

LaTeX Cookbook

Eureka

2023 年 6 月 26 日

目录

1	常见符号	2	8.2	图片任意放置	11
1.1	书籍章节符号	2	8.3	图片文字环绕	12
1.2	其他符号	2	8.4	单双栏排版	13
2	首字下沉	3	8.4.1	使用 <code>\twocolumn</code> 命令	13
3	各个长度单位	3	8.4.2	使用 <code>multicol</code> 宏包	16
4	代码环境	4	8.5	浮动体	17
4.1	Cpp 代码	4	9	盒子	17
4.2	Python 代码	4	9.1	水平盒子	17
5	自定义命令的坑	5	9.2	自定义水平盒子	18
6	公式相关	5	9.3	竖直盒子	19
6.1	上色	5	9.4	标尺盒子	21
6.2	强制上下标	6	9.5	缩放盒子	21
6.3	常见的 <code>limits</code> 变种	6	9.6	旋转盒子	21
6.4	加边框	6	10	一些 primitive	22
6.5	<code>\ensuremath{}</code> 命令	7	10.1	\TeX 的背后	22
7	TikZ 和 Gnuplot 结合	7	10.2	primitive 介绍	23
8	页面布局相关设置	10	11	魔改系列	23
8.1	打印页面布局	10	11.1	进制	23
			11.2	\TeX 的字符与编码	23
			11.3	字符 & 编码的转换	24
			11.4	类别码	25
			11.5	@ 溯源	26
			11.6	catcode 背锅	27

1 常见符号

经常我们都可以在别人的 pdf 中看到许多的符号，然后你不知道符号的名字，去网上查询也不方便，所以这里给出部分的常见符号的命令。详情可以参见官方文档 The comprehensive LaTeX symbol list。

1.1 书籍章节符号

使用它定义的命令，但是位置很奇怪



或者是使用\textikz 命令



1.2 其他符号

这里给出其他的一些常用符号：

所需宏包	命令	效果
none	<code>\copyright</code>	©
none	<code>\dag</code>	†
none	<code>\P</code>	¶
none	<code>\varpi</code>	ϖ
none	<code>\bullet</code>	•
none	<code>\clubsuit</code>	♣
wasysym	<code>\RHD</code>	►
wasysym	<code>\smiley</code>	☺
marvosym	<code>\Biohazard</code>	☠
marvosym	<code>\CEsign</code>	Ⓒ Ⓔ
marvosym	<code>\Stopsign</code>	⛔
marvosym	<code>\Radioactivity</code>	☢
marvosym	<code>\Pointinghand</code>	☞
marvosym	<code>\Writinghand</code>	✍
pifont	<code>\ding{227}</code>	➤
dingbat	<code>\smallpencil</code>	✎
dingbat	<code>\ding{229}</code>	➡
manfnt	<code>\lhd bend</code>	⚡ (缩放 0.8)






2 首字下沉

SOME text First-time space explorers welcome. Our universe is a wild and wonderful place. Join NASA astronauts, scientists, and engineers on a new adventure each week —all you need is your curiosity.

3 各个长度单位

在 \LaTeX 中有着各种各样的单位，下面对这些单位进行可视化记忆：

```
\begin{tabular}{l|l}
\hline\hline
1.pt(point) & \red{\rule{1pt}{1pt}}\hline\hline
2.mm(毫米) & \red{\rule{1mm}{1mm}}\hline\hline
3.ex(size of 'x') & \red{\rule{1ex}{1ex}}\hline\hline
4.em(size of 'M') & \red{\rule{1em}{1em}}\hline\hline
5.cm(厘米) & \red{\rule{1cm}{1cm}}\hline
\hline
\end{tabular}
```

1.pt(point)	
2.mm(毫米)	
3.ex(size of 'x')	
4.em(size of 'M')	
5.cm(厘米)	

4 代码环境

之前一直使用的是一个在线的网站，把导出的 pdf 插入 tex 文档，但是太麻烦了。我之前也用过 `lstings` 环境来定制我的代码抄录环境，但是效果还是不太理想，终于，使用 `mint` 宏包可以较为满意。下面是两段插入代码示例：

4.1 Cpp 代码

```
1  int main()
2  {
3      int i = 0;
4      int j = 0;
5      int N = 0;
6      scanf("%d", &N);
7      int ls[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
8      for(i=0; i<N; i++){
9          scanf("%d", &ls[i]);
10     }
11     return 0;
12 }
```

4.2 Python 代码

```
1  def add(x, y)
2      return x+y
3  x = 1
4  y = 2
5  res = add(x, y)
```

5 自定义命令的坑

我定义了一个常用的生成函数序列 ($y = x_1 + \cdots + x_n$) 的宏:

$$y = x_1 + \cdots + x_n \quad (1)$$

$$y = x_1 + \cdots + x_i \quad (2)$$

错误的调用例子: `\fn[i]{x}`

..... 注: 调用 LaTeX 中自定义命令时: 默认参数是放在 `[]` 内部的, 不是 `{}` 内部

颜色演示【注意: 染色用 `textcolor`, 别用 `color`(会影响你之后的所有内容)】

命令: `\textcolor{red}{hello 你好}`, 结果: hello 你好

默认参数演示

错误命令:

`\Test{第一个参数}{第二个参数}`

结果:

第一个参数: default *args

第二个参数: 第一个参数第二个参数

正确命令:

`\Test[第一个参数]{第二个参数}`

结果:

