

科技论文排版技术 参考书

庞冬青

2025 年 7 月 14 日

前言

本书遵循 GPL 协议发布。

本书是 L^AT_EX 入门教材，内容涵盖了天津大学研究生公共课“科技论文排版技术”的讲课内容，并包含了一部分由于时间关系课堂上略去的但是很实用的例子。本书的内容编排强调实用性，而不是系统性。

本书的 tex 分发版本为 texlive，这个版本是 linux 下的默认 tex 版本，同时可以在 windows 下使用。texlive 的下载地址是：<https://tug.org/texlive/acquire-iso.html>。推荐选择最近的镜像下载，iso 比较大，但是下载速度一般情况下是很快的。

安装速度依赖于电脑配置和系统，Windows 下比较老的电脑要安装很久。texlive 大约每年 4 月下旬更新，推荐使用可移植安装方式。全选安装后占用空间约 7G，系统分区不够大的情况下建议放在其他分区。早期 Windows 系统下安装曾经出现过命令行不可用的问题，需要用手动添加环境变量的方式解决，最近两年没有同学报告过这个问题。

苹果系统下的安装参见手册。

本书的例子都在 linux 系统下经过实际编译测试，测试版本为注明的 texlive 分发版本。大部分基础代码一般不会因为分发版本的不同产生编译问题，一些功能包因为版本更新缘故，会导致少量的旧代码失效，一些 bug 也可能会在新版本中产生或被修复。极少量的代码涉及操作系统，比如系统字体或超链接打开视频文件这样的代码，给出的代码例子仅仅在笔者电脑系统 linux 下可以正确运行，这点需要额外注意。

书后的索引主要是包和运行命令，以及 tikz 的子库。

虽然本书的代码可以 copy 后使用，强烈建议自己输入代码方式学习，尤其是数学公式。不推荐初学者依赖代码补全功能。

本书一般每年在考试完毕后会发布上一年版本的终结版（包括代码），每年八月初开始基于当年的 texlive 版本修订新版，pdf 文件根据情况会不定期更新。本书计划中还有第八章内容，将给出一些常用设置代码，但是这部分工作预计会晚两年开始。

写书是一个永无止境的事情，修改进度有很大的不确定性。2025 年（2024 版本）有很大的不同，明确了一些细节问题，长期困扰笔者的几个实用问题今年想出了更好的解决方法。

本书将以开源方式分发，在下面的链接中免费下载。

<http://ull.tju.edu.cn/education/course/latex.html>

这个链接下还准备添加三个例子：天大本科生试卷、天大本科生毕设任务书和天大硕士研究生论文。

目录

前言

目录	I
1 编译方式	1
1.1 使用 pdf \LaTeX 编译	2
1.2 加载 xeCJK 包且使用 x \LaTeX 编译	6
1.3 使用 latex 编译	7
1.4 参考文献	8
1.4.1 natbib 包和 bibtex 编译方式	8
1.4.2 bib \LaTeX 包和 biber 或 bibtex 编译方式	10
1.5 索引 index 的编译	11
1.6 符号表 nomencl 的编译	12
1.7 专业术语表 glossaries 的编译	14
1.8 画图包 pstricks 的编译	15
1.9 文件格式转换	16
1.10 texmf-local 的使用	17
2 排版	19
2.1 基础设置	19
2.1.1 字号	19
2.1.2 行间距	21
2.1.3 字体	21
2.1.4 缩进	23
2.1.5 横向空白和纵向空白	23
2.1.6 对齐	24
2.1.7 分页	25
2.1.8 边注	25
2.1.9 图文混排	26
2.1.10 常用类的选项	27
2.1.11 划线	28
2.1.12 条目	29
2.2 页面设置包 geometry	31
2.3 颜色包 xcolor	33
2.4 盒子 box	34
2.5 插图	38
2.5.1 基本用法	38
2.5.2 设置 caption 形式	40

