LaTeX 学习笔记

殷元昊

1 前言

这份笔记的前身来自笔者学习吴康隆的《简单高效 LATEX》的笔记。笔者按照本书的结构学习了大部分代码，在学习的同时进行了大量的随文注释。但是在回看的时候，这些代码和随文注释混杂在一起，极大地影响了可读性。因此笔者深感有必要将这些随文注释整理出来，形成单独的 LaTeX 文档。

在吴书之外，笔者还另外单独学习了一些有用的 LaTeX 宏包，并也做了大量的随文注释，这些注释也需要整理出来。

值得一提的是，由于 AI 工具的兴起，代码学习的效率被大大提高，笔者在编辑本文档时，也仍然在使用 AI 学习新的知识，并立刻将其应用到本文档中

笔者曾经在学习制作表格和插入图片的相关宏包时制作过两份 LaTeX 文档，但是回看的时候仍然觉得可读性不强，这是因为在编写文档的时候，尽管笔者使用了详细的文字描写相关命令，但由于缺少命令的参数结构，因而在回看时仍不免陷入“迷失在文字的海洋里”的尴尬境地，因此，该笔记将严格按照“宏包-位置-参数-注释”的架构搭建。

2 正文

封面

命令

`\title{}`、`\author{}`、`\date{}`、`\thanks{}`

宏包 基础命令

位置 略

参数 略

注释

1. `\date{}`如果省略，仍然会自动打印出编译当天的日期，如果不想显示日期，可以保留`\date{}`，但不输入参数。

2. 标题页的脚注用\thamks{}完成。

标题

命令

\maketitle

宏包 基础命令

位置 略

参数 略

注释 略

目录

命令

\tableofcontents、\listoffigures、\listoftables

宏包 基础命令

位置 略

参数 略

注释 略

目录标题

命令

\contentsname、\listfigurename、\listtablename

宏包 基础命令

位置 略

参数 略

注释 这三条命令用于\renewcommand{}{}结构中，通过重定义来修改目录的标题、图片目录的标题、表格目录的标题。

保留字符

命令

(1) \# % 输出 #
(2) \$ % 输出 $
(3) \% % 输出%
(4) \& % 输出 &
(5) _ % 输出 _
(6) \{ % 输出 {
(7) \} % 输出}
(8) \^{} % 输出 ^

宏包 基础命令

位置 略

参数 略

注释

1. 关于% 符号，如果在 LaTeX 的代码中换一行，打印时会在换行处插入一个空格，而此时如果在该行末尾插入一个% 符号，就能移除这个空格。
2. 关于 ^ 符号，该命令后面如果不加一对大括号，单独打印时会报错，而如果在大括号中填入一个字母（此时也可以不需要大括号），输出的就是一个扬抑符（circumflex），比如\^{a}输出 â。

保留字符：反斜杠

命令

(1) \textbackslash
(2) \textrm{\char92}、{\rmfamily\char92} % 使用 ASCII 码进行输出
(3) $backslash$ % 使用数学环境

宏包 基础命令

位置 略

参数 略

注释

1. 方法二中的\textrm{}也可以替换为其他字体命令。
2. 方法二、方法三输出的反斜杠和方法一输出的反斜杠不完全相同：如果将这几种方法输出的反斜杠排列在一行上，会发现方法一输出的反斜杠和后面反斜杠符号的间距更小，但是有意思的是，如果将反斜杠后面的符号替换成其他符号，这些反斜杠后的这一间距又会恢复相同，因此在实际情况中，只需要在需要连续使用两个反斜杠符号时，注意选择命令即可，比如，如果要输出“\\”符号，应该选用两个连续的\textbackslash命令。
3. 顺便在此可以讨论\texttt{}和{\ttfamily}这两类不同的命令，这两类命令的大括号的位置和功能是不同的：前一种命令的大括号置于命令之后，是强制性的，用来放置相应的参数，只有大括号内的内容才会受到该命令的影响，其特点有点类似于环境。有些教程会将这种命令称为“argument style”（参数型）。