第一个参数: 第一个参数

第二个参数: 第二个参数

6 公式相关

- 0. 普通形式: \int_i^∞ 样例
- 1. `\displaystyle{*args}` 形式 (行间公式尺寸)
 - 1.1 一个: \int_i^∞ 样例
 - 1.2 命令封装 $\int_{i=0}^\infty f(x, y) \Delta \xi_i$
- 2. text 形式 (行内公式尺寸): \int_i^∞ 样例
- 3. scriptstyle 形式 (上下标尺寸): \int_i^∞
- 4. scriptscriptstyle (次级上下标尺寸): \int_i^∞
- 5. Large 参数尝试: 你好

6.1 上色

1.1 文本本生上色

红色的文字

1.2 文本背景上色

浅蓝色背景的字

2.1 公式本身上色

$$\int_{i=0}^\infty \sum_{i=0}^\infty f(\xi_i, \varepsilon_i) dx \quad (3)$$

2.2 公式背景上色

$$X_k = \frac{1}{N} \sum_{n=1}^{N-1} x_n e^{i2\pi k \frac{n}{N}} \quad (4)$$

$$\sum_{i=0}^{\infty} \quad (5)$$

上色命令的封装

浅绿色背景的文字

$$\sum_{i=0}^{\infty}$$

6.2 强制上下标

$$\text{强制上下标: } \sum_{i=0}^{\infty} \frac{1}{i^2}$$

但是`\limits` 只能够应用于 Operator(操作符). 比如 \propto 就不是操作符, 使用`\limits` 强制下标就会报错. 此时可以使用`\mathop{*args}` 把它转化成一个操作符, 然后就可以使用`\limits` 了.

使用演示:

$$\sum_{i=0}^{+\infty} \int \sum_{j=0}^{\text{上标}} \frac{1}{i \cdot j \cdot x} d \left\{ \int_{\Omega}^{\text{Emoji}} \frac{\phi}{x} \right\} \quad (6)$$

\propto
 $A \theta B$

6.3 常见的 limits 变种

注: 以下的展示均为行内公式

第一种, `limits` 选项: $\int_a^b f(x) dx$ $\sum_{i=1}^{+\infty}$

第二种, `nolimits` 选项: $\int_a^b f(x) dx$ $\sum_{i=1}^{+\infty}$

第三种: `displaylimits`: $\int_a^b f(x) dx$ $\sum_{i=1}^{+\infty}$

6.4 加边框

直接使用 `shadowbox` 就行了, 如果最求简单一点的盒子, 可以直接使用 `boxed` 命令, 此命令内部会自动进入数学环境:

$$\sum_{i=1}^{+\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$$

$$\sum_{i=1}^{+\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$$

6.5 `\ensuremath{}` 命令

先看两个例子:

命令: `\ensuremath {hello 你好}`

结果: *Hello*

命令: `{hello 你好}`

结果: Hello 你好

由此可以看出`\ensuremath{*args}`会自动进入数学环境,这一点在自定义命令和环境时是十分有用的。因为在`align`等常见环境中可以嵌套此环境,但是你自定义命令中的含有`$ ** $`等,嵌入前面的环境就会报错。

7 TikZ 和 GnUPlot 结合

在介绍怎么使用 TikZ 与 Gnuplot 之前,我们先说怎么编译。在 TeX 发行里, Gnuplot 并不包含在内。它的原理是 TikZ 调用 Gnuplot 之后,在编译的过程中,我们将会得到两个中间文件

- 1. filename.pgf-plot.gnuplot
- 2. filename.pgf-plot.table

其中,前者是 pgf-plot.table 的设定文件,而后者就是调用 Gnuplot 之后产生的一个数据表(点的坐标),然后 TikZ 调用这些点的坐标得到函数图。

为了能够使用 TikZ 与 Gnuplot,需要做两件事

- 1. 在电脑安装 Gnuplot
- 2. 命令行:`pdflatex --shell-escape filename.tex`,并且在 vscode 的配置文件中加上`-shell-escape`.

```

{
  "name": "xelatex",
  "command": "xelatex",
  "args": [
    "-synctex=1",
    "--shell-escape",
    "-interaction=nonstopmode",
    "-file-line-error",
    "%DOC%"
  ],
},

```

新加的

图 1: VScode 配置 GNU Plot

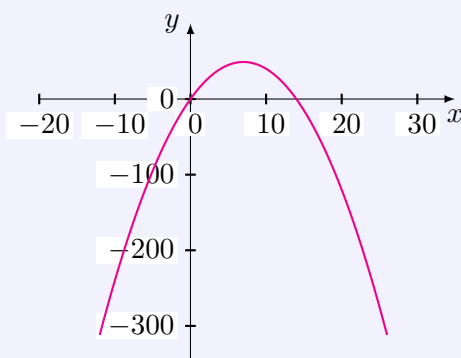
代码

```

\begin{tikzpicture}[>=stealth]
  \tkzInit[xmin=-20,xmax=30,
    ymin=-350,ymax=50,
    xstep=10,ystep=100]
  \tkzAxeXY
  \tkzFct[domain=-12:26,color=magenta,thick]{14*\x-\x**2}
\end{tikzpicture}

```

运行结果:



