

# X<sub>E</sub>L<sup>A</sup>T<sub>E</sub>X 指南

在 *Ubuntu* 下排版专业的 *pdf* 文章

万泽<sup>1</sup> | 万泽<sup>2</sup>

版本： 1.19

---

<sup>1</sup>作者：万泽，湖南常德人氏。

<sup>2</sup>编者：万泽，邮箱：[a358003542@gmail.com](mailto:a358003542@gmail.com)。

# 前言

我试图收集整理各方面来的 `xelatex` 相关的信息，包括自己的实践经验总结，作为后学者的指南手册。

一切在尊重版权的基础上出于爱好兴趣和相信自由分享的价值原则上进行。

全书主要分为五大块：

## 1. ubuntu 知识

主要是一些最基本的 `ubuntu` 知识或者和本文其他内容相关的知识。当然有时出于全面性考虑会加入一点其他的 `linux` 知识进来，不过这部分需要做到不过分扩充内容。

## 2.xelatex 基础

这部分内容会就 `xelatex` 或者 `latex` 的基础知识内容做全面的整理归类排版，争取做到言简意赅并面面俱到和实用性很强的手册性质。

## 3.latex 高级篇

这部分内容会就 `latex` 的一些较高级的知识做出说明，有时就会 `tex` 的一些命令做出说明。这部分内容主要以各个不同的专题议题的形式展开，不求全，但求就某一个实用的问题解释清楚。

#### 4. 其他问题的讨论

这部分如标题所言，是我根据实际需要接触到的很有用的技巧，以问题的形式展开。这部分内容能够做到分解成为前面三部分的将会分解，觉得不好分解的就放到这里，这部分内容不应过分扩充。

#### 5. 后面的珍宝

这部分会将本书涉及到的有用的资源收集起来给出下载链接或者打印出来。

由于文档变大使得编译有点延迟了，这很浪费时间，而我不喜欢那种分章的写法，一是现在的编辑器目录导航功能都很好了，根本不需要另外分章，二是书本来就是一个整体，相同的东西我不会说上两遍的，所以需要大量建立 `label` 和 `ref`，而现在的编辑器让文章在一个文档中各个小工具帮助都很大，让文档分开只是削弱了那些文档的功能。况且我也希望编译出来是什么样子就是什么样子。

所以我现在约定如下，不写 `part` 部分，文章大小控制在一两百页左右范围内，同时这个文章内部链接互引非常的精彩浑然一体。至于其他的内容建议写入另外的文章中。如果你非常需要将更多的内容合并，我的建议是将最紧密的部分单独写成一个 `pdf` 文档，然后几个 `pdf` 文档再合并吧。毕竟没有分离就没有紧密。不过这种合并之后超链接可能就没有了。所以网络版的建议文档还是不要做的太大。

为了减小文章大小，除了精炼文字之外，和文章相关的几行代码写出来，大段的源码就不列出来了，最多把 `github` 地址写出来。

# 目 录

前言	i
目录	iii
<b>1 ubuntu 知识</b>	<b>1</b>
1.1 通过 U 盘安装 ubuntu	1
1.2 初入 ubuntu	2
1.3 linux 命令介绍	3
1.4 通过 ppa 安装软件	4
1.5 软件推荐	5
1.6 rime 输入法	7
1.6.1 备份和还原	8
1.6.2 把 QQ 拼音上你的词库导入 rime	8
1.6.3 定制 rime 输出特殊符号	9
1.7 ubuntu 美化	13
1.8 去除登录界面的白点	13
1.9 配置 L <sup>A</sup> T <sub>E</sub> X 编写环境	14
1.10 ubuntu 下新宏包的安装	14
1.11 texmaker 技巧	15
1.11.1 自动补全命令	15
1.11.2 自定义命令	15
1.11.3 保存你的设置文件	16

1.12 用 rime 快速输出 $\text{\LaTeX}$ 命令	16
1.13 其他 $\text{\TeX}$ 编辑器评测	16
1.14 gedit 的一些技巧简要说明	17
1.15 结合 github 进行项目管理	18
1.15.1 基本命令	18
1.15.2 日常改动提交流程	20
1.15.3 本地仓库管理文件丢失	20
1.16 安装字体	21
1.16.1 找字体文件	21
1.16.2 放置字体文件	21
1.16.3 命令行安装字体	21
1.16.4 通过 fontmanager 安装字体	22
1.17 ubuntu 系统字体的配置	22
1.18 ubuntu 的备份和还原	24
1.19 更多 ubuntu 知识	25
1.20 注解	25
<b>2 latex 基础</b>	<b>27</b>
2.1 背景知识	27
2.1.1 $\text{\TeX}$	27
2.1.2 关于 Knuth 教授	28
2.1.3 $\text{\LaTeX}$	30
2.1.4 $\text{\XeLaTeX}$	31
2.2 基本情况说明	31
2.3 查看宏包帮助文档	33
2.4 从 documentclass 说起	33
2.4.1 一般的可选项	35
2.4.2 书籍的通用结构	35

2.4.3 页码	36
2.5 页面布局	37
2.5.1 geometry 宏包详细讨论	37
2.5.2 临时改变页面布局	39
2.5.3 多栏环境	41
2.6 字体	41
2.6.1 字体的五种属性	42
2.6.2 调整五种属性	44
2.6.3 调整字号	45
2.7 本文字体配置	47
2.7.1 fontspec 宏包详解	47
2.7.2 基本的命令	48
2.7.3 font features 的讨论	50
2.7.4 最后两个命令	52
2.7.5 设置数学字体	52
2.7.6 xeCJK 宏包详解	52
2.7.7 只能在加载宏包时填入的选项	53
2.7.8 可以在 xeCJKsetup 命令中设置的选项	53
2.7.9 xeCJK 宏包的一些命令	53
2.7.10 设置 CJK 字符大小	54
2.7.11 UTF-8 编码	54
2.8 特殊的符号	55
2.8.1 去掉符号命令后面的空格	55
2.8.2 基本的特殊符号	55
2.8.3 更多特殊符号	58
2.9 初步中文化	60
2.10 颜色	61

2.10.1 本文颜色的配置	61
2.10.2 颜色的心理学	61
2.10.3 有关颜色的基本命令讨论	62
2.10.4 xcolor 宏包	63
2.11 段落	65
2.11.1 换行和分段	65
2.11.2 断字	66
2.11.3 段落中的行距	66
2.11.4 段落间距	68
2.11.5 段首缩进	68
2.11.6 段落对齐	69
2.12 页眉页脚设计	69
2.12.1 观察	70
2.12.2 全部信息归零	70
2.12.3 继续定制 plain 样式	71
2.12.4 本文页眉页脚配置代码	72
2.13 章节标题设计	74
2.13.1 本文章节标题设计	75
2.13.2 titleformat 命令说明	75
2.13.3 章节编号数值修改	76
2.13.4 章节编号深度修改	76
2.13.5 章节编号形式修改	77
2.13.6 章节标题上插入脚注	77
2.13.7 text	77
2.14 目录设计	77
2.14.1 目录深度控制	78
2.14.2 前言加入目录	78

2.14.3 目录行间距拉大 . . . . .	79
2.14.4 目录数字和标题间距调整 . . . . .	80
2.14.5 titletoc 宏包 . . . . .	80
2.15 封面设计 . . . . .	80
2.15.1 mytitle.sty . . . . .	80
2.15.2 一个简单的封面 . . . . .	82
2.16 引用 . . . . .	84
2.16.1 文章内部引用 . . . . .	84
2.16.2 hlabel 命令 . . . . .	84
2.16.3 flabel 命令 . . . . .	85
2.16.4 hyperref 宏包简介 . . . . .	85
2.17 脚注 . . . . .	86
2.18 旁注 . . . . .	86
2.19 尾注 . . . . .	87
2.19.1 endnotes 宏包说明 . . . . .	87
2.20 文字强调 . . . . .	96
2.20.1 emph 命令 . . . . .	96
2.20.2 重新定义 emph . . . . .	97
2.20.3 underline 命令 . . . . .	97
2.20.4 ulem 宏包 . . . . .	97
2.20.5 reduline 命令 . . . . .	98
2.20.6 reddotuline 命令 . . . . .	98
2.21 插入列表 . . . . .	99
2.21.1 基本使用 . . . . .	99
2.21.2 enumerate 环境标签的修改 . . . . .	100
2.21.3 itemize 环境标签的修改 . . . . .	101
2.22 插入图片 . . . . .	101



2.22.1 图片标题的修改 . . . . .	102
2.22.2 设置图片寻找文件夹 . . . . .	102
2.22.3 图片格式的讨论 . . . . .	103
2.22.4 图片格式选择总结 . . . . .	107
2.22.5 导入 pdf 页面 . . . . .	107
2.22.6 本文插入图片的一些 DIY . . . . .	108
2.23 插入表格 . . . . .	109
2.23.1 基本情况的讨论 . . . . .	109
2.23.2 booktabs 宏包 . . . . .	111
2.23.3 booktabs 宏包详解 . . . . .	112
2.23.4 拉宽一行行的距离 . . . . .	114
2.23.5 带颜色的表格 . . . . .	114
2.24 插入代码 . . . . .	115
2.24.1 小代码 . . . . .	115
2.24.2 稍微大点的程序代码 . . . . .	115
2.24.3 fancyvrb 宏包 . . . . .	116
2.24.4 xverbatim 宏包说明 . . . . .	116
2.24.5 cverbatim 宏包说明 . . . . .	122
2.25 语录, 引用和诗歌环境 . . . . .	122
2.26 插入摘要 . . . . .	122
2.27 参考文献 . . . . .	123
2.28 注释 . . . . .	123
<b>3 L<sup>A</sup>T<sub>E</sub>X 高级篇</b>	<b>125</b>
3.1 长度量 . . . . .	125
3.1.1 单位 . . . . .	125
3.1.2 默认的长度量 . . . . .	126
3.1.3 使用长度量 . . . . .	126

3.1.4 定义自己的长度量	126
3.1.5 修改长度量	126
3.2 计数器	126
3.2.1 latex 默认的计数器	126
3.2.2 使用计数器	127
3.2.3 计数器变成带圈的数字系列	127
3.2.4 定义自己的计数器	129
3.2.5 修改计数器的值	129
3.2.6 计数器和其他计数器绑定	130
3.3 新的命令或环境	131
3.3.1 新的命令	131
3.3.2 新的环境	132
3.4 盒子和 glue	133
3.4.1 基本知识	133
3.5 盒子命令介绍	136
3.5.1 makebox	136
3.5.2 raisebox	136
3.5.3 parbox	136
3.5.4 minipage	136
3.6 带颜色或者线框的盒子	137
3.6.1 framebox	137
3.6.2 colorbox	137
3.6.3 fcolorbox	137
3.6.4 fancycolorbox	138
3.7 保存盒子	139
3.7.1 savebox	139
3.8 线条	139

3.9 大文档管理 .....	139
3.10 注释 .....	139
<b>4 其他问题的讨论</b>	<b>140</b>
4.1 plain TeX 命令 .....	140
4.2 TeX 中的程序结构 .....	140
4.2.1 条件控制语句 .....	140
4.3 文本中的上标还有下标 .....	141
4.4 多张图片并列显示 .....	141
4.5 生成字体所有已有的字形 .....	143
4.6 方便手机上观看的 pdf .....	144
4.7 注释 .....	145
<b>5 后面的珍宝</b>	<b>146</b>
<b>6 源码和参考资料打包下载地址</b>	<b>148</b>
<b>插图目录</b>	<b>149</b>
<b>表格目录</b>	<b>150</b>
<b>参考文献</b>	<b>151</b>

# ubuntu 知识

## 通过 U 盘安装 ubuntu

1. 在安装之前请先把硬盘中的资料做一些调整，空出一个大于 20G 的硬盘做将来要安装 **ubuntu** 根目录的地方。然后还需要一个大约为你内存两倍的硬盘分区等下要作为 **linux** 系统的 **swap** 交换分区。
2. 到 **ubuntu** 的官网上下载系统的光盘映像。
3. 用 **ultriso** 软件（其他具有类似功能的软件也行）（注意 **ultriso** 最好选择最新的版本，**ubuntu10.10** 之后的一定要用 9.3 版本之上的）将该光盘映像写入到你的 U 盘中去。
4. 重启计算机，**BIOS** 稍作改动使计算机变成从 U 盘启动。
5. 进入安装过程，其他过程都比较直观，就是硬盘分区设置上我们选择高级手动，然后将你分出来的那个 20G 硬盘作为／的加载点，并设置格式化成 **ext4** 日志系统（其他文件系统如 **ext3** 等也行）。然后设置交换分区，安装完成。

## 初入 ubuntu

- 设置 ubuntu 的 root 用户，这个以后可能会有用。按住 `ctrl+alt+t` 弹出终端，然后输入：

```
sudo passwd
```

设置好密码之后，以后输入：

```
su
```

就可以进入 root 账户了。

- 安装一个有用的工具：

```
sudo apt-get install nautilus-open-terminal
```

这个工具可以让你在视窗背景下鼠标右键之后打开当前目录下的终端，有时还是很快捷的，尤其是遇到一些怪怪的文件名的时候。

- 系统安装好之后第一件要做的事是选一个好的源，然后安装更新。我的 ubuntu 版本是 12.04，在右上角的哪里，软件更新，软件源设置，在下载自的哪里就是软件源的服务商，你最好还是自己搜索一个速度最快的源。然后在终端中执行以下命令来升级系统软件包：

```
sudo apt-get update （更新源）
```

```
sudo apt-get upgrade （升级源下已经安装了的软件）1
```

- 一些闭源的有用的东西：

```
sudo apt-get install ubuntu-restricted-extras
```

这个安装到 `adobe-flashplugin` 的时候会有点卡住，最好还是耐心点吧。或者在 ubuntu 软件中心（Ubuntu 额外的版权受限程序）中安装也是可以的。

---

<sup>1</sup>如果有很多软件需要升级的时候推荐使用命令：`sudo apt-get dist-upgrade` 这样不容易出错些。

- ubuntu 系统清理，运行以下两个命令即可：  
sudo apt-get autoremove （清理软件残余）  
sudo apt-get autoclean （清理缓存）。
- 删除通过 apt-get 安装的软件：  
sudo apt-get remove 要删除的软件名  
删除相应的软件配置文件：  
sudo apt-get purge 软件名
- 长按 super 键或者说是 win 键会弹出一些 ubuntu 桌面常用的快捷键，比如说 ctrl+win+d 是显示桌面，按住 win+f 搜索文件，win+a 搜索程序等。

## linux 命令介绍

**\$ 和 ~** 当我们打开终端的时候，看到一个美元 \$ 符号，如果我们输入 su 命令，然后进入 root 账户，看到开头有一个 # 符号，其中 \$ 表示普通用户，# 表示现在是超级用户。然后我们看到一个波浪号 ~，这个波浪号的意思就是当前用户的个人主文件夹，比如现在我这里 ~ 的意思就表示目录 / home / wanze。

**cd** 一般 dir 或者 ls 之后列出该目录所包含的文件夹或者文件，然后执行如下命令：

cd 某个文件夹名

就进入这个文件夹了。如果 cd 一个文件就会报错。然后特殊的有：

cd ~ 回到个人主文件夹

cd . 其中点表示当前目录

cd .. 表示返回上一级目录

**cd** 某个目录，比如 **cd / etc** 那么就直接跳到系统的 **/ etc** 目录下了。

**cp** 复制文件命令：

**cp** 要复制的文件 目标文件目录

**which** 查看系统某个命令的位置

**touch** 创建某个文件或者对已存在的文件更新时间戳

**rm** 删除某个文件，一些具体选项我就不罗嗦了，如果有不明了的请自己用 **-- help** 来查看帮助信息。比如这里的 **-r** 选项就可以强制删除一个目录下所有的文件。

**mkdir** 创建文件夹

**rmdir** 删除文件夹

**mv** 移动某个文件或者重命名某个文件

## 通过 ppa 安装软件

ppa 也就是软件源，一般推荐 ppa 安装软件即该软件还处于活跃开发期中，所以推荐尽快更新到官方最新的版本。过程就是添加 ppa 源，然后用前面讲的 **apt -get upgrade** 来更新源下的软件。添加 ppa 源的格式如下：

**sudo add-apt-repository ppa:** 后面就是什么官方源地址了

然后更新源下的软件列表：

**sudo apt-get update**

接下来就是通过 **apt-get install** 来安装该软件了。**upgrade** 是更新源下已经安装了的软件，所以第一次装还是要用 **install** 命令。<sup>2</sup>

---

<sup>2</sup>[how-install-any-software-ubuntu-ppa](#)

## 软件推荐

以下编号：Ⓐ 表示可以从 `ubutnu` 软件中下载到，Ⓑ 表示可以从终端 `apt-get` 命令中下载到，Ⓒ 表示推荐从 `ppa` 安装。

**新立得软件包管理器** 这个有时安装一套软件组合很有用的。Ⓐ

**chromium** 不清楚和 `google-chrome` 的区别。它的插件很方便，有的时候网页显示相比较于 `firefox` 不会出错些。Ⓐ

`chromium` 插件推荐：

**Daum Equation Editor** 本文暂时不会讨论数学公式的输入问题，这里有个简单的解决方案，就是在这个插件里面编辑公式，下面就有对应的 `LaTeX` 的代码。

**代理助手** 好吧，我讲这个完全是因为天朝的防火墙，不学会翻墙严重干扰我们搜索信息。具体设置是 `http` 代理填写相关地址，`china list` 这个选项也勾上吧。下面还有详细说明。

**翻墙软件** 要搜东西真的要用 `google`，在天朝你懂的，因为要跑防火墙慢死了。轮子的自由门还可以。<sup>3</sup>翻完墙之后到弹出来的那个网站或者 `voa` 网站哪里多下几个类似的翻墙软件吧，有备无患。`freegate` 在 `ubuntu` 下需要使用 `wine` 模拟，在模拟前先用 `wine` 装好 `mfc42.dll`，然后就行了。`freegate` 进去之后在通道哪里选择经典模式。`google-chrome` 那边的设置在代理助手哪里，就是使用 `freegate` 提供的那个 `http` 地址加上端口号即可——在内容那一页。一般代理助手就点系统自动连接吧，毕竟是免费代理，别想质量有多好。

---

<sup>3</sup>[下载 freegate 的链接](#)



**texmaker** 写  $\text{\LaTeX}$  非常好的界面环境，和 windows 下的 **texstudio** 类似，但是感觉界面更加的漂亮。④

**shutter** 一个非常好的截图软件。**ubuntu** 系统自带的截图软件截完图不能拖动改变大小，这个可以。④

**ubuntune** 在 **ubuntu** 系统下很好的文档同步软件，在手机上安装之后可以方便进行某些文档的传输。云诺和坚果云也还可以。115 网页版也行。④

**wine** 虽然 **wine** 软件一直在更新，但是老实说总感觉不是十分令人满意。这里提到 **wine** 是因为前面谈到的翻墙软件需要用 **wine** 来模拟。**wine** 的安装不同于其他软件，尽量下载最新版本的吧。到官网上查看最新的稳定版本号，比如 **wine1.6**。然后在终端上通过 **ppa** 来安装，**ppa** 地址见脚注。④<sup>4</sup>

**workrave** 人们对久坐的危害总是低估了，这个软件帮助提醒人们做一段时间之后最好起来活动活动。④

**Gcolor2** 这个小软件取色，RGB 转换等很有用。④

**wps for linux** 目前写毕业论文不得已还是要用 doc 格式，而在 **ubuntu** 下我试过很多解决方案，比如 **wine**，**libreoffice**，或者网上的在线编辑。结果都不尽如人意。因为我需要的是最好完全全兼容 **microoffice2003** 的格式。金山公司的这个 **wps for linux** 目前还在开发中，但是效果还是可以了。下载地址见脚注。⑤<sup>5</sup>

其他还有很多有用的软件看官自己探索吧。

---

<sup>4</sup>[ppa:ubuntu-wine/ppa](#)

<sup>5</sup>[wps for linux](#)

## rime 输入法

更多信息请参考 rime 官方 [wiki](#)<sup>(1)</sup>

。还有一种 `ppa` 安装方法，`ppa` 地址见尾注<sup>(2)</sup>。

rime 输入法也可以在 `fcitx` 输入法框架下进行，就目前来说（2013-12-04）感觉 `fcitx` 输入法框架要比 `ibus` 稍微好一点点。

### 1. 安装 `ibus-rime`:

```
sudo apt-get install ibus-rime
```

### 2. `ibus` 输入框不见了:

```
ibus-daemon -drx
```

### 3. 重启 `ibus`: 有时 `ibus` 会出现一些奇怪的问题，在终端运行下面命令重启 `ibus` 也许会解决:

```
killall ibus-daemon
```

```
ibus-daemon -d
```

### 4. Rime 输入法的其他设置: 在输入框调到 `Rime` 的状态的时候，按一下 `F4`，就弹出一些设置选项。

### 5. 用 `Rime` 输入常见的特殊符号: 比如我要输入 `*`，按下 `shift` 和数字 `8` 按键，就出来一些选项供你选择。其他很多符号类似，具体请参考官网上的说明。

### 6. `Rime` 输入法是架构在 `ibus` 框架之上的，所以在系统输入法首选项哪里，可以设置 `Rime` 候选项是横着显示或者竖着显示，还有下面可以自定义显示的字体和字体大小。推荐用 `sans-serif` 字体。然后字体大小要比你输出之后的效果稍微大一点，这样醒目些。

## 备份和还原

输入法用久了，你的自造词还有你输入相关词的词频这些信息是非常有用的，能够极大地提高你的工作效率。rime 备份的核心词是命令：**rime\_dict\_manager**。你需要在 rime 的用户配置文件夹里面打开终端，然后输入这个命令，你就可以看到提示信息了。

首先输入 **-l** 选项，你可以看到现在已经有的词典，比如 `luna_pinyin`。然后用 **-s** 同步，这样你就会看到一个 `sync` 文件夹，里面放着快照文件。**-b** 是选择性是备份某一个词典。**-r** 是还原快照文件。**-e**，**-i** 是 `txt` 格式操作，`txt` 保存数据会有点损失。将用户 `rime` 配置文件夹整个复制到同步盘即可。

## 把 QQ 拼音上你的词库导入 rime

1. 在 windows 下在 QQ 拼音设置里面找到那个导出词库的命令。就是一个 `txt` 文件。
2. 将这个 `txt` 文件另存为 `utf-8` 编码
3. 进入 linux（以下 windows 用户原理类似）找个在线的简繁转换网站，如果你使用的是简体略过这步。然后转换成为繁体，由于词库比较大，建议你耐心点，推荐笨笨网站简繁转换，不容易卡死。实在不行就只好下载软件了。我的一万多条还是能够转换出来。
4. 打开 `libreoffice`，将前面的 `txt` 文件，复制，粘贴。然后会弹出一个选项，分隔符注意勾选空格。
5. 前三列有用，后面的意义不大，统统删除。

6. 按 `ctrl+f`，下面弹出查找于替换。输入，后面输入空格全部替换。输入' 后面输入空格全部替换。
7. 将第一列第二列位置互换，就是剪切粘贴操作。
8. 全部选择前三列，复制，粘贴到文本。
9. `rime_dict_manager -i luna_pinyin` 上面谈及的你创建的文本名字。
10. 注意上面要在 `rime` 用户目录下操作即`.config/ibus/rime`。
11. 备份 `dict_manager -b luna_pinyin`

## 定制 rime 输出特殊符号

我希望通过 `rime` 输入法能够快速地将 `Unicode` 中的符号打出来。本文主要关注这个问题，其他定制请参阅官网 [wiki](#)。

## rime 基本情况说明

`Rime` 的 `ubuntu` 版本叫做中州韵，不管这么多，一般称作 `ibus-rime`，下面简称 `rime`。下面的讨论只使用于 `ubuntu`，我目前的版本是 13.04，12.10 试过没有问题。

`rime` 的程序在哪里我不关心，在 `ubuntu` 下所有程序的配置是用户配置优先级高于程序自带的配置，所以我们在 `rime` 的用户文件夹里面 `DIY` 就是了。`rime` 的用户资料夹在：

`~/.config/ibus/rime`

进入这个文件夹之后，我们可以点击看以下里面的内容，这些文件情况说明如下：

**cangjie5.schema.yaml** 仓颉五代配置文件。我用的是朙 (míng) 月拼音，这个文件就不管了。

**default.yaml** 这个文件存放着输入法的一些全局设定，重新部署 rime 时会重新生成。

**installation.yaml** 这个文件存放着 rime 的安装信息，不用管它。

**luna\_pinyin.schema.yaml** 这个是默认的朙月拼音的配置文件，要仔细看看。

**luna\_pinyin\_fluency.schema.yaml** 这个是朙月拼音语句流的配置文件，我不关心。

**luna\_pinyin\_simp.schema.yaml** 这个是明月拼音简化字的配置文件，也就是输入法切换到简体字输入之后起作用的。

**luna\_pinyin\_tw.schema.yaml** 这个是朙月拼音台湾正体的配置文件，没用过。

**symbols.yaml** 这个是 rime 自带的一个输出 symbol 的配置文件，但是奇怪的是刚开始并没有配置好。

**user.yaml** 这个是用户状态信息。

## 开始 DIY

其中官网 wiki 中讲了一些 default.custom.yaml 的定制，那些我似乎都没有需要，然后略过。现在主要讨论 luna\_pinyin.custom.yaml 的配置问题。具体配置如下：

---

```

1 patch:
2   punctuator:
3     import_preset: symbols
4   recognizer:
5     import_preset: default
6   patterns:
7     reverse_lookup: "'[a-z]*'?$"
8     punct: "^/[a-z\\[\\]]*$"