   而后一种命令的大括号不是强制性的，如果没有这一对大括号，则该命令之后的所有内容都会受到该命令的影响，如果需要控制该命令的统辖范围，则可以在该命令和需要统辖的范围两边加上大括号。有些教程会将这种命令称为“declarative style”（声明型）。

   本文档中会反复出现这两类命令，比如自定义距离命令、两类对齐方式命令等，以下只通过大括号的位置来体现这两类命令的类型，如果没有必要，不再就这一点展开具体讨论。

保留字符：波浪线

命令

(1) \textasciitilde
(2) \~{}
(3) sim % 实际是一个数学符号“约等于”（similar）

宏包 基础命令

位置 略

参数 略

注释

1. 方法一在不同的编辑器中输出效果不同，在 TeXstudio 中输出的是一个腭化符，而在 VS Code 中输出的却是一个正常的波浪线。

2. 方法二和上文中的 ^ 符号一样，该命令如果后面跟上一个字母，实际输出的就是一个波浪号/腭化符（tilde），如果要单独输出这一符号，必须在后面添加一对大括号，在 TeXstudio 中输出的是一个单独的腭化符，但是在 VS Code 中输出的就是一个正常的波浪线，比如`\~{a}`输出 ã。

大于号和小于号

命令

(1) `\textgreater`、`\textless`
(2) `>`、`<`
(3) `$>$`、`$<$`

宏包 基础命令

位置 略

参数 略

注释

1. 文本中的大于号和小于号需要使用方法一中的`\textgreater`和`\textless`命令。

2. 方法二和方法三输出的都是数学符号中的大于号和小于号。TeXstudio 中直接输入大于号和小于号不会正确打印相应的符号，但是 VS Code 中正常，效果等同于两边加上数学环境。

引号

命令

“你好，‘世界’!” v.s. ``` ``\thinspace`Max' is here.'' ```

宏包 基础命令

位置 略

参数 略

注释

1. 中文下的单引号和双引号可以用中文输入法直接输入。英文的左单引号是重音符“`”，右单引号是常用的引号符“'”。
2. 吴书中提到上述“英文下的引号嵌套需要借助\thinspace命令分隔”，但实际上这和语言无关，\thinspace的效果是在命令处略微扩大外部引号和内部引号之间的距离，只是一个细节问题。

连字符、破折号和省略号

命令

```
(1) - % 连字符：daughter-in-law
(2) -- % 数字起止符：1--2
(3) --- % 英文破折号：Listen---I'm serious.
(4) —— % 中文破折号
(5) …… % 中文省略号
(6) \ldots % 英文省略号
```

宏包 基础命令

位置 略

参数 略

注释 \ldots = lower dots, v.s. \cdots = center dots。注意连打三个句点“...”输出的不是真正的英文省略号：\ldots输出…，“...”输出...。

强调

命令

```
(1) \emph{强调} % 输出：强调
(2) \emph{empasis} % 输出：empasis
```

宏包 基础命令

位置 略

参数 略

注释 中文的效果等同于`\textsl{}`，西文的效果等同于`\textit{}`。

下划线和删除线

命令

`\underline{}` % 输出：下划线

宏包 基础命令

位置 略

参数 略

注释 略

下划线和删除线

命令

(1) `\uline{}` % 下划线
(2) `\uuline{}` % 双下划线
(3) `\dashuline{}` % 虚下划线
(4) `\dotuline{}` % 点下划线
(5) `\uwave{}` % 波浪线
(6) `\sout{}` % ~~删除线~~、
(7) `\xout{}` % ~~删除线~~

宏包 ulem

位置 略

参数 略

注释 “ulem”来自“ul (underline)”和“em (emphasis)”的合写。“sout”来自“strike out”的缩写。“xout”来自“cross out”的缩写。

长度单位

命令

```
(1) pt % point, 磅
(2) pc % pica, 1 pc = 12 pt, 四号字
(3) in % inch, 英寸, 1 in = 72.27 pt
(4) bp % bigpoint, 1 bp = 1/72 in
(5) cm % centimeter, 厘米, 1 cm = 1/2.54 in
(6) mm % millimeter, 毫米, 1 mm = 1/10 cm
(7) sp % scaled point, TeX 的基本长度单位, 1 sp = 1/65536 pt
(8) em % 当前字号下, 大写字母 M 的宽度
(9) ex % 当前字号下, 小写字母 x 的高度
(10) \textwidth % 页面上文字的总宽度, 即页宽减去两侧边距
(11) \linewidth % 当前行允许的行宽
```