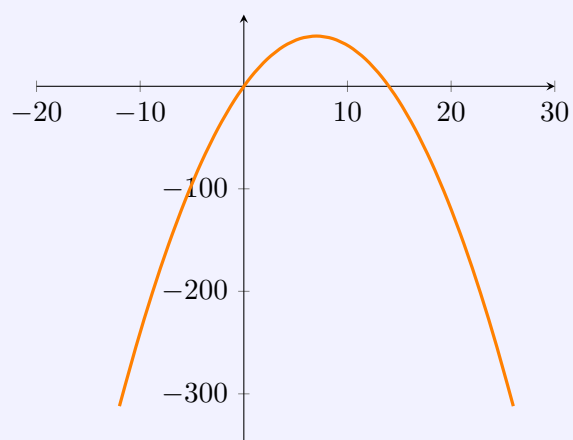
这个例子用到了 tikz-fct 这个宏包，这个宏包的缺陷:

- 1. 只有法文的手册，没法进行进一步的修饰。
- 2. 会出现字体的缺少警告。

由于上面的宏包没有英文版的手册，我们也只能看看别人的例子，下面给出 Gnuplot + pgfplots 结合的方法：代码

```
\begin{tikzpicture}
\begin{axis}[axis x line = middle,
             axis y line = middle,
             xmin=-20,xmax=30,
             ymin=-350,ymax=70]
\addplot[domain=-12:26,color=orange,very thick,smooth]
          gnuplot {14*x-x^2};
\end{axis}
\end{tikzpicture}
```

运行结果：



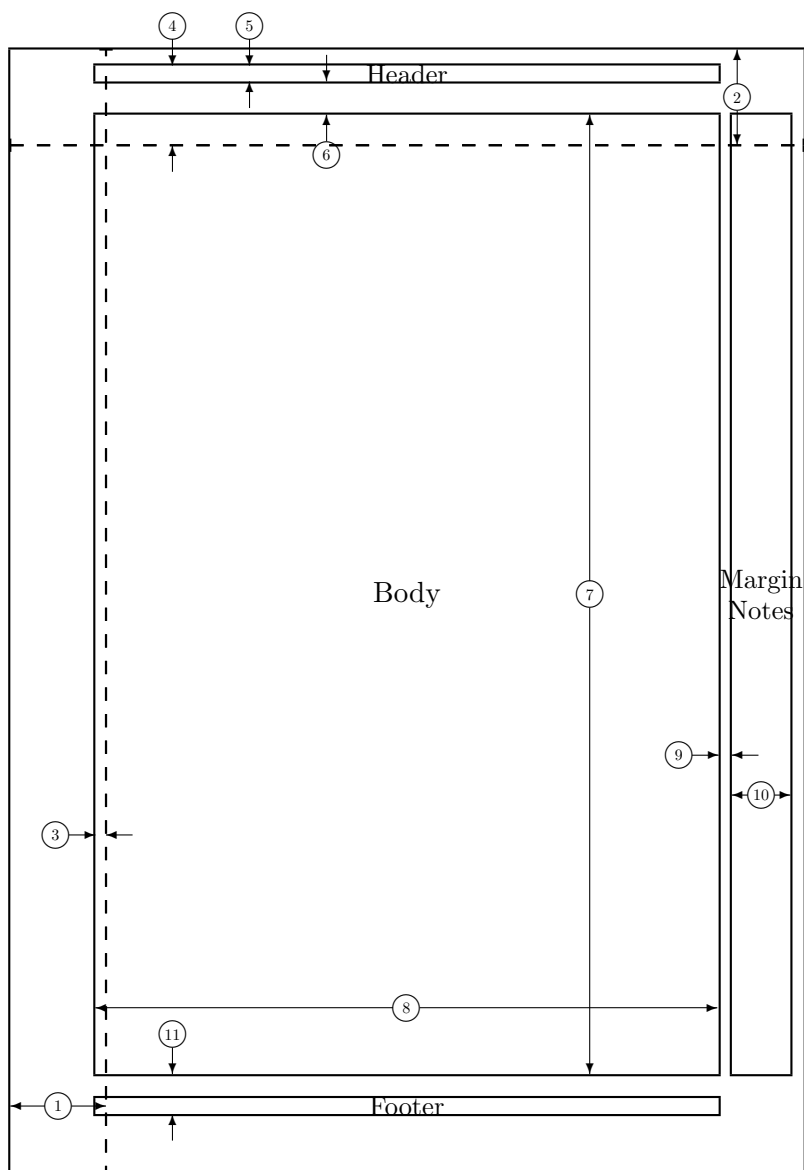
Remarque

Our universe is a wild and wonderful place. Join NASA astronauts, scientists, and engineers on a new adventure each week —all you need is your curiosity. Visit the Amazon rainforest, explore faraway galaxies and dive into our astronaut training pool. First-time space explorers welcome.