```

---

我想你已经看到了这些配置文件都是.yaml 后缀，它们是用一种什么 yaml 数据描述语言写的，类似于 XML 的标记语言。然后里面还有一些正则表达式的东西。关于这些内容我是能够略过就略过吧。

刚开始我写这个 patch 的时候总是不成功，然后看到佛振的那个帖子，链接到另外一个源码上。[请参看这个网站](#)。然后直接修改 luna\_pinyin.schema.yaml 文件，就是把 default 改成 symbols，也就是指向那个 symbols.yaml 文件，然后将 patterns 的 punct: 那一行复制过去。就发现可以了。

现在 patch 可以正常工作了，原因是之前我的缩进不正确，由于我对 yaml 语言不清楚，就此打住。

其中 punct 那一行涉及到的正则表达式知识有符号 ^ 表示匹配开始，\$ 表示匹配结束，[] 和里面的内容一起表示一个字符，这个字符可能是 a-z 所有的小写字母，然后我想加上符号 [和]，因为我想新建一个命令 /[] 就能输入很多形式的括号。最后发现这个问题还有点小麻烦，[请参看这个网站](#)。他解释了最好前面加上两个 \，然后才能更好的工作。最后那个 \* 表示重复零次或者更多次。(3)

## symbols.yaml 文件的说明

我在 `github` 网站上新建了一个项目。里面有我编辑的 `symbols.yaml` 文件。现在将我所做的修改工作简单说明如下：

前面是全角和半角部分，如果你有需要可以自己定制下，这里我略过。然后看到 `symbols:` 哪里，开始自己定制命令。前面那个代码我们看到了匹配是以 `/` 开始的，所以下面所有的命令都要以符号 `/` 开始，如果你想有其他模式修改正则表达式就可以了，比如说可以换成命令以符号 `\` 开始。

`yaml` 的缩进表达我不太清楚，实际上我不太喜欢那种缩进的语法。`yaml` 还提供了另外一种语法形式。就是一个系列可以用一个方括号明确标识出来。<sup>6</sup>在这里格式很简单，就是建立了很多符号调出命令，对应的后选词选项用方括号包围起来。

然后还需要一提的就是 `yaml` 语言支持锚点，意思就是某一个变量，跟着 `&` 任意的名字，后面就可以用 `*` 那个名字来表示那个词条了，这个有点类似于取别名的意思。anyway，用这样的方法可以给你建立的命令取几个别名。

## 重新部署 rime

最后将上面两个文件放入用户 `rime` 配置文件夹哪里，然后删除 `default.yaml` 文件，然后运行 `ibus-daemon -drx`。等着 `rime` 重新部署完毕即可。

---

<sup>6</sup>请参看这个网站

## 具体输入符号的命令

现在我在 `rime` 下输入 `/gcs`，就会弹出很多符号。类似的还有很多，就不一一介绍了。

## ubuntu 美化

1. 不要 `unity` 环境，我一直都用 `unity` 环境，觉得自己还是有权发言了，真的不好。推荐安装 `gnome` 环境。我不喜欢什么桌面特效，用无特效环境，毕竟系统流畅才是最重要的东西。
2. 安装 `vareity` 软件自动换桌面，推荐在加点其他的网站壁纸资源。然后登录界面也选上。
3. 按住 `alt` 键右键点击 `gnome` 面板，然后优化。推荐把上面的设置到底部，因为上面的视角很重要，然后设置为隐藏带按钮最好。
4. 顺便谈谈快捷键，在系统设置 → 键盘 → 快捷键那里。你可以设置各种程序的快捷键。比如 `ctrl+alt+t` 是打开终端设置 `ctrl+alt+w` 打开默认网页浏览器等等。

## 去除登录界面的白点

如果你放一张非常漂亮的图片，比如 `variety` 可以设置，那么这些白点将会让你很不舒服。<sup>7</sup>

通过运行下面的 `bash` 脚本，就是一行行输入进去即可，重启之后确实去除了。

---

<sup>7</sup>参考了这个网站



---

```
1 sudo xhost +SI:localuser:lightdm
2 sudo su lightdm -s /bin/bash
3 gsettings set com.canonical.unity-greeter draw-grid false
4 exit
```

---

## 配置 $\text{\LaTeX}$ 编写环境

1. `sudo apt-get install texlive` (下载安装 ubuntu 下有名的 texlive)
2. `sudo apt-get install texlive-full` (下载安装 texlive 的各个包)
3. 在 ubuntu 软件中心中下载安装 texmaker 软件。
4. ibus 似乎在这里面有点小问题，有点不稳定。在终端中运行  
`sudo apt-get install ibus-qt4`  
则问题解决。
5. 在 texmaker 的选项和配置 texmaker 里面<sup>8</sup>，设置快速构建，点上用户自定义那一栏，然后输入如下命令：  
`xelatex -interaction=nonstopmode %.tex|`  
这是使用 xelatex 来对目标 tex 文件进行编译，而不是传统的 latex 或者 pdflatex 方式，之所以这样是因为多方对比之后，觉得其在字体处理方面是未来的趋势。

## ubuntu 下新宏包的安装

安装 texlive-full 之后，如果还遇到没有的宏包，可以先到 CTAN 官网上下载到这个宏包之后，然后将这个宏包解压到系统目

---

<sup>8</sup>现在 texmaker 已经加入  $\text{\LaTeX}$  编译命令。

录：

`/usr/share/texmf/tex/latex` 里面即可。

当然你也可以在另外一个文件夹里面，这里必须是你的主文件夹下，新建一个文件夹 `texmf`，然后里面新建一个 `tex`，然后再新建一个目录 `latex`，然后在这个 `latex` 里面放着你下载下来的宏包。具体比如说有一个宏包名字叫做 `config`，那么 `latex` 下面就是 `config` 文件夹，然后里面就是 `config.sty` 文件。你自己写的宏包扔进去一样有效。唯一要额外做的操作就是在 `texmf` 目录之下运行命令：

```
sudo texhash
```

让 `texlive` 把这个目录也加入搜索范围。

## texmaker 技巧

### 自动补全命令

在菜单里找到用户自定义的 `customize completion` 也就是自动完成，里面加入你想要的命令。比如：

`\textbackslash` 然后点击 `add`，这样以后想输入显示命令前面的那个斜线的时候会方便点。如果括号里面加入 `@` 符号，那么就会出现类似系统自带命令 `\section{•}` 的那个黑点 `•`。

### 自定义命令

在菜单哪里用户自定义，你看到可以用户自定义命令，填好之后就是快速构建下面那些备用的 `1:2:3:` 哪里将成为有意义的命令。

## 保存你的设置文件

在选项哪里有保存设置文件的功能，主要是自定义命令，自动补全命令等可以保存下来。

## 用 rime 快速输出 $\text{\LaTeX}$ 命令

具体效果就是我按下 `/tex` 等命令，就会弹出很多  $\text{\LaTeX}$  命令，这个也是修改 `symbols.yaml` 文件来达到的，也是在我新建的那个 [github 项目](#) 里面，我做了很多优化工作，有兴趣的可以研究下。

(4)

## 其他 $\text{\TeX}$ 编辑器评测<sup>9</sup>

**emacs** 安装 `emacs24` 之后，用 `package manager` 安装 `auctex` 宏包。然后一些基本的 `latex` 编辑器功能都有了。稍微熟悉下就行了。唯一遗憾的是没有那种左侧很方便的目录导航功能。然后 `emacs` 的各种快捷键让我压力好大。

**winefish latex** 太高端了，刚进去一片空白？

**gedit** `gedit` 安装好 `latex` 的插件之后也很好。因为 `latexila` 编辑器在编译大文件的时候速度好慢，而 `gedit` 是直接调用终端模式所以没有损伤的。值得一提的是 `latexila` 编辑器的一些功能在 `gedit` 里面可以通过外部工具这个插件自己编写终端命令来实现。

**texworks** 没怎么评测。

---

<sup>9</sup>虽然 `texmaker` 软件也有些小缺陷，但综合起来我觉得是最棒的。

**kile** 没怎么评测。

**texstudio** 我刚进去什么都没动，它好像中毒一样到处乱操作。。

**gummi** 打开 tex 文档或者做些小的测试还是可以的，但是似乎不支持 include 文档。

**latexila** 这个软件刚进去我就感觉设计理念非常的简洁，和 gedit 一样。然后我特别在意的文档结构图显示，语法染色都有。[\(5\)](#)

## gedit 的一些技巧简要说明

在编辑 → 首选项那里请安装好 latex 插件和外部工具插件，恩，代码注释插件和文件浏览器插件推荐。

我在这里重点介绍一下外部工具插件的用法。看到工具 → 外部工具。请选择最下面的那个管理外部工具，看到在此处打开终端，gnome-terminal 后面去除掉，然后就能正常工作了。[\(6\)](#)

现在我们新建一个命令，名字叫做 xelatex，具体内容如下：

---

```
1#!/bin/sh
2filename=$GEDIT_CURRENT_DOCUMENT_NAME
3shortname='echo $filename | sed 's/\(.*\)\\.tex$/\1/'
4xelatex -interaction batchmode -src $filename
```

---

那么这个小工具就实现 xelatex 编译功能了。

然后我们也可以再新建一个小工具，名字叫清理：

```
1#!/bin/sh
2filename=$GEDIT_CURRENT_DOCUMENT_NAME
3shortname='echo $filename | sed 's/\(.*\)\\.tex$/\1/'
4rm -f $shortname.aux $shortname.ent $shortname.out
5$shortname.lot $shortname.idx $shortname.lof
6$shortname.ilg $shortname.ind $shortname.log
7$shortname.toc $shortname.bbl $shortname.blg
```

---

这样运行它就可以清理临时文件了。

## 结合 github 进行项目管理

以下主要参考[这个网站<sup>\(7\)</sup>](#) 到 github 上注册, 创建新的项目, 还有安装 git 软件 (ubuntu12.10 自带的有) 都很简单的, 我就不多说了。下面就 git 命令使用的基本流程说明如下:

### 基本命令

#### 远程仓库文件到本地

网上创建项目之后, 你需要将网上的存档下载到本地, 在你希望下载的地点, 打开终端:

```
git clone https://github.com/a358003542/xelatex-guide-book.git10
```

git init 命令用于本地创建的文件夹上传到远程仓库, git clone 下来的仓库文件已经索引了。<sup>(8)</sup>

---

<sup>10</sup>后面的这个链接地址在你创建的项目的右下角哪里, 写着 HTTPS clone URL

## 本地仓库文件进入索引

下载下来的本地仓库文件进入 git 的索引，该文件夹内的所有文件都进入索引则在终端中输入如下命令：

```
git add .
```

因为我们在 github 创建项目的时候已经创建了一个配置文件，比如我选择的是 latex 语言，然后它会自动会处理将某些文件不上传。

如果你本地删除了文件，你希望远程仓库也删除这些文件，那么使用命令：

```
git add --all .
```

## 将索引中改动的文件提交到 head

不太清楚这个索引，head 具体是什么意思，anyway，过程就是这样的。

```
git commit -m '2013-08-25:19:00'
```

后面的文字串等下在 github 网站中会看到的，表示这个文件的标示符吧，你也可以取其他的名字，比如 version0.1 之类的。

## 将 head 中改动的文件更新到远程仓库

第一次你需要给你的远程服务器取个简单点的名字：

```
git remote add origin https://github.com/a358003542/xelatex-guide-book.git
```

然后以后都可以用这个简单的命令来更新了：

```
git push origin master11
```

---

<sup>11</sup>这里的 origin 的意思前面说了，master 是远程仓库默认的一个分支，后面会讲到你可以创建其他的分支。

## 远程仓库的改动更新到本地

下面这个命令 `git` 对文件的操作是合并式的，也就是只是替换最新改动的文件。如果你希望远程仓库所有改动包括删除也更新到本地，使用可选项 `--all`。

```
git pull origin master
```

## 日常改动提交流程

一般情况下就用前面讲的的三步，`add .` `commit -m` 然后 `push`。这是基本的日常维护提交流程。

如果你在网站上对远程仓库做了一些修改，记得先用 `pull` 命令将远程仓库的改动更新到本地。

## 本地仓库管理文件丢失

如果你把本地仓库隐藏的 `.git` 文件夹删除了，但是本地的更改你又想上传到远程仓库，你首先需要 `git init`，然后添加远程服务器名字，`git remote add origin` 地址。然后建立本地索引，`git add --all`。然后 `commit` 和 `push`。这里可能远程服务器会拒绝，`push` 的时候加入 `-f` 选项会强制 `push`，但是 `github` 网页里面所有之前 `commit` 的记录都没有了，如何无缝对接我还不清楚。

## 安装字体

### 找字体文件

如果你装了 windows 系统，那么你可以到 windows 下 copy 这些字体文件。比如 windows 常用的宋体，times new roman 等，在 C 盘的 windows 的 fonts 文件夹里面。本文用的就是 adobe 中文系列：adobe 宋体 std，adobe 黑体 std，adobe 楷体 std。很奇怪，在 pdf 上我觉得这几个字体感觉很好，但是在屏幕上就觉得不太好了。

### 放置字体文件

推荐都放在 ubuntu 的主目录的 .fonts 文件夹里面<sup>12</sup>，如果没有请新建一个。这是通常默认用户新加字体放置的目录。当然你也可以放在其他目录里面，比如你的同步盘里面，然后用 font-manager 安装字体也是可以的。<sup>13</sup>

### 命令行安装字体

运行命令：

```
fc-cache -f -v
```

字体就安装好了，如果你要看现在你的系统上有那些可用的中文字体，在终端运行命令<sup>14</sup>：

```
fc-list :lang=zh | sort >ziti.txt
```

---

<sup>12</sup>这是一个隐藏目录

<sup>13</sup>这是本文推荐的方式，安装卸载字体都方便些。

<sup>14</sup>| 表示 linux 命令中的通道，第一个命令的输出信息流会流向 sort 命令，排序之后重定向到 ziti.txt 文件里面。然后终端的数据就保存在这个文件里面了。



打开 `ziti.txt`，里面就是你的可用中文字体的信息，比如：

```
/home/wanze/.fonts/simsun.ttc:
```

```
宋体,SimSun:style=Regular
```

其中第一个是字体文件所在的目录，第二列信息是可以调用的名字，有宋体和 `SimSun`。

## 通过 fontmanager 安装字体

你可以安装其他软件来安装和管理字体，比如 `fontmanager`：

```
sudo apt-get install font-manager
```

这个软件查看安装卸载或者禁用某些字体都很方便的，需要提醒的是这个软件占用了默认的用户配置文件 `.fonts.conf`。然后你的字体 DIY 需要到 `~/.config/font-manager/local.conf` 哪里去设置。这个下面会讲到。

## ubuntu 系统字体的配置

不是特别难看的情况就没必要改动系统字体，因为我们不要低估了人眼的适应能力。这里的配置主要是指由于系统升级带来的字体的改变，特别是中文字体的改变，手动配置将其固定下来。

(9)

---

```
1 <?xml version="1.0"?>
2 <!DOCTYPE fontconfig SYSTEM "fonts.dtd">
3 <fontconfig>
4
5 <match target="font">
```

```

6 <edit name="rgba" mode="assign"><const>rgb</const></edit>
7 </match>
8
9 <match>
10 <test name="lang" compare="contains"><string>zh</string></test>
11 <test name="family"><string>serif</string></test>
12 <edit name="family" mode="prepend"><string>微软雅黑</string></edit>
13 </match>
14
15 <match>
16 <test name="lang" compare="contains"><string>zh</string></test>
17 <test name="family"><string>sans-serif</string></test>
18 <edit name="family" mode="prepend"><string>微软雅黑</string></edit>
19 </match>
20
21 <match>
22 <test name="lang" compare="contains"><string>zh</string></test>
23 <test name="family"><string>monospace</string></test>
24 <edit name="family" mode="prepend"><string>微软雅黑</string></edit>
25 </match>
26
27 </fontconfig>

```

---

有很多内容没有深究，第一个是打开 `rgba` 模式，优化液晶显示的。然后下面就是对三大字族设置，如果是 `zh` 中文的话那么就 `prepend` 也就是插入微软雅黑，也就是在搜索队列中微软雅黑优先级最高。这个可以通过 `fc-match -s serif` 等来查看。然后其他字体设

置不想涉及了，只希望他们能够稳定下来。感觉设置雅黑字体了，系统的主题换为 **Radiance** 更好看些。然后用 **font-manager** 针对微软雅黑高级设置加上 **AA** 和 **AH**。一个是反锯齿一个是自动粗细设置吧。就这样了。<sup>(10)</sup>

## ubuntu 的备份和还原<sup>15</sup>

首先是利用系统自带的备份软件将 **home** 文件夹里面的一些内容备份好，注意配置文件夹排除法则。

备份步骤简介如下：

1. 备份你的 PPA，也就是你通过 PPA 装的软件。有的可能都没有通过 PPA 装过软件，那么这一步和下面的备份 PPA 的 key 都可以省略。命令如下：

```
sudo cp -r /etc/apt/sources.list.d ~/Nutstore/ubuntu-config
```

2. 备份你的 PPA 的 key:

```
sudo apt-key exportall > ~/Nutstore/ubuntu-config/myppakey
```

3. 通过新力得软件包管理器生成你安装的所有软件包的列表：选择文件 **F**→ 将标记的项目另存为 **A**→ 然后下面保存全部状态，不仅仅是变更选项勾上 → 然后选择一个地址取一个有意义的名字。

如果重装系统了，还原步骤相当于前面备份步骤的反向操作吧：

---

<sup>15</sup>以下主要参考了这个网站

1. 还原你的 PPA: 这里就不写命令了, 和上面类似也是 `cp -r`, 不同的是现在将你保存的 PPA 还原到新系统的 `etc` 的 `apt` 目录下。
2. 还原你的 PPA key, 假设对照上面的情况, 我们有命令:  

```
sudo apt-key add ~/Nutstore/ubuntu-config/myppakey
```
3. 通过新力得软件包全部安装你之前保存的软件包列表: 选择文件 `F`→ 读取标记的项目 `R`→ 然后类似正常的通过新力得安装软件包的步骤, 点击应用即可。
4. 通过上面的步骤, 你的新系统的所有软件全新安装好了, 至于其他配置文件前面说了一些你在意的配置保存, 然后选择相应的位置, 一般在 `.config` 文件夹里面, 复制进去就行了。

## 更多 ubuntu 知识

更多 ubuntu 知识将被放在 `ubuntu 技巧` 一书中。

我又扫了一遍。这一部分进入冷处理状态, 即除非特别的需要, 不再新加入内容了。

## 注解

- (1) [rime 官方 wiki](#)
- (2) ppa 地址是: `[ppa:lotem/rime]`
- (3) 你可以到这个网站继续学习正则表达式: [正则表达式 30 分钟入门教程](#)
- (4) 在 `rime` 中文模式下, 直接按 `enter` 键输出的也是英文, 所以一些简单地命令直接输入也是很快捷的。

- (5) latexila 编辑器使用 xelatex 引擎生成文档，一样要自己编写一个生成命令，在创建 → 首选项那里 → 标签就写 xelatex，写其他的也可以。扩展名写.tex，然后命令和 texmaker 一样写上这个：xelatex -interaction=nonstopmode \$filename 。
- (6) 本信息来自 ubuntu13.04，gedit 版本号为 3.6.2。
- (7) 在项目网站右下角 settings 哪里进去有很多项目管理内容，其中最下面有删除项目的功能，请慎重使用。
- (8) 参考了[这个网站](#)
- (9) 这个代码主要参考了[这个网站](#)
- (10) chromium 字体也都设置成为微软雅黑吧，然后我感觉页面稍微放大点更好看。

# latex 基础

## 背景知识

### TeX

以下完全按照 wikibook 中的 latex 翻译的。[wikibook-latex](#)

TeX 是一个底层的标记式的编程语言，Donald Knuth 发明的排版系统，可以用来排版出很漂亮的文章。当初 Knuth 看到自己的文章和书籍被排版的丑陋不堪，于是在 1977 年开发了这个排版引擎，这个引擎深深地改变了出版业，大力扩展了数字打印设备的潜能。1989 年 TeX 支持了 8 位字符，然后 TeX 的开发就被冻结了，只限 bug 的修复。TeX 作为一种编程语言，是支持 if-else 结构的：你能够在里面执行数学运算（他们在编译文件的时候被执行），等等。。不过你会发现要做其他的还是很困难的除了排版文字。TeX 对于文章的结构和格式提供了良好的解决方案，使得它成为一个强大的神器。TeX 是出了名的稳定，可以运行在各种计算机上，几乎没有 bug。TeX 的版本是按照 的序列扩展的，目前到了 3.1415926。

## 关于 Knuth 教授

摘自台湾同胞良心作品 latex123。latex123

- 1938.01.10 诞生。Milwaukee, Wisconsin; U.S. citizen。
- 1956 进入凯思工学院 (Case Institute of Technology) 学习物理。
- 1960 毕业后进入加州理工学院 (California Institute of Technology) 研究所，此时转向数学领域的研究方向。
- 1961.06.24 和 Nancy Jill Carter 结婚。他的中文名字是高德纳，他老婆名叫高精兰，老婆小他一岁。两个小孩，一男一女。中文名字是 1977 去中国大陆时取的。
- 1963 拿到 Ph.D.，并留校任教。
- 1968 开始任教于 Stanford 大学，信息科学系 (Computer Science)。同年开始撰写名闻遐迩的 TAOCP(The Art of Computer Programming)。有人曾说，看了这部书，往后对写程序的话题都会变得谦虚。:)
- 1977 不满意书商所印出的 TAOCP，因此，自行开发 TeX 排版系统，这可就影响了往后的排版、出版界，至今不坠。但也因此拖延了 TAOCP 第四册的完成时间。
- 1986 荣获 ACM 软件系统奖。

他可说是著作等身，书籍、论文都有，他的任何著作有个奇怪的『副作用』，那就是任何人发现书上的错误，都可以向他举发，并领取 \$2.56 (美金)！想试试看「手气」吗？台湾就有人领过。:-) 为什么是 \$2.56？Knuth 教授的答案是：

“256 pennies is one hexadecimal dollar.” 发现 TeX 系统的臭虫也是一样，这个奖金每年倍数增加，直至 \$655.36( $2^{16}$  pennies) 为止。

他也很推崇自由软件基金会 (Free Software Foundation) 及 GNU/Linux，把一些希望都寄托于 GNU/Linux，尤其是 Unicode 环境，他希望 GNU/Linux 很快就能在网页上显示他的中文名字，而不必使用图档。其实是可以做到了，只是 Unicode 环境还不算普及罢了。他曾在 1996 年接受 Dr. Dobbs's Journal 访问时英雄惜英雄的公开表示，倡导自由软件 (Free Software) 的 Richard M. Stallman 是他心目中的英雄之一，他认为自由软件基金会这些人所做的贡献很不错，虽然和他的方式不一定一样，但许多贡献是互有认同的。

在发展 TeX 时就同时思考 WEB（这个词比目前使用的 WWW Web 还早使用），那是一种 *literate programming* 的程序方法。他认为目前已成熟的可以提出含有文件的程序方法，使写程序就像写文学作品（小说？）一般的艺术表现。后来也把他由 C 改写（和 Silvio Levy 合作），名为 CWEB。TeX 就是由 WEB 写成的，WEB 可视为 Pascal 语言的一个子集。

一个 *literate* 程序师可被视为文学作家、评论写作者、随笔作者……，程序的表现不仅仅是搬弄符号，而是展现自己的风格，当然也是指达成目的的风格、甚至程序中变量运用的风格。

这样一来就可以展现让人类较能理解的程序码。使用形式及非形式的融合，而且两者间相辅相成，目的达成了，也让阅读的人就好像读文学作品般的去抓住作者的心，使程序创作提升至更高的（文学）艺术境界，而不再是死板板的 code 了。

Knuth 大师已于 1992 年自大学退休，但仍在 Stanford/Oxford 等大学有授课。目前正处于隐居的生活，他这么早退休的原因，就



是因为 TAOCP 这部书，他估计大约要花 20 年来完成，因此目前的工作重点是完成他的 TAOCP（分成好几册，目前真正出版的只有三大册）。他认为 email 会影响他的思路，因此，宁愿留住址，要和他联络就只好写信，传真。给他的秘书的 email，是最后有时间才会去看，他曾公开的表明，这部书是他这一生中最重要的工作。

虽然 Knuth 教授写了许多严肃艰深的书籍、论文，但是他也是有风趣的一面，在 1996 年，Mathematisch Centrum (MC, 为庆祝五十周年庆改称为 Centrum voor Wiskunde en Informatica, CWI) 曾邀请他演讲，并知会荷兰的 TeX User Group(NTG)，NTG 见机会难得，就邀请 Knuth 教授另开个 TeX 及 Metafont 讨论会，并接受大家的提问，他说：『不对，我也是可以问你们问题的！』。而且，他还说：『这种问答的内容，很可能在不同场合重复过，所以，如果我对同一个问题，曾有过两种答案的话，你们必需取其平均值。』他的妙语如珠，在许多类似的场合常常引起哄堂大笑，但实际的内容却绝非泛泛之言。:-)

## LaTeX

LaTeX 是一个宏包, 其目的是使作者能够利用一个预先定义好的专业页面设置, 从而得以高质量地排版和打印他们的作品。LaTeX 最早是由 Leslie Lamport 编写的, 并使用 TEX 作为其排版系统引擎。<sup>1</sup>

---

<sup>1</sup>again, form the lshort

## XeLaTeX

关于 XeLaTeX 第一是文档是 UTF-8 编码的，第二是它对各种字体多语言输出文章的解决方案是最完美的，第三是 LaTeX 里面能够用的命令它一般都能用，第四是编译生成 pdf 文件使用的命令是 xelatex 什么什么 tex 文件，第五是需要知道它内置引擎现在一般是 xdvipdfmx。