宏包 基础命令

位置 略

参数 略

注释 略

空格

命令

~

宏包 基础命令

位置 略

参数 略

注释 ~输出效果等同于一个空格，并且在此空格之后不会换行，这样可以使空格前后内容始终在同一行上。

换行和分段

命令

\par、\\、\newline、\mbox{}

宏包 基础命令

位置　略

参数　略

注释

1. 分段有两种方式：（1）在两段之间空一行；（2）在两段之间使用\par命令，新的段落开头空两格打印。
2. 强制换行有两种方式：（1）在换行处使用\\命令；（2）在换行处使用\newline命令，下一行顶格打印。
3. 打印一个空白段落的方式：在空白段落处输入\mbox{}命令。注意空白段落和前后段落之间也要有空行或者\par命令。该命令还有一个功能是输入参数，防止放入其中的词在换行时断开。

段落间距

命令

```
\parskip % 当前默认值：0.0pt plus 1.0pt
\setlength{\parskip}{30pt} % 自定义段落距离
```

宏包　基础命令

位置　略

参数　略

注释

1. 当前默认值可以通过\the\parskip得到。“0.0 pt plus 1.0 pt”是弹性距离，意即普通情况下是 0pt，在有需要的时候可以拉伸到 1pt，相应的也可以设置收缩距离（minus）
2. “段落间距”其实是一个差值，是【前一段的最后一行的箱子和后一段的第一行的箱子的基线间距】减去【同一段内的基线间距】之后的差值，具体可以参见本文档中关于“基线间距”（baselineskip）的讨论。
3. 顺便可以讨论一下\setlength{}{}这一类命令的统辖范围，\setlength{}{}如果放到导言区，统辖范围是正文，直到在正文中设置新的相应长度为止。{\setlength{}{}...}如果放在正文当中，则为声明型命令，可以通过在命令和统辖范围两边加大括号来控制统辖

范围。本文档中还会出现其他的\setlength{}{}命令，不再另外讨论。

4. \parskip控制的对象是：（1）统辖范围内的每一段和其前面一段之间的距离；（2）统辖范围内的第一段和统辖范围之前的最后一段之间的距离，但不包括其和统辖范围之后的第一段之间的距离，这段距离会恢复为原来的长度，因为这个段落间距是由统辖范围之后的第一段来控制的，关于行距的设置的也具有类似的特点。

首字下沉

命令

```
\lettrine % 输出效果：THIS is an example.
```

宏包 lettrine

位置 略

参数 略

注释 略

分页

命令

```
\newpage
\newpage\mbox{}\newpage % 可以达到空出一整页的效果
```

宏包 基础命令

位置 略

参数 略

注释 \newpage命令的功能是从当前行开始到当前页的最后不再打印内容，而直接从下一页开始打印后续内容。

段首缩进

命令

```
\parindent % 当前默认值：15.0pt
\setlength{\parindent}{10em} % 自定义段首缩进距离
\noindent % 强制取消缩进
```

宏包 基础命令

位置 略

参数 略

注释

1. 当前默认值可以通过`\the\parindent`得到。
2. `\noindent`只会影响到其后的一个段落，而不会导致其后所有的段落都取消缩进，也不需要大括号来控制其统辖范围。

对齐方式

命令

```
(1.1) \begin{flushleft}...\end{flushleft} % 左对齐
(1.2) \begin{flushright}...\end{flushright} % 右对齐
(1.3) \begin{center}...\end{center} % 居中对齐
(2.1) {\raggedright...\par} % 左对齐
(2.2) {\raggedleft...\par} % 右对齐
(2.3) {\centering...\par} % 居中对齐
```