8 页面布局相关设置

8.1 打印页面布局

我们可以使用 `layouts` 宏包的 `layout` 和 `layout*` 命令可视化当前的页面布局



1	<code>one inch + \hoffset</code>	2	<code>one inch + \voffset</code>
3	<code>\oddsidemargin = -8pt</code>	4	<code>\topmargin = -60pt</code>
5	<code>\headheight = 12pt</code>	6	<code>\headsep = 25pt</code>
7	<code>\textheight = 722pt</code>	8	<code>\textwidth = 469pt</code>
9	<code>\marginparsep = 10pt</code>	10	<code>\marginparwidth = 44pt</code>
11	<code>\footskip = 30pt</code>		<code>\marginparpush = 7pt (not shown)</code>
	<code>\hoffset = 0pt</code>		<code>\voffset = 0pt</code>
	<code>\paperwidth = 597pt</code>		<code>\paperheight = 845pt</code>

8.2 图片任意放置

我们可以使用 `tikzpicture` 环境来进行图片的放置，本质上就是把 `node` 中的内容换位一张图片，然后在这个环境上添加一个 `overlay` 参数，基本参数的作用解释：

- `north west`: 页面的左上角
- `anchor`: 指定参考点来进行偏移
- `at`: 没有偏移时的位置

```
{  
  "name": "xelatex",  
  "command": "xelatex",  
  "args": [  
    "-synctex=1",  
    "--shell-escape",  
    "-interaction=nonstopmode",  
    "-file-line-error",  
    "%DOC%"  
  ],  
},
```

```
{  
  "name": "xelatex",  
  "command": "xelatex",  
  "args": [  
    "-synctex=1",  
    "--shell-escape",  
    "-interaction=nonstopmode",  
    "-file-line-error",  
    "%DOC%"  
  ],  
},
```

```
{  
  "name": "xelatex",  
  "command": "xelatex",  
  "args": [  
    "-synctex=1",  
    "--shell-escape",  
    "-interaction=nonstopmode",  
    "-file-line-error",  
    "%DOC%"  
  ],  
},
```

```
{  
  "name": "xelatex",  
  "command": "xelatex",  
  "args": [  
    "-synctex=1",  
    "--shell-escape",  
    "-interaction=nonstopmode",  
    "-file-line-error",  
    "%DOC%"  
  ],  
},
```

8.3 图片文字环绕

输出段落间距: \parindent

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

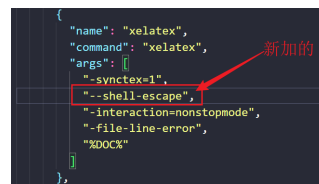


图 2: FIG 1

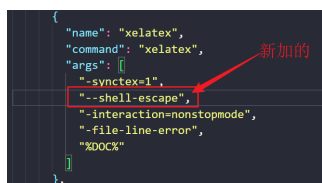


图 3: FIG 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

8.4 单双栏排版

L^AT_EX 支持简单的单栏或双栏排版。标准文档类的全局选项 `onecolumn`、`twocolumn` 可控制全文分单栏或双栏排版。L^AT_EX 也提供了切换单/双栏排版的命令：

```
\onecolumn
\twocolumn[<one-column top material>]
```

`\twocolumn` 支持带一个可选参数，用于排版双栏之上的一部分单栏内容。

切换单/双栏排版时总是会另起一页（`\clearpage`）。在双栏模式下使用 `\newpage` 会换栏而不是换页；`\clearpage` 则能够换页。

双栏排版时每一栏的宽度为 `\columnwidth`，它由 `\textwidth` 减去 `\columnsep` 的差除以 2 得到。两栏之间还有一道竖线，宽度为 `\columnseprule`，默认为零，也就是看不到竖线。

双栏排版时每一栏的宽度为 `\columnwidth`，它由 `\textwidth` 减去 `\columnsep` 的差除以 2 得到。两栏之间还有一道竖线，宽度为 `\columnseprule`，默认为零，也就是看不到竖线。

一个比较好用的分栏解决方案是 `multicol`，它提供了简单的 `multicols` 环境（注意不要写成 `multicol` 环境）自动产生分栏，如以下环境将内容分为 3 栏：

```
\begin{multicols}{3}
...
\end{multicols}
```

8.4.1 使用 `\twocolumn` 命令

缺点：无法在一页中使用单栏，双栏并存。使用 `\newpage` 进行换列

双腿瘫痪后，我的脾气变得暴怒无常。望着望着天上北归的雁阵，我会突然把面前的玻璃砸碎；听着听着李谷一甜美的歌声，我会猛地把手边的东西摔向四周的墙壁。母亲就悄悄地躲出去，在我看不见的地方偷偷地听着我的动静。当一切恢复沉寂，她又悄悄地进来，眼边红红的，看着我。“听说北海的花儿都开了，我推着你去走走。”她总是这么说。母亲喜欢花，可自从我的腿瘫痪后，她侍弄的那些花都死了。“不，我不去！”我狠命地捶打这两条可恨的腿，喊着：“我活着有什么劲！”母亲扑过来抓住我的手，忍住哭声说：“咱娘儿俩在一块儿，好好儿活，好好儿活……”可我却一直都不知道，她的病已经到了那步田地。后来妹妹告诉我，她常常肝疼得整宿整宿翻来覆去地睡不了觉。

那天我又独自坐在屋里，看着窗外的树叶“唰唰啦啦”地飘落。母亲进来了，挡在窗前：“北海的菊花开了，我推着你去看看吧。”她憔悴的脸上现出央求般的神色。“什么时候？”“你要是愿意，就明天。”她说。我的回答已经让她喜出望外了。“好吧，就明天。”我说。她高兴得一会坐下，一会站起：“那就赶紧准备准备。”“唉呀，烦不烦？几步路，有什么好准备的！”她也笑了，坐在我身边，絮絮叨叨地说着：“看完菊花，咱们就去‘仿膳’，你小时候最爱吃那儿的豌豆黄儿。还记得那回我带你去北海吗？你偏说那杨树花是毛毛虫，跑着，一脚踩扁一个……”她忽然不说了。对于“跑”和“踩”一类的字眼儿。她比我还敏感。她又悄悄地出去了。