## 基本情况说明

这一段内容参考了有名的不太短的 LaTeX 手册。[lshort-cn](#)

在 LaTeX 的代码中最重要的是理解各种各样的命令的功能，正是这些各种各样的命令让你输入的文字显得与众不同。比如说我现在在打很长的一段文字，LaTeX 会自动换行的，而我在这里按下 Enter 键，实际上并没有换行的效果。理解这一点很重要，LaTeX 不同于微软的 word 软件或者其他 openoffice 之类，不是采用的所见即所得模式，我在这里打的是奇奇怪怪的东西，但是最后显示出来的却可能是很美观的东西。LaTeX 的一个设计理念就是所想即所得，它甚至有点偏执地要求你组织好你自己的文章的结构，而这正是 LaTeX 的爱好者所推崇的。

同样在代码中你空一个格或者空很多个格都是没有区别的，都是一个空格。

在 LaTeX 中空一行和空很多行的效果是一样的，都是空一行，表示另起一段。

LaTeX 的命令用到了一些特殊的符号，所以你就不能按照常规用到它们了，这些符号如下：

# \$ % ^ \_ & { } ~ \

更详细的说明请参见后面的特殊符号[2.8](#)

$\LaTeX$  的命令是 **case sensitive** 的。也就是命令是区分大小写的。

现在我把最基本的代码说明一下， $\LaTeX$  的代码的通用格式是这样的，\ 开头，然后跟上命令符号，然后跟上 []，方括号中放的是该命令的可选参数，然后跟上 {}, 花括号里面跟的是该命令的必填参数。具体如下：

```
\command [optional parameters]{parameters}
```

前面第一行代码是：

`\documentclass [12pt,oneside]{book}`, 意思是描述文章模板的类型为 **book**, 也就是一本书, 除此之外还有 **article**, **report**, **slide** 类型等, 更详细的讨论参见 `documentclass` 说明[2.4](#)

然后我们看到第二行代码：

```
\usepackage{什么宏包}
```

这个 `usepackage` 命令后面跟上你想加载的库文件, 等你使用  $\LaTeX$  久了, 就会接触到更多的宏包的。

后面的代码：

```
\begin{document}
```

文章内容

```
\end{document}
```

描述文档开始, 文档结束。在文档结束命令之后, 你写的任何东西都会被  $\LaTeX$  忽略掉。文档环境里面就写着你的文章的内容。

## 查看宏包帮助文档

这个我先讲了，实际上沉下心来阅读文档是最好的学习  $\text{\LaTeX}$  的方法了。比如我要查看 `xeCJK` 文档，就在终端中输入：

```
texdoc xecjk
```

在 `texmaker` 的帮助菜单下面有个功能类似的小插件。

## 从 documentclass 说起

文档刚开始是 `preamble` 区域，放着文档的一些配置，从 `\begin{document}` 开始进入正文区，出了 `\end{document}` 这句话之后后面写的什么程序都不管了。`document` 是一个环境，后面我们会接触很多的环境的。

`documentclass` 命令的必选参量有 `article`, `report`, `book`, `slides`, `beamer` 等，一般了解这几个就够用了。他们之间有很多细微的差别，这个后面慢慢了解。

分节命令带星号表示该分节不进入目录，也不编号。

文档的章节分级结构如下：

```
\part {partname}
\chapter {chaptername}
\section {sectionname}
\subsection {subsectionname}
\subsubsection {subsubsectionname}
\paragraph {paragraphname}
\subparagraph {subparagraphname}
```

一般 `paragraph` 和 `subparagraph` 分节不怎么使用，就在文档中一行一行空出来即可。还有 `subsubsection` 这个分节也不常使用，因为 `section` 之下有 `subsection` 已经很好地满足了思想的分级结构，再加上一个 `subsubsection` 只会让人们更加困惑罢了。<sup>(1)</sup>

所以结合前面 `book`，`article` 分类我在这里为了简单起见约定如下：文档中一个小段落就是 `subparagraph` 不需要用命令再标识一次，几个段落构成一个 `paragraph`，这从原则上就是某一个问题的阐述，也就是一个 `section` 级别，当我们对某一个课题反复思考之后，积累的资料越来越多，然后我们发现某几个 `section` 可以合并起来，这样出现了 `section` 和 `subsection` 两个级别。目录只需要显示 `section`，如果是大型文档有 `part` 的时候可以考虑加入少量的 `subsection`，这样目录才不至于过于庞大反而失去了实用性。这所有的 `section` 潜在的一个大的分类是 `chapter`，但是这里不需要写出来，因为这个时候整个文档的级别是 `article`。也就是通常所见的小容量的书小册子，某一个专门课题的讨论就按照 `article` 来处理。如果上升到某一个学科不同课题的讨论，那么上面 `article` 隐藏的 `chapter` 写出来，然后将他们合并为 `book` 类，这个时候这个 `book` 潜藏的最高级别为 `part`。如果是不同学科的合并书籍那么级别就上升到 `part` 了。目录在 `book` 类的时候有 `part` 写上 `part`，然后 `chapter` 和 `section` 都显示出来，结构也不会太复杂的。

当然以上讨论只是泛泛而论，你需要根据自己的实际情况来，但总的原则是自己心里应该有一个划分标准，毕竟一本书最有价值的部分就是目录了，如果一本书的目录结构是乱七八糟的那么这本书不值一看。

## 一般的可选项

10pt, 11pt, 12pt 设置文档所使用字体的大小, 默认是 10pt。

a4paper, letterpaper... 定义纸张的大小, 此外还有 a5paper, b5paper, executivepaper, legalpaper 等。

fleqn 设置该选项将使数学公式左对齐, 而不是中间对齐。

leqno 设置该选项将使数学公式的编号放置于左侧而不是右侧。

titlepage, notitlepage 指定是否在文档标题后开始新一页, article 文档类不开始新页, report 和 book 开始。

onecolumn, twocolumn 指定 LaTeX 以单栏或双栏方式排版文档。

twoside, oneside 指定 LaTeX 排版文档为双面或单面格式, article 和 report 默认为单面, book 默认为双面。

openright, openany 定义 chapter 开始时仅在奇数页或者随意, book 类默认 openright, report 默认 openany, article 没有 chapter。

## 书籍的通用结构

通常一本书是由好几部份构成的, 包括封面、扉页、书名页、目次、序、内文、补充或参考资料、版权页。

出版的书籍的封面和扉页这里我们不考虑。电子书籍就从书名页开始说起。也就是我们的 maketitle 命令。这个时候也可以认为书名页作为了通常意义上的封面。maketitle 可以生成多页, 你可以

考虑把版权页也算在里面，因为出版的书籍那个版权页刚好在背面，而电子书籍我觉得版权页还是放到最后面合适一些，当然多页封面你还可以自己加点名言警句页，这个看自己喜好了。

然后接下来是序言部分，自序或者他序都可以。自己编的电子书籍就是自序了，自序内容不宜过长，相当于论文的摘要部分，用最简短的话让别人对这本书有一个大概的印象。

接下来如果有 listoftables 和 listoffigures 的就放到这里，这个看个人喜好，似乎现在一般都喜欢放在书后面吧。

然后是 tableofcontents, 目录。

然后是正文部分，包括引言，前言等。

然后是 appendix，附录部分：载于一书后面之文字或图表，是用来提示一些与内容有关而不便载于正文里的资料。

然后是参考资料部分。

然后是索引：是针对这本书中重要资料如人名、地名、概念等的查检。将本文的重要概念列出，并注明出现在文中的页次。它是依一定的方法排列，通常中文是按字体的笔划多寡决先后顺序，西文则按字母的顺序排列，以便检查。通常附录是直接资料，索引则是提供查询资料的线索。

## 页码

frontmatter 命令跟在 begin document 后面，接下来页码为罗马数字。

mainmatter 命令放在正文开始的前面，表示页码的阿拉伯数字开始计数。

`appendix` 命令表示附录开始, 后面各章节改为字母标记。页码没有变化

`backmatter` 命令放在参考文献或者索引的前面。章节编号关掉, 页码没有变化。<sup>2</sup>

## 页面布局

页面布局最好用`geometry`宏包调节。

### geometry 宏包详细讨论

页面布局尺寸由 `geometry` 宏包指定, 页面布局包含很多参量也就是 `geometry` 的可选项, 请看下图2-1:

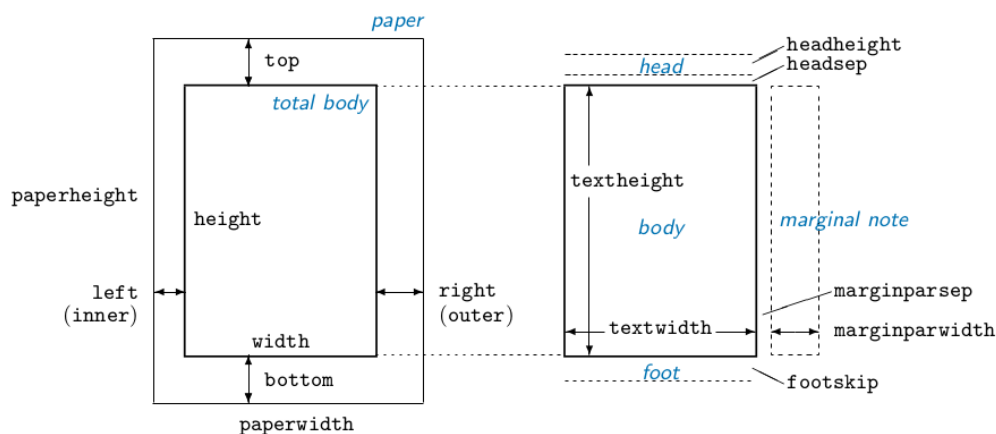


图 2-1: geometry 选项 1

`geometry` 提供的纸张类型很多, 从 `a0paper` 一直到 `a6paper` 都有, 还有 `b1paper` 到 `b6paper` 系列等等。纸张类型指定了后面的 `paperwidth` 和 `paperheight` 就都确定了。

<sup>2</sup>`backmatter` 不能在 `appendix` 前面, 请参考[这个网站](#)。



我们先来看横向参量，`paperwidth` 是纸张的宽度，`textwidth` 是正文宽度，`marginparsep` 指旁注和正文之间的间距，`marginparwidth` 指旁注宽度，`left` 指左边空白宽度，`right` 指右边空白宽度。如果 `book` 类型是 `twoside` 的，那么 `left` 最好命名为 `inner`，也就是类似出版书籍靠里面的那段空白宽度，类似的 `right` 最好命名为 `outer`，靠外面的那段空白宽度。其中默认情况下  $\text{width}=\text{textwidth}$ ，如果加入选项 `includemp=true`，那么： $\text{width}=\text{textwidth}+\text{marginparsep}+\text{marginparwidth}$ 。然后还有： $\text{paperwidth}=\text{left}+\text{width}+\text{right}$ 。

再来看竖向参量，`paperheight` 为纸张高度，`textheight` 是正文高度，`top` 为上面的空白高度，`bottom` 为下面的高度，默认 `top` 包含页眉高度 `headheight` 和页眉于正文见的一段小空白 `headsep`，`bottom` 包含页脚高度 `footskip`，所以你的 `top` 至少要大于 `headheight` 高度。然后中间的区域高度为 `height`。默认情况下  $\text{height}=\text{textheight}$ ，请看下图2-2:

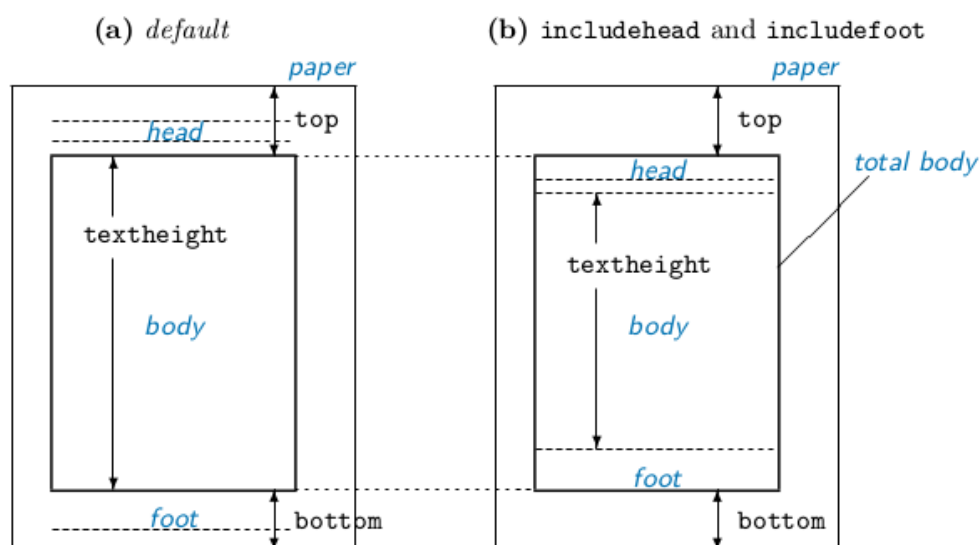


图 2-2: geometry 选项 2

如果加上 `includehead=true` 选项，那么就和上图右边描述的类似，页眉部分计入 `height`，类似的有 `includefoot=true`，那么页脚部分也计入 `height`。

`geometry` 的机制是以上讨论的横向或者竖向参量指定足够的数量之后，剩下的可以自动计算得到。没有明确指定的参量虽然可以通过计算得到，但是在后面似乎是不能够作为变量使用的？

## 临时改变页面布局

临时改变页面布局前面讲的 `geometry` 宏包也有一种实现机制，但是不太好用而且会把后面的段落格式打乱。这里推荐使用 `change page` 宏包。

进入 `adjustwidth` 环境就可以调整旁注宽度了。比如这里我新建一个 `widetext` 环境代码如下：

---

```

1 \newenvironment{widetext}
2     {\begin{adjustwidth}{-70mm}}%marginparwidth+marginparsep
3     {\end{adjustwidth}}

```

---

新建环境请看3.3.2。我们在这里建立了一个 `widetext` 环境，主要是 `adjustwidth` 环境操作，我们看到后面有两个花括号，第一个指左 `margin`，第二个指右 `margin`。注意这里不要和前面 `geometry` 宏包里面设置的 `includemp` 弄混了。<sup>3</sup>这里左 `margin` 就是前面设置的 `left`，右 `margin` 是前面设置的：  
`right+marginparwidth+marginparsep`。所以可以考虑 `geometry` 宏包不要设置 `includemp`，我前面只是为了理解简单才如此设置的，当然这里理解这个概念了就没有什么问题了。

然后正数表示 `margin` 宽度增大，负数表示 `margin` 部分宽度减小。这里设置 `-70mm` 即右边 `marginparwidth+marginparsep` 的宽度。然后自己注意进入这个环境了就不要使用 `marginpar` 或者其他旁注的命令了，这是显而易见的。不设置数值即表示不改变，出了 `adjustwidth` 环境，一切复原。

还有一个 `adjustwidth*` 环境，意思表示 `margin` 改变随着页面奇偶数变化而定，这个宏包的页面奇偶数设定及相关讨论后面都略过了，因为本文只关注于 `oneside` 模式，毕竟电子书籍设置成为 `oneside` 更好一些。

`changepage` 宏包还有 `changetext` 和 `changepage` 命令，有兴趣的可以看下，感觉并不太好用。

---

<sup>3</sup>`geometry` 宏包里面设置 `includemp=true` 让 `marginpar` 部分进入 `width`，也就是 `right margin` 并不包含旁注部分了，但是只适用于 `geometry` 宏包。

## 多栏环境

多栏环境推荐使用`multicol`宏包。<sup>(2)</sup> 这个宏包很厉害，支持两栏到十栏的环境。就作为通用的一般形式如下：

在这里每一栏的宽度表示为 `linewidth`，这个可以用来控制放进去的图片宽度。这里用 `columnbreak` 命令手动调整栏的跳转。你也可以不用 `columnbreak` 命令，而让  $\text{\TeX}$  自动计算栏的高度和分布等。不过似乎用 `columnbreak` 只能近似控制，并不是那种完全严格的跳转命令。这是最基本的应用，请参看多张图片并列显示这一小节。[4.4](#)

如果你希望整个文档都分为两栏，那么在前面 `documentclass` 命令可选项里面加上 `twocolumn` 即可。这里的分栏环境是分两栏，然后加入的分栏线。就是在分栏环境中用 `setlength` 调整长度量 `columnseprule` 为 `0.4pt`<sup>4</sup>。觉得这个命令应该重新修改下，直接

`\columnseprule` 就是加分栏线，然后后面跟个可选参数表示线的宽度。

还有一个长度量 `columnsep` 表示栏之间的间距宽度，一样用 `setlength` 调节。一般没啥好调整的。分栏环境就这样简单说下吧。

## 字体

我们知道 `xelatex` 的机制可以调用系统内的任意字体，当然系统没有的字体就要自己安装了，请看[1.16](#)。

<sup>4</sup>参考了[这个网站](#)

## 字体的五种属性

$\text{\LaTeX}$  的字体有五种属性，这五种属性是：字型编码，字族，字型系列，字形，字号，即：`encoding,family,series,shape and size`。

### 字型编码

字型编码即各个个别的字在一个字型里头的排列顺序以及安排方式。原始的  $\text{\TeX}$  字型编码我们就称为 OT1 (Old  $\text{\TeX}$  text encoding)，这是预设的，如果都不指定字型编码，那所使用的就是 OT1 编码。在目前新一代的字型编码里头，字的安排方式及内容和 OT1 不一样，例如 T1 等。

### 字族

字族分为三大类，`roman or serif (rm)`，`sans serif (sf)` 和 `monospace`。(tt) <sup>(3)</sup>

**serif** Serif 中文译为「有衬线字体」，衬线即是印刷字体在每个笔划起始与终止处，加上短线或三角突起等，以便于快速辨认字符，利于阅读，为印刷专用字体。旧版 Windows 与较旧的 Linux 发行版以此为预设显示字体，而英文新版则改为 Sans-Serif，中文新版则是：当字体大于某一程度时，则将 Serif 的明体或宋体，以 Sans 的黑体取代。

Serif 字体著名的有：Times New Roman、DejaVu Serif、宋体、明体。

**sans-serif** Sans-Serif 中文译为「无衬线字体」，专用于荧幕、简报、艺术字体、展示等，较美观，但不适于长时间阅读。多数

英文语系的作业系统多以此为预设字体，而采用此种字体为预设的中文作业系统，以 Mac 系统最为著名。

Sans-Serif 字体著名的有：Arial、DejaVu Sans、Helvetica、Verdana、楷体、圆体、黑体、ubuntu logo 体。

**monospace Mono** 意思是「单一的」，space 意思是「空间」，中文翻为「等宽字体」。等宽字是打字机时代下的遗产，每个英文字母皆设计为同一宽度，以便于排版；现代为节省不必要空间的浪费，依字母形状分配其宽度，如 **m** 与 **i** 其宽度便不相同，不相信可以拿尺来量看看。**Monospace** 现多用于终端机显示、程序码表示等。

Monospace 字体著名的有：Andale Mono、DejaVu Sans Mono、Courier、AR PL New Sung Mono。

## 字型系列

正常用的是 medium, **m**。粗体是 bold, **b**。然后还有 Bold extended, **bx**。还有 Semi-bold, **sb**。。和 Condensed, **c**

## 字形

字形有正常的 normal, **n**。还有意大利斜体 Italic, **it**。斜体 Slanted, **sl**。和 Small Caps, **sc**。

## 字号

比如说本文设置的就是 11pt。

## 调整五种属性

### 调整字型编码

X<sub>Y</sub>LaTeX 只处理 UTF-8 编码，那个调整字体编码的 `inputenc` 和 `fontenc` 宏包都不要用了。

### 调整字族

有两种方法设置，一种是命令式的，一种是环境式的。`roman font family` 是默认的字族，一般不需要明确设置。

命令式	环境式	描述
<code>\textrm{text...}</code>	<code>{\rmfamily text...}</code>	roman 字族
<code>\textsf{text...}</code>	<code>{\sffamily text...}</code>	sans-serif 字族
<code>\texttt{text...}</code>	<code>{\ttfamily text...}</code>	monospace 字族

表 2.1: 调整字族

### 调整字型系列

默认的 `medium`，一般不需要设置，然后还有一个 `bold`，即字体加粗。

命令式	环境式	描述
<code>\textmd{text...}</code>	<code>{\mdseries text...}</code>	正常字体粗细
<code>\textbf{text...}</code>	<code>{\bfseries text...}</code>	bold 粗体

表 2.2: 调整字型系列

## 调整字形

默认是 upright shape，常用的字形如下：

命令式	环境式	描述
<code>\textup{text...}</code>	<code>{\upshape text...}</code>	正常字形
<code>\textit{text...}</code>	<code>{\itshape text...}</code>	意大利斜体
<code>\textsl{text...}</code>	<code>{\slshape text...}</code>	斜体
<code>\textsc{text...}</code>	<code>{\scshape text...}</code>	small caps

表 2.3: 调整字形

## 调整字号

### 相对字号调整

$\text{\LaTeX}$  里面自带的调整相对字号命令如下：

命令	输出
<code>{\tiny test line}</code>	test line
<code>{\scriptsize test line}</code>	test line
<code>{\footnotesize test line}</code>	test line
<code>{\small test line}</code>	test line
<code>{\normalsize test line}</code>	test line
<code>{\large test line}</code>	test line
<code>{\Large test line}</code>	test line
<code>{\LARGE test line}</code>	test line
<code>{\huge test line}</code>	test line
<code>{\Huge test line}</code>	test line

表 2.4: 调整字体大小命令

然后我们看下图，不同字号下这些命令确切的大小：



size	10pt (default)	11pt option	12pt option
<code>\tiny</code>	5pt	6pt	6pt
<code>\scriptsize</code>	7pt	8pt	8pt
<code>\footnotesize</code>	8pt	9pt	10pt
<code>\small</code>	9pt	10pt	11pt
<code>\normalsize</code>	10pt	11pt	12pt
<code>\large</code>	12pt	12pt	14pt
<code>\Large</code>	14pt	14pt	17pt
<code>\LARGE</code>	17pt	17pt	20pt
<code>\huge</code>	20pt	20pt	25pt
<code>\Huge</code>	25pt	25pt	25pt

表 2.5: 字体具体大小

## 绝对字号调整

上面的命令基本上够用了，并不鼓励使用绝对字号。不过有的时候还是有用的，比如旁注环境需要同时调整字体大小和行距，使用这个命令注意不要对文本中某一小段文字使用，否则会造成行距的不一致。比如这个旁注使用的命令是：

---

```

1 \newcommand{\sidenote}[1]{\marginpar{
2   \fontsize{10pt}{20pt}\selectfont #1}}

```

---

其中 `fontsize` 就是调整字号的命令，第一个参量是字体的大小，第二个参量是行距。然后后面 `\selectfont` 必须写上，就理解为表示对后面的字体进行操作吧。类似的还有 `fontencoding`，`fontfamily`，`fontseries`，`fontshape` 命令，这些本文略过。

## 本文字体配置

本文字体配置代码如下，后面有相关介绍：

---

```

1 \usepackage{xltextra,fontspec,xunicode} % 必备三件套
2 \usepackage[CJKnumber=true]{xeCJK} % 中文环境宏
3 \xeCJKsetup{PunctStyle=plain}
4 \defaultCJKfontfeatures{Scale=1.2} % 中文字应该稍微高于英文字
5 \renewcommand\CJKglue{\hspace 0pt plus 0.12\baselineskip}
6 \setCJKmainfont[BoldFont=Adobe 黑体 Std,ItalicFont=Adobe 楷体 Std]
7   {Adobe 宋体 Std}% 影响 rmfamily 字体
8 \setCJKsansfont{Adobe 黑体 Std}% 影响 sffamily 字体
9 \setCJKmonofont{Adobe 楷体 Std}% 影响 ttfamily 字体
10 % 设置英文字体
11 \setmainfont[Mapping=tex-text]{DejaVu Serif}
12 \setsansfont[Mapping=tex-text]{DejaVu Sans}
13 \setmonofont[Mapping=tex-text]{DejaVu Sans Mono}

```

---

上面代码第一行就是 `xelatex` 必备的三件套，其中 `xltextra` 宏包是专门处理  $\text{\LaTeX}$  的一些问题的，它会自动加载后面的 `fontspec` 和 `xunicode` 宏包。`xunicode` 是处理某些特殊字符的。上面代码的具体解释详见下面的 `fontspec` 宏包和 `xeCJK` 宏包详解。

### fontspec 宏包详解

当然来自 `fontspec` 宏包的帮助文档。后面关于宏包的相关信息来源于自身帮助文档我就不说明了。`fontspec` 宏包主要用于英文字

体的设置，中文字体的设置建议用 `xeCJK` 宏包来管理。`XYLATEX` 只能使用 `opentype` 和 `truetype` 字体。

## 基本的命令

`fontspec` 宏包最基本的应用就是用 `setmainfont` 来设置文档的默认 `roman` 字族字体的，`setmainfont` 原来的名字叫 `setromanfont`。<sup>5</sup>`setsansfonts` 是设置文档默认 `sans-serif` 字族字体的，`setmonofont` 是设置 `monospace` 字族字体的。然后我们看到前面都有一个可选项 `Mapping=tex-text`，这个说是让 `XYLATEX` 文字分布更好的，可能和正确断行有关系，不大确切。

`fontspec` 宏包有一个和这个宏包名字一样的命令，这个命令非常的基本，大约相当于引擎的入门口，我估计前面三个命令实际上是建构在 `fontspec` 命令之上的。`fontspec` 命令的作用不光临时改变字体。还可以加上很多可选项，比如字体尺寸，颜色等等。总之 `fontspec` 命令的优先级要高于前面的三大默认字体设置命令。请看下面的例子：

---

```
1 {\fontspec[Scale=4,Color=magenta]{Comic Sans MS} this is a test.}
```

---

this is a test.

