个 item。一个 frame 生成的幻灯片在页数编码上只占一个，也就是这一组幻灯片共享一个页码。

frame 中可以有 frametitle, 如`\frametitle{幻灯片题目}`。至于题目出现在什么位置，这就是各个模板自己的事情了。

刚才我们说道动画效果，在 beamer 中也可以使用 item 环境，如下:

```
\begin{itemize}
\item<1-> a
\item<2-> b
\item<3-> c
\item<4-> d
\end{itemize}
```

这样就生成了一组四个幻灯片。这是我目前用到的动画效果，beameruseguide 好像还介绍了另外的效果如何实现，有兴趣的可以去试一试。我认为重要在与内容而不是形式，动画并不是越花哨越好，淡雅朴素也是一种风格。

beamer 就介绍这么多，又想要深入了解如何制作的同学可以参考 beamer 给出的文档，或者放狗搜。前边提到的几个交流的地方也有不少的资料可供使用。总之，还是要多用，用过一两次自然也就熟悉了。

2.3 流程图绘制

流程图选择使用 Pgf 宏包和 Tikz 前端来做，入门和介绍参加 lnotes，这里就不再多说了。

对于 pgf 绘制流程图，有三个主要要解决的问题。其一是节点的绘制，其二是节点相对位置的防治，其三则为连线和标注问题。这三个问题解决了，一个完美的流程图也就搞定了。本文绘制的流程图按照 Dia 的格式，圆角矩形为开始和结束框，矩形为中间过程框，而菱形为判断框。

2.3.1 节点的绘制

节点的绘制从来都不是一个问题，事实上 pgf 提供了圆形和矩形，并且加载合适的库的话你还可以绘制菱形。我们所需要考虑的，或许仅仅是一个美观的问题，这就上升到了艺术的层次，而不是我们这么一片就能够搞定的。我只是做了示范，诸如配色和阴影，或者如果更炫的话来点儿渐变色，那就是你自己的事情了。

以下为设置 tikz 样式的源码，本着方便以及复用的原则，我们选择了 tikzset，而不是每一个环境都声明一次。你可以把它加入到导言区，然后就可以用另外给出的代码来引用这些已经定义好了样式的节点了。

```
\tikzset{
passprocess/.style={
rectangle,
draw=blue,
minimum width=50pt,
minimum height=20pt,
font=\ttfamily,
text centered
},
startstop/.style={
rectangle,%
rounded corners=5pt,%
minimum width=50pt,
minimum height=20pt,
text centered,
fill=orange,
font=\ttfamily,
```

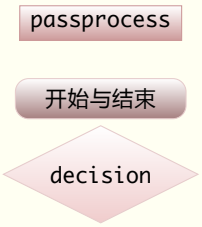

图 2.1: 节点绘制示例

```
draw=red
},
decision/.style={
diamond,%
shape aspect=2,%aspect value is the ratio of width and height for diamond
draw=green,%the color of line
fill=lime,%filled color
font=\ttfamily,%set font
text centered%surely you know what it means
},%here is a ",",if you forget it,tex will crash
line/.style = {
draw,
->,
%shorten>=2pt,
thick
}
}
```

2.3.2 节点位置的放置

接下来就是各个节点位置的放置了。说到这里我不禁想腹诽一下 pgf，介绍说 pgf 画流程图有独到之处，可是等接下来放置节点的时候就知道这种说法有多么的错误了：你必须手动调节各个节点的位置，有时候可能会出现新增加一个节点却不得不修改整个设计的问题。所以，首先你应该确定你要用 pgf 画流程图么？其实你可以有其他的选择，比如说 dia，也有 windows 版的哦。好吧，如果你坚持，那么请在施行之前，先规划好你需要画流程图各个节点的相对位置。在这方面 dia 这样的工具有独到之处：你可以先把各个节点弄好，然后用鼠标摆来摆去。瞧，你还坚持么？YES，I promise。OK，在这里我们用 matrices 来放置节点，好学，但不好用，和 M$ 的大多数软件一个特点。

matrix 的用法也很简单，row sep 和 column sep 分别设置行和列的间距，建议行的值稍微设置的大一些，否则节点们挤到一块儿不好看。然后 & 符号作为分隔符，分割各个节点。如果没有节点，则两个 & 之间没有内容。换行是由\\ 完成的，最后一行依然需要换行。源码如下所示

```
\begin{tikzpicture}
\matrix[row sep=5mm,column sep=5mm]{
%First row:
\node (p1) [startstop] 开始{};&\\
%Second row;
\node (p2) [passprocess] 问问题咯{};&\\
```