2.5.3	子图	41
2.6	表格	43
2.6.1	基本功能	43
2.6.2	三线表	45
2.6.3	表格颜色	45
2.6.4	单元格内换行 <code>makecell</code> 包	47
2.6.5	同行合并列	48
2.6.6	同列合并行 <code>multicol</code> 包	49
2.6.7	表格 <code>tabular</code> 环境的嵌套	49
2.6.8	表格脚注	50
2.6.9	跨页表格 <code>longtable</code>	51
2.6.10	表格单元格内的公式环境	51
2.6.11	参考书和一些有用的包	52
2.7	超链接包 <code>hyperref</code>	52
2.8	文章 <code>article</code> 类	55
2.9	参考文献	57
2.9.1	<code>bib</code> 格式的文献条目	57
2.9.2	<code>natlib</code> 包 + <code>bibtex</code> 编译	59
2.9.3	<code>natbib</code> 包情况下分章参考文献	60
2.9.4	<code>biblatex</code> 包 + <code>biber</code> 编译	62
2.9.5	<code>biblatex</code> 包 + <code>bibtex</code> 编译	63
2.9.6	<code>biblatex</code> 的常见功能介绍	64
2.10	版面设计 <code>titlesec</code> 、 <code>titleps</code> 和 <code>titletoc</code>	66
2.10.1	<code>book</code> 类的结构	66
2.10.2	页眉页脚的样式	68
2.10.3	章节名称的样式	71
2.10.4	目录样式	72
2.10.5	扉页设计	77
2.10.6	其他常用书籍排版	77
2.10.7	中文书的常用公式图表编号样式	79
2.10.8	天津大学论文的格式要求	80
2.11	页眉页脚包 <code>fancyhdr</code>	86
2.12	脚注 <code>footnote</code>	87
2.13	信 <code>letter</code> 类	88
2.14	背景水印	88
2.14.1	<code>background</code>	88
2.14.2	<code>eso-pic</code>	88
2.15	文章内分栏	89
2.15.1	多栏 <code>multicol</code> 包	89
2.15.2	交互两栏 <code>paracol</code>	91
2.15.3	更复杂的多栏版面 <code>flowfram</code>	91
3	数学公式	93
3.1	行间公式	93

3.2	独立公式环境	93
3.3	amsmath 包的基本用法	94
3.3.1	自定义函数	94
3.3.2	amsmath 包的公式环境	95
3.3.3	一些格式调整	100
3.4	分式和 displaystyle	102
3.5	开根号的细节	104
3.6	括号	105
3.7	积分	107
3.8	矢量	108
3.9	矩阵	109
3.10	狄拉克符号	111
3.11	复杂上下标	112
3.12	一些比较常见的特殊形式	113
3.13	结构型公式	114
3.14	标注重点	114
3.15	数学符号	115
3.16	mathtools	117
3.17	定理	118
3.18	网页公式	120
4	画图包 tikz	123
4.1	直线	124
4.1.1	坐标	124
4.1.2	线参数	127
4.1.3	箭头和 arrows.meta 库	129
4.2	矩形、圆、椭圆和弧	131
4.2.1	矩形	131
4.2.2	圆和圆弧	132
4.2.3	椭圆和椭圆弧	133
4.3	曲线	135
4.3.1	任意曲线 controls 方式	135
4.3.2	任意曲线 bend 方式	136
4.3.3	抛物线	136
4.3.4	正弦余弦曲线	137
4.3.5	plot 语句	137
4.4	一些功能	138
4.4.1	标识角度 angles 库	138
4.4.2	剪切 clip	139
4.4.3	网格 grid	139
4.5	循环 foreach	139
4.6	渐变色和混色	140
4.6.1	渐变 shade	140
4.6.2	透明 transparent	142