她出去了。就再也没回来。

邻居们把她抬上车时，她还在大口大口地吐着鲜血。我没想到她已经病成那样。看着三轮车远去，也绝没有想到那竟是永远的诀别。

邻居的小伙子背着我去看她的时候，她正艰难地呼吸着，像她那一生艰难的生活。别人告诉我，她昏迷前的最后一句话是：“我那个有病的儿子和我那个还未成年的女儿……”

又是秋天，妹妹推我去北海看了菊花。黄色的花淡雅、白色的花高洁、紫红色的花热烈而深沉，泼泼洒洒，秋风中正开得烂漫。我懂得母亲没有说完的话。妹妹也懂。我俩在一块儿，要好好儿活……

赏析

世界上最痛苦的事情莫过于“子欲养而亲不待”，这样的痛楚太过绵长，痛得难以名状。

随着年龄的增长，时间长河的流逝，我们会慢慢离开父母温馨的怀抱。为学习，为工作，为生活，和父母甚至远隔万水千山。一家人在一起的日子终究是有限的，珍惜和父母在一起的日子。在面临突如其来的困难时，不要太过任性，不要自暴自弃，要知道如果你觉得痛苦了，父母所承受的痛苦是加倍的，爱自己，“好好活”，也是爱父母。

作者寥寥几百字就把自己对母亲的爱与自己少不更事的追悔挥洒得淋漓尽致，表现了母爱的无私、理解与伟大。

8.4.2 使用 multicolumn 宏包

- 1. 在 multipulcols* 环境中 -> 默认一个分栏排满以后在换栏。
- 2. 在 multicols(*) 环境中使用 \newpage 只会换到新的一页
- 3. 使用 \columnbreak 来换栏

伟大与渺小--臧克家

我们有太多的伟人。写在历史上的被渲染过的，不必说他们了；和我们同时代，向我们显示伟大的，已经够数了。这些人，凭了个人的阴谋机诈、凭了阴险与残酷，只要抓住一个机会使自己向高处爬一级，他是决不放弃这个机会的，至于牺牲个人的天良与别人的利害甚至生命，他毫不顾惜。这些伟人的伟大，是用个人的人性去换来的，是踏在人民大众的骨骼上升高起来的。当他站得高、显得伟大的时候，一般有肉没有骨头，有躯壳没灵魂的人中狗，便成群地蜷伏在他脚下，仰起头来望望他，便“伟大呵，伟大呵”地乱叫一阵子。当别人靠近他的时候，它们便猖狂吠起来，在壮主子的声威之余，自己仿佛也有威可畏了。这些伟人与臣侯是相依为命，狼狈为奸的。主子为了获取权势的兔，是不能没有走狗的，在走狗的瞳孔里，主子的尊容也许并非那样庄严，然而在他们口里又是另一回事了。为了一块骨头，它们出卖了自己。在伟人自己，眼睛看的是逢迎的脸色，咄咄赳赳的情感，耳朵听的是谄媚阿佞的声音，左右的人钢壁铁墙一样把他围在一个小天地里，眼看过不过咫尺，耳听不出左右，久而久之，也只能以他人之耳为耳，以他人之目为目。

而这些他人，又正是以他为法宝而有所贪图的人，他们所说的话，所报告的见闻，全是以自己的利害为标准而取舍，改窜，编辑的，不但与事实不符，常常会整个相反。

信假为真，以真为假，是非颠倒，黑白不分。古时候有这样的皇帝，天下大饥，他怪罪人民何不食肉糜，今日的伟人吃的鸡蛋也许还是一块钱一个。

这样的伟人，拔地几千尺，活在半空里，和群众、和现实，脱离得一干二净。在别人眼前，他作势，他装腔，他在别人眼里不是“人”，而是“伟人”。他自己，喜怒哀乐，不能自由，不愿自由，不敢自由，硬把人之所以为人的一些天性压抑，闷死，另换上一些人造的东西，这样弄得长久了，自己也觉得自己不是“人”了，而成了“人”以上的另一种人的“人”。勉强解释，就是孤家“寡人”之“人”。这样的“人”，是“性相近也，习相远也”，远的是民众，是人性。这样的人是刚愎的，残暴的，虚伪的，反动的，半疯狂的，自欺欺人的，存心“不令天下人负我，我负天下人”的。把一个国家，一个世界，交给这样一个半疯子去统治，那会造成个什么样子呢？

赏析

苍茫大地，芸芸众生中，每个人都是渺小的，不需要因为自己的渺小而感到自卑，渺小是渺小的妙处，每个人都有自己的位置，都能找到自己的光源，发出自己的声音。要知道蓝天有自己的深邃，白云有自己的飘逸，草原有自己的芬芳，就连不起眼的小草也有属于自己的翠绿，每个人都是独特而优秀的

8.5 浮动体

内容丰富的文章或者书籍往往包含许多图片和表格等内容。这些内容的尺寸往往太大，导致分页困难。L^AT_EX 为此引入了浮动体的机制，**令大块的内容可以脱离上下文，放置在合适的位置**。L^AT_EX 预定义了两类浮动体环境 figure 和 table。习惯上 figure 里放图片，table 里放表格，但并没有严格限制，可以在任何一个浮动体里放置文字、公式、表格、图片等等任意内容。

各个参数的理解

$\langle placement \rangle$ 参数提供了一些符号用来表示浮动体允许排版的位置，如 **hbp** 允许浮动体排版在**当前位置、底部或者单独成页**。table 和 figure 浮动体的默认设置为 tbp。