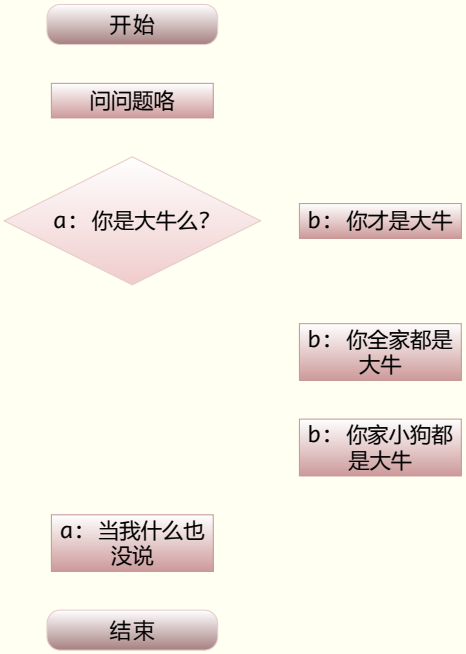

图 2.2: 节点位置放置示例

```
%Third row:
\node (p3) [decision] {a你是大牛么:?};&
\node (p32) [passprocess] {b你才是大牛:};\\
%Fourth row:
&
\node (p4) [passprocess] {b你全家都是大牛:};\\
%Fifth row:
&
\node (p4) [passprocess] {b你家小狗都是大牛:};\\
%Sixth row:
\node (p5) [passprocess] {a当我什么也没说:};&\\
%Seventh row:
\node (p6) [startstop] 结束{};&\\
};
\end{tikzpicture}
```

另外一个是关于文本的换行问题，直接使用\\ 是不行的，编译无法通过。后来让我找到了解决的方法：设置节点的 text width 属性。节点的 text width 属性可以控制文本的宽度，达到此宽度则自动换行。不过这样设置以后貌似节点的宽度也就固定死了，所以流程图中语言还是越少越好，能不换行尽量不要换行。

OK，除了用 matrices 放置节点外，pgfmanual 的第四个 tutorial 中还提到了两种，一个使用节点的相对位置，另外一个则是用 chain。如果有兴趣的人可以研究一下如何来做。对于我来说，我还是倾向于用 dia 画流程图的：简单，好用，免费，小巧，直接导出图片。还能有什么要求呢？？

2.3.3 连线和标注问题

pgf 中有 scope 来绘制连线，这样一来也就方便多了。连线的时候需要引用节点的名称，”- -” 表示绘制直线，”-” 便是绘制先直线，后折线，”-” 表示先折线，后直线。老实说，我也不知道”-” 和”-” 的区别，幸好选择不多，如果一个不对，换另外一个就是了。流程图中常用的另一种双折线的画法在接下来的那一小节中介绍。

而标注的问题就简单多了，其实还是加了一个 node, 不过是文字形式的。node 可以选择 near start, 或者 near stop，或者 midway，这是在选择位置。

scope 的源码如下，只需要放到 tikzpicture 环境里，matrix 之后就行了。

```
\begin{scope}[every path/.style=line]
\path (init) -- (p2);
\path (p2) -- (p31);
\path (p31) -- node [near start,above] {yes} (p32);
\path (p32) -- (p4);
\path (p4) -- (p5);
\path (p5) |- (p6)表示折线先折线后直线，而先直线后折线;%|,|--|
\path (p31) -- node [midway,left] {no}  (p6);
\path (p6) -- (end);
\end{scope}
```