4.6.3	光学混色 <code>blend</code>	143
4.6.4	奇偶原则	143
4.7	分层	144
4.8	数学库 <code>math</code>	145
4.9	<code>node</code>	147
4.9.1	基本功能	147
4.9.2	命名 <code>node</code> 并使用 <code>node</code> 名坐标	156
4.9.3	<code>edge</code> 操作和 <code>quotes</code> 子库	158
4.9.4	<code>node</code> 的形状和 <code>shapes.geometric</code>	159
4.9.5	<code>node</code> 的其他库	164
4.10	定义样式	165
4.10.1	修改默认样式 <code>style</code>	165
4.10.2	自定义样式	166
4.10.3	自定义带参数的样式	166
4.11	修饰库	167
4.12	三维坐标	169
4.12.1	默认三维坐标	169
4.12.2	<code>tikz-3dplot</code> 包	171
4.12.3	3d 库	172
4.12.4	<code>perspective</code>	173
4.13	坐标计算 <code>calc</code> 库	173
4.14	几何做图	176
4.14.1	<code>through</code>	176
4.14.2	相交 <code>intersections</code>	176
4.14.3	相切	177
4.14.4	<code>calc</code> 库和几何做图	177
4.15	<code>matrix</code> 库	178
4.16	思维导图	179
4.17	电路	179
4.17.1	<code>tikz</code> 的电路子库	179
4.17.2	基于 <code>tikz</code> 的电路包 <code>circuitikz</code>	182
4.18	<code>pgfplots</code>	183
4.18.1	二维函数画图	183
4.18.2	二维数据画图	185
4.18.3	三维函数画图	185
4.18.4	三维 <code>map</code> 图	185
4.18.5	从数据画三维图	185
4.19	基于 <code>tikz</code> 和 <code>pgf</code> 的科学包或 <code>tikz</code> 库	186
4.19.1	高亮数学公式	186
4.19.2	物理类	186
4.19.3	量子	187
4.19.4	光路图包 <code>tikz-optics</code>	187
4.19.5	工程	188

4.19.6	chart	188
4.19.7	其他	188
4.20	基于 tikz 和 pgf 的其他包	189
4.20.1	装饰纹包 pgfornament	189
4.20.2	tikz-among-us	189
4.20.3	tikz-truchet	189
4.20.4	动物包	189
4.20.5	符号	189
4.21	几个光路图的例子和一些小技巧	190
4.21.1	直线中间添加箭头	190
4.21.2	光路图技巧	190
4.21.3	tikz 自带的镜像参数	194
4.21.4	一些实用技巧	195
5	一些有用的包	197
5.1	standalone 类	197
5.2	计算机程序语法高亮: listings	199
5.2.1	样式设置	199
5.2.2	lstlisting 环境和行间程序代码	201
5.2.3	显示 tex 代码及其结果	202
5.3	tcolorbox	202
5.3.1	单个、上下和并排	203
5.3.2	数学公式	204
5.3.3	theorems	204
5.3.4	tcblisting	205
5.4	动图: animate	206
5.4.1	利用多张图片实现动图	206
5.4.2	tikz 直接画动态图	207
5.4.3	tikz 多张画图方式	209
5.4.4	时序控制	210
5.4.5	转换成 svg 动图	212
5.5	试卷: exam 类	212
5.6	其他一些不务正业的包	218
6	化学分子式 chemfig	219
6.1	基本设置	219
6.2	键的基本语法	220
6.3	分子结构	223
6.4	结构复用	228
6.5	对称	229
6.6	Lewis 形式	230
6.7	一些细节	231
6.8	宏	231
6.9	高分子与宏	232
6.10	在 tikz 的 node 里加载高分子	233

6.11 chemmove、debug、内置 node 和 schemestart/schemestop	233
6.11.1 chemmove 和 standalone 类使用 [chemfig] 选项	233
6.11.2 debug 和内置 node 名	234
6.11.3 内置 node 名和宏 node 名混用	235
6.11.4 schemestart/schemestop 与宏	236
6.12 arrow 与分子位置的精确放置	237
6.13 chemfig 的几个例子	239
7 演示稿 beamer	241
7.1 beamer 模板	241
7.2 frame	243
7.3 beamer 自定义的分栏环境	244
7.4 目录	245
7.5 复用、暂停、显示顺序和跳转	246
7.6 图片	250
7.7 参考文献	251
7.8 简单的设置语句	252
7.9 beamer 内置的 box 环境	255
7.10 beamer 模板设计	256
7.11 两个重新定义 frame 的例子	256
7.12 一些其他功能	258
索引	261

第一章 编译方式

本章主要介绍常用的编译步骤。最开始的三个小节是上课前必须掌握的，后面的编译技巧则随着学习各种包逐渐掌握。编译方式这章的内容，有些是逐渐学习的，有些是课堂没有时间详细讲，需要课下练习的。这个辅助教材的内容不是完全按照讲课的顺序进行的。