另外还有一个有用的命令就是 `newfontfamily`，这个命令相当于把 `fontspec` 命令包起来再新建一个命令。新建的那些字体命令就像 `\sffamily` 之类的命令一样使用。，在后面讲到引入特殊符号的

---

<sup>5</sup>参考了这个网站

时候我们还会接触到这个命令。

```
\newfontfamily{\ubuntu}[Scale=3]{Ubuntu}
```

以上基本命令总结如下：

---

```
1\fontspec [ font features ] { font name }  
2\setmainfont [ font features ] { font name }  
3\setsansfont [ font features ] { font name }  
4\setmonofont [ font features ] { font name }  
5\newfontfamily{ cmd }[ font features ] { font name }
```

---

上面的 font name 在安装字体的时候说明清楚了，比如说用 fc-list 命令调出来的『宋体,SimSun:style=Regular』中字体名字就是宋体或者 SimSun。而在 fontmanager 里面比如第一行显示『Comic Sans MS Regular』，字体名就是 Comic Sans Ms。接下来主要讨论 font features 的内容，所讨论的内容以上几个基本命令都适用。由于 xeCJK 的 fontfeatures 可选项是继承自 fontspec，所以下面的讨论也适用于 xeCJK 宏包。

## font features 的讨论

### 字形

一个字体的字族之下还分为不同的字形，默认的字形设置可能并不能满足你的要求。有一些字体甚至没有粗体或者意大利体这样的字形。一般的玩家就选用默认字形设置足有了，最多在 mainfont 哪里设置下 boldfont 和 italicfont。见上面例子的第 4, 5 行。

**BoldFont** = font name

**ItalicFont** = font name

**BoldItalicFont** = font name

**SlantedFont** = font name

**BoldSlantedFont** = font name

`SmallCapsFont = font name`

## Color

前面的例子我们看到了 `Color` 属性的定制，在这里推荐适用 `xcolor` 宏包。 $\text{\LaTeX}$  默认使用的是 `xdvipdfm`，不支持透明颜色。然后 `xcolor` 宏包对于颜色的讨论请参看颜色一节[2.10](#)

## 字体大小

(4) 在前面的那个例子也用到了 `Scale` 选项，选个数字来整体调整这个字体的尺寸大小。还有两个常量值，一个是 `MatchLowercase`，一个是 `MatchUppercase`。就是变成小写字母一样的或者大写字母一样的大小。

## 词间距——WordSpace

觉得默认的设置更有弹性吧，一般的玩家没必要动它。

## 标点后间距——PunctuationSpace

从零开始可以加点距离。

## 断字符号——HyphenChar

就是要换行了选择从哪里断字的符号，比如设置 `HyphenChar = +`，那么标明 `+`，就从哪里断字。默认的是 `\-`，对英文的断字还不怎么关心。

## 字母之间的距离——LetterSpace

从零开始加点距离，而且只适用于小写字母，感觉很累赘。而且这个 feature 只适用于  $\text{\XeTeX}$ 。

## 最后两个命令

最后还有两个命令 `defaultfontfeatures` 和 `addfontfeatures`。有什么优先级：

`addfontfeatures>fontspec>defaultfontfeatures`。觉得太花哨了暂时应该用不到。

## 设置数学字体

本文不深究数学领域排版知识的讨论。

---

```

1 \setmathrm [ font features ] { font name }
2 \setmathsf [ font features ] { font name }
3 \setmathtt [ font features ] { font name }
4 \setboldmathrm [ font features ] { font name }

```

---

## xeCJK 宏包详解

`xeCJK` 宏包只处理 CJK 字符在这里指中文字，也就是英文字还是用前面的 `fontspec` 宏包来处理。

## 只能在加载宏包时填入的选项

`CJKnumber = <true | false>`

默认是 `false`，如果设置为 `true`，那么可以适用 `CJKnumber` 命令。比如这个命令 `\CJKnumber{1}` 的输出结果是：一。还有一个命令比如 `\CJKdigits{1545}` 的输出结果是：一五四五。

`indentfirst = <true | false>`

章节下面第一段首行缩进不缩进，默认是 `true`，缩进。这个没啥好修改的，一般都统一缩进吧，如果有某个段落你不想缩进加上 `noindent` 命令就是了。

## 可以在 `xeCJKsetup` 命令中设置的选项

`CJKmath = <true | false>`

默认是 `false`，如果设置为 `true`，那么可以在数学模式下输入 CJK 字符。

`PunctStyle = {quanjiao|·····|plain}`

其他选项请参看文档，这个是设置标点处理格式的。在本文字体配置第 3 行那里你可以看到我设置为 `plain`，也就是是什么就是什么，不做处理。默认是 `quanjiao`。

## xeCJK 宏包的一些命令

现在对照 `fontspec` 宏包对 `xeCJK` 新建的只针对 CJK 字符处理的命令说明如下，其中各个命令的用法和 `fontfeatures` 都是类似的。

**`setCJKmainfont`** 类似于 `setmainfont`。



**setCJKsansfont** 类似于 `setsansfont`。

**setCJKmonofont** 类似于 `setmonofont`。

**CJKfontspec** 类似于 `fontspec`。

**newCJKfontfamily** 类似于 `newfontfamily`。

**defaultCJKfontfeatures** 类似于 `defaultfontfeatures`。

**addCJKfontfeatures** 类似于 `addfontfeatures`。

**setCJKmathfont** 估计应该类似于 `setmathrm`。

## 设置 CJK 字符大小

在本文字体配置中第 4 行，我将 CJK 字体都放大了 1.2 倍，为的是让中文字比英文字稍微大一点，这样更好看一点。具体原来为 11pt，放大后为 13.1pt。

## UTF-8 编码

`texmaker` 软件并没有这个问题，不管怎么样，加上这样一行代码没什么害处喽。确保文档以 `utf-8` 编码打开和保存。在文档开头加上如下代码：

```
% !Mode:: "TeX:UTF-8"
```

本文只关注于 UTF-8 编码。

## 特殊的符号

### 去掉符号命令后面的空格

值得一提的是命令如果后面跟上一个花括号，后面的字符紧跟花括号，那么命令显示的符号和后面的字符就没有空格了。比如这里 `&a` 之间就有一个空格，加上花括号`\&{ }a`，`&a` 就没有了。

### 基本的特殊符号

\

\, 我们知道命令的开头表示就用它，所以在文档中是不能直接使用的，如果需要显示 \, 需要输入 `\textbackslash` 来显示。

{和}

{和}, 同上，作为命令的格式。如要显示前面加上 \ 符号。

%

%, 我们知道这个符号在文档是用来标记注释信息的开始的，所以在文档也不能直接使用。在前面加个 \ 即可。

~

~; 这个符号在文档中产生一格空白，如要显示在前面加上 \ 符号。这样显示的波浪号有点小。还可以进入数学模式下输入 `\sim`,

在 `texmaker` 左边的关系符号一栏中也可以找到。所以为了美观的话就用数学模式吧。然后在 `Rime` 输入法里面我们看到还有一种符号是全角下的波浪号 $\sim$ 。我比较了下这个和前面两个都不相同，简单起见用这个全角的波浪号也是可以的。

\$

\$, 美元符号之所以不可以用是因为它标记了数学模式的开始和结束。比如说我要输入字母  $\Pi$ , 就是在两个美元符号中间输入 `\pi` 即可。理论上输入 `\pi` 就可以显示  $\Pi$  了, 但是和中文字混合处理的时候会出错, 而英文字混合输入会引起字体不协调的问题, 而进入数学模式之后就不会。同样还有一种希腊字母的符号  $\pi$ , 这个可以正常显示, 只是没有前面数学模式下的美观。我查了一下, 数学模式下的那个  $\Pi$  也是 `unicode` 区域里面的那个希腊字母, 我想只是字体问题了。<sup>6</sup>

#

#, 这个符号没怎么接触, `latex123` 说是定义巨集的, 前面加上`\`即可。

^ 和 \_

和`_`, 这两个符号表示进入数学模式之后进入上标和进入下标。这两个符号在文本中后面还要跟一对花括号, 否则显示会出问题。

---

<sup>6</sup>[这个网站](#)用来查找 `unicode` 和 `LaTeX` 命令的对应关系

&

& , 这个符号表示表格中的分隔符。要显示前面加上 \ 即可。

## 单引号和双引号

你可以用 `\textquotedblleft` 来表示左双引号，用：  
`\textquotedblright` 来表示右双引号。用 `\textquoteleft` 来表示左单引号，用 `\textquoteright` 来表示右单引号。

然后还有敲键盘的方法，要左边和右边区别对待，比如说左单引号就是点击 **ESC** 下面那个键，右单引号是点击分号右边那个键；而左双引号是点两次 **ESC** 下面那个键，右双引号是按住 **shift**，然后点击分号右边那个键。

这两个符号倒不怎么用到，因为中文的输入法调试好会很方便的，至于英文左单引号和右单引号好像都没区分。

## 破折号和连字号

连字号就是一个这个 - ，可以直接在键盘上输入。  
短破折号就是两个 - 连续输入  $\Rightarrow$  -  
长破折号就是三个 - 连续输入  $\Rightarrow$  —  
负号就是数学模式下的 -  $\Rightarrow$  -1 如果有时嫌麻烦就直接 -1 应该可以接受把。

## 温度的度

在 `texmaker` 左边的关系符号哪里我们可以看到一个小圆圈，输入命令是 `\circ`，当然需要在数学模式下。现在就需要让这个圆圈位于上标即可。前面的符号介绍我们提及 `^`，就是进入上标显示，上标的内容在 `^` 后面的那个花括号内，于是我们就知道三十度怎么画了。30°，注意 `texmaker` 左边有快捷键。点起来很方便的。具体代码是：`$ 30^{\circ}`。当然你可以直接用 `rime` 输入法打出这个温度的度：°，℃。在输出% 候选项哪里。

最后八一句，不明白全角形式的百分号怎么在 `LATEX` 文档里面也是注释作用？我确认是全角的 `unicode`。

## 省略号

this... that...

三个点... `\ldots` 命令...

老实说没看出什么区别，所以省略号就跟着点三个点也是可以的，别用中文句号就行了。

## 更多特殊符号




上面是 `Ubuntu` 的图标符号，可不是一张图片哦，是一个字

符。这个图标符号只在 Ubuntu 字体里面才有。现在让我们将思路先理清一下。首先是 Unicode 码，这个只是一个理论上的编码规则，具体的实现是字体。但是每一个字体都只专注于某一个领域，并没有把 Unicode 所有的码都画出字形来，那么系统是如何显示字体的呢？系统是安装了很多字体，如果一个字体并不包含它要显示的 Unicode，它就搜索打开下一个字体文件，找相关的 Unicode 的字形。只有从字体文件中具体找了这个 Unicode 的字形，才有办法将其显示出来。<sup>(5)</sup>

我的加入新符号的配置代码如下：


---

```

1 \newfontfamily{\libertine}[Scale=1.5]{Linux Libertine O}
2 \newfontfamily{\ubuntu}[Scale=3]{Ubuntu}
3 \usepackage{newunicodechar}
4 \newunicodechar{@}{\libertine{@}}
5 \newunicodechar{B}{\libertine{B}}
6 \newunicodechar{C}{\libertine{C}}
7 \newunicodechar{D}{\libertine{D}}
8 \newunicodechar{

值得提醒的是目前系统的文档里面那个 Ubuntu 图标都不能正确显示出来，而我们在 pdf 中却能显示出来。我们来看看代码。(6)


```

首先第 1, 2 行前面以及谈及了，主要是看这个新引入的宏包 `newunicodechar`。好吧，这个宏包就只有一个命令，这个命令就是这个宏包的名字。其实我们能够猜到，这个命令的作用就是将这个字符变成类似  $\text{\TeX}$  命令的东西，然后替换为后面的那一串，而后面的那一串单独提出来也是能够正常显示的。如 。

所以过程还是很简单的，但是整个过程让人紧张，如果一篇文章有很多这样的特殊符号，那不要写上长长的这样一串？我想到的第一个解决方案就是最好有这么一个字体包含所有的 Unicode 的字形，然后我试着用 `fontforge` 将一些字形复制粘贴调大小混到一起，然后发现这个方案既满足不了美观的要求也满足不了效率的要求。后来认识到也许我多虑了，如果一个符号经常出现，那么那个字体设计者肯定会优先考虑把它加进去的。

这里插播一段广告……

这里有一段代码生成该字体所有已有的字形<sup>4.5</sup>。当然你可以直接在字符映射表里面选择只显示已有字形查看，不过有时并不方便。生成那个 pdf 查看之后可以复制粘贴，到那个字符映射表里面搜索，还是很方便的。

## 初步中文化

主要是些原  $\text{\LaTeX}$  常量是英文单词，然后改成中文字即可。看看代码就清楚了。

---

```

1 \renewcommand\contentsname{目 ~ 录}
2 \newcommand\econtentsname{Contents}
3 \renewcommand\listfigurename{插图目录}
4 \renewcommand\listtablename{表格目录}
5 \renewcommand\bibname{参 ~ 考 ~ 文 ~ 献}
6 \renewcommand\indexname{索 ~ 引}
7 \renewcommand\figurename{图}
8 \renewcommand\tablename{表}
9 \renewcommand\partname{部分}

```

```

10 \renewcommand\appendixname{附录}
11 \renewcommand{\abstractname}{摘 ~ 要}
12 \renewcommand\today{\number\year~ 年 ~\number\month~ 月 ~\number\day~ 日}

```

---

## 颜色

### 本文颜色的配置

---

```

1 \usepackage{xcolor}
2 %===== 在网上找的配黑色文字比较好的背景色 =====%
3 \definecolor{bgcolor-co}{RGB}{255,255,255} % 引用 cite observe
4 \definecolor{bgcolor-dd}{RGB}{255,255,200} % 怀疑 doubt desire
5 \definecolor{bgcolor-tp}{RGB}{215,255,240} % 理论 theory 实践 practice
6 \definecolor{bgcolor-bf}{RGB}{240,218,210} %believe %faith
7
8 \definecolor{defaultbgcolor-0}{RGB}{199,237,204} %for eye
9 \definecolor{defaultbgcolor-1}{RGB}{240,240,240} %gray25 recommend
10
11 \pagecolor{defaultbgcolor-0}

```

---

### 颜色的心理学

颜色心理学是一门学问，这里不会深究，只是在文档里面字体或者背景使用什么颜色是一门大学问。这里主要参照[这个网站](#)简单说下。本文定义的几个颜色也是参考的这个网站。



**红色** 一种激奋的色彩。刺激效果，能使人产生冲动，愤怒，热情，活力的感觉。

**绿色** 介于冷暖两中色彩的中间，显得和睦，宁静，健康，安全的感觉。它和金黄，淡白搭配，可以产生优雅，舒适的气氛。

**橙色** 也是一种激奋的色彩，具有轻快，欢欣，热烈，温馨，时尚的效果。

**黄色** 具有快乐，希望，智慧和轻快的个性，它的明度最高。

**蓝色** 是最具凉爽，清新，专业的色彩。它和白色混合，能体现柔顺，淡雅，浪漫的气氛 (像天空的色彩:)

**白色** 具有洁白，明快，纯真，清洁的感受。

**黑色** 具有深沉，神秘，寂静，悲哀，压抑的感受。

**灰色** 具有中庸，平凡，温和，谦让，中立和高雅的感觉。

关于具体选择什么颜色，我也纠结了很久，但最后无果而终，可选的颜色太多了，我说不上什么意见。

## 有关颜色的基本命令讨论

首先推荐使用 `xcolor` 宏包。<sup>(7)</sup>

**定义新的颜色** 前面谈及本文颜色配置的代码中有很多 `definecolor` 命令就是定义新的颜色的，然后在后面要用到颜色的地方使用你这里定义的新的颜色名字就可以了。第一个花括号就填着你定义的新颜色的名字。第二个花括号填着你要定义的颜色模式，比如 `RGB`, `rgb`, `HTML`, `cmymk` 等。最后就是填着对应的模式的对应的数值。其中 `HTML` 模式不要 `#` 号。

**用软件查看颜色** 你可以用手机照相，或者某个在某个网页某个文档上截图。然后用 Gcolor2 软件来捕捉某个点的颜色。

**改变文章的背景颜色** 上面代码第 11 行 `pagecolor` 命令就是，我试着在 `minipage` 模式下使用也会改变整个文章的背景颜色。

**改变字体的颜色** 有两个命令，`textcolor` 和 `color` 命令。

```
\textcolor{colorname}{some text}  
{\color{colorname} some text...}
```

我更喜欢 `color` 命令。

## xcolor 宏包

`xcolor` 宏包虽然是 `color` 宏包的扩展集，但对于我这个颜色知识盲来说多少有点不知所云，可能专业弄颜色的对那些颜色模式的增加还有颜色混合的表达扩展觉得很感动吧。如果你对颜色有更高的要求，请详细阅读 `xcolor` 宏包文档，这里不赘述了。

就一般用户还是用 `definecolor` 命令吧，支持的模式有 `gray`, `rgb`, `RGB`, `HTML`, `cmyk`。其中只要加载 `xcolor` 宏包就能使用的颜色名字如下：




















颜色	效果	颜色	效果
black		olive	
blue		orange	
brown		pink	
cyan		purple	
darkgray		red	
gray		teal	
green		violet	
lightgray		white	
lime		magenta	
yellow			

表 2.6: 直接可以使用的颜色名字

这个表格里面的小格子涂上颜色是使用的 `xcolor` 宏包里面的 `cellcolor` 命令，就是在表格相应的格子位置使用这个命令就可以了。上面表格类似的一行我写出来吧：

```
yellow & \cellcolor{yellow}\\ \bottomrule
```

虽然颜色混合我弄不大明白，不过单个颜色调百分比<sup>7</sup>还是很有用的。比如 `gray` 灰色后面跟个 `!20`，就表示 20% 的灰，那种淡淡的灰色做背景颜色挺好的。请看下面不同百分比的灰色。

<sup>7</sup>不清楚和谁调？白色？



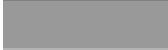
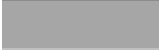
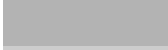
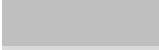
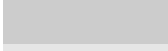
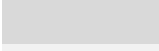
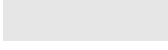
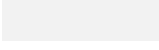
灰色百分比	效果	灰色百分比	效果
gray		gray!90	
gray!80		gray!70	
gray!60		gray!50	
gray!40		gray!30	
gray!20		gray!10	
gray!0			

表 2.7: 不同百分比的灰色

这个表格最后一行是：

```
gray!0 & \cellcolor{gray!0}\\ \bottomrule
```

值得一提的是 `xcolor` 宏包还支持一种表格颜色交替模式，看上去不错。请看带颜色的表格这一小节[2.23.5](#)。

## 段落

### 换行和分段

一个意思写一个段落，意思没说完就不要分段，在文章中一个段落的书写就是直接写就是了， $\text{\LaTeX}$  会自动处理好一切的。简单的分段的做法就是空一行。这样分段首行缩进仍然存在，表示不同的段落。

如果你只是想换行而不分段，那么用命令：

```
\
```

可以满足你的要求。

`\` 命令还有一个用法，比如后面跟上 `[10pt]`，表示在原有行距的

基础上再加上额外的空白距离，参数也可以是负数。

## 断字

如果你编译的时候出现了 `overfull hbox` 什么的错误，一般是断字出现问题了，有字越过文档边界了。命令 `\` 是强制换行。还有一个换行命令 `\linebreak`，这个命令后面还可以跟个可选项，但意义我不大明确。一般就是观察生成的文档，然后选择合适的位置插入 `\linebreak` 命令。这个断行命令断行之后，前面那行的文字将会扩展充满整行，在越界小距离的情况下还是可行的，但是距离大了这一行扩展导致字间距过大也不是十分美好。

你可以考虑改变文字，或者直接用 `\newline` 换行，主要是英文的断字问题，还有一个命令是手动调整英文单词断字的，这里先略过。

## 段落中的行距<sup>(8)</sup>

```
\setlength{\baselineskip}{22pt}
```

上面的代码就是设置行间距的。一行一行之间的间距也是一个 `glue`，我们知道 `glue` 有基本的 `space` 和伸缩量。行距的基本 `space` 由命令 `\baselineskip` 控制，伸缩量有 `\baselinestretch` 命令。<sup>8</sup>

本文目前控制段落行距模板采用的方案是使用宏包 `setspace`。因为这个宏包使用双倍行距选项之后尾注部分行距也有调整，更加好看些：

```
\RequirePackage[doublespacing]{setspace}
```

<sup>8</sup>实际上这个多少有点揣测的意思。

## baselineskip

段落中行之间上下间距<sup>9</sup>由三个命令控制：`baselineskip`，`lineskip` 和 `lineskiplimit`。简单的说明就是首先间距是 `baselineskip`，但是如果上面的盒子伸的太下或者下面的盒子伸的太高，那么他们就可能会碰到一起。`lineskiplimit` 控制的的就是盒子之间最小间距，比如 `0pt`。当 `baselineskip` 减去上面的盒子的深度 `depth` 再减去下面盒子的高度 `height` 然后得到的值比 `lineskiplimit` 小，那么跳转方案就会选择 `lineskip` 模式。也就是上面盒子最低的点和下面盒子最高的点之间的距离是 `lineskip` 那么多。<sup>(9)</sup>

我感觉自己设置一个 `baselineskip` 已经很满足要求了，尤其个别字用 `Huge` 命令的时候效果也还行。

## baselinestretch

<sup>10</sup> `baselinestretch` 量相当于行间距 `glue` 的伸缩量，也就是对前面的 `baselineskip` 做一定的伸缩。这个命令要使用的话格式如下：

```
\renewcommand{\baselinestretch}{伸缩量}
```

这个命令等价于：

```
\linespread{伸缩量}
```

不过这不是故事的全部。完整的设置格式如下：

```
{\linespread{伸缩量}\selectfont sometext}
```

其中伸缩量人们说设为 1.3 就是 1.5 倍行距，1.6 就是双倍行距。放在导言区里面才有效果，不能在文档中临时改变行间距。还是推荐使用 `setspace` 宏包，除非 `setspace` 宏包不知怎么失效了。

---

<sup>9</sup>一行一行就是横向的盒子，行间距就是横向的盒子之间的上下间距。

<sup>10</sup>参看的[这个网站](#)

## 段落间距

```
\setlength{\parskip}{1.6ex plus 0.2ex minus 0.2ex}
```

段落间距也就是一段和一段之间的空白距离。上面就是本文段落间距的设置代码。`\parskip` 是一个 `length` 量。其中距离设置设置一个固定量和一个加量还有一个减量，`wikibook` 中说很有用，不太清楚。

## 段首缩进

### 段首缩进量调整

---

```
1 \setlength{\parindent}{
2 {\baselineskip * \real{0.06} + \textpt * \real{2.4}}}
```

---

上面代码就是设置段首缩进量的，其中 `\parindent` 是一个 `length` 量。具体设置我用了一个算法，需要加载 `calc` 宏包<sup>11</sup>。也就是希望设置的距离更加相对化。其中 `\textpt` 是我新建的一个长度量：

```
\newlength{\textpt}
```

```
\setlength{\textpt}{12pt}
```

本文的默认字体大小值是 `11pt`，然后传递到了 `\textpt` 这里。这里 `\textpt` 乘以 `2.4` 的意思是之前我放大 `CJK` 字符 `1.2` 倍了，再乘以 `2` 相当于乘以 `2.4` 即两个字的距离。然后我在字体设置哪里中文字之间间距为 `0.12` 个 `\baselineskip`。由于 `glue` 有点波动，所以在这设置 `0.06` 个 `\baselineskip`。因为我设置为 `0.18` 的时候距离过大了，总之大概在 `30pt` 左右，在这里用公式表示就是为了距离表达更加相对

---

<sup>11</sup>注意乘法乘以小数的时候需要用 `real` 命令处理。

化。

## 缩进还是不缩进

在 `xeCJK` 宏包加载的时候默认章节下面第一段缩进打开了，然后后面基本上都段段有缩进，也没怎么管了。如果你希望某一段不缩进就用 `noindent` 命令吧，类似的有 `indent` 命令。如果你希望都不缩进，我觉得简单点的办法就是把缩进量设置为 0 吧。

## 段落对齐

`flushleft` 环境为左对齐环境，`flushright` 环境为右对齐环境，`center` 为居中环境。类似的命令样式有 `raggedleft`，`raggedright` 和 `centering`。这些命令还可以控制表格图片 (也许是一切盒子?) 的位置。简单示例如下：

---

```
1 \begin{flushright}
2 \fbox{右边才是王道。}
3 \end{flushright}
```

---

右边才是王道。

## 页眉页脚设计

页眉页脚设计推荐用 `fancyhdr` 宏包。



## 观察

我什么都没设置，生成的文档有章名的那一页中间有页码，其他页右上角有页码。有章名的那一节可能样式是 `plain`，其他页可能是默认的 `myheadings` 样式。

## 全部信息归零

### plain 样式重置

`fancyhdr` 宏包提供了一个命令 `fancypagestyle` 来重新或者定义一个新的页眉页脚样式。现在我将 `plain` 样式所有信息全部清除，宁愿没有也不愿出现其他别样的信息。

---

```
1 \fancypagestyle{plain}{
2   \fancyhf{}
3   \renewcommand{\headrulewidth}{0pt}
4   \renewcommand{\footrulewidth}{0pt}}
```

---

该代码第一行是提出要重新定义 `plain` 样式，然后里面包含新的定义。第二行是所有的页眉页脚信息换为空值，第三行是将页眉那条线宽度设为 `0pt`，也就是不显示了，第四行类似是页脚那根线不显示了。