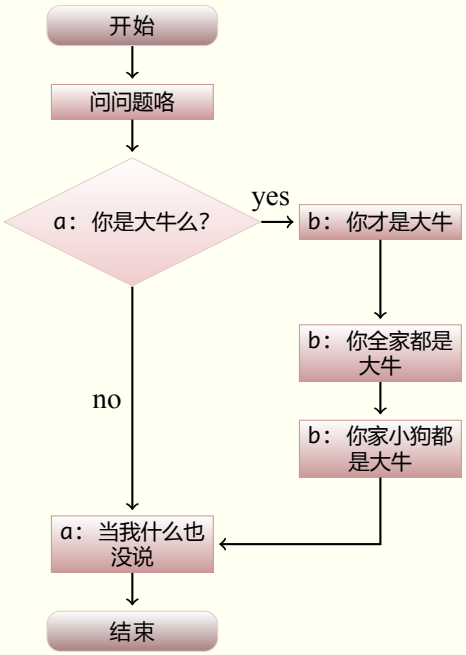

图 2.3: 连线以及标注问题

2.3.4 双折线的实现

双折线的效果如下所示，由于 tikz 中没有提供现成的绘制双折线的方法，所以可能需要一些小小的技巧。不用担心，不是很难。

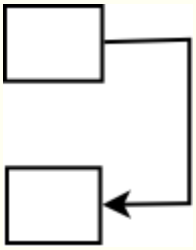

图 2.4: 双折线示意图

让我们还以上一个流程图为蓝本，不过这一次换一下结构。

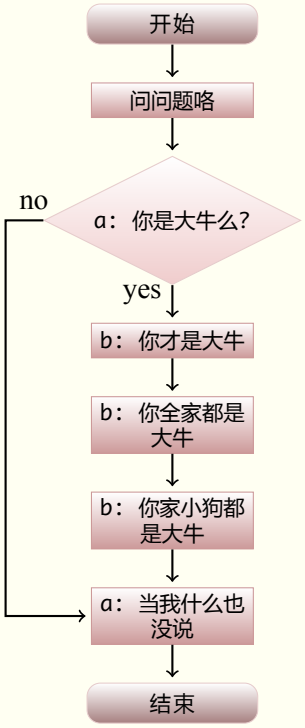

图 2.5: 流程图中双折线的实现

哦，效果依然实现了。在此我把 scope 部分的源码奉上，关于节点的命名，第一个是 init, 第二个是 p2，依次排下，最后一个叫做 end，以方便理解。

```
\begin{scope}[every path/.style=line]
\path (init) -- (p2);
```

```
\path (p2) -- (p3);
\path (p3) -- node [near start,left] {yes} (p4);
\path (p3) -- node [near start,above] {no} ($(p3.west) + (-0.5,0)$) |-
    (p7);
\path (p4) -- (p5);
\path (p5) -- (p6)表示折线先折线后直线，而先直线后折线;%|,|--|
\path (p6) --  (p7);
\path (p7) -- (end);
\end{scope}
```

一切的关键就在于 p3 与 p7 之间的连线。node [near start] {no}当然是标注，可以不用理会。然后这一句就变成了 (p3) – ($(p3.west) + (-0.5,0)$) |- (p7) 可以看出双折线实质上由两部分组成。第一部分是由 (p3) 到 ($(p3.west) + (-0.5,0)$) 的直线，第二部分则是由后者到 p7 之间的折线。那么现在问题来了，那么一长串的表达式到底表示哪个点呢？哈哈，你差不多猜到了吧。(p3.west) 引用 p3 节点左边线中点的坐标 (pgfmanual[1]的 312 页有详细的坐标图), 然后将此坐标与 (-0.5,0) 相加，即左移 0.5cm。接下来的事情就简单多了不是么？至于那一对 dollar 符号，则是为了引入实时计算的功能。如果你的文件编译出错在这个地方，试一试使用 xkeyval 和 calc 宏包。

对于矩形，也就是 rectangle，有 east，west，south，north 等 8 个可以引用的坐标点。了解了这些，绘制各种曲线也就简单易行了。详细信息以及其他图形可获得的参考点请自行查阅 pgfmanual.pdf。

2.4 Minimal 文档类使用

Minimal 是一个与 beamer 类似的 document class, 不是标准而是一个环境。它的名字揭示了这个文档类的作用：短小，精悍。你可以使用如下的方式来声明接下来创建的是一个 minimal 类的文档:

```
\documentclass[a4paper]{minimal}
```

Minimal 文档类适用于测试性场所或者草稿，基本上就是那种不需要很严谨的结构的场所。为了达到精简的目的，这个文档类抛弃了标准文档类中的一些结构，比如说 title，比如说目录。所以之前使用的那些习惯在这里有可能就会编译出错。因而对于 minimal 的使用也需要认真对待。

以下所介绍的均为自己的经验，使用 XeLaTeX，并且只引用宏包而不对未定义部分进行修改的情况下的来的。如果你的测试结果与我的有出入，或者有更好的见解，欢迎和我联系。测试的时候我只选择了我常用的命令和宏包，如果你有其他命令和宏包的测试结果，也欢迎和我联系。我致力于搜集几乎所有能用得到的命令和宏包在 minimal 环境下的兼容性。

与其介绍那些可以使用的倒不如专心介绍那些不受欢迎的家伙，毕竟前者貌似的确有那么点儿或许比后者多一些吧。当然，如果评定为不兼容，那么事实上它就是不兼容；但如果没有评定不兼容，它倒还真不一定可用。道理浅显易懂，因为我不可能测试完所有的命令与宏包。所有有些时候从敌友不辨的环境中介绍一些熟悉的面孔还是很有必要的，可能是敌人，也可能是朋友，总归比那些不清不楚的好。