表 1: 浮动体的位置参数

参数	含义
h	当前位置（代码所处的上下文）
t	顶部
b	底部
p	单独成页
!	在决定位置时忽视 限制

注 1: 排版位置的选取与参数里符号的顺序无关，L^AT_EX 总是以 h-t-b-p 的优先级顺序决定浮动体位置。也就是说 [!htp] 和 [ph!t] 没有区别。

注 2: 限制包括**浮动体个数**（除单独成页外，默认每页不超过 3 个浮动体，其中顶部不超过 2 个，底部不超过 1 个）以及**浮动体空间占页面的百分比**（默认顶部不超过 70%，底部不超过 30%）。

9 盒子

一般的水平盒子，竖直盒子在 lshort 上面都讲地清清楚楚的，具体参见 lshort。盒子是 L^AT_EX 排版的基础单元，虽然解释上去有些抽象：每一行是一个盒子，里面的文字从左到右依次排列；每一页也是一个盒子，各行文字从上到下依次排布，颇有一些活字印刷术的味道。

不管如何，L^AT_EX 提供了一些命令让我们生成一些有特定用途的盒子。

9.1 水平盒子

```
\mbox{...}
```

```
\makebox[ $\langle width \rangle$ ][ $\langle align \rangle$ ]{...}
```

\mbox 生成一个基本的水平盒子，内容只有一行，不允许分段（除非嵌套其它盒子，比如后文的垂直盒子）。外表看上去，\mbox 的内容与正常的文本无二，不过断行时文字不会从盒子里断开。

测试

生成一个基本的水平盒子，内容只有一行，不允许分段（除非嵌套其它盒子，比如后文的垂直盒子）。外表看

结论：从上便可以看出，\\ 不起作用，也不会自动换行

9.2 自定义水平盒子

对齐选项有：

- c: 居中对齐 (默认)
- l: 左对齐
- r: 右对齐
- s: 分散对齐

测试

1. Test 10em box
2. Test 10em box
3. Test 10em box¹

水平盒子加边框

```
\fbox{...}
```

```
\framebox[⟨width⟩][⟨align⟩]{...}
```

`\boxed{...}` 通过 `\setlength` 命令调节边框的宽度 (向外拓展) `\fboxrule` 和内边距 (保证上下内边距) `\fboxsep`

```
%% 1.\boxed{}会自动进入数学环境
%% \boxed{} 和 \fbox{} 不能指定宽度和对齐
%% 2. ~会输出硬空格
\boxed[10em]{Hello~}\
\boxed{Hello~}\
\boxed{\sum_{i=0}^{\infty}\frac{1}{i^2}}\
\fbox{Hello Jack}\
\framebox[3em][l]{Helloa}\
\framebox[3em][l]{MMM}\

\framebox[6em][l]{Hello}

\setlength{\fboxrule}{5mm}
\framebox[6em][l]{Hello}\

\setlength{\fboxsep}{10mm}
\framebox[6em][l]{Hello}
```

[10em]Hello

Hello

$$\sum_{i=0}^{\infty} \frac{1}{i^2}$$

Hello Jack

Helloa

MMM

Hello

ello

Hello

¹分散对齐方式强行拉开单词的间距，往往会报 Underfull \hbox 的消息

9.3 竖直盒子

一般是可以换行的

```
\parbox[⟨align⟩][⟨height⟩][⟨inner-align⟩]{⟨width⟩}{...}  
\begin{minipage}[⟨align⟩][⟨height⟩][⟨inner-align⟩]{⟨width⟩}  
...  
\end{minipage}
```

测试

```
测试:\parbox[b][2em][r]{3em}%  
{你好, what you do, Thanks}
```

你 好,
what
you

do,
测试:Thanks

你 好,
what
you

```
测试:\parbox[b][2em][r]{3em}%  
{你好, what you do, Thanks}
```

do,
测试:Thanks

你 好,
what
测试:
you do,
Thanks

```
测试:\parbox[t][2em][r]{4em}%  
{你好, what you do, Thanks}
```

测试:你 好,
what
you do,
Thanks

```
测试:\parbox[t]{4em}%  
{你好, what you do, Thanks}
```

测试:你 好,
what
you do,
Thanks

```
\fbox{  
  测试:\parbox[b][2em][c]{8em}%  
  {你好, what you do, Thanks}  
}
```

你好, what you do,
测试:Thanks

```
\textcolor{red}{测试}:\parbox[t]{8em}%
{北美洲原始居民为印第安人。16-18世纪，
正在进行资本原始积累的西欧各国相继入侵北美洲。
到了十八世纪中期，在北美大西洋沿岸建立了
十三块殖民地，殖民地的经济，文化，政治相对
成熟。}
```

测试:北美洲原始居民为印第安人。16-18世纪，正在进行资本原始积累的西欧各国相继入侵北美洲。到了十八世纪中期，在北美大西洋沿岸建立了十三块殖民地，殖民地的经济，文化，政治相对成熟。

```
\textcolor{red}{测试}:\parbox[b]{8em}%
{北美洲原始居民为印第安人。16-18世纪，
正在进行资本原始积累的西欧各国相继入侵北美洲。
到了十八世纪中期，在北美大西洋沿岸建立了
十三块殖民地，殖民地的经济，文化，政治相对
成熟。}
```

北美洲原始居民为印第安人。16-18世纪，正在进行资本原始积累的西欧各国相继入侵北美洲。到了十八世纪中期，在北美大西洋沿岸建立了十三块殖民地，殖民地的经济，文化，政

测试:治相对成熟。

总结, 一般使用格式

1. `\parbox` [`[<parbox 盒子和前边盒子的对齐方式>]`]{<text>}

2.