### 选择默认样式为 `empty`

使用命令：

```
\pagestyle{empty}
```

也就是明确指定页眉页脚样式为已经有的样式 `empty`，即什么都没有。现在文章所有的页眉页脚信息都被清除了。现在摆在我们面前有两条路，一条是继续 DIY `plain` 样式，然后全部页面设置为 `plain` 样式。一条是 `plain` 继续归零，然后自己 DIY 一个新的样式并使用这个样式。我在这里选择第一条道路了。

### 继续定制 `plain` 样式

我不太喜欢那一条横线，有需要的请自己将前面的什么 `rulewidht` 命令横线宽度设为 `0.4pt` 左右。

我是一个喜欢简单的人，`fancyhdr` 宏包里面还有很多命令：

---

```

1 \head[<even output>]{<odd output>}
2 \chead[<even output>]{<odd output>}
3 \rhead[<even output>]{<odd output>}
4 \lfoot[<even output>]{<odd output>}
5 \cfoot[<even output>]{<odd output>}
6 \rfoot[<even output>]{<odd output>}
7 \fancyhead[selectors]{output you want}
8 \fancyfoot[selectors]{output you want}

```

---

这些命令都可以用 `fancyhf` 命令来达到，所以我不做介绍了，有需要的请参考 `fancyhdr` 宏包文档。

## fancyhf 命令的可选项

字母	意义
H	页眉 (head)
F	页脚 (foot)
L	左边 (left)
C	中间 (center)
R	右边 (right)
E	偶数页 (even)
O	奇数页 (odd)

表 2.8: fancyhf 可选项字母意义

比如偶数页的页眉左边就是 OHL，本文没有区分偶数页和奇数页，而且也不想有过多的信息，比如就想页眉左边和页脚右边有点内容，那么可选项为 HL 和 FR。

## fancyhf 必填项定制

必填项就是在那个位置要输出的内容，可以是一段简单的文字比如书名。而 FR 也就是右边页脚处哪里简单填上 \thepage 即表示当前页码。具体格式不用设置，请参看页码这一小节[2.4.3](#)。

显然直接写上书名没多大意思，现在决定定制如下，页眉左边用 ttfamily 字体，字体大小是 footnotesize，左边写上章名，右边写上节名。

## 本文页眉页脚配置代码

```

1 \RequirePackage{fancyhdr} % 页眉页脚
2 \pagestyle{fancy}
3 \fancypagestyle{plain}{
4     \fancyhf{}
5     \renewcommand{\headrulewidth}{0pt}
6     \renewcommand{\footrulewidth}{0pt}
7     \renewcommand{\chaptermark}[1]
8         {\markboth{第 \CJKnumber{\arabic{chapter}} 章 ~~#1}{} }
9     \renewcommand{\sectionmark}[1]
10        {\markright{第 \CJKnumber{\arabic{section}} 节 ~~#1}{} }
11 %     \fancyhf[HL]{\ttfamily \footnotesize \leftmark }
12     \fancyhf[HR]{\ttfamily \footnotesize \rightmark }
13     \fancyhf[FR]{\thepage}
14     \fancyhfoffset[R]{\marginparwidth+\marginparsep}
15 }
16 \pagestyle{plain}

```

---

### 一些基础情况说明

请看到第 11 行，设置的是页眉左边的格式，然后设置为 `ttfamily` 字族和 `footnotesize`，这个大家都清楚。然后后面是一个命令 `leftmark`，要使用这个命令，前面必须先加上 `fancy` 样式。<sup>12</sup>

`leftmark` 就是目前的章名，`rightmark` 就是目前的节名。而重新定义则需要通过 `chaptermark` 和 `sectionmark`，不太清楚为什么。我们再看上面对 `chaptermark` 和 `sectionmark` 的重定义。其中 `chaptermark` 影响 `leftmark`，他的参数就是具体的章名。而他使用

---

<sup>12</sup>似乎可以理解为这个命令继承自 `fancy` 样式。

的是 `markboth`，`sectionmark` 使用的是 `markright`，具体意义也不大清楚。

我起初试图如下重新定义 `chaptername` 和 `sectionname`，然后重新定义 `thechapter` 和 `thesection` 也就是当前章节编号。在这里先将编号用 `arabic` 命令转化为 1,2,3... 的形式，然后用 `CJKnumber` 命令转化为一, 二, 三... 的形式。可是目录形式也跟着改变了，还有图片编号。影响范围太大了，所以就直接用上面代码的形式组合了。

---

```

1 \renewcommand{\thechapter}{\CJKnumber{\arabic{chapter}}}
2 \renewcommand{\thesection}{\CJKnumber{\arabic{section}}}
3 \renewcommand{\chaptername}{第 \thechapter 章}
4 \renewcommand{\sectionname}{第 \thesection 节}

```

---

## 页眉页脚宽度

如果你文章旁注比较宽，你会注意到下面的页码并不是在最右边。如果你希望页码移到最右边可以通过命令 `fancyhoffset` 来调节。同时相应页眉页脚那条线也会加长。代码类似于：

```
\fancyhoffset[R]{\marginparwidth+\marginparsep}
```

## 章节标题设计

推荐使用 `titlesec` 宏包进行章节标题设计，当然还有其他的宏包可以设计出更加花哨的章节标题这里忽略。

## 本文章节标题设计

---

```

1 \usepackage{titlesec}
2 \titleformat{\part}{\huge\sffamily}{}{0em}{}
3 \titleformat{\chapter}{\LARGE\sffamily}{}{0em}{}
4 \titleformat{\section}{\Large\sffamily}{}{0em}{}
5 \titleformat{\subsection}{\large\sffamily}{}{0em}{}
6 \titleformat{\subsubsection}{\normalsize\sffamily}{}{0em}{}

```

---

本文章节标题设计非常简单，几乎就是 `titleformat` 命令各个选项的空值。

### titleformat 命令说明

`titlesec` 宏包还提供了其他一些命令，不过一切设置都可以通过 `titleformat` 命令来获得：

---

```

1 \titleformat{\chapter}[shape]{格式}{label}{sep}{before-code}[after-code]

```

---

第一个花括号是选择你要修改的目标，也就是是章啊，还是节啊，还是小节。从 `part` 到 `subparagraph` 都可以的。

后面是 `shape`，一个可选项，没看懂要干嘛的。估计也不太重要吧。

第二个花括号是重点，里面放着格式命令，比如设置字体字族，字体大小，颜色等都可以的。这个格式影响后面要讲的 `label`

标签还有标题文本。<sup>13</sup>

第三个花括号是标签，空着就是没有标签。你也可以——比如说 `section` 标题——填上 `\thesection` 表示节的编号。

第四个花括号是标签和后面标题文字之间的空隙，这里因为没有 `label` 所以设为 0 了，如果有 `label` 还是加点距离。

第五个花括号和后面的可选项是什么标题盒子之前和之后的代码，这里忽略。有兴趣请参看文档。

## 章节编号数值修改

`part`, `chapter`, `section` 等这些都是 `counter` 量，通过 `setcounter` 命令直接修改——比如说 `section`——为 2，那么接下来 `counter` 自动计数是从 2 开始，下一个 `section` 编号是 3。

## 章节编号深度修改

通过设置 `secnumdepth` 这个 `counter` 的量可以设置章节编号深度。

```
\setcounter{secnumdepth}{1}
```

默认是 2，编号到 `subsection`，你可以设置为 3，`subsubsection` 都有编号，或者设置为 1，`section` 有编号。我想设置为 0 的话 `section` 也没有编号了。

---

<sup>13</sup>这里对齐命令，`vspace` 命令等都是可以用的??

## 章节编号形式修改

`thepart`, `thechapter`, `thesection`, `thesubsection` 等你可以称他们为标签吧，也就是通常我们看到的编号那部分内容。通过对这些命令重定义就可以对他们进行修改了。

```
\renewcommand{\thesection}{\arabic{section}.}
```

那么就会有 1. 的形式。

## 章节标题上插入脚注

一般使用章节标题命令就是 `\section{text}`，前面还是可以加个可选项的，这个可选项的文本将作为目录中实际显示的文本，而后面则是实际执行的文本。比如：

```
\subsection[text]{\color{red} text}\footnote{脚注}}
```

text<sup>14</sup>

## 目录设计

一般在封面之后插入目录，用 `tableofcontents` 命令即可。本文没有对目录做太多的修改，默认的目录格式挺好的。唯一使用的技巧是新建一个 `comman-format` 环境，用这个将除了目录和封面之外所有的其他内容都包括进去。

---

```
1 \newenvironment{common-format}{ %
```

```
2   \setlength{\parskip}{1.6ex plus 0.2ex minus 0.2ex} % 段落间距
```

---

<sup>14</sup>脚注



```

3 \setlength{\parindent}{\baselineskip * \real{0.06} + \textpt * \real{2.4}}
4 {}

```

---

## 目录深度控制

`\setcounter{tocdepth}{1}` `tocdepth` 是一个 counter 量，默认是 2，显示到 subsection。这里为了是目录更加简洁，设置为了 1 即显示到 section。0 显示到 chapter，-1 显示 part，-2 就什么都没有了。

## 前言加入目录

本文定义了两个命令为了做到这点：

---

```

1 \newcommand{\addchtoc}[1]{ % 目录中加入新章节
2   \cleardoublepage
3   \phantomsection
4   \addcontentsline{toc}{chapter}{#1}}
5 \newcommand{\addsectoc}[1]{ % 目录中加入新的 section
6   \phantomsection
7   \addcontentsline{toc}{section}{#1}}

```

---

两个命令大致类似吧，一个是针对 chapter 的，一个是针对 section 的。其中 chapter 的需要加上一个 `cleardoublepage` 命令，其具体解释参看[这个网站](#)。意思是首先结束本页，然后将所有图片和表格都显示出来。在两面 `twoside` 模式里，要确保下一页从奇数页开始，必要时插入空白页。

后面的 `phantomsection` 命令<sup>15</sup>是由 `hyperref` 宏包提供的，需要加上它使链接有效。<sup>(10)</sup>

然后 `addcontentsline` 命令就是将目前章节加入目录，第一个花括号里面的选项有：`toc`，`lof`，`lot`。也就是目录，图目录和表目录。第二个花括号里面如果是 `toc` 的话就是 `part`，`chapter` 之类的，如果是 `lof`，好吧，一般加入 `label` 就可以了，还没接触这种情况，说不上什么。最后那个花括号里面放着要在目录上面显示的文字。

本文目录的前面加入如下命令即可：

```
\addchtoc{目录}
```

同理，参考文献处理情况类似。值得提醒的是 `section` 的时候注意换命令。然后 `part`，`section` 之类的命令带个星号 `*` 表示不编号不进入目录，这个前面说过的。

## 目录行间距拉大

我试图建立一个通用的格式环境横跨目录的时候会失效，只好将其分开。不过这样做带来一个好处，那就是你在目录前面设置一些格式只对目录起作用。比如：

```
\addtolength{\parskip}{8pt}
```

这里对段落之间的间距增加了一点宽度，这个会影响一条条目录之间的行间距。

---

<sup>15</sup>这个命令名字真不好记。。

## 目录数字和标题间距调整

### titletoc 宏包

titletoc 宏包可以自己 DIY 目录格式，这里忽略。

## 封面设计

基础的封面就是用 `title` 输入题目，用 `author` 输入作者，用 `date` 命令输入日期，默认是输出当时编辑的日期的。`author` 里面可以用 `and` 命令连接几个作者或者用 `\\` 命令换行。然后用 `maketitle` 命令插入一个封面即可。

要做出好看的封面大概是很费心思的事情，本文使用了这样的策略，新建了一个模板文件 `mytitle.sty` 文件，然后里面内容如下。

### mytitle.sty

---

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{mytitle}
3
4
5 \def\titlea#1{\gdef\t@titlea{#1}}
6 \def\t@titlea{\@latex@warning@no@line{No \noexpand\titlea given}}
7 \def\titleb#1{\gdef\t@titleb{#1}}
8 \def\t@titleb{\@latex@warning@no@line{No \noexpand\titleb given}}
9 \def\titlec#1{\gdef\t@titlec{#1}}
```

```

10 \def\@titlec{\@latex@warning@no@line{No \noexpand\titlec given}}
11
12 \def\authorinfo#1{\gdef\@authorinfo{#1}}
13 \def\@authorinfo{\@latex@warning@no@line{No \noexpand\authorinfo given}}
14 \def\email#1{\gdef\@email{#1}}
15 \def\@email{\@latex@warning@no@line{No \noexpand\email given}}
16 \def\editorinfo#1{\gdef\@editorinfo{#1 邮箱: \href{mailto: \@email}{\@email}。}}
17 \def\@editorinfo{\@latex@warning@no@line{No \noexpand\editorinfo given}}
18
19 \def\editor#1{\gdef\@editor{#1}}
20 \def\@editor{\@latex@warning@no@line{No \noexpand\editor given}}
21 \def\version#1{\gdef\@version{#1}}
22 \def\@version{\@latex@warning@no@line{No \noexpand\version given}}
23
24
25 % 默认的 title 样式 继续添加的有 mytitlea, mytitleb, mytitlec。。。
26 \newcommand{\mytitle}{
27 \begin{titlepage}
28 \begin{flushleft}
29 \vspace*{\stretch{2}}
30 {\HUGE\bfseries \@titlea}\vspace{\stretch{1}}
31 {\Huge\bfseries \@titleb}\vspace{\stretch{1}}
32 {\LARGE\itshape \@titlec}\vspace{\stretch{2}}
33 {\Large \@author\footnote{\@authorinfo}}\quad\rule{0.8pt}{3ex}\quad
34 {\large \@editor\footnote{\@editorinfo}}\vspace{\stretch{2}}
35 \vfill
36 {\ttfamily 版本: \@version}

```

```

37 \end{flushleft}
38 \end{titlepage}
39 }
40
41 \endinput

```

---

## 一个简单的封面

这个代码牵涉到的内容较多，为了说明基本知识我将使用本文早期的一个简单的封面作为例子讲解。

---

```

1 %===== % 封面設計 ===== %
2 \makeatletter
3 \renewcommand\title[1]{\def\@title{#1}}
4 \renewcommand\author[1]{\def\@author{#1}}
5 \newcommand\email[1]{\def\@email{#1}}
6 \newcommand\version[1]{\def\@version{#1}}
7 \newcommand\editor[1]{\def\@editor{#1}}
8
9 \renewcommand{\maketitle}{
10   \begin{titlepage}
11     \begin{flushleft}
12
13       \vspace*{\stretch{1}}
14       {\Huge\sffamily \@title}\[10pt]
15       {\sffamily\large 作者: \@author}\[
16

```

```

17 \vspace{\stretch{1}}
18 {\sffamily 编者: \@editor}\[\[10pt]
19 {\sffamily 邮箱: \href{mailto: \@email}{\@email}}\[\[
20
21 \vspace{\stretch{1}}
22 {\large\ttfamily 版本號: \@version}\[\[10pt]
23 {\large\ttfamily 完成日期: \today}\[\[
24
25 \end{flushleft}
26 \end{titlepage}
27 }
28 \makeatother

```

---

这个代码一些小的细节我就不说明了，现在就主要内容说明一下。这个封面主要是通过重新定义 `maketitle` 命令来完成的，然后前面还加上了一些新的输入命令。其中 `makeatletter` 和 `makeatother` 一前一后表示他们夹著的内容 `@` 这个符号就是一个符号，这样 ‘`@abc`’ 和 ‘`abc`’ 是不同的。

`def` 命令是  $\text{\TeX}$  的原始定义命令，比如说：

```
\newcommand\email[1]{\def\@email{#1}}
```

就是定义 `@email` 命令，然后这个命令的输出就是输出 `#1`。`#1` 也就是第一个参数。也就是 `email` 命令接受到的参数。

然后封面设计就是进入 `titlepage` 环境进行一些排版操作即可。这里涉及到的居左对齐，`vspace` 等命令就不多说了。值得一提的是好的封面设计需要大量的  $\text{\LaTeX}$  的高级排版知识。

## 引用

### 文章内部引用

某一个特别的章节图片或者表格等需要被引用时，你就在它哪里加上 `label` 命令。然后就可以用 `ref` 命令在文章内部建立链接引用他们了。值得一提的是 `texmaker` 的提示功能非常好。建立 `label` 的时候方便你管理，`section` 部分前面就加一个 `sec:` 前缀，图片加一个 `fig:` 前缀，表格加个 `tab:` 前缀等。然后 `label` 中文英文都是可以的，方便自己管理最好写的很明晰。

比如在这里我插入了一个参考文献的引用，是用的是 `cite` 命令。 `\cite{lshort}` 请参见文献 [1]

还有一个 `pageref` 命令和 `ref` 命令差不多，文档上写法都类似，不同的是在文章上显示的是页码。

### hlabel 命令

`\newcommand{\hlabel}[1]{\phantomsection \label{#1}}` 本文定义了一个 `hlabel` 命令，就是在前面加上了 `hyperref` 宏包提供的精确定位命令 `phantomsection`<sup>16</sup>。一般 `ref` 图片或者 `section` 或者 `subsection` 等文档片段都不需要这么做。就是比如有时一大段 `section` 里面有十几个段落，占了几页的篇幅，那时你如果直接用 `label` 命令建立链接，这几个小段的链接都会挂在 `section` 开头那里。如果使用这个 `hlabel` 命令那么就会精确定位到你想要的位置。

<sup>16</sup>对这个命令具体作用还不太清楚。

## flabel 命令

我在写 `endnotes` 宏包的时候新建了一个命令，还不清楚在外面能不能用。一般为了省心不怎么使用吧。`flabel` 命令有两个必填参数，第一个参数是真实建立的 `label` 标签，第二个参数影响你后面用 `ref` 命令引用这个标签显示的文字。

---

```
1 \DeclareRobustCommand*\flabel}[2]
2 {\phantomsection \def\@currentlabel{#2}
3 \label{#1}}%label styl customization
```

---

## hyperref 宏包简介

### 如何插入超链接

这里插入一个超链接到 `google`，`google`  
 首先要在前面加载 `hyperref` 库文件

```
\usepackage {hyperref}
```

然后在你想要插入超链接的地方使用命令：

```
\href {https://www.google.com/} {google}
```

### 链接字体颜色控制

---

```
1 \usepackage[colorlinks=true,linkcolor=blue,urlcolor=red,citecolor=blue]{hyperref}
```

---

设置书签和目录链接等 for ebook 目录蓝色，对外链接红色，还可以有更多颜色设置 `linkcolor` 影响目录颜色和脚注和内部引用，



`ulrcolor` 影响对外链接, `citecolor` 影响对文献的链接。`anchorcolor` 超链接源码 `hyperref` 宏包目前我的 DIY 只限于一些颜色设置, 有兴趣的请自己研究文档。

## 脚注

加入脚注还是很有用的<sup>17</sup>。具体方法就是:

```
\footnote{这就是一个脚注}
```

## 旁注

本文<sup>18</sup>旁注新建了一个命令 `endnote`, 对字体和行距做了一下调整, 然后使用的是  $\text{\LaTeX}$  自带的 `marginpar` 命令, 我也接触过另外一个宏包的 `marginnote` 命令, 虽然多了一个可选项可以调整旁注竖向位移, 但是带来的麻烦更多, 有时候页尾的旁注会向下越界, 如果我想写了一个旁注, 又在下面继续写一个旁注命令也会出现重叠出错。而 `marginpar` 就没有这样的问题。写了一个 `marginpar`, 后面可以继续再开一条 `marginpar` 命令。至于竖向位移的问题, 在 `endnote` 里面使用 `vspace` 命令即可。

---

```
1 \newcommand{\endnote}[1]{\marginpar{
2   \fontsize{10pt}{20pt}\selectfont #1}}
```

---

在 `maginpar` 命令里面进行各种  $\text{\LaTeX}$  命令都是可以的, 比如改变字体大小, 字体颜色, 插入图片等等。

---

<sup>17</sup>这就是一个脚注

<sup>18</sup>早期版本。。

## 尾注

尾注也就是每一章后面的注释部分，这种排版形式显得更加专业。而且如果尾注写的好参考文献一章或者其他附录部分都可以不用写了。

本来已经有一个 `endnotes` 宏包了，但是这个宏包不支持超链接，然后后来我按照[这个网站](#)的说明操作加入另外一个 `sty` 文件之后是能够正常双向链接的，但是为了解决一个问题引入两个宏包，而且这两个宏包还存在交互问题只是让问题更加复杂了。

我试著慢慢将这两个宏包融合起来，遇到了很多困难，因为我对 `tex` 原初命令并不是十分熟悉，最终似乎解决了这一问题。其中有些细节我也不大清楚，下面就这个新的 `endnotes` 宏包详细说明。

### endnotes 宏包说明

整个宏包还是很简洁的，满打满算八九十行。宏包的文件在 `github` 源码的 `texmf` 文件夹里面。

首先是一些文档的注释说明信息，这里就不写了。

然后是标准的宏包 `sty` 文件的开头样式，这里填上 `endnotes` 宏包即可。

---

```
1 \NeedsTeXFormat{LaTeX2e}[1994/12/01]
2 \ProvidesPackage{endnotes}[2013/11/29 v0.12 fancy endnotes print]
```

---

这里使用 `RequirePackage` 命令引入了一个宏包 `etoolbox`，这里主要是引入了 `expandonce` 命令，这个命令是确保后面的一个命令只运行一次。

然后接下来新建了一个命令 `flabel`，这个命令是为了文本中的尾注标记符号和尾注中的标记符号区分开来，比如文本中的尾注标记符号是上标的，而尾注中的标记符号则不是。`flabel` 命令里面使用了 `hyperref` 宏包的精确定位命令 `phantomsection`，因为 `hyperref` 宏包在 `myconfig` 宏包中已经加载了，这里就不加载了。

`flabel` 命令接受两个参数，第一个参数 `#1` 表示真实的 label，而第二个参数 `#2` 表示显示的 label。所谓真实的 label 指等下生成的 pdf 中的精确定位 label 标记。命令中重新定义了 `\@currentlabel`，也就是显示的 label。这样将这两者区分开来，本来他们都是一样的。

---

```

1 \RequirePackage{etoolbox}
2 \DeclareRobustCommand*\flabel}[2]%
3 {\phantomsection \def\@currentlabel{#2}
4 \label{#1}}%
5
6 \gdef\theenmark{\arabic{@EndnoteCounter}}%
7 \gdef\endnotemark{\textsuperscript{(\theenmark)}}%
8 \gdef\endnotemarkback{(\theenmark)}%

```

---

然后上面三行定义了一些对外的接口命令，方便 DIY。`theenmark` 命令是尾注标记符号的计数器部分，这里使用了 `arabic` 命令转化了。`endnotemark` 命令是文本中尾注标记符号的形式，这里简单加上了 `()` 然后使用 `textsuperscript` 命令将其转化成为上标形式，

之所以使用这样的形式，是因为使用数字可以没有任何限制，而使用其他奇怪的符号标记要为可能超过界限操心。而 `endnotemark-back` 命令则是尾注中的标记符号形式，这里和前面的区别就是没有转化成为上标。

接下来下面的代码首先用 `newwrite` 命令新建了文件，`tex` 中的文件操作命令如下：

---

```
1 \newwrite\tempfile
2 \immediate\openout\tempfile=lists.txt
3 \immediate\write\tempfile{this is test}
4 \immediate\closeout\tempfile
```

---

这里的 `newwrite` 是新建一个写文件，然后后面 `immediate` 命令表示立即执行，`openout` 表示打开某个文件，这里打开了新建的这个写文件 `\tempfile`，后面的等号跟著你想要的这个新文件的文件名，等下电脑中新建出来的这个文件就是这个文件名。

然后 `write` 命令是往这个文件里面写入某个文本信息。接下来是 `closeout` 命令把这个文件关闭，只有文件正常关闭了写的内容才能从缓冲区真正进入系统中的文本。

我们再看下面新建了三个计数器 `@EndnoteCounter` 等，其中 `@EndnoteCounter` 标记每一章尾注内部的计数，而 `AllEndnoteCounter` 标记了所有的尾注的计数，这个影响真实 `label`。`ShowEndnoteCounter` 标记了每一个分章的章数（这里主要指按照每一章，不过实际操作并不局限于此。）的计数。然后用 `setcounter` 命令给这些计数器一个初值。

```

1 \newwrite\@entout
2
3 \newcounter{@EndnoteCounter}%
4 \newcounter{@AllEndnoteCounter}%
5 \newcounter{@ShowEndnoteCounter}%
6 \setcounter{@EndnoteCounter}{0}
7 \setcounter{@AllEndnoteCounter}{0}
8 \setcounter{@ShowEndnoteCounter}{1}
9
10 \newif\if@entopen \global\@entopenfalse

```

---

上一段代码最后还新建了一个条件控制变量，具体请参看 `tex` 条件控制语句一小节4.2.1。这里的 `global` 命令是让接下来的定义或者声明是全局性的。

---

```

1 \def\@openent{%
2 \def\@entname{\jobname
3 \ifnum\value{@ShowEndnoteCounter}<10 0\fi
4 \ifnum\value{@ShowEndnoteCounter}<100 0\fi
5 \arabic{@ShowEndnoteCounter}}%
6
7 \immediate\openout\@entout=\@entname.ent%
8 \global\@entopenttrue
9 }%

```

---

上一段代码新建了一个命令 `@openent`，这个命令等下要完成打开新建的 `ent` 文件任务，然后正如前面谈及的文件操作，在 `openout`

命令打开文件的时候，这里对具体生成的 `ent` 文件的文件名进行了一些 DIY。其实我可以将文件名简单设置成为 `jobname-1`，`-2` 之类的形式，之所以弄成以上这种形式多少是我在之前探索过程中弄出的一种 `jobname001`，`jobname002` 这种形式，然后方便使用 `linux` 系统的 `cat` 命令来将他们汇总，后来修改修改着这个功能发现不需要了，但这个技巧还是很有用的就没删除了。

下一段的解释先跳过，看到下下一段定义的 `endnote` 命令。

---