宏包 基础命令

位置 略

参数 略

注释

1. “ragged”是“凹凸不平”的意思，以`\raggedleft`为例，其意思是文本的左侧是凹凸不平的，意即“右对齐”。
2. (1) 中都是参数型命令，(2) 中都是声明型命令。吴书中提到，`\centering`（包括`\raggedleft`、`\raggedright`）命令“常常用在环境内部（或者一对花括号内部）”，但这些都是“老旧的命令”，还是建议优先使用参数型命令。

3. \centering、\raggedleft、\raggedright如果用大括号控制了统辖范围，则统辖范围内的最后一段后面需要空行或者加上\par才能产生效果，否则最后一段的对齐方式将无法生效，如果没有用大括号控制统辖范围，则其后的所有内容的对齐方式都生效。另一条\selectfont...命令也有类似特点。

字体：字族（family）

命令

```
\familydefault % 默认字族：lmr，“lmr”意即“Latin Modern Roman”
\textrm{}、{\rmfamily...} % 罗马字族：Roman
\textsf{}、{\sffamily...} % 无衬线字族：Sans Serif
\texttt{}、{\ttfamily...} % 等宽字族：Typewriter
```

宏包 基础命令

位置 略

参数 略

注释

1. \familydefault控制的是当前文档中的默认字族，一般就是\rmdefault，而\rmdefault的默认值就是“lmr”，同理，也有\sfdefault和\ttdefault，其默认值分别为“lmss”和“lmtt”。而\textrm{}/{\tmfamily}则是字体命令，会调用\rmdefault进行打印，其余两类字体命令同理。可以通过\renewcommand{}{}来重定义\familydefaut或者\rmdefault等命令的值。
2. \mintinline{}{}中的字族会变为 tt 字族（如上文所示），在正文中，默认使用 rm 字族。

字体：字系（series）

命令

```
\seriesdefault % 默认字系：m，“m”代表“Medium weight and width”
\textbf{}、{\bfseries...} % 粗体字系：BoldSeries
\textmd{}、{\mdseries...} % 中粗体字系：MiddleSeries
```

宏包 基础命令

位置　略

参数　略

注释　`\mintinline{}{}`中的字系默认使用 md 字系，和正文中相同，但由于字族变为了 tt 字族，因此不容易看出 bf 字系和 md 字系的区别，此处分别给出 rm 字族和 sf 字族下搭配 bf 字系和 md 字系的区别：
rm 字族：MiddleSeries v.s. **BoldSeries**
sf 字族：MiddleSeries v.s. **BoldSeries**

字体：字形（shape）

命令

```
\shapedefault % 默认字形：n，“n”代表“Normal”
\textup{}、{\upshape...} % 竖直字形：Upright
\textsl{}、{\slshape...} % 斜体字形：Slant
\textit{}、{\itshape...} % 强调体字形：Italic
\textsc{}、{\scshape...} % 小号大写体字形：SMALLCAPITAL
```

宏包　基础命令

位置　略

参数　略

注释　`\mintinline{}{}`中的字形默认使用 up 字形，和正文中相同，此处分别给出 rm 字族和 sf 字族下搭配各个字形的区别：
rm 字族：Upright、*Slant*、*Italic*、SMALLCAPITAL
sf 字族：Upright、*Slant*、*Italic*、SMALLCAPITAL
可以发现，sf 字族下的 sc 字形和 rm 字族下没有区别，这是因为 sf 字族下的 sc 字形没有相应的设计。

相对字号

命令

```
(1) {\tiny...} % tiny
(2) {\scriptsize...} % scriptsize
(3) {\footnotesize...} % footnotesize
(4) {\small...} % small
(5) {\normalsize...} % normalsize
```

(6) {\large...} % large
(7) {\LARGE...} % LARGE，注意\LARGE全大写
(8) {\huge...} % huge
(9) {\Huge...} % Huge，注意\Huge只有“H”大写

宏包 基础命令

位置 略

参数 略

注释 正文当中默认使用{\normalize}字号。这些字号命令的默认值取决于\documentclass{}，比如 article 中{\normalsize}的默认值是 10 pt。可以通过定义一个显示当前字体的命令\showfontsize来获得：

```
\makeatletter
\newcommand{\showfontsize}{\f@size{} pt}
\makeatother
```