`\begin{minipage}`

`[<minipage 盒子和前边盒子的对齐方式>]`

`[<[<minipage 盒子的高度>]>]`

`[<minipage 盒子内部的对齐方式>]`

`{<[<minipage 盒子的宽度>]>}`

<text>

`\end{minipage}`

9.4 标尺盒子

`\rule` 命令用来画一个实心的矩形盒子，也可适当调整以用来画线（标尺）：

```
\rule[⟨raise⟩]{⟨width⟩}{⟨height⟩}
```

Black `\rule{12pt}{4pt}` box.

Upper `\rule[4pt]{6pt}{8pt}` and
lower `\rule[-4pt]{6pt}{8pt}` box.

A `\rule[-.4pt]{3em}{.4pt}` line.

Black  box.

Upper  and lower  box.

A  line.

本质上也就是画一个粗的实心矩形，具体的使用样例如下：我绘制了一个宽`0.8\linewidth`，高`5pt`，离开基线`0pt`的一个居中矩形。

代码如下：

```
\begin{center}
  \textcolor{blue}{
    \rule[0pt]{0.8\linewidth}{5pt}
  }
\end{center}
```

绘制效果：



9.5 缩放盒子

我们可以使用缩放盒子完成一些很伟大的操作。还记得有一次我想用 LaTeX 的 logo 绘制一个封面，结果绘制出来的那个 logo 太小了，只有放大 AI 里面操作一番之后再在插入进来，极尽狼狈。现在直接使用 LaTeX 的 `graphicx` 宏包使用的缩放功能就行了，可以实现对几乎任何的对象进行缩放：图片，文字，数学符号，... 自定义的源代码演示：

```
\newcommand{\scale}[3]{
  \scalebox{#1}[#1]{#2}
}
```

具体的使用效果：放大了 3 倍的 TeX logo

TeX

Σ

9.6 旋转盒子

这个 `rotatebox` 命令必须放在 `figure` 环境中，从下面也可以看出，`box` 对象，旋转的基点是左下角，在基线上

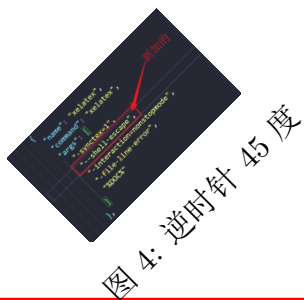


图 4: 逆时针 45 度

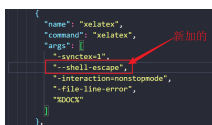


图 5: 没有旋转

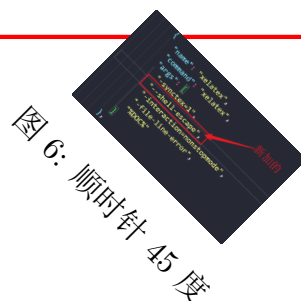


图 6: 顺时针 45 度

基线

10 一些 primitive

当我们想要魔改一些东西的时候，我们必须的使用内部的宏，这是不可避免的。下面介绍几个基本的原语 (primitive): 我们通过一个例子来引入:

10.1 \TeX 的背后

下面为一些尝试的命令

命令一: $\text{\TeX}\kern-0.2em\lower.25em\hbox{E}\kern-0.1em X$

命令二: $\text{\TeX}\ensurmath{\mathrm{L}\kern-.325em{\scriptstyle A}}\kern-.17em\}$

运行效果: \TeX \LaTeX

正如上面的代码所示，主要就是一个 \lower 宏，使用 $+$, $-$ 可以实现上下偏移。例如，我们把 \lower 后面的正号变为 $-$ 时，E 机会出现在上面， \LaTeX 中的 A 同理

效果演示一: \TeX

效果演示二: \LaTeX

10.2 primitive 介绍

- 1. `\kern`: 其中 - 表示向左移动
- 2. `\lower`: 其中 - 表示向上移动
- 3. `\raise`: 其中 - 表示向下移动
- 4. `\hbox`: 表示水平盒子

LaTeX 的原始命令

```
\DeclareRobustCommand{\LaTeX}{L\kern-.36em%
  {\sbox\z@ T%
    \vbox to\ht\z@{\hbox{\check@mathfonts
      \fontsize\sf@size\z@
      \math@fontsfalse\selectfont
      A}%
    \vss}%
  }%
  \kern-.15em%
  \TeX}
```

11 魔改系列

11.1 进制

在 TeX 的内部使用 ' (单引号) 后加数字表示八进制, " (双引号) 后接数字表示十六进制。下面便有几个示例。注: 我们使用原语 (primitive) 的 `\number` 命令进行十进制输出。

八进制: `\number'10` 结果: 8

十进制: `\number10` 结果: 10

十六进制: `\number"10` 结果: 16

11.2 TeX 的字符与编码

TeX 内部的字符是有各自的编码的, 下面先演示一句话:

神秘代码: `^^(^^^%^^,^^,^^/^^1^^^7^^/^^2^^,^^$^^a`

运行结果: hello, world!