[1]对于 texlive2009，文档路径为 2009/texmf-dist/doc/generic/pgf/pgamanual.pdf

2.4.1 受到影响的命令与环境们

minimal 文档类中抛弃了基本类的一些命令与环境，因为对打草稿而言，似乎这些命令并不需要。想象一下你打草稿的时候从来不会做什么事情，是你在写标准文档的时候常常做的，然后你就知道为什么 minimal 要抛弃它们了。

一个显而易见的就是`\maketitle`命令了，因为打一个草稿几乎不需要那么正式的写上标题作者和日期（虽然我很怀疑）。`\title`，`\author`，`\date`这几个都是可用的，但他们只是设置值，而不会显示出来。一旦你决定 maketitle，minimal 马上就告诉你丫犯抽了，不知道这个 control sequence。

接下来的倒霉蛋是目录结构，包括 chapter,section,subsection,subsubsection 等。这也就意味着你在写`\subsection`来为你的草稿划分段落（多么愚蠢的事情啊）的时候 minimal 也会向你抱怨的。同样的，既然没有了目录结构，那么`\tableofcontents`也就不再需要了，你不应当为你的草稿还列出目录 :-)，当然，还有 abstract。

目录不是第一个倒霉蛋，也不是最后一个，接下来猜猜是谁？嗯，是 footnote，脚注兄。想象一下打草稿的时候你还煞有介事的标上一个 1，然后在最后写上你的那些八卦，唔，哥们，你出门忘吃药了吧。草稿其精髓就在一个草字[1]，何必那么规矩呢。

噢，here we find another guy！figure 环境[2]。figure 浮动环境在 minimal 中也取消了，同时意味着如`\listoffigure`命令也变得不那么受欢迎了，虽然正式文档中我也很少用到它。但看着老朋友的离去，总是让人这么伤感，不是么？

figure 浮动环境还有一个难兄难弟，table 浮动环境。minimal 中 begin{table} 是无法识别的，so 放弃浮动吧。但即使只使用 tabular，`\hline`却画不出线来了，虽然可以编译通过。可怜的娃，别用表格了，丫就注定被丢弃。

接下来被抛弃掉的这位仁兄就有点儿不清不楚了，大小控制命令，包括 tiny，scriptsize，footnotesize，small，large，LARGE，Huge 系列命令。其实大部分情况下在草稿中玩一字大小的变换还是蛮有意义的。或许因为 minimal 认为验证性的工作专注于内容就可以了，不必有那么多的花哨，于是乎，这个家族，灭亡鸟。。。

其他阵亡的据说还有参考文献和索引，我了个去，打个草稿都这么正式，先生贵姓啊？？不过这些我没用过，只能据说，据谁说呢？也是推测而已，因为 minimal 已经无法识别 `\refname`这个小东西了。

既然说到 refname，那就不能不提他的那些兄弟们了。`\figurename`无法使用，理所应当，figure 都没了还要个毛 name。`\abstractname`，`\partname`，`\appendixname`，`\listfigurename`，`\listtablename`，`\contentsname`统统挂掉。

虽然是草稿，可 minimal 支持页眉页脚，这点让我啧啧称奇。这就意味着 `\pagestyle{}`这条命令是可用的。

[1] 请自行理解，谢绝 YY

[2] 我觉得似乎还是需要的

2.4.2 池鱼：宏包的传说

看完了命令，我们接着看宏包们的状况。宏包的使用也和 minimal 抛弃的环境相关，所以了解上一小节所讲的内容后，这一节的内容可能会更加简单。

既然没有了目录结构，定制目录的 titletoc 和 titlesec 也就显然不被支持了。同样不受欢迎的还有因为 footnote 而受到牵连的 footnote，footmisc 和 footnpag 等定制 footnote 的宏包。

因为没有了 figure 浮动环境，subfig 和 picinpar 也就没用处了。事实上包含这两个宏包的 minimal 可以编译通过，但是使用就是另外一回事情了。picinpar 使用的时候引用 figure 环境，会导致编译出错。而 subfig 宏包我也不知道除了 figure 环境以外还能有什么用法，也没法测试。graphicx 宏包可以使用，其实你能使用的就是一个 includegraphics 了，它不能和 figure 携手飘来飘去，实在让人遗憾。同理，关于表格的那些宏包也都无法在 minimal 环境中使用了。