```

1 \begingroup \catcode '|=0 \catcode '['=1
2 \catcode']=2 \catcode '\{=12 \catcode '\}=12
3 \catcode'\}=12
4 |gdef|@writeent[|immediate|write|@entout[
5 {\footnotesize
6 \hspace{-2\parindent}\makebox[\parindent-3pt][l]
7 {\flabel{|\linknameb}{|\showlinknameb} \ref{|\linkname}}
8 \leftskip=\parindent |expandonce|@include \par } ] ]%
9 |endgroup

```

---

上面这段代码就是定义了那个 `@writeent`，但是以一种十分绕脑袋的形式，我也没完全弄明白。。首先看到 `catcode` 命令，这个是改变 `tex` 系统内部字符列表的命令。在 `tex` 中字符分为以下几类：

0. 跳脱符号，一般是 \
1. group 开始符号，一般是 {
2. group 结束符号，一般是 }
3. 数学符号，一般是 \$

4. 表格分隔符，一般是 &
5. 一行结束符号，一般是 <return>
6. 参数符号，一般是 #
7. 上标符号，一般是 ^
8. 下标符号，一般是 \_
9. 忽略符号，一般是 <null>
10. space，一般是 <space> 和 <tab>
11. 字母，一般是 a-z 和 A-Z
12. 其他，不是其他分类就在此类
13. 激活 (Active) 符号，比如 ~
14. 注释符号，一般是 %
15. 无效 (Invalid) 符号，一般是 <delete>

也就是说 `tex` 引擎看到 `\` 符号就会认为接下来将是一个命令，然后 `catcode` 命令可以改变这个列表。比如上面这段代码中，`\catcode '|=0` 就将 `|` 符号归为 0 类，这样 `tex` 看到 `|` 也会跟看到 `\` 一样处理。后面的类似，于是我们有：

`|` 类似 `\` `[` 类似 `{` `]` 类似 `}` `}` `\` 都是其他

`gdef` 命令是 `global` 和 `def` 命令的组合缩写，这里定义了下面要用到的 `@writeent` 命令。就是立即向打开的那个 `ent` 文件写入花括号内的信息。其中的符号变化法则上面谈及了，于是这里的 `\footnote` 之类的都不会处理，而是直接写入 `ent` 文件之中。但是因为我们在这里要建立 `label`（等下插入 `ent` 文件之后，在尾注那里要

标记 `label` 和 `ref` 命令。) 因此 `label` 和 `ref` 内的参数必须立即运算出来。在这里有一些尾注的格式美化命令，比如用 `makebox` 命令调整尾注标记符号格式等，这里介绍略过。

现在主要看到从下一段代码中定义的 `@include` 命令接受的是 `endnote` 命令传递过来的文本，本来在下一段代码中新建一个 `group` 然后用 `let` 命令将 `endnote` 命令中所有的 `protect` 命令改成了 `string`（字符串），应该是可以了。实际上操作中确实各个命令没有展开直接以文本进入 `ent` 文件了，但是环境命令还是展开了，于是这里加上了 `etoolbox` 宏包提供的 `expandonce` 命令，也就是控制只展开一次，这样环境命令也成功传递过去了。这里的具体实现细节我实际上也没弄太明白。

---

```

1 \DeclareRobustCommand*\endnote[1]{%
2 \stepcounter{@EndnoteCounter}%
3 \stepcounter{@AllEndnoteCounter}%
4 \edef\@linkname{enote:\arabic{@AllEndnoteCounter}}%
5 \edef\@linknameb{enote:\arabic{@AllEndnoteCounter}b}%
6 \edef\@showlinknameb{\endnotemark}%
7 \edef\@showlinkname{\endnotemarkback}%
8 \ignorespaces\flabel{\@linkname}{\@showlinkname}%
9 \ignorespaces\hspace{-0.5ex}\ref{\@linkname b}%
10 \if@entopen \else \@openent \fi%
11 \begingroup%
12 \let\protect\string%
13 \gdef\@include{#1}%
14 \endgroup%
15 \@writeent%
```



显然你看到这里定义的命令 `endnote` 就知道这是这个宏包的主体部分，程序一般遇到的就是两个命令，`endnote` 引入尾注，`showendnotes` 插入尾注。首先这个命令将两个主要计数器加 1，其中 `@EndnoteCounter` 计数器是标记一章尾注内部的计数的，这里加 1 了就开始从 1 开始，然后后面这里的处理技巧是如果你需要插入尾注了，那么就将 `@EndnoteCounter` 这个计数器归为零。这样做和将 `@EndnoteCounter` 和 `chapter` 或者 `part` 绑定起来的好处就是更加的自由，你不需要考虑什么，甚至在某一个 `section` 部分内，插入 `showendnotes` 命令就插入了尾注，然后内部计数又重新开始。

接下来定义了四个命令，我们看到这里有一个 `edef` 命令，`edef` 命令和 `def` 命令类似，区别在于定义某个命令的时候立即展开，比如在这里我 `edef` 了一个 `@linkname` 命令，那么后面的 `@AllEndnoteCounter` 当时是什么立即赋值计算出来。因为这里要建立 `label` 的名字比如是独一无二的，然后看到后面 `@showlinkname` 和 `@showlinknameb`，这是在决定文档中尾注标记符号和尾注中标记符号的样式，这里名字弄翻了，因为之前我在编写的时候也不大确切，然后后面就交换了一下。

然后我们注意到这个 `endnote` 命令后面每一行都有一个%，这是不得已而为之。`endnote` 命令在它所在地地方我需要插入 `label` 和 `ref`，然后每一行换行之后都会增加一个空格。只好用百分号消去。

这里插入的 `ignorespaces` 命令为了消去空格可能是多余的，然后使用 `flabel` 命令植入 `label`，前面谈及了第一个参数是真实的 `label`，第二个参数是显示的 `label`。还有我们需要把 `label` 命令和

`ref` 命令的关系弄清楚，`label` 只是在文档中植入一个记号，而 `ref` 命令才会在文档中显示某些字符。我在测试的时候发现尾注标签上标记号有点偏后，然后用 `hspace` 命令稍作调整，注意我们调整距离尽可能使用 `ex` 或者 `em` 这样的相对距离。然后就是使用 `ref` 命令。

接下来一行语句的意思就是 `ent` 那个文件打开了吗？打开了什么也不做，没打开打开这个文件，使用之前定义的 `openent` 命令打开。

然后我们进入一个 `group`，`tex` 命令中 `\begingroup \endgroup` 之间夹著一个内环境。这个内环境的作用就是为了把 `endnote` 命令中的文本内容传递出去，为什么要这么做？我们的思路是将 `endnote` 中的文本写入到某个 `ent` 文件中去，然后继续写，直到要插入尾注了再引入这个 `ent` 文件即可。思路很简单，但是要实现起来并不简单。

这里最大的问题是 `endnote` 命令里面的命令和环境都要不执行，当作文本一行写入进去。因为 `tex` 默认参数都要执行计算出来的，这就是问题的所在。这里面的细节甚是复杂，我也没完全弄明白。我们再看到上面那段代码定义的 `@writeent` 命令。

---

```

1 \DeclareRobustCommand*\showendnotes{%
2 \immediate\closeout\@entout%
3 \global\@entopenfalse
4
5 \input{\@entname.ent}
6
7 \setcounter{@EndnoteCounter}{0}

```

```

8 \stepcounter{@ShowEndnoteCounter}%
9
10 }
11
12 \endinput

```

---

这是宏包的最后一段了，非常容易理解了。定义了一个 `showendnotes` 命令，命令首先将那个打开的 `ent` 文件关闭，这样之前写入的内容都写进去了。然后改变那个条件变量表明文件关闭了。然后引入尾注。然后 `@EndnoteCounter` 计数器归零，这个前面谈到过。然后 `ShowEndnoteCounter` 计数器加一。然后结束宏包使用 `endinput` 命令。

我在这里使用了 `cverbatim` 环境来描述这个 `endnotes` 宏包，总的文件生成在本 `github` 目录下 `codehome` 文件夹里面。进入尾注宏包文件夹之后把那个 `endnotes_sty.tex` 文件改名为 `endnotes.sty` 然后放入该放的位置即可使用。

## 文字强调

### `emph` 命令

`emph` 命令一般是英文换成斜体，中文换成楷体 (也就是前面设置的意大利字形的字体)。不过不同字族会有不同的表现。如果在强调环境之内有强调 (一般没有这种情况把。) 那么文字又会换成常规形态。`emph` 命令是 `LATEX` 自带的最基本的用于文字强调的方法。

## 重新定义 emph

本文档中的 `emph` 命令被重新定义了：

```
\renewcommand\emshape{\color{red}}
```

比如：我觉得字体设为红色更加起到强调作用

## underline 命令

这个也是  $\text{\LaTeX}$  自带的命令，就是加上下划线，不过在中文中并不能正确换行。所以往下看。

## ulem 宏包

下面的内容来自 `ulem` 宏包。<sup>19</sup>

`ulem` 宏包提供如下命令：`uline`，`uuline`，`uwave`，`sout`，`xout`，`dashline`，`dotuline`。主要用于文字的强调，其中 `uline` 是下划线，`dotuline` 是加点强调，`uwave` 是波浪线。其他命令请参看 `ulem` 宏包文档。

这是一段很长的测试文字，主要用于说明中文情况下加入下划线并且能够正确的换行。用的是 `uline` 命令。

注意如果单纯加载 `ulem` 宏包，原有的 `emph` 命令也会成为类似 `uline` 命令的效果，也就是加下划线。可以后面跟上命令 `\normalem`，也就是 `emph` 命令还是原来的处理效果。

<sup>19</sup>现在 `ulem` 中文可以正确换行了。

## reduline 命令

---

```

1 \newcommand\reduline{\bgroup\markoverwith
2   {\textcolor{red}{\rule[-0.5ex]{1em}{0.4pt}}}}
3   \ULon}

```

---

这个命令来自 `ulem` 文档，其中除了 `reduline` 命令名字可以自己 `diy` 之外，你能改动的就是第二行了。`rule` 部分简单说明下，第一个可选项是竖向位移，第一个参量是线条横向宽度，我为了好理解就设置成了 `1em`，似乎这个值设置成其他的值影响也不大。第二个参量是线条竖向高度。

## reddotuline 命令

我结合 `ulem` 宏包中的代码，然后修改自造一个命令如下：

---

```

1 \makeatletter
2 \def\reddotuline{\bgroup
3   \UL@setULdepth
4   \markoverwith{\begingroup
5     \advance\ULdepth0.168ex
6     \lower\ULdepth\hbox
7     {\kern.168em \textcolor{red}{.} \kern.168em}%
8     \endgroup}%
9   \ULon}
10 \makeatother

```

---

我对  $\text{T}_{\text{E}}\text{X}$  原生命令不太熟悉，就这样借用原代码简单地修改了下凑合着用吧。就是将原来代码中的小点改变了颜色，然后两边 `kern` 的距离稍微拉大了点。其实我心目中是希望每个中文字刚好最下面加个红点，先就这样吧。

## 插入列表

`itemize` 和 `enumerate` 环境其实也支持 `item` 后面跟上可选项的形式，只是从格式上他们常常出界，不建议使用。

### 基本使用

#### `itemize` 环境

---

```
1 \begin{itemize}
2 \item 这是一个列表
3 \item 这又是一个列表
4 \end{itemize}
```

---

- 这是一个列表
- 这又是一个列表

#### `enumerate` 环境

---

```
1 \begin{enumerate}
2 \item 这是一个列表
```

```
3 \item 这又是一个列表
4 \end{enumerate}
```

---

1. 这是一个列表
2. 这又是一个列表

### description 环境

---

```
1 \begin{description}
2 \item[鸭子] 是一种动物
3 \item[苹果] 是红色的
4 \end{description}
```

---

鸭子 是一种动物

苹果 是红色的

### enumerate 环境标签的修改

`enumerate` 环境中各个 `item` 标签自动生成是依赖各个对应的计数器，然后通过 `labelenumi`, `labelenumii`, `labelenumiii` 和 `labelenumiv` 这几个命令控制的，也就是重新定义这些命令，标签样式就被修改了。这几个中最常用的是第一级标签 `labelenumi`，下面就这个命令给出例子。

首先我希望这个 `enumerater` 环境的 `item` 计数从 0 开始，然后我希望它的格式是 `<0>`。

<0> 这是一个列表

<1> 这又是一个列表

上面形式的实现就是在这个 `enumerate` 环境中加入了这样两行代码：

---

```
1 \setcounter{enumi}{-1}
2 \renewcommand{\labelenumi}{<\arabic{enumi}>}
```

---

估计计数器使用的时候都会先加一，所以为了初值为 0，只好先设置为 -1 了。

### itemize 环境标签的修改

`itemize` 环境里面当然没有计数器，不过它类似的也有：`labelitemi`，`labelitemii` 等命令，通过重定义这些命令就可以影响第一级第二级等标签样式。

## 插入图片

要插入一个图片， $\text{\LaTeX}$  文档开头那里要加载库文件：

```
\usepackage[dvips]{graphicx}
```

然后在你想要插入图片的地方输入如下命令：

```
\includegraphics [scale=1]{图像名字}
```

插入图片有很多参数可以设置，这里就最有用的宽度和高度设置说明一下：



具体就是 `height` 表示高度，`width` 表示宽度。然后等于多少 `in`，英寸。参数放在可选参数那里。不过我觉得下面这个参数设置挺实用的，如下：

```
\includegraphics[width=\textwidth, keepaspectratio]{texstudio.png}
```

其中 `\textwidth` 表示让图片和文字一般宽，然后 `keepaspectratio` 参数意思是缩放的时候保持宽高比不变。

## 图片标题的修改

本文的图片环境标题通过输入 `caption{text}` 即实现如下形式的图片标题：图 1-1: text。其中图这个字是通过重定义 `figurename` 命令来完成的。

```
\renewcommand\figurename{图}
```

而后面本来默认的是 1.1 这样的形式，但是常见的都是 1-1 这样的形式，通过重定义 `thefigure` 命令即可达到目的。

---

```
1 \renewcommand{\thefigure}{\arabic{chapter}-\arabic{figure}}
```

---

## 设置图片寻找文件夹

```
\graphicspath{{figures/}}
```

可以设置多个文件夹。

## 图片格式的讨论

### 支持的图片文件格式

使用 `xelatex` 和 `graphicx` 宏包，目前我测试支持的图片格式有：`eps`，`png`，`jpg`，`pdf`。

### 搜索图片后缀名控制

经测试上面的 `eps`，`png`，`jpg`，`pdf` 格式的图片后缀名默认都支持，也就是只要写上前面的文件名即可。

### eps 图片格式

如果你使用矢量绘图软件画的图片，推荐用 `eps` 或者 `pdf` 格式。下面给出一个 `eps` 图片，是 `pyx` 宏包画出来的，然后用 `inkscape` 软件用 300dpi 另存为 `pdf` 格式，并用 `inkscape` 软件用 300dpi 另存为 `png` 格式。

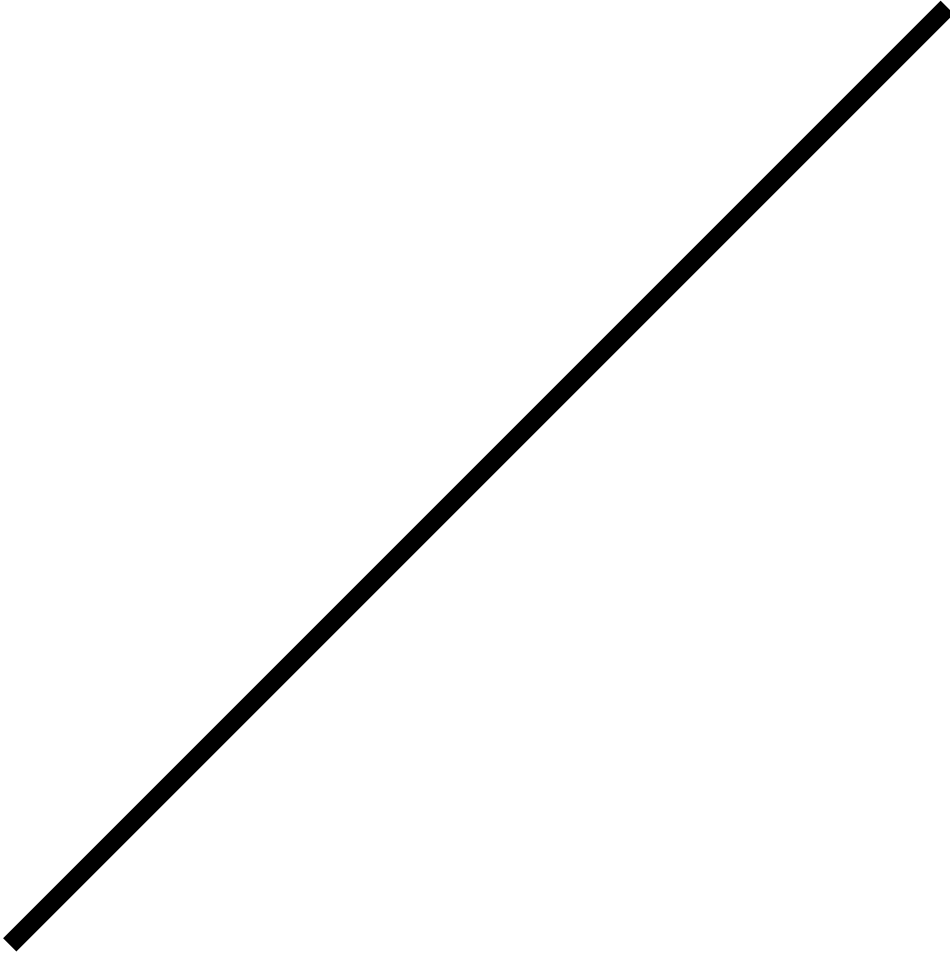


图 2-3: line.eps

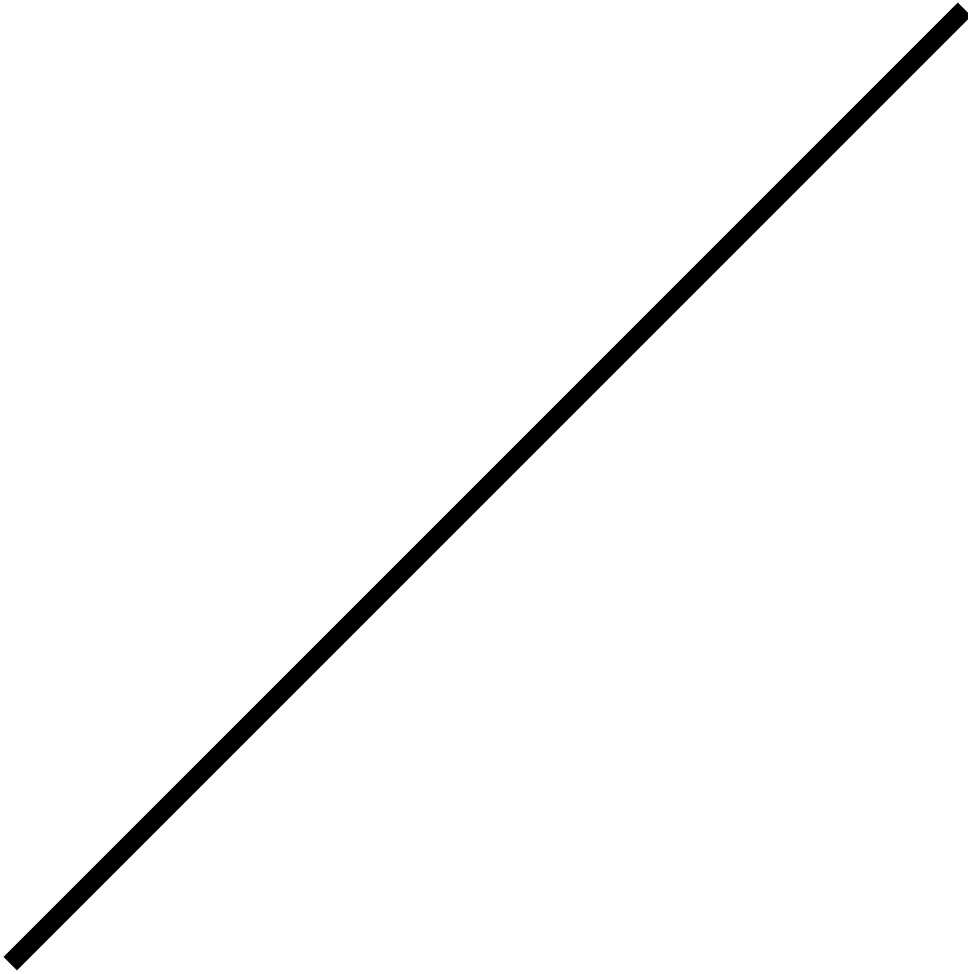


图 2-4: line.pdf

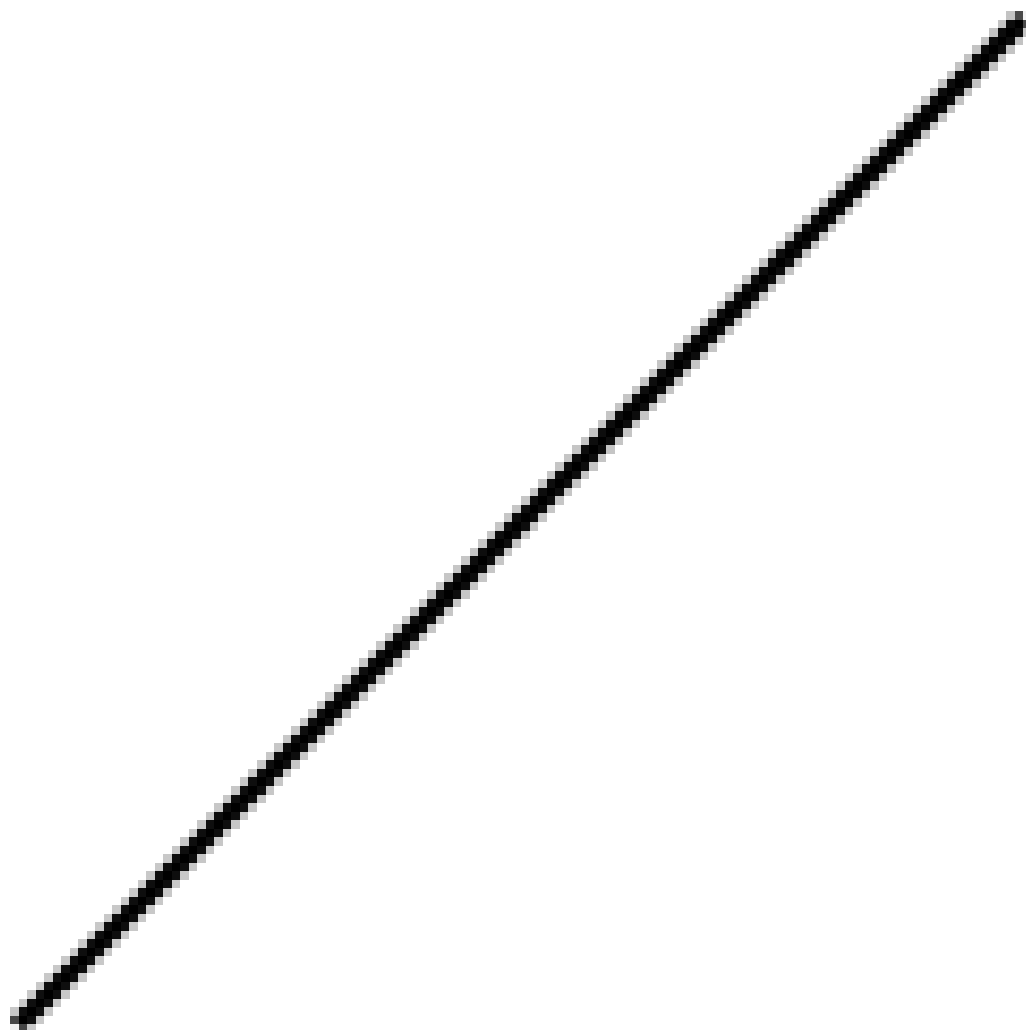


图 2-5: line.png

其中 pdf 和 png 格式文件要比原来的 eps 格式（300 字节）文件大 600 字节，可以看出如果选择 png 格式图片将会变得很模糊，而选择 pdf 虽然图片一样清晰但是图片文件更大了。

### svg 图片格式

有时在网上另存为出来的图片格式是 svg 图片格式，是一种什么可缩放矢量图形。下面是一个小测试，原 svg 图片是一个很小的

文字，然后放大。其中 pdf 和 png 都是 inkscape 软件用 300dpi 导出的，可以看出 pdf 格式明显更胜一筹。

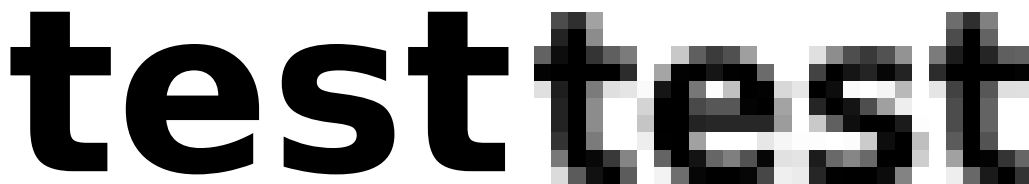


图 2-6: svg 图片测试.pdf

图 2-7: svg 图片测试.png

### jpg 图片格式

他们说相片最好用 jpg 格式，一般相片什么的都较大，所以也不存在缩放问题。

### 图片格式选择总结

结论是如果图片本来很大不存在缩放问题，那么能够使用的就直接使用，其他不支持的图片格式考虑生成 png 图片，记得 dpi 至少设到 300 以上，一般到 1200dpi 是绝对够用了。如果源图片很小，需要缩放，那么 eps 和 pdf 文件格式不用做进一步处理，其他文件格式优先考虑 pdf 文件格式，导出的时候我们看到 300dpi 已经绰绰有余了。

### 导入 pdf 页面

这里要讲的是使用 pdfpages 宏包来实现多个 pdf 页面的导入。