本课程的例子，除了 `pstricks` 和 `listings` 的例子外，都是采用 `pdflatex` 编译的。汉字都是采用 `CJKutf8` 包。本书中的极少部分例子需要 `xelatex` 编译，更换编译方式时会特别指出。本书则是 `xelatex` 编译的，汉字采用 `xeCJK` 包。

本书编译命令行是 linux 下的结果，Windows 下命令可能需要加 `.exe` 后缀名。Texworks 是一个集成环境，可以用鼠标方式选择编译命令。最新版 Windows 的 linux 子系统 WSL 可能更方便使用命令行，请自行测试。

安装目录

Windows 下几乎是可移植安装的，linux 下推荐使用可移植安装，安装目录为

```
texlive/2024/ %2024是当前版本号
texlive/texmf-local/ %默认本地文件目录，安装时可以指定其他目录
```

安装时，可以指定 `texmf-local/` 的位置，现在默认的可移植安装方式，已经将不同版本的 `texlive` 和 `texmf-local/` 目录分开了，linux 下直接选择可移植安装即可。`texmf-local/` 目录下的设置，重新安装不会覆盖。

一般不建议直接修改安装目录下的文件，在 `texmf-local/` 下建立自己的设置文件时，先阅读手册，明确目录位置，很多时候需要新建目录。或者查看安装目录框架也可以，基本上是相同的结构，目录具有对应关系。例如上面的例子，

```
texlive/2024/texmf-dist/texdoc/texdoc.cnf %安装后自动生成
texlive/texmf-local/texdoc/texdoc.cnf %自己的配置文件
```

调用文档

调用文档的命令为 `texdoc`，下面的命令可以调用 `texlive` 分发版的文档，获得安装说明。Windows 下命令行调用文档成功可以说明命令行语句是可用的，安装成功后可以先测试一下 `texdoc` 命令。

```
texdoc texlive
```

`lshort-zh` 是 `texlive` 自带的 `LATEX` 介绍 `lshort` 的中文翻译版，没有 `-zh` 获得的是英文版。这里特别感谢 `ctex` 小组持续地更新该文档的中文版。调用中文手册

```
texdoc lshort-zh
```

基本上所有导言中用`\usepackage{包名}`语句加载的包，都可以用 `texdoc` 调用相关文档。几乎所有文档都是 pdf 格式的，除了一些特别古老的包，如 CJK，是其他格式的，文本格式或网页格式。这些格式的文档也都可以通过 `texdoc` 调用相应的阅读器（一般是浏览器）自动显示。

`texdoc` 命令会自动调用 pdf 阅读器，linux 下是 `xpdf`，可以在安装目录下的`texmf-local/texdoc/texdoc.cnf`文件里修改为自己选用的 pdf 阅读器。首次需要新建 `texdoc` 目录和新建 `texdoc.cnf` 文件（使用文本编辑器），不需要 copy 初始的 `texdoc.cnf` 文件里的内容，自动地优先使用本地设置。

```
viewer_pdf = acroread %s &
```

下次安装后不会影响这个设置，会优先读取`texmf-local/`下的本地设置文件。

1.1 使用 pdflatex 编译

tex 文件是文本文件，原则上可以用任何文本编辑器编写。下面是一个简单的中文 tex 文件的例子，为了简单起见，本书中我们将所有 tex 文件的例子都命名为 `a.tex`。

pdf 格式是最通用的文件格式，使用 `pdflatex` 编译方式可以一步到位获得 pdf 文件。目前绝大部分包都支持 `pdflatex` 编译方式。下面的例子包含了中文环境 CJK。

```
\documentclass[12pt,a4paper]{article} %默认是10pt

\usepackage{CJKutf8} %最基础的中文包，与pdflatex配合使用
\usepackage[T1]{fontenc}%解决复制f,d等问题

\usepackage{lipsum} %随机英文假文
\usepackage{zhlipsum} %随机中文假文

\begin{document}
\begin{CJK}{UTF8}{gkai} %gkai是楷体，宋体是gbsn
```

下面9个字符，必须以下面的方式才能正确显示。其中百分号是单行注释符。

```
\%
\& %表格单元的分隔符等
\# %变量
\$ %行间公式符号，必须配对
\{ \} %模块符号，必须配对
\^{} %公式上标
\_{} %公式下标
\~{} %留间距但不换行，xelatex编译时分隔命令和文字
```