经过测试可用的宏包有如下。xeCJK 和 fontspec，中文和字体必备的宏包，对于 minimal 如此深明大义我还是颇感欣慰的。hyperref 超链接宏包，可以定制链接和 pdf 元数据。xcolor 宏包，增强型的色彩支持。shapepar 宏包，一个能把页面排版成有趣的形状的宏包。ams 数学宏包，包括 amsmath,amssymb,amsthm,mathrsfs。listings 宏包，支持代码的排列。identfirst 宏包，提供首行缩进功能。lastpage 宏包，提供关于最后一页的信息，特别是页码数。fancyhdr 宏包，提供了页眉页脚的定制显示，在使用此宏包的时候注意不要引用包括章节名称或者其他已经被取消的结构的相关命令。tikz 宏包，强大的在线矢量绘图宏包。

这里基本列出的支持或者不支持的宏包都是我经常使用的几个，而我没使用过的自然也就不知晓其兼容性如何了。因而我是热烈欢迎你和我交流你的测试结果，一切都是为了共建河蟹社会，不是么？

Chapter 3

TEX 周边篇

TEX 的东东就暂时说到这里，让我们先休息一下，看点儿别的有趣的。既然本篇是 TEX 总结，这点儿别的东西自然也不可能离题万里了。嗯，我们谈一下关于 TEX 文件的编辑以及编译的一些小玩意儿。

3.1 为 TEX 写一个 Makefile

Makelove 是最伟大的一项运动，它可以让你 ******。等等，什么，你说我刚才说了 makelove ？？你一定听错了，我说的是 Makefile，瞧瞧你，小小年纪，整天都想着啥呢。好吧，我们接着说。Makefile 是很伟大的一项工具，他可以让你管理项目变得更加容易。

写好 Makefile 之后，编译文档只需要一句 make 就足够了，而清理中间文件可以用 make clean, 如果下边的源码所展示的。的确很方便哦。这个方法也是我从 caspar 那里偷来的，嘘，别告诉他哦。

示例的 makefile 如下，因为 win 下木有 make，所以这个也没有测试，有 Lin 的童鞋请帮忙测试一下，谢谢了:

```
.default: all

doc = texsummary
latex = xelatex

all:
  $(latex) $(doc).tex

clean:
  rm -f $(doc).{fot,out,aux,log,ptc,toc,bbl,blg,out,aux,lof}

deepclean: clean
  rm -f $(doc).pdf
```

3.2 Vim 编辑 TEX 文件

Vim 之强大，相信熟练的人不用我多说也知道，没用过的人任我吹个天花乱坠也不会相信，所以这里我就不再做一个传教士了。我自己用 Vim

编辑 TEX 文件，所以更多的还是总结 Vim 下的一些。如果你用 Emacs 或者 WinEdt 或者 TXT[1]，抱歉，请君自谋生路罢。

3.2.1 Vim 乱码问题

Vim 编辑文件最烦的莫过于乱码问题，尤其是 Lin 和 Win 下的文件。Windows 采用 GBK 编码，你在 Vim 中用命令 set fenc 看一下，fileencoding=cp936，据说这就是 GBK。而 Linux 采用 UTF-8 编码，原始的 Vim 编辑 utf-8 会显示乱码，幸好我找到了一个方法，可以用来自动识别编码的。

```
set encoding=utf-8
set termencoding=utf-8
set fileencoding=chinese
set fileencodings=ucs-bom,utf-8,chinese,cp936
set langmenu=zh_CN.utf-8
source $VIMRUNTIME/delmenu.vim
source $VIMRUNTIME/menu.vim
language messages zh_cn.utf-8
language messages zh_cn.utf-8
```

将这段话写入.vimrc 文件中 (对于 windows 版的 gvim 则是_vimrc 文件)，就可以完美解决乱码问题了。这段话设置了挺多的东西，据我所知包含了 term 编码，gui 编码等。有兴趣的童鞋可以对着 vim 手册认真研究一下，对于我们这些懒人，能用就行，嘿嘿。

3.2.2 vim 自动换行问题

对于 TEX 而言，换行并不依赖于源文件中的换行，而是自动进行的。源文件中只要两行数据之间没有空行，TEX 就认为是一个段落。换言之，源文件中的换行并不影响最终生成文件中的换行。所以我们在源文件中可以任意换行，每行一个汉字也没问题的！如果有雅兴，可以来个长短疏落有致，更是美不胜收哇。

同时，Vim 中如果一行过长，那么从行的一段移动到另一端也就变得更加浪费时间，而且源文件本身也不好看。幸而 Vim 的作者们或许也碰到过我们的烦恼，他们提出了自动换行这一个概念 —比方说，80 个字符就自动换行，不用手动敲击 Enter 键了。自动换行挺简单，set textwidth=80 就可以。等等，如果这么简单的话还写这么多干吗？？YES！你得到它了。试一下这个，对于英文字符可以正确换行，对于中文，仅仅的 set textwidth=80 可就没什么效果了。