```
\usepackage[options]{pdfpages}
```

默认是插入所有 pdf 页面，

## 本文插入图片的一些 DIY

---

```

1 \newenvironment{fig}[2][1]
2     {\begin{figure}[H]
3     \centering
4     \includegraphics[scale=#1 , keepaspectratio]{#2}
5     \caption{#2}}
6     {\end{figure}}
7 \newenvironment{linefig}[2][1]
8     {\begin{figure}[H]
9     \centering
10    \includegraphics[width=#1\linewidth ,totalheight=\textheight , keepaspectratio]{#2}
11    \caption{#2}}
12    {\end{figure}}

```

---

本文新建了两个新的图片环境命令，第一个 `fig` 环境默认使用 `scale=1`，也就是那些较小的图片的居中对齐效果较好，如果图片较大可以考虑 `linefig` 环境，这个环境会让图片宽度对齐到 `linewidth`（之所以选择 `linewidth` 是因为在多栏环境下也会自动适应到那个栏的宽度。）。

然后通过第一个参数结束图片文件名，这个参数传递给 `caption` 命令作为输出标题。这样我要插入一个图片代码如下：

---

```

1 \begin{fig}{geometry 选项 1}
2     \label{fig:geometry 选项 1}
3 \end{fig}

```

---

这样插入图片代码就很简洁了。然后第二个插入图片环境命令主要针对那些稍微小点的图片，给它设计了一个可选参数<sup>20</sup>缩放比。这样方便调整图片大小。

## 插入表格

### 基本情况的讨论

一般情况下一些小的表格就用 `tabular` 环境处理即可。下面看这个例子：

---

```

1 \begin{table}[h]
2 \centering
3 \begin{tabular}{|c|c|}
4 \hline
5 l & l 表示该列格子左对齐 \\
6 \hline
7 c & c 表示该列格子居中 \\
8 \hline
9 r & r 表示该列格子右对齐 \\
10 \hline
11 \end{tabular}
12 \caption{tabular 参数}
13 \label{tab:tabular 参数}
14 \end{table}

```

---

<sup>20</sup>记住可选参数在参数中是排前面的



例子显示如下：

l	l 表示该列格子内容左对齐
c	c 表示该列格子内容居中
r	r 表示该列格子内容右对齐

表 2.9: tabular 参数

在 `table` 环境那里我加了一个可选参数 `h`，意思是在这里就在这里。这个表格还有图片环境都是什么浮动体。我们看到他们可以加上 `caption` 命令从而有一个标题，然后 `table` 和 `figure` 后面有个可选参数来控制这个浮动体的位置。默认是 `tbp`。不过我喜欢用 `h`，也就是在这里如果可能。有的时候 `h` 的表现效果可能不太让你满意。那么你可以尝试 `float` 宏包，它提供了 `H` 参数，会更加强制地控制浮动体，`H` 的意思是一定要在这里。

这段代码中 `centering` 命令是让表格居中，类似的命令还有 `raggedleft` 和 `raggedright`。

`caption` 命令是加上标题，`label` 命令是方便引用。

最重要的就是 `tabular` 环境，前面 `|` 符号表示画一个竖线，也就是每一列刚开始画一条竖线，你也可以不画表示每一列开始不画竖线。然后是字母 `c`，表示每一列第一个格子居中对齐，类似的还有字母 `l(left)` 和 `r(right)`。还有一种格式 `pwidth`，表示该格子具有 `width` 的宽度，然后里面的文字自动断行。

`hline` 命令表示画一条横线。`&` 这个特殊符号表示进入下一列。`\\` 表示进入下一行，后面跟上可选项距离，那么下一行的高度将拉高。

## booktabs 宏包

如果你只是想要一个简洁明了的表格，目前大家公认的好的表格标准就是三线表式，你也可以称之为 **booktabs** 风格吧。简单来说有以下规则：① 不要垂直线；② 不要双横线；③ 每一行的各个格子都有足够的空间；④ 一律左对齐；⑤ 三线，**toprule**，**midrule**，**bottomrule**。 <sup>(11)</sup>

在这里我们还有个捷径，不一定要手工将表格代码全部敲出来。首先我们找一个表格软件，**libreoffice** 的或者 **gnnumeric** 都行。然后将数据输入进去保存好。然后我们将数据选择复制，打开网站：<http://www.tablesgenerator.com/>。在那个网站的表格的开头哪里按下 **Ctrl+v**。这个网站还有一些设置可以调整，稍微摸索下就知道了。唯一要说的是那个 **activate/deactivate custom grid edit** 选项，可以选择绘制某几根线框显示。在这里不做调整，直接选择最右边的 **booktabs table style**。然后把多余的行或者列删除掉，就可以点击下面的 **generate** 了。生成的代码如下：

---

```

1 % Booktabs require to add \usepackage{booktabs} to your document preamble
2 \begin{table}[h]
3 \begin{tabular}{@{}ll@{}}
4 \toprule
5 参数      & 描述      & \\ \midrule
6 l      & 左对齐    & \\
7 c      & 居中      & \\
8 r      & 右对齐    & \\
9 p{width} & 一定宽度自动换行 & \\ \bottomrule
10 \end{tabular}

```

```
11 \end{table}
```

---

表格的显示效果初步如下: (12)

参数	描述
l	左对齐
c	居中
r	右对齐
pwidth	一定宽度自动换行

接下来就是微调整了, 比如说上面的花括号没有正常显示, 还有想要居中, 开头加上 `centering` 命令, 还有 `caption` 标题命令, 还有 `label`。修改之后结果如下:

参数	描述
l	左对齐
c	居中
r	右对齐
p{width}	一定宽度自动换行

表 2.10: tabular 参数 -2

## 一个三线表模板

---

```
1 \begin{table}[H]
2 \centering
3 \label{tab:}
4 \caption{}
5 \medskip
6 \begin{tabular}{@{}ll@{}}
```

```

7 \toprule
8 选项 & 说明 \\ \midrule
9 H & 页眉 (head) \\
10 O & 奇数页 (odd) \\ \bottomrule
11 \end{tabular}
12 \end{table}

```

---

上面就是一个简单的三线表模板，然后填充内容即可。

## booktabs 宏包详解

上面已经算是一个 `booktabs` 风格的表格了，为了进一步深度定制这里对 `booktabs` 宏包做一些说明。

### 线条粗细

前面我们看到了 `booktabs` 宏包新加了三个命令 `toprule`，`midrule` 和 `bottomrule`。这三个命令后面都可以跟个可选项调整线条粗细。后面跟的参数 `1pt` 就表示线条粗 `1pt`，没有加法的意思。上面的例子中 `toprule` 和 `bottomrule` 设置为 `1.2pt`，比原来的稍微粗了一点。嫌麻烦不改动也可以，觉得原来的也还好。

### cmidline 命令

`cmidline` 类似于原来的 `cline` 命令，简单来说就是你希望第几个到第几个格子画一条线。这个命令后面一样可以跟个描述粗细的可选项。请看下面的例子：

---

```

1 \begin{table}[h]
2 \begin{tabular}{@{}lll@{}}
3 \toprule
4 slices      & \multicolumn{2}{l}{abs.error(slices)} \\ \cmidrule{2-3}
5             & avg.           & max           \\ \midrule
6 <5000       & 116           & 625           \\
7 5000-10000  & 209           & 1807          \\
8 10000-15000 & 297           & 2133          \\
9 >15000     & 317           & 1609          \\ \bottomrule
10 \end{tabular}
11 \end{table}

```

---

显示效果如下：

slices	abs.error(slices)	
	avg.	max
<5000	116	625
5000-10000	209	1807
10000-15000	297	2133
>15000	317	1609

表 2.11: cmidrule 例子

我觉得没必要折腾成居中，还有也没必要调整页面布局，和图片一样会自动越界。其实完全没必要列这个大的一个例子，`cmidrule` 也很好理解，就是转行从几个格子上面画到第几个格子。主要是这种表头分线的例子还是画一个出来更直观些。

### 拉宽一行行的距离

```
\renewcommand{\arraystretch}{1.3}
```

通过上面这个命令，就在导言区整体设置就行了。文章的表格一行行距离稍微拉宽了一点，看上去更加美观了些。

### 带颜色的表格<sup>(13)</sup>

`xcolor` 宏包提供了一种颜色交替的表格模式，还挺好看的。好吧，我对颜色搭配不太擅长。首先需要在加载 `xcolor` 宏包时填上 `table` 可选项，即`\usepackage[table]{xcolor}`。

实现本表步骤	备注
<code>table</code> 环境下加上 <code>rowcolors</code> 命令	让整个表格交替显色
<code>rowcolors</code> 第一个选项	决定颜色从那一行开始显示
<code>rowcolors</code> 第二个选项	奇数列的颜色
第三个选项	偶数列的颜色
本表具体代码	<code>\rowcolors{2}{}{lightgray!50}</code>
第一行用 <code>rowcolor</code> 命令控制	在表头行上面

表 2.12: 带颜色的表格

这样形式的表格建议不用三线表式了，反倒是纯颜色样式不带线条更美观。

## 插入代码

### 小代码

有的时候一行之内的小代码就不需要大动干戈用 `xverbatim` 环境，用 `\verb+` 这里放着小代码 `+`，这里的 `+` 号可以换成其他任何的符号表示小代码开始和结束，除了 `*` 和空格不行。带星号的 `verb` 有其他用途，在里面空格以符号显示出来了，比如：

```
_this_is_a_test_.
```

感觉好丑陋，应该没啥用处。[\(14\)](#)

某一行小代码有时候需要缩小字体从而防止字出界，这里似乎没有办法 DIY 出一个新的命令，只能在 `verb` 命令之外套个改变字体的环境。

```
this is a test line in verb command and outside is the footnotesize command
```

### 稍微大点的程序代码

在环境 `verbatim` 之间的任何文本是什么就是什么，不执行任何 `LATEX` 命令，包括所有的空白和断行，如下：

```
(defmacro with-gensyms (syms &rest body)
  '(let ,(mapcar #'(lambda (s)
    '(',s (gensym)))
    syms)
    ,@body))
```

同样这里也外面包围了一个 `footnotesize` 环境改变字体大小。

## fancyvrb 宏包

`fancyvrb` 宏包提供了一个 `Verbatim` 环境，本文具体的配置代码如下：

---

```
1 \fvset{numbers=left,frame=lines,tabsize=4 ,baselinestretch=2,
2   xleftmargin=6pt, fontsize=\footnotesize , numbersep=2pt}
```

---

更多说明请参见 `fancyvrb` 宏包文档，这里只就涉及到的简单说一下：

表 2.13: `Verbatim` 环境一些设置

配置	说明
<code>numbers=left</code>	左边显示数字
<code>frame=lines</code>	框框是两条线
<code>tabsize=4</code>	<code>tab</code> 符号是四个空格
<code>baselinestretch=2</code>	行间距拉伸
<code>xleftmargin=6pt</code>	左边间距 <code>6pt</code>
<code>fontsize=\footnotesize</code>	字体大小设置
<code>numbersep=2pt</code>	数字和框框间距

## xverbatim 宏包说明

我新建了一个 `xverbatim` 宏包，提供了一个 `xverbatim` 环境，能够对代码进行染色显示输出，并对某些编程语言具有执行和显示输出结果的功能。这个环境还提供集中可选模式让实际操作更加灵活。由于这个宏包包括很多行 `pygments` (`python` 的一个宏包) 的输出配置信息，这里就不将宏包全部列出来了。完整的宏包内容在本 `github` 的 `texmf` 目录之下。



虽然整个宏包有两百多行，但主干结构也就一百行左右。具体实现结构和前面谈及的 `endnotes` 宏包有点类似，在编写的时候也参考了 James Brothie 的 `python` 宏包。

正常运行这个宏包你需要安装 `python` 的 `pygments` 宏包，在 `ubuntu` 下运行：

```
sudo apt-get install python-pygments
```

或者 `python3` 的版本：

```
sudo apt-get install python3-pygments
```

首先我们看到程序的入口，定义 `xverbatim` 环境的这一段程序，`xverbatim` 宏包的其他部分都是围绕着这个环境命令展开的。

---

```

1 \newenvironment{xverbatim}[2][1]
2 {
3 \immediate\stepcounter{@xverbatimCounter}
4 \def\@modestatus{#1}
5 \def\@codefilename{#2}
6 \StrBefore{\@codefilename}{.}\@codefile]
7 \StrBehind{\@codefilename}{.}\@codemode]
8
9 \immediate\write18{
10 if [ ! -d codeexport/\arabic{@xverbatimCounter} ];
11 then mkdir -p codeexport/\arabic{@xverbatimCounter} ; fi}
12
13 \newcommand{\@outname}
14 {codeexport/\arabic{@xverbatimCounter}/\@codefile }
15

```

```

16 \immediate\openout\@codeout=\@outname.\@codemode
17 \newlinechar='15
18 \begingroup \catcode'\^^M=12 %
19 \let\do\@makeother\dospecials\obeyspaces%
20 \@xverbatim}
21 {\endgroup
22 \immediate\closeout\@codeout
23 \ifnum \@modestatus = 0
24 \else \ifnum \@modestatus = 1
25 \@showcode
26 \else \ifnum \@modestatus = 12
27 \@showcode \@executecode \@writefeedback
28 \else \ifnum \@modestatus = 129
29 \@showcode \@executecode \@showresult \@writefeedback
30 \else \ifnum \@modestatus = 19
31 \@showcode \@showresult
32 \else
33 \errmessage{you typed wrong optional parameter,
34 the choice is 0 1 12 129 19 .}
35 \fi\fi\fi \fi \fi
36
37 }

```

---

首先新建 `xverbatim` 环境，然后接下来那么一大段都是 `xverbatim` 环境的定义。`xverbatim` 接受两个参数，一个可选参数 `#1`，只能接受 0, 1, 12, 129, 19 这几个数字，其他将会报错。然后还要接受一个必填参数 `#2`，（这里出于简单的美学考虑），这个参数接受等下要创建的文件的完整文件名。比如 `test.py`。这个文件将会放在

`codeexport` 目录的第几个例子计数文件夹里面。

这里先将 `@xverbatimCounter` 这个计数器加一，为的是它的第一次出现是 1。然后每次遇到一个 `xverbatim` 环境都会加一。

接下来定义了 `@modestatus` 命令接受参数 `#1`，为了将这个参数传递到环境的结尾部分。然后定义了 `codefilename` 接受第二个参数。

接下来的两行需要详细说明一下，这里使用的 `StrBefore` 和 `StrBehind` 命令来自 `xstring` 宏包，用处是从某个字符串中提取出某个字符前面的字符串或者后面的字符串。这里是处理 `@codefilename` 命令也就是接受的文件名，字符分隔符为 `'.'`，在 `'.'` 的前面（默认是第一次出现，所以为了防止出错文件名只能出现一次 `'.'`）的字符串将会赋值给 `\@codefile`，下面的 `StrBehind` 命令类似，不同的时候将后面出现的字符串赋值给 `\@codemode`，这个后面方便 `pygments` 处理代码用的。

接下来的 `write18` 命令是直接向终端输送某个命令：这个命令首先是 `if` 开头表示进入条件语句，然后方括号表示某个条件判断语句，`-d` 表示判断某个文件夹是不是存在，`!` 表示逻辑否。分号表示换行，`bash` 命令是有一定的格式的，如果你需要一行表示某一个整个命令，可以用分号表示换行。`then` 表示条件语句进入如果判断真那么执行下面的语句，然后 `mkdir -p` 表示创建某个文件夹包括父目录。然后分号然后 `fi` 表示条件语句结束。整个命令就是确保 `codeexport` 中的 `@xverbatimCounter` 命令的那个目录存在。

接下来新建一个 `@outname` 命令，这个命令等下要控制写入文件的所在目录和文件名的。

然后打开那个文件，其中文件名已经包含了目录，文件名和文

件后缀。

接下来的几行可能有点多余，这是 `python` 宏包里面的，可能和写入文档有关，然后接下来的 `@xverbatim` 命令由前面那一系列符号转码规则确定，太难懂了，我也没弄明白，只知道它的作用就是将环境内的所有内容写入文件中，改动的话就是 `xverbatim` 这几个字对应环境。

然后我们看到 `xverbatim` 环境结尾部分，那个 `\endgroup` 命令让我不知所以，但是少了它程序就会出错。

接下来就是关闭文件，这个没什么问题。然后接下来就是有点稍显花哨的东西，其实就是前面 `@modestatus` 接受的那个参数，在这里构建了一个类似 `switch` 语句的结构。

首先如果 `@modestatus` 等于 0，那么什么也不做，在这里的意思就是既不显示代码也不做其他的。如果 `@modestatus` 等于 1，那么执行 `@showcode` 命令。我们再看到 `@showcode` 命令。

这里有个 `ifdefstring` 命令是 `etoolbox` 宏包提供的，意思是如果 `\@codemode` 等于 `tex` 的话，那么执行命令，否则执行下一个花括号里面的命令。这里主要是 `tex` 代码后缀和 `pygmentize` 命令生成的 `tex` 代码有点容易混淆，就单独开出来，区别只是文件名。

`pygmentize` 命令就是 `pygments` 提供的，它有很多选项，最常用的有：

表 2.14: pygmentize 常用选项

选项	说明
-o	输出到某个文件
-f	告知要输出的文件类型，比如 <code>tex</code> ，如果 <code>-o</code> 后面跟着的文件名带着后缀，命令会自动推测的。
-l	告知要染色的方案，比如 <code>-l python</code> ，就是 <code>python</code> 的染色方案，如果处理的文件带着后缀，命令会自动推测。
-O	全部信息，这里主要 <code>tex</code> 下 <code>Verbatim</code> 环境中的颜色支持命令。
--help	帮助信息

然后我们再往下看，如果选项是 12，那么将会执行

```
\@showcode \@executecode \@writefeedback
```

这三个命令。其中 `@writefeedback` 和程序如果发生错误并把错误信息写入文档有关，这里略过了。

而 `@executecode` 命令就是根据不同的 `codemode` 来选择不同的终端命令，这部分 `tex` 和 `c` 我有点考虑删除掉，最多支持 `python3` 和 `common-lisp` 和 `scheme` 和 `bash` 这几个脚本高级语言吧。

然后就剩下 `@showresult` 命令，值得一提的是我之前就是简单使用 `verbatiminput` 命令导入生成的结果文件，而在这里我则是使用的 `input` 命令，这是有深意的。因为我在探索 `sympy` 宏包的时候发现，这个 `python` 宏包使用 `latex` 命令可以直接生成 `tex` 文档下的数学模式下的代码，也就是我直接 `input` 将会显示很漂亮的数学公式。这里加了个 `group` 方便调整下 `input` 之后的一些格式，比如使用 `obeylines` 命令 `input` 的时候空格换行都将保留，然后取消了行缩进。

简要的说明就到此为止吧。哦，对了，你看到这个宏包前面有

很大一段定义命令，这个是 `pygmentize` 对于 `Verbatim` 环境的颜色支持，通过选项 `-O` 获得的，复制到这里以后就不用引入了。

## cverbatim 宏包说明

在 `texmf` 文件夹里面，我还新建了 `cverbatim` 宏包，支持 `cverbatim` 环境。这个环境主要是针对很大型的文档的，这个宏包的编写大体和 `xverbatim` 类似，属于简化版本。在前面只是显示代码，然后后面通过 `endcodefile` 命令来做一些总结工作，比如将 `temp` 文件夹删掉，新建一个 `codehome` 文件夹，将前面一些小片段的 `cverbatim` 环境里的代码合并成为一个文件。前面在介绍 `endnotes` 宏包的时候使用的就是 `cverbatim` 环境。这个宏包并没有什么新东西，有兴趣的可以自己研究下。

## 语录，引用和诗歌环境

`quote` 为语录环境，`quotation` 用于超过几段的引用环境，`verse` 用于诗歌环境。下面是 `quote` 语录环境：

“When the winds of change blow, some people build  
walls and others build windmills.”

—anonymous

## 插入摘要

---

```
1 \begin{abstract}
```

```
2 这是一段摘要文字。
```

```
3 \end{abstract}
```

---

## 参考文献

首先设置 `thebibliography` 环境，然后用 `bibitem` 命令插入文献，这里 99 的意思是编号宽度不超过 99. 在你想要引用的地方用 `cite` 命令。如下所示：

---

```
1 \begin{thebibliography}{99}
2 \bibitem{1} Bachmann W , 1973.
3 Verallgemeinerung and Anwendung der Rayleighschen
4 Theorie der Schallstreuung.
5 Acustica,28 (4): 223-228
6 \end{thebibliography}
```

---

## 注释

- (1) `paragraph` 命令可以构建出类似 `description` 环境的效果，不同的是后面不缩进，装载内容容量更大。
- (2) 在 `beamer` 类下有一种看上去不错的多栏环境，就是使用的 `columns` 环境，不过只适用于 `beamer` 类的 `frame` 框架下。
- (3) 下面参考了[这个网站](#)
- (4) 我查看了 `log` 文件，`Scale=1.2` 之后的 `pt` 值就是原 `pt` 值乘以 1.2。
- (5) 主要参考了[这个网站](#)
- (6) 如果你输入了某个 `Unicode` 的字形你目前的 `pdf` 字体并不包括，那么将会产生这样的一个错误信息：WARNING invalid CMap mapping entry. 然后你编

译 pdf 之后发现某个字符没有正常显示出来只是一个方框那么可能是目前字体不包括这个字形。

- (7) 在[这个网页](#)里，谈到原 `color` 宏包的所有特性基本上 `xcolor` 都支持，同时又加了很多新特性，相当于扩充集吧。
- (8) 参考了[这个网站](#)
- (9) 参看了 a beginner's book of  $\text{\TeX}$  的 `spacing between boxes` 一节。在 google book 哪里可以看到这一段，不过似乎这本书网上并不能自由下载。
- (10) 参看了[这个网页](#)
- (11) 按照 Markus Püschel 的 `small guide to making nice tables` 里面的介绍。
- (12) 你注意到这里有 `@{}`，意思是每一列前面有一段空白，可以被花括号中的字符填充，这里是完全取消掉那点空白。`booktabs` 的风格是开头那点空白和最后一列最后那点空白全部取消掉。
- (13) 本小节除了参看 `wikibook` 之外还参看了[这个网站](#)。
- (14) 在 `xverbatim` 环境或者 `verb` 命令之内，字体是 `ttfamily`。



# L<sup>A</sup>T<sub>E</sub>X 高级篇

## 长度量

### 单位

L<sup>A</sup>T<sub>E</sub>X 中常见的长度单位如下表所示：

表 3.1: L<sup>A</sup>T<sub>E</sub>X 中的常见长度单位

单位	说明	换算法则
pt		
mm	一毫米	1mm=2.84pt
cm	一厘米	1cm=28.4pt
in	一英寸	1in=72.27pt
ex	(相对长度) 大约相当于当时字体的 x 高度	
em	(相对长度) 大约相当于当时字体的 M 宽度	

## 默认的长度量

## 使用长度量

## 定义自己的长度量

## 修改长度量

## 计数器

计数器 (counter) 之前也接触过一些了，现在基于 wikibook 和[这篇博文](#)做一些整理工作。

### latex 默认的计数器

- **part chapter section subsection subsubsection paragraph subparagraph** 这些计数器我们在前面的章节编号形式修改那一小节中有所接触，具体请参看章节编号形式修改这一小节：[2.13.5](#)。
- **equation figure table** 这三个是浮动体环境的计数器，其中 figure 我们在前面的图片标题的修改那一小节有所接触，具体请参看图片标题的修改这一小节：[2.22.1](#)。
- **page footnote mpfootnote** page 看得出来是页面的计数器，还没怎么接触。footnote 是脚注的计数器，这个在 DIY 脚注的时候会接触到，mpfootnote 是 minipage 下的脚注的计数器，这个知识点参看的[这个网站](#)。

- 此外 `enumerate` 环境里面还有专门的计数器：**`enumi enumii enumiii enumiv`**。`enumi` 记录的是当前 `enumerate` 环境第一级的 `item` 出现的次数，后面的依次是第二级第三级等等，不怎么常用。具体请参看 `enumerate` 环境标签的修改这一小节：

[2.21.2](#)

## 使用计数器

计数器并不可以直接使用，需要使用以下命令来获得某个特殊形式的值。

**value** 将计数器转化为数值方便后面的计算，不能作为字符直接显示。

**arabic** 将计数器转化为 1 2 3... 的形式，可以直接显示。

**alph** 将计数器转化为 a b c... 的形式，可以直接显示。

**Alph** 将计数器转化为 A B C... 的形式，可以直接显示。

**roman** 将计数器转化为 i ii iii... 的形式，可以直接显示。

**Roman** 将计数器转化为 I II III... 的形式，可以直接显示。

**fnsymbol** 将计数器转化为一系列的特殊符号。

## 计数器变成带圈的数字系列

我想新建一个命令，这个命令可以把计数器转化为 ①②③... 这样的形式，可以直接显示。可以看得出来这个命令会很实用的。

## 第一种实现方法

---