实际的对应规则: 就是下面那个 ASCII 码表左右两列对应字符的相互对应具体的规则如下:

- 1. 连续两个上标符号 `^^` 后跟一个字符 `c`, 可以表示另一个字符 `c'`
 - 1.1 $64 \leq \text{ASCII}_c \leq 127 \Rightarrow \text{ASCII}_{c'} = \text{ASCII}_c - 64$
 - 1.2 $0 \leq \text{ASCII}_c \leq 63 \Rightarrow \text{ASCII}_{c'} = \text{ASCII}_c + 64$

ASCII 码表

控制字符				符号、数字				大写字母、符号				小写字母、符号			
00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	\$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[6B	k	7B	{
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

18

图 7: ASCII

- 2. `^^` 后跟两个小写十六进制数字，可以表示任意 ASCII 符号

下面演示第二点：

代码：`^^68^^65^^6c^^6c^^6f^^2c^^20^^77^^6f^^72^^6c^^64^^21`

运行结果：hello, world!

11.3 字符 & 编码的转换

我们可以使用原语 `\number` 轻松的输出字符对应的 ASCII 数值¹

- 1. 字符 \Rightarrow 编码：``<字符>`
- 2. 编码 \Rightarrow 字符：`\char<编码>`

注意

编码转换使用的是前引号：```，但是 8 进制使用的是后单引号：`'`

下面展示关于字符到编码的几个使用样例：

代码：`\number`A` 运行结果：65

代码：`\number`\A` 运行结果：65

代码：`\number"41` 运行结果：65

代码：`\number`%`` 运行结果：37

但是如果我们输入了 `\number`人`，即一个非 ASCII 字符²那么会得到什么东西呢？

¹注：ASCII 中的字符和十进制之间的相互转换

²注意：pdf_lat_ex 不支持 Unicode 编码

代码: `\number`人` 运行结果: 20154
 其实此时它会按照 Unicode 字符集进行映射。

下面再展示几个编码到字符的转换样例:

代码: `\char65` 运行结果: A

代码: `\char"41` 运行结果: A

代码: `\char20000` 运行结果: 北

我们可以使用 `\char` 命令进行一些不方便使用输入法键入的繁体字的录入, 前提是我们得知道它的 Unicode 编码³。所以必要时你需要指定一个支持中文, 且支持中文数量比较的字体, 使用如下的语句进行设置:

```
\usepackage{fontspec}
\setmainfont{SimSun}
```

11.4 类别码

相关的注意事项, 主要就是下面的两张表

类别码			
类别码	含义	符号	ASCII 码 (HEX)
0	转义符	<code>\</code>	5C
1	组开始	<code>{</code>	7B
2	组结束	<code>}</code>	7D
3	切换数学环境	<code>\$</code>	24
4	表格对齐	<code>&</code>	26
5	回车	<code>CR</code>	0D
6	参数	<code>#</code>	23
7	上标	<code>^</code>	5E
8	下标	<code>_</code>	5F
9	可忽略字符	<code>NUL</code>	00
10	空格	<code>SP</code>	20
11	字母	<code>A ~ Z, a ~ z</code>	-
12	其他字符	本表未列出的其他符号	-
13	活动字符	<code>~</code>	7E
14	注释	<code>%</code>	25
15	无效符	<code>DEL</code>	7F

21

图 8: catcode

³默认情况下 xelatex 使用的字体是 latin morden, 其中只有拉丁字母, 不含中文

- 9 (可忽略字符)
TeX 遇到它后将直接跳过，就仿佛它不在那里一样
- 13 (活动字符)
本身就是一个宏。例：`~ = \nobreakspace`
- 15 (无效符)
TeX 遇到它也会跳过，但会打印如下错误信息：

```
! Text line contains an invalid character.
```

22

图 9: catcode2

11.5 @ 溯源

首先展示一段代码：

```
\makeatletter
\number\@ne

\number\thr@@
\makeatother
```

它的运行结果如下：

```
1
3
```

其实在 L^AT_EX2_ε 中：

```
\@ne = 1          \tw@ = 2
\thr@@ = 3        \sixt@@n = 16
```

还有其它的很风骚的命名，具体可以参见
官方文档：macros2e

tex 将字符分成 16 类 (catcode)，其中第 11 类是 letter(a-z, A-Z)，第 12 类是 other，就是除了其它 15 类的那种。tex 命令只能由 letter 组成（这也是很多人在 macro 名中用数字出错的原因，数字就属于 other 类）。@ 默认属于 other 类，故不能在 macro 的名字中出现。LaTeX 的内部命令中，\makeatletter 命令的实质就是修改字符 @ 的 catcode 为 11，这样 @ 就可以出现在命令名称中了。 \makeatother 重新修改 @ 的 catcode 为 12，不允许 @ 出现在命令的名字中。

11.6 catcode 背锅

catcode 就是 TeX 的一个黑魔法，当你学会了使用 catcode，并且随意使用这玩意之后，你的 LaTeX 代码就没有人能够看得懂了。

下面开始使用 catcode 命令魔改 TeX：

\catcode ——类别码的切换

- 语法：

```
\catcode⟨字符编码⟩=⟨类别码⟩ % “=” 可以忽略
```

- 例子：

```
\catcode`\\"=\active
%      0=|      7=#      3=}      2={
\catcode`00 \catcode`76 \catcode`32 \catcode`21
% |def"#1"{{\textbf{#1}}
\def"71"20\textbf{27133
This is an "important" word.
```

⇒ This is an **important** word.

24

图 10: catcode3