经过我细致的搜寻，终于找到了能够同时胜任中英文自动换行的代码了。如下所示，btw，本段代码也同时把 Vim 初始窗口的大小给设定好了，当然，我们说的是 GVim 咯:

```
set lines=30
set columns=88
set textwidth=80
set fo+=Mm"设置这个之后对中文可以正确断行
```

[1]我不相信会有人这么傻

如果不想要自动换行，可以 set textwidth=0 就行了。自动换行对你新加入的字符有效，但并不会影响你之前的记录。如果有一段话你也想要达到自动换行的效果，确保本行前后都有空行，然后用 gqap 命令就可以了。如果你想对全文实行这种效果，gggqG 命令，看着挺长的，但分解一下就很简单了。gg 移动到第一行，gq 给出了断行命令，而所选的范围则从当前行，也就是 gg 移动到的第一行，到 G 所选定的最后一行。类似的命令还有 gg=G，你知道这个命令是干什么的么？

3.2.3 vim 的 LaTeX 插件

用 vim 编辑 tex 文件的人不在少数，毕竟 vim 之强大是有目共睹哇。但每次都需要编辑完，打开 texworks editor 编译一下，毕竟不是很方便，如果能在 vim 环境里直接编译查看结果，应该是很爽的一件事情吧。有这想法的人不在少数，有需求自然也有蛋疼乳酸的人去做啦。在这里我要介绍的就是一个 vim 关于 TEX 文件的一个插件：vim-latex suite。

vim-latex suite 是 vim 的一个插件，什么意思呢？That means 要想用这个，你得先装上一个 vim 才行。vim-latex suite 的安装过程就和普通的 vim 插件安装一样，在此不再多说。如果有不清楚的或者不明白的，放狗 or 看 readme。

vim-latex suite 为 tex 文件提供了一系列令人心动的功能和特性。我只使用了 vim-latex suite 很少的一段时间，之后重装了电脑，没了网上，原来下载的安装文件也不知道飞到哪里去了，于是乎也就放弃了。所以现在描述的 vim-latex suite 的特性仅凭当时的印象，如有不妥或不尽详实的敬请指正。

一个特性是折行。折行原本是 vim 的概念，如果有不懂的接着放狗。其实这个概念很简单，你看过折行的效果就知道怎么回事。vim-latex suite 把同一个结构的折起来，方便作者对整体结构的把握。这些结构包括 chapter,section,subsection,figure,table 等，总之就是几乎所有的结构都包括了。要想了解是什么效果？自己下一个试试不就行咯。

如果仅仅提供这行功能，可能 vim-latex suite 就不能称之为强大了。事实上有很多人，包括我，都不喜欢折行功能的。虽然有助于整体上的把握，可却丢失了细节。这个插件所提供的另外一个心动的功能可以 vim 中直接编译和查看结果。安装完 vim-latex suite 之后，编辑 tex 文件，保存，然后在 visual 模式下输入\ll，就可以将 tex 文件编译成 dvi 文件。然后\lv 就可以查看 dvi 文件了。如果你想直接生成 pdf 文件并查看，只需要按照说明修改某些小东西就行了。

此外，vim-latex suite 还提供了包括环境补全[1]，自动匹配，工程管理等特性。需要了解的童鞋请到官网上查询，或者安装插件，认真阅读帮助文档。的确很不错的一个插件，值得一试。

[1]听起来像不像人类补完计划？？

3.2.4 vim 的自带小工具

如果你和我一样，不喜欢 vim-latex suite 的缩进风格，或者其他理由（比如，suite 自动调用的是 latex, 而我习惯使用的是 xelatex，虽然改动起来不是难事，但毕竟不想花那么多的时间），不愿意选择 vim-latex suite，那么强大的 Vim 还提供了一些小小的工具，可以用来帮助我们实现愿望。

一个小工具的名称叫做 make，熟悉吧。对于 linux 下编辑文件的童鞋可以无视了，因为你们有强大 GNU make，对于我这个在 win 下打转的人来说，从来不知道如何安装 GNU make 的，一无必要，二，也是没有时间。幸而 vim 有一个 make，虽然小，足以满足我们的需要。btw:vim 好像还自带了一个 grep，有兴趣的可以去看看。

关于 vim 自带的 make，需要说的就是它执行 makeprg 所指定的命令。这个小小的 make 语法可不是 gnu make 的语法。我们的需求也不是很多，瞧瞧我们上面给出的 Makefile，其核心也不过是一句 xelatex *.tex。