```

1 \newcommand*\circled[1]{%
2   \tikz[baseline=(char.base)]\node
3     [shape=circle,draw,inner sep=1pt,minimum size=8pt] (char) {\#1};}
4 \newcommand*\circledarabic[1]{\circled{\arabic{\#1}}}
```

---

上面这段代码参考了[这个网站](#)。看得出来代码大体过程就是利用 `tikz` 宏包画出那个符号出来。

这个方法的好处是自由度比较高，根据 `circled` 命令，类似的可以定义 `circledalph`——圈圈里面带个字母等。这个方法还有一个好处，虽然 ①② 这样的符号 `Unicode` 里面有，但是字体没有，如前面所述，还是需要写上一连串的代码，多少有点费事，而且到后面数字大了总会没有这个字体符号了，就会出错，而这个方法可以避免这些麻烦。推荐使用这个方法。

① 这是一个列表

② 这又是一个列表

## 第二种实现方法

第二种实现方法比较直观，是我第一眼想到的解决方法。就是新建一个列表数据结构，然后引入进来。

`arrayjobx` 这个宏包就提供了这个数据结构。

---

```

1 \newarray\circlednumber
2 \readarray{circlednumber}{①&②&③}
3 \newcounter{testcounter}
4 \setcounter{testcounter}{1}
5
6 \circlednumber(\arabic{testcounter})%
7 \stepcounter{testcounter}%
8 \newcommand{\osn}[1]{\circlednumber(\arabic{#1})}%
9 \osn{testcounter}%

```

---

①②

这个方法的缺点就是前面第一个方法的优点，首先是字体符号要引入进来，并且符号总会有不够用的时候。不过这个方法有一个优点，那就是引入的字体符号和文档中其他类似的符号形体上都差不多，前面第一个方法通过一些 **DIY** 应该可以达到接近类似的效果，但较为繁琐了。不过通过脚注单页编号的设置或者某些小型的列表使用这个方法也未尝不可，这个看读者自己的取舍了。

## 定义自己的计数器

用 `newcounter` 命令来定义自己新的计数器。

```
\newcounter{countername}
```

## 修改计数器的值

系统默认的计数器都是默认设置初始值为 0，如果你需要使用这个计数器，首先加一，然后使用这个计数器。

## 计数器设值

直接将计数器设为某个值用 `setcounter` 命令。

```
\setcounter{countername}{number}
```

## 计数器加一

将计数器数值加一使用 `stepcounter` 命令。

```
\stepcounter{countername}
```

## 计数器加多少

将计数器加上一个数值使用 `addtocounter` 命令。

```
\addtocounter{countername}{number}
```

## 计数器和其他计数器绑定

在使用 `newcounter` 命令定义新的计数器的时候可以后面跟个可选项，可选项里填著另外一个计数器的名字，每当这个计数器加一的时候你定义的新的计数器就会归为零。

比如.....

它的实现机制应该是使用了这个 `\@addtoreset{counter}{chapter}` 命令。

## 新的命令或环境

### 新的命令

新建命令一般就用 `newcommand` 命令

```
\newcommand{name}[num]{definition}
```

用法都差不多，第一个必填选项是新建命令的名字，一般前面都加一个 `\` 符号。第二个可选选项是新建的这个命令接受的参数，比如你填一个 2，那么表示这个命令接受两个参数，在后面 `definition` 中引用的时候 `#1` 表示第一个参数，`#2` 表示第二个参数。后面还可以跟一个可选项表示这个命令的某个参数为可选参数并且可以给出默认值，这个可选参数默认对应后面的 `#1`。

$\text{\LaTeX}$  的 `newcommand` 命令是由原  $\text{\TeX}$  的 `def` 命令 (还有 `edef` `gdef` `xdef` 这里先不管。) 而来的。作用原理有点类似于替换展开的功能。比如：

```
\newcommand{\test}{this is a test line}
```

你定义了这个 `test` 命令，然后后面你输入 `\test`，系统遇到 `\test` 之后就会进行替换操作，就成了后面的 `this is a test line` 了。理论上只是简单的文本替换，后面可是跟上随意的命令参数或者环境。不过这只是理论上，实际编写宏包命令的时候会遇到很多复杂的问题，主要是 `expand` 展开的控制。

还有一个命令 `providecommand` 和 `newcommand` 差不多的，区别就是 `providecommand` 只是提供命令，如果这个命令已经定义了它就什么都不做。而 `def` 一定是完全不动声响的覆盖了。

在编写宏包文件的时候推荐使用 `DeclareRobustCommand` 命令，但是有些任务似乎只有 `def` 或者 `edef` 才能完成，这个我也没弄

明白。

## 新的环境

新建环境命令格式如下：

```
\newenvironment{name}[num]{before}{after}
```

和 `newcommand` 命令其他用法都差不多，区别就在于后面有两个宏替换。这里值得一提的是原  $\text{T}_{\text{E}}\text{X}$  并没有新建环境这个命令，而是  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  新加进去的。

我们假设我们要新建 `env` 这个环境，那么  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  的工作流程是，创建了两个命令，一个是 `\begin{env}` 命令；一个是 `\endenv` 命令。???。所以简单的来看 `newenvironment` 命令就是把它看作开辟了一个 `group`（原  $\text{T}_{\text{E}}\text{X}$  里面的概念），然后首先执行了 `before` 中的命令，然后是环境中的内容，然后是 `after` 中的命令，然后退出环境。

这里只是说明一下 `newenvironment` 命令的工作原理，具体我们新建环境还是用  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  的风格，即用 `newenironment` 命令，只有在不得已的情况下（通常很复杂的情况）才考虑用  $\text{T}_{\text{E}}\text{X}$  里面的 `group` 概念，那个时候估计是要使用 `catcode` 命令。



## 盒子和 glue

这里参考了 Knuth 的 The Texbook，但是我并没有将其写入参考文献，因为只是觉得关于 `box` 和 `glue` 的概念最好参考原初定义，但是并不推荐读者阅读这本书，现在已经是  $\text{\LaTeX}$  时代了，不推荐读者使用原始的  $\text{\TeX}$  命令，一是不太实用，二是兼容性可能不太好。除非你是宏包编写者，但就作为一般的使用者真的没有必要接触那些原始命令了。还参考了 [4]

(1)

### 基本知识

最小的盒子就是基于 `Unicode` 的字符，这些字符然后组成更大的盒子——单词，然后单词组成更大的盒子——行等等。行是一个盒子，段落也是一个盒子，图片是一个盒子，表格也是一个盒子。而这些盒子按照 Knuth 的描述都是用 `glue` 胶水粘合起来的，或者我们称之为这些盒子之间都存在着空间胶合层。下面就是一个盒子的详细参数：

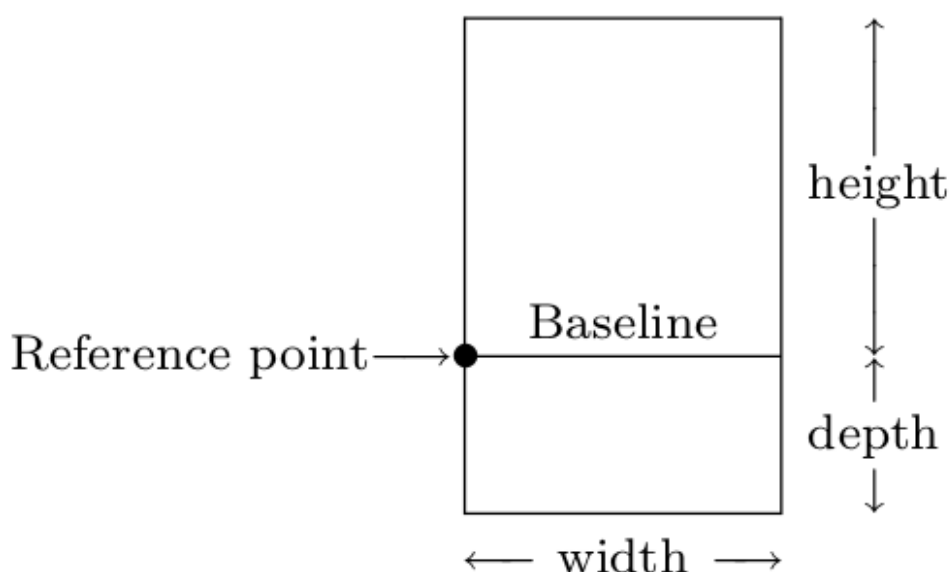


图 3-1: box 参数

盒子就是这么一个长方形的区域，如上图所示，它有参数：`height`，`width`，`depth`。`baseline`和`reference point`在后面讲的`hbox`和`vbox`中会用到。在 $\text{\TeX}$ 看来，从字体而来的 Unicode 字符就是一个最简单的盒子。字体的设计者已经决定了这个字符的高度，宽度和深度以及它在这个盒子里面看起来如何。 $\text{\TeX}$ 就是用这些维度将盒子黏合到一起，并最终决定所有字符的`reference point`参考点在页面上的位置。

$\text{\TeX}$ 的盒子如果全部涂上颜色，一般是黑色，那么就成了一个黑盒子。这样的黑盒子还有一个名字叫做`rule box`。也就是线条。这个在后面会谈论到的。

不管是字符盒子还是黑盒子，他们要某是水平排列要某是垂直排列。水平排列要做的就是让这些盒子的参考点在一条水平线上。类似的垂直排列要做的就是让这些盒子的参考点在一条垂直线上。

好吧，介绍两个  $\text{T}_{\text{E}}\text{X}$  命令：`hbox` 和 `vbox`。`hbox` 命令就是让所有的盒子在一条水平线上，而 `vbox` 命令就是把一些 `hbox` 命令垂直排列，比如下面的代码：

```
\vbox{\hbox{恭}\hbox{喜} \hbox{发}\hbox{财}}
```

恭

喜

发

财

`glue` 也就是各个盒子之间的间距。下图是 `glue` 的具体图示：

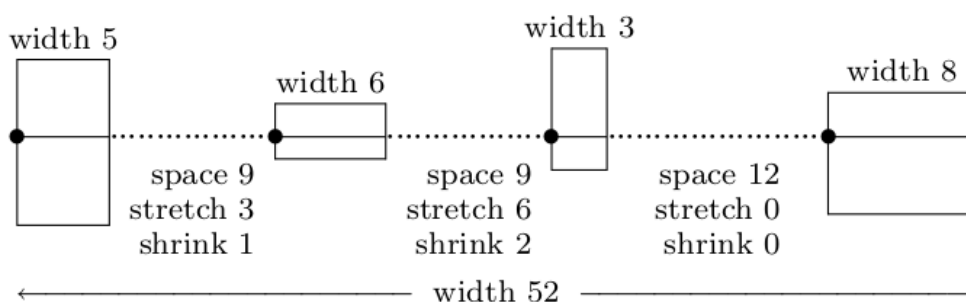


图 3-2: `glue` 说明

前面说到盒子的 `reference point` 水平排列，然后他们之间还有叫做 `glue` 的间距。间距有三个属性：正常间距量 (`space`)，拉伸量 (`stretch`)，缩减量 (`shrink`)。比如这个图片中第一个 `glue` 的正常间距是 9 个单位，拉伸量为 3 个单位，缩减量为 1 个单位。而总的情况是正常间距是  $5(\text{box1})+9+6+9+3+12+8=52$  个单位。现在假设一行宽 58 个单位， $\text{T}_{\text{E}}\text{X}$  就要调整使得这一行盒子的宽度刚好等于 58 个单位，于是还需要增加 6 个单位的宽度，而这 6 个单位的宽度<sup>1</sup>需要从这一行所有 `glue` 里的拉伸量中找出来。于是总的拉伸量加法是  $3+6+0=9$ 。也就是 6 个单位的宽度要分成 9 等分再分配给

<sup>1</sup>为了简单起见这里不考虑缩减量，具体缩减量如何计算我也不大清楚。

他们，即第一个 `glue` 的拉伸量是  $3 * \frac{6}{9}$ 。这样第一个间距的总长度就是  $9 + 3 * \frac{6}{9} = 11$ 。

经过计算所有的 `glue` 间距都确定下来了，那么整个页面布局就确定了。我在这里就戛然而止了，毕竟这里只是对 `box` 和 `glue` 的基本概念的阐明。

## 盒子命令介绍

[参考了这个网站](#)

### `makebox`

`\makebox[width][alignment]{some text}` 还有一个 `mbox`，不过 `makebox` 命令更加全面，推荐使用。`makebox` 就是制造一个水平的盒子，注意这个 `box` 里面的文本是不能换行的，也就是一行之内的盒子。第一个可选项 `width` 指这个盒子的长度，第二个可选项 `alignment` 是里面文本的对齐方式，有 **【l c r s】** 几个选项，**l**表示左对齐；**c**表示居中；**r**表示 right；**s**表示两端对齐。

### `raisebox`

### `parbox`

### `minipage`

#### `minipage`

## 带颜色或者线框的盒子

### framebox

`framebox` 和 `makebox` 命令类似，除了加上了一个线框。

```
\setlength{\fboxsep}{10pt}
```

```
\setlength{\fboxrule}{5pt}
```

如上面描述的这里有两个长度量，`fboxsep` 控制线框和文字之间的距离，`fboxrule` 控制线框的宽度。值得一提的是 `framebox`，`fbox` 或者 `fcolorbox` 命令的线框的这两个参数都是类似上面的代码控制的。

### colorbox

```
colorbox \colorbox{yellow}{this is a test line.}
```

```
this is a test line.
```

### fcolorbox

`fcolorbox` can also be tweaked with `\fboxsep` and `\fboxrule` 这两个长度量。

## fancycolorbox

本文定义了一个 fancycolorbox 命令，具体代码如下：

为什么是这么复杂的一个代码？这个需要说明一下。首先系统自带的 colorbox 命令不能换行，必须 colorbox 里面加上 minipage 环境才行，9,10 行是 minipage 环境的格式优化，然后整个构成一个新环境我还想传递颜色，就成这个样子了。主要参考的[这个网站](#)。

---

```

1 \definecolor{bgcolor-0}{HTML}{CCFFCC}
2
3 \newsavebox{\tempbox}
4 \newenvironment{fancycolorbox}[1][bgcolor-0]
5 { \noindent%%
6   \renewcommand{\tempcolor}{#1}
7   \begin{lrbox}{\tempbox}%
8   \begin{minipage}{\linewidth-10pt}
9     \setlength{\parskip}{1.6ex plus 0.2ex minus 0.2ex} % 段落间距
10    \setlength{\parindent}{\baselineskip * \real{0.06} + \textpt * \real{2.4}}
11    {\ignorespacesafterend%
12     \end{minipage}%
13     \end{lrbox}%
14     \colorbox{\tempcolor}{\usebox{\tempbox}}}
```

---

fancycolorbox 环境的效果前面你应该已经看过了。这里就不做演示了。

## 保存盒子

savebox

## 线条

`\rule[depth]{width}{height}` 这三个选项都要填长度量，第一个长度量是线条竖向偏移量，第二个长度量是线条横向长度，第三个长度量是线条竖向宽度。

## 大文档管理

## 注释

- (1) 在这里 `box` 翻译为盒子没什么问题，就是这个 `glue` 翻译为胶水或者橡皮都让我不太满意。在后面我都使用的术语是距离或者间距或者干脆用英文 `glue`。从某种意义上讲 `glue` 的直译确实应该译为胶或者胶水，不过觉得距离这个词汇更能够让人们有感觉些：`TeX` 排版就是不同的盒子编写和彼此之间距离的设置问题。

## 其他问题的讨论

plain TeX 命令

## TeX 中的程序结构

### 条件控制语句

[参考了这个网站](#)

条件控制语句的首先要新建一个条件控制变量：

`\newif\iflogvar` 默认这个变量是设置为假，然后通过以下模式来建立条件语句：

---

```
1 \iflogvar
2   aaaa
3 \else
4   bbbb
5 \fi
```

---

条件语句可以嵌套，然后你可以通过这样的命令来调整条件控制变量的真假：



`\logvartrue`

`\logvarfalse`

`ifnum` 命令

`ifnum` 命令用于判断数字。`\ifnum number = 0`

## 文本中的上标还有下标

上标使用命令 `textsuperscript` 命令，下标感觉自己缩小点就差不多了吧。示例如下：

上标<sup>上标</sup> 下标<sub>下标</sub>

更常见的是化学分子式里面的上标和下标，推荐使用 `mhchem` 宏包，至于数学模式里的上标和下标自不必说了。

## 多张图片并列显示

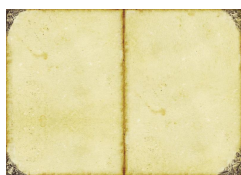


图 4-1: temp

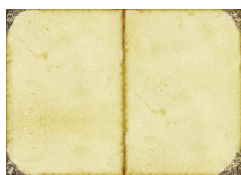


图 4-2: temp

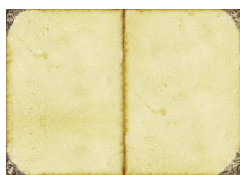


图 4-3: temp



图 4-4: temp

图 4-5: denosie fig

具体代码如下：

```

1 \begin{figure}[h]
2
3 \label{fig: 四栏图片}
4 \begin{multicols}{4}
5 \includegraphics[width=\linewidth ,totalheight=\textheight ,
6     keepaspectratio]{temp.jpg}
7 i want some test to show there is a text and not to column break.\\
8 \centerline{test}
9 \columnbreak
10 \includegraphics[width=\linewidth ,totalheight=\textheight ,
11     keepaspectratio]{temp.jpg}
12 \centerline{test}
13 \columnbreak
14 \includegraphics[width=\linewidth ,totalheight=\textheight ,
15     keepaspectratio]{temp.jpg}
16 \centerline{test}
17 \columnbreak
18 \includegraphics[width=\linewidth ,totalheight=\textheight ,
19     keepaspectratio]{temp.jpg}
20 \centerline{test}
21 \end{multicols}
22 \caption{denosie fig}
23
24 \end{figure}

```

---

这段代码的分栏还有插入图片知识都已经介绍了，值得一提的就是 `widetext` 环境，也就是有 `changepage` 宏包而来的临时改变页面布局环境和浮动体环境 `figure` 以及 `table` 不兼容。比如放入浮动体

环境内才能起作用。然后 `caption` 命令似乎只是默认的 `linewidth` 居中。所以如果想要图片和表格在扩大的文本布局中居中对齐，那么需要在浮动体环境内部使用改变页面布局命令，然后使用居中命令。

## 生成字体所有已有的字形<sup>(1)</sup>

---

```

1 \documentclass[landscape]{article}
2 \usepackage{geometry}
3 \usepackage{fontspec}
4 \setmainfont{DejaVu Sans}
5 \usepackage{multicol}
6 \setlength{\columnseprule}{0.4pt}
7 \usepackage{multido}
8 \setlength{\parindent}{0pt}
9 \begin{document}
10 \begin{LARGE}
11 \begin{multicols}{5}
12 \multido{\i=0+1}{"196607"}{% from U+0000 to U+2FFFF
13 % 后面的还会继续扩展，目前一般还没使用。
14   \iffontchar\font\i
15     \makebox[4em][l]{\i}%
16   \symbol{\i}\endgraf
17   \fi
18 }
```

```
19 \end{multicols}
20 \end{LARGE}
21 \end{document}
```

---

## 方便手机上观看的 pdf

原文档内容都不需要修改，只需要修改 `geometry` 的设置就可以满足要求，然后在手机上看的时候选择 `adobe pdf` 软件的连续观看功能会有更好的体验。具体配置如下： [\(2\)](#)

---

```
1 \ifphone
2 %for phone
3 \RequirePackage[
4   paperwidth=105mm, % 除去旁註其他沒變,115, 再稍微小點
5   paperheight=190mm,% 太長了縮短點,
6   bindingoffset=0mm,% 裝訂線
7   top=15mm, % 上邊距 包括頁眉
8   bottom=15mm,% 下邊距 包括頁腳
9   left=5mm, % 左邊距 or inner
10  right=5mm, % 右邊距 or outer
11  headheight=10mm,% 頁眉
12  footskip=10mm,% 頁腳
13  includemp=true,% 旁註寬度計入 width
14  marginparsep=0mm, % 沒有旁註
15  marginparwidth=0mm, % 沒有旁註
16 ]{geometry}
17 \else
```

```

18
19 \RequirePackage[a4paper, %a4paper size 297:210 mm
20   bindingoffset=0mm,% 裝訂線
21   top=45mm, % 上邊距 包括頁眉
22   bottom=40mm,% 下邊距 包括頁腳
23   left=35mm, % 左邊距 or inner
24   right=40mm, % 右邊距 or outer
25   headheight=25mm,% 頁眉
26   footskip=25mm,% 頁腳
27   includemp=true,% 旁註寬度計入 width
28   marginparsep=0mm, % 旁註與正文間距
29   marginparwidth=0mm, % 旁註寬度
30   ]{geometry}
31 \fi

```

---

这里使用了一个条件命令。在加载 `myconfig.sty` 宏包的时候已经新建了一个条件变量：

```
\newif\ifphone
```

`\phonefalse` `\ifphone` 就是一个条件判断命令，这里涉及到的命令我也不大懂，总之，如果我改成：

```
\phonetrue
```

就会自动生成适合在手机上观看的 pdf。

## 注释

- (1) 主要参照了[这个网站](#)
- (2) 主要参考了[这个网站](#)。

## 后面的珍宝

“The opera ain’t over until the fat lady sings.”

好戏还在后头，等著瞧吧。

我在 Ubuntu 的模板文件夹里面新建了一个 `tex.tex` 文件。方便我用鼠标右键快速创建模板。

第一个模板我打算使用一个通用的配置，写一个 `myconfig` 宏包，用于加载这个通用配置。还有一个 `mytitle` 宏包用于加载封面。然后整个模板就专注于写作，很简洁的样子。

然后我在 Ubuntu 的 `home` 文件目录或者称之为主文件夹下面新建了一个文件夹 `texmf`，然后里面新建了一个文件夹 `tex`，然后里面新建了一个文件夹 `latex`，然后在里面新建了一个文件夹 `myconfig`。`myconfig.sty` 文件就放在里面。然后在终端上运行命令：`sudo~ texhash` 即可。`mytitle` 宏包的安装类似。

`myconfig` 宏包的内容是我以前之前的 XeLaTeX 指南的精华部分，很多冗余删除了。关于 `myconfig` 和 `mytitle` 宏包的编写，我参考了[这个网站](#)。

三个文件的代码在本文后面列出。现在就一些必要的信息说明如下：

首先 sty 文档里面 @ 符号不需要用 `makeatletter` 这个命令了，@ 在里面已经默认就是一个符号。其次文档的基本结构

---

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{mytitle}
3 ...
4 \endinput
```

---

这个不用多说。

然后就是 sty 文档里面所有 `usepackage` 都换成 `RequirePackage`。这样 sty 文档兼容性更好，前面没有 `documentclass` 也行。

`myconfig` 宏包的详细讨论我在 `XeLaTeX` 指南一书中会详细讨论。`mytitle` 宏包前面那部分是定义新的输入命令，第二行是用于处理可能出现错误的反应。就照这个模式写。然后 `gdef` 命令后面就跟著新定义的命令的形式，`gdef` 和 `def` 命令的不同是他定义的是 `global` 的，我估计之前我那种写法不行可能就是没有 `gdef`。后面就是基本的封面设计了。

特别提醒在 `Ubuntu` 下右键新建之后，是未命名然后空格然后 `tex`，这种文件名 `XeTeX` 处理似乎不很好。建议改成没有空格的文件名。

源码和参考资料打包下载地址



# 插图目录

2-1 geometry 选项 1 . . . . .	37
2-2 geometry 选项 2 . . . . .	39
2-3 line.eps . . . . .	104
2-4 line.pdf . . . . .	105
2-5 line.png . . . . .	106
2-6 svg 图片测试.pdf . . . . .	107
2-7 svg 图片测试.png . . . . .	107
3-1 box 参数 . . . . .	134
3-2 glue 说明 . . . . .	135
4-1 temp . . . . .	141
4-2 temp . . . . .	141
4-3 temp . . . . .	141
4-4 temp . . . . .	141
4-5 denosie fig . . . . .	141

# 表格目录

2.1 调整字族 . . . . .	44
2.2 调整字型系列 . . . . .	44
2.3 调整字形 . . . . .	45
2.4 调整字体大小命令 . . . . .	45
2.5 字体具体大小 . . . . .	46
2.6 直接可以使用的颜色名字 . . . . .	64
2.7 不同百分比的灰色 . . . . .	65
2.8 fancyhf 可选项字母意义 . . . . .	72
2.9 tabular 参数 . . . . .	110
2.10 tabular 参数 -2 . . . . .	112
2.11 cmidrule 例子 . . . . .	113
2.12 带颜色的表格 . . . . .	114
2.13 Verbatim 环境一些设置 . . . . .	116
2.14 pygmentize 常用选项 . . . . .	121
3.1 L <sup>A</sup> T <sub>E</sub> X 中的常见长度单位 . . . . .	125

## 参考文献

- [1] 有名的《一份不太简短的  $\text{\LaTeX}2\text{e}$  介绍》，原版作者：Tobias Oetiker, Hubert Partl, Irene Hyna 等，版本：2001-08-09，中文翻译：CTeX 用户小组，版本：2002-05，pdf 下载链接：<http://www.ctan.org/tex-archive/info/lshort/chinese>。
- [2] 《大家来学  $\text{\LaTeX}$ 》，原版作者：李果正，版本：2004-03-08，网站链接：<http://edt1023.sayya.org/tex/latex123/>。
- [3] 维基图书 latex 篇英文版，版本：2013-08，网站链接：<http://en.wikibooks.org/wiki/LaTeX>。这本书甚至都没有 `cite` 它一下，因为不知道插在哪里，几乎到处都有对它的引用，谢谢所有 `wikibook` 的编写者。
- [4] `using boxes and glue in  $\text{\TeX}$  and  $\text{\LaTeX}$` ，原版作者：Nelson H. F. Beebe，版本：2009-03-28，pdf 下载链接：链接地址不好复制，请 google 搜索书名之。
- [5] The Latex Companion