当前字体为：10 pt

默认字体

命令

{\normalsieze...} v.s. {\normalfont...}、\textnormal{}

宏包 基础命令

位置 略

参数 略

注释 {\normalsize...}用来将字体还原回默认字号，但不能修改其他的字族、字形、字系等特征，{\normalfont...}或者\textnormal{}用来将字体还原回除了字号以外的默认字体特征，因此这两条命令是互补的，如果搭配使用，则将字体全方面还原回默认值，可以比较下面的字体：

{\large\bfseries\itshape ABC}：***ABC***
{\large\bfseries\itshape\normalsize ABC}：***ABC***
{\large\bfseries\itshape\normalfont ABC}：ABC
{\large\bfseries\itshape\normalsize\normalfont ABC}：ABC

设置字号和行距

命令

```
\fontsize{fontsize}{baselineskip}{\selectfont...\par}
```

宏包 基础命令

位置 略

参数

1. fontsize = 字号
2. baselineskip = 行距（基线间距）

注释

1. `\fontsize{}{}`只能放在正文当中，放在导言区不会生效。
2. `\selectfont`类似一个声明型命令，只有其后面的所有内容才会受到`\fontsize{}{}`的影响，为了控制其统辖范围，可以在该命令和统辖范围的两边加上大括号。与同为声明型命令的`{\raggedleft...\par}`、`{\setlength{}{}...}`相比，`{\selectfont...\par}`更像前者，这两个命令都要求用大括号控制的统辖范围内的最后一段在最后空行或者加上`\par`，否则最后一段不会被包括进统辖范围。总的来说，如果出现用大括号控制声明型命令统辖范围的情况，在最后一段的末尾空行或者加上`\par`是最保险的做法，可以避免很多不必要的麻烦。
3. 类似关于段落间距`\parskip`的设置，`\fontsize{}{}`的控制对象是：（1）统辖范围内的每一行和其前面一行之间的距离；（2）统辖范围内第一行和统辖范围外之前的最后一行之间的距离，但不包括统辖范围内的最后一行的末尾和统辖范围之后的第一行之间的距离都会恢复为原来的长度，这段距离会恢复为原来的长度，因为这个行距是由统辖范围外之后的第一行来控制的。

行距（基线间距）

命令

```
\baselineskip % 当前默认值：12.0pt
\renewcommand{\baselinestretch}{1.5} % 间接控制\baselineskip
\linespread{1.5}  % 间接控制\baselineskip
```

宏包 基础命令

位置 略

参数 略

注释

1. 当前默认值可以通过`\the\baselineskip`得到，其默认值是默认字号的 1.2 倍，article 中默认字号为 10 pt，相应的基线间距的默认值为 12 pt。

2. 除了可以通过`\fontsize{}{}`的第二个参数直接控制基线间距，还可以通过重定义`\baselinestretch`或者`\linespread{}`修改默认字号的倍数，间接控制基线间距，要注意这两条命令控制的数字需要乘到原来默认的 1.2 倍的基础上，最后得到的数字才是默认字号的倍数，比如如果设置`\renewcommand{\baselinestretch}{1.5}`或者`\linespread{1.5}`，基线间距就会变成字体大小的 1.2 * 1.5 = 1.8 倍。修改行距的命令如果放在导言区，统辖范围就是正文，如果放在正文当中，需要使用`{\selectfont...\par}`来控制统辖范围。