将下一句话加入 vim 启动脚本文件 (lin:.vimrc;win:_vimrc)，然后再 visual 模式下键入:make，就 OK 了，等待最终编译完成吧。如果等待很长的时间依然没有完成，很可能是你的代码出错了[1]，关闭新出来的那个窗口，然后用:clist 查看一下最后的输出是什么，就知道错在什么地方了。

```
请
"help 查看工具的用法，自带的一个小小的也挺好用的呢makemakevimmake
set makeprg=xelatex\ %<.tex
```

如果要查看完整的输出列表，请使用:clist!。事实上 vim 里的 make 原始就是为 c 语言设计的，从很多命令以 c 开头就可以看出。简单的:clist 只列出 Vim 认为[2]与编译错误相关的条例，要查看所有的条例必须加上感叹号。vim 中已经内置了 C 语言错误的范例，所以可以找出错误。对于 tex 的错误样式则没有。这意味着你使用:clist 也可以得到完整的输出列表，因为 vim 根本不知道哪条是有用的错误信息，哪条则是不相干的。这个命令是我从手册里找到的。观察输出列表的时候，和 vim 操作一样，hjkl 可以用来移动光标，gg 首行而 G 最后一行。不过需要注意的是 q 键才是退出 list，而不是:q，后者退出编辑文件。

这貌似有一个小小的 bug，当你编辑一个以上的 tex 文件，然后键入 make 的时候，可能会有一些小小的问题。比如我把我导言区的内容放入一个 sty 文件，然后用 usepackage 来引用这个文件。如果当我编辑一个以上的 vim(指用同一个 vim，而不是多个，你的明白？？)，编译就会不出结果。clist 查看说找不到 sty 文件。如果有时候你也出了这些莫名其妙的错误，单独为要编译的 TEX 文件开一个 vim 试试。

就像注释里说的一样[3]，想要了解更多？？Plz,:help make 可以给你足够多的内容。

遵循所有 tex 文件编译的规则，有时候你可能需要编译两遍甚至多遍才能得到正确的文档，即使对于 make 工具而言也不例外。*That means, Plz type*

[1] 另一种可能是你的文档太大，需要更多的编译时间 :-)
[2] 从这里我们可以看出感叹号只是警告 Vim 别自作聪明
[3] 注释很飘逸，是吧？可恨的 listings 居然不支持中文

"make" for two times or more, then you get the result you want. When you type "make", plz shutdown the target PDF file.Surely I know some of you won't until you find xelatex.exe gives an error,just like "** ERROR ** Unable to open "texsummary.pdf"".

3.3 给你的文档打上自己的印记

其实，严格来说这个并不算是 TEX 周边了，因为我们要介绍一些宏包。但从效果上来看，这里所讲的和版面无关，而纯粹是一些小小的玩意儿，类似于旁门左道性质的，差不多用来自娱自乐的。

3.3.1 PDF 文件属性的修改

使用 Adobe Reader，或者 Adobe Acrobat，打开一个 pdf，在文件选项中选择属性，如果你想要更快捷的方式，CTRL+D 可以达到同样的效果。文档属性中关于标题，作者，主题以及关键字的选项是在 pdf 生成之初就加进去的，之后不可更改。利用这个我们可以给所有我们自己生成的 pdf 搭上自己的烙印 —无损美观，依然实用，不是么？想象一下，同学拿着你的报告交给老师，老师一看[1]，擦，这作者怎么不是他的时候，是多么棒，多么美妙的一件事情啊。当然，如果谁连抄都懒得抄，直接拿 pdf 交上去的话，他一定蠢得无可救药了。而你所做的，也是在挽救他堕落的灵魂，多么高尚的一件事情啊。

OK,no target,no magic。明白了要做什么之后，接下来就是怎么做了。修改 PDF 文件属性，我们需要用到 hyperref 宏包，如果你有一个 TEX 发行版装在你的电脑上，你可以到 texmf-dist/doc/latex/hyperref 文件夹下找到一个 manual.pdf，在这个文件的第八页有关于设置的详细信息，如果你没有 doc，或者你也如同我一样懒的话，我在这里给出了设置的例子。

```
\hypersetup{
pdftitle = {\modeltitlecn},%标题pdf
pdfsubject = {\modeltitlecn \modelversion},%主题
pdfkeywords = {\modelkeyword},%关键字
pdfauthor = {\modelauthoren(\modelmailen)},%作者
pdfcreator = {xelatex},%不能使用的latexlogo
pdfproducer = {Texlive},
}
```