3. 所谓的“行距”在 LaTeX 中称为“基线间距”（baselineskip），这一称呼建立在 LaTeX 的另一个重要概念“箱子”（box）的基础之上，每一行的内容都可以看成处于一个大箱子内，有一条固定的基线（由这一行上处于默认状态的箱子的基线确定）穿过这个箱子，箱子顶部到基线的距离称为“高度”（height），箱子底部到基线的距离称为“深度”（depth），“行距”这一称呼常常会被误认为是【上一行箱子的下边线和下一行箱子的上边线之间的距离】（以下称为“相邻两行箱子的上下边距”，这其实是另外一个概念，下文会提到），因此，即使要称呼“行距”，也要意识到其指的是相邻两条基线之间的距离（注意，这只是默认情况，特殊情况见下文讨论），而相邻两行箱子的基线间距实际也等于这两行箱子的上边线间距或者下边线间距。另外，要注意“基线间距”的概念是在同一个段落内进行定义的，当开始一个新的段落时，一般不要求上一段的最后一行的箱子和下一段的第一行的箱子的“基线间距”额外增加新的距离，这也是 LaTeX 定义的“段落间距”（parskip）的默认值为“0 pt + 1 pt”中“0 pt”这一部分的由来（“1 pt”的部分见下文讨论），因此，“段落间距`\parskip`”其实是一个差值，是在“基线间距”的基础上增加的长度，它也像“行距”一样容易被误认为是【上一段的最后一行的箱子的下边线和下一段的第一行的箱子的上边线之间的距离】，如果要在页面上为其找到一个对应的距离的话，可以将其理解为设置了新的段落间距后的段落从原来的位置向下移动的长度。

   现在来讨论一下前文提到的“相邻两行箱子的上下边距”，这段距离会随着基线间距的变化而变化，变化的方式有一些复杂，需要考察基线间距和箱子上下边距（转换成【上一行箱子的深度加下一行箱子的

高度】会更便于理解）的大小关系：如果前者大于后者，即上一行箱子的深度加下一行箱子的高度小于基线间距，则 LaTeX 会在两个箱子之间自动插入一段距离（称为“glue”），即“相邻两行箱子的上下边距”，以满足规定的基线间距；如果前者等于后者，即上一行箱子的深度加下一行箱子的高度等于基线间距，则相邻两行箱子的上下边距为 0 pt，这两行箱子上下无缝堆叠在一起；如果前者小于后者，即上一行箱子的深度加下一行箱子的高度大于基线间距，此时如果还要满足规定的基线间距，理论上必然要求上一行箱子和下一行箱子出现重叠部分（想象一下最极端的情况，基线间距越来越小直到 0，则理论上每一行的箱子都会重叠在一起），第一种情况（两行箱子没有重叠部分）对应的是相邻两行箱子的上下边距大于 0 pt，第二种情况（上一行箱子的下边线和下一行箱子的上边线重合）对应的是相邻两行箱子的上下边距等于 0 pt，第三种情况（两行箱子预计会出现重叠部分）对应的是相邻两行箱子的上下边距预计会小于 0 pt（可以理解为距离方向为负），这个“0 pt”可以看作一个界限，由`\lineskiplimit`控制，其默认值为 0 pt，为了避免第三种情况的发生，LaTeX 的处理方式是在上一行箱子的下边线和下一行箱子的上边线之间加上一段固定的距离，而不再要求基线间距达到规定的长度，这段距离由`\lineskip`控制，其默认值为 1 pt，也就是说，此时实际的基线间距并不是那个设置的小于箱子上下边距的值，而是【箱子上下边距加上 1 pt】。从设置基线间距的角度总结一下，如果设置的基线间距大于等于箱子的上下边距，实际的基线间距就是这个设置的值；如果设置的基线间距小于箱子的上下边距，实际的基线间距是箱子的上下边距加上 1 pt。

有的教程中会提到，“基线间距”其实也像“段落间距”一样，是弹性距离，通过以上讨论，可以更好地理解这一论断，弹性距离的性质就体现在第三种情况当中：一旦由于某种原因，上下两行箱子的上下边距预计会小于 0 pt，则这两行箱子会被 LaTeX 分开，并在上一行箱子的下边线和下一行箱子的上边线之间插入 1 pt 的距离，这很让我们很自然地联想到了段落间距的默认值 0 pt + 1 pt，这也是一个弹性距离，很显然，这两者之间是有象似性的，段落间距默认值当中的“1 pt”其实就相当于此处基线间距当中的`\lineskip`的默认值，这个“1 pt”的适用条件其实就相当于此处基线间距当中的`\lineskiplimit`，只不过这段距离从同一段落的相邻两行之间转移到了上一段的最后一行和下一段的第一行之间。