显然，所有\model 开头的都是预先定义好的，你也可以这样做，或者直接写入你想写入的内容。pdftitle,pdfsubject,pdfkeywords,pdfauthor 就是我们之前看到的那几个栏目，而关于其他的含义，你可以参照 manual 文件。这里并没有列出全部，还有更多有趣的东西在帮助文档里。

3.3.2 文字水印：不厚道的行为

首先声明，打水印在绝大多数情况下都是一种很不厚道的行为。前一节介绍关于使用 PDF 属性做印记应该是 PDF 文件本身设计的内容。比如关键

[1] 如果有这么认真细致的老师的话

字，我想这部分可以让我们在互联网上更快地找到自己需要的文字。但水印就是另外一回事情了：污染视野，有损 RP，没有共享精神。当然我说的是绝大多数情况。存在即合理，水印的存在还是有一定的好处的。我不否认这一点，But I hold on my opinion: Water-Marking will cut down your RP value.

发完牢骚说正事。这里介绍的是文字水印，你可以把像“这是一个草稿”，“我爱你，XXX”，“Hello Kitty,Hello Moto” 之类的水印打在你的文档中，让所有人都能看到，一遍又一遍。为达到这种效果，我们所使用的宏包是 draftwatermark，瞧这名字，威风吧。

OK，在导言区\usepackage{draftwatermark} 之后，编译一遍，就可以看到每一页都有一个灰色的，大大的”DRAFT” 在文档的背景中。你也可以定制一些参数，如内容啊，角度啊，大小啊什么的。

- `\SetWatermarkAngle{}`设定角度，默认 45 度
- `\SetWatermarkLightness{}`设定灰度，默认 0.8
- `\SetWatermarkFontSize{}`设定大小，默认 5cm, 最大 5cm
- `\SetWatermarkScale{}`设定缩放，默认 1.2
- `\SetWatermarkText{}`设定内容，默认，厄，DRAFT[1]

在绝大多数情况下，我建议都不要用水印。将心比心，谁愿意自己拿到的文档充满这些无趣的东西呢。东西用在正确的地方才能发挥其作用，比如打个“草稿”告诉别人我还没完工呢，打个“I love you,XXX”来展现你澎湃的热情，等等等等，不一而足。我期待着能出现特别有创造性的用法，这样一来我也可以偷偷地学几招，嘿嘿。

[1]我觉得这个宏包的作者设为”I LOVE YOU,HONEY” 可能会更讨人喜欢

写在最后

经过五天不懈的努力，终于将这份总结性质的文档弄出草稿来了。对于我来说，这篇文档不仅仅是总结，更多的还是新的尝试吧。流程图的那一部分内容是设定好要写的，但之前我也没有搞定。趁着这次写文档的功夫，把一些旧的内容又重新梳理一遍，而一些新的内容，则是边写边做的。最终文档草稿完成了，自己的 XeLaTeX 知识也长进了不少，可谓一举多得。

本篇文档背景色采用了 RGB(255,255,242)，如果使用 xcolor 宏包的话则为 HTML 模式，值为 FFFFF2。米黄色的背景也比纯白色看起来更舒服一些吧。字体正文部分，中文以 SimSun 为 CJKmainfamily，英文以 Times New Roman 为 mainfamily。首页的签名是 Kunstler Script 字体，第18页3.2.4节花体英文部分则是 Vladimir Script 字体。

按照国际惯例，一般在文档的前一部分，或者后一部分是要有致谢的。鉴于本人要感谢的人不是很多，所以就不单独开一页了，和这篇写在最后一起吧。

首先要感谢的自然是父母和姐姐了。在我暑假在家的这些日子他们认真照顾我，把我养得白白胖胖。感谢他们精心的照顾，我才有这个蛋疼的功夫去总结这篇文档。

其次，感谢 Linux 版版主 caspar，你的 ya-buptthesisbachelor[1] 让我受益匪浅，很多我的做法都是从你那里偷师的。感谢 Linux 版大牛和曾经的版主 yegle[2]，在我疯狂灌水的那些日子你没有封我的 ID 真的让我感激不尽。感谢 Linux 版大牛 wks，虽然在别的版面也常常碰到你。你一稿多投的关于 LaTeX 的感悟我也曾一再阅读，并颇有“与我心有戚戚焉”之触感。也要感谢清华的包老师（黄老师??），您的 lnotes 引导我进入了 LaTeX 的世界，很多疑惑也是在您的教程里找到了答案，预祝第二版早日杀青，我已经等得迫不及待了。另外，感谢一众陪我在 Linux 版灌水的童鞋们，让那段无聊的日子不再孤单。

[1] Sorry, 不过这么长的名字也难怪我没记住

[2] 你的 ID 每每让我想起诛仙里的野狗道人