四个反斜杠的打法，显示效果略有不同，两个用了行间公式方法。

```
\verb||a
\textbackslash a
$\backslash$a
$\setminus$a

\lipsum[1-3] %插入一段英文假文
```

```
\zhlipsum[3-6] %插入一段中文假文
\end{CJK}
\end{document}
```

这个例子除了注释外，还有几点需要额外说明。

1. 使用 CJKutf8 包，不要使用 CJK 包，这样生成的 pdf 文件里的中文可以 copy。
2. 百分号是单行注释符，多行注释用下面的模块

```
\iffalse
需要注释掉的语句
\fi
```

3. 例子中的符号，绝不能出现在 tex 文件的文本中，否则会报错且有时难以查找错误。

CJKutf8 包的固有缺陷

下面的情况需要改用 xeCJK 包，并且使用 xelatex 方式编译。

1. 偏僻汉字。
CJKutf8 包的汉字集有限，如果汉字无法显示，则说明超出了汉字集的范围，需要 xeCJK 包。
2. 代码中有汉字。
使用 CJKutf8 包时，汉字只能出现在文本中，不能出现在代码中。常见的代码中出现汉字的情况为
 - 代码中包含中文路径。
 - listings 包（计算机语言高亮包）显示的代码中有中文。
 - 汉字画图。（画图包文本框里的中文除外）。
3. 多种字体。
CJKutf8 只有两种字体，楷体 gkai 和宋体 gbsn。xeCJK 包允许使用计算机系统中的所有可用字体。对于中文文档，xeCJK 包 + xelatex 编译方式是最优选择。

pdflatex 编译方式的优点

虽然有这样那样的不足，CJKutf8 包和 pdflatex 编译方式也有几个优点，除了写本书和代码中有中文的笔记外，笔者基本都采用 pdflatex 和 CJKutf8 包。

1. 支持度最广。可能有些工具包不支持 xelatex 编译方式或者不支持 xeCJK。这是两种可能，不能将 xelatex 和 xeCJK 等价看待。xeCJK 必须采用 xelatex 编译，但是 xelatex 可以支持全英文或者其他语言。xelatex 编译在一些特殊代码情况下还可能出现不正确的结果，原因基本上是包之间的代码冲突。
2. 与基础命令最融洽。不需要特别设置，latex 基本语法对汉字有效，比如加粗。
3. 编译速度快。xelatex 第一次的编译速度比 pdflatex 略微慢一点，不过后续编译速度差别也不是很大。

后两点其实不太重要，主要是第一点，有些情况 xeCJK 包会产生微小偏差，比如字符无法显示，这种错误来自包匹配问题，只能通过切换编译方式检查。特殊情况 xelatex 编译结果也会出现微小偏差，而 pdflatex 是最稳定的。

附属文件

一般情况下代码文件包括源代码文件.tex 和参考文献数据文件.bib, 除了自动生成的 pdf 文件外, 编译还会自动生成其他附属文件, 最常见的如下:

- .aux 是辅助文件, 所有情况都存在
- .log 是编译日志文件, 所有情况都存在
- .bbl 文件, bibtex 或 biber 编译参考文献时存在
- .toc 是目录文件, 生成目录时存在
- .ilg、.idx 和 .ind 是索引文件, 生成索引时存在
- .ilg、.nlo 和 .nls 是符号表文件, 生成符号表时存在
- .glg、.glo、.gls 和 .ist 是术语表文件, 生成术语表时存在
- .nav 和 .snm, beamer 类时存在
- texput.log 文件, 一般在编译文件名称错误时出现

a.log 里保存了编译过程的信息, log 文件可以方便我们查看错误信息, 帮助我们修正 tex 文件中的错误。需要注意的是: a.log 里的信息和编译过程中调试窗口显示的信息虽然有大量重合, 但是内容并不是完全一致的。

在编译过程中生成的附属文件不要删除, 否则会增加编译时间。多次编译情况, 附属文件必须存在才能生产正确结果, 因为需要读取附属文件里的信息。因此一般建议一个 tex/pdf 一个目录, 方便文件管理。如果图比较多, 可以再建一个或多个子目录放插入的图片。

关于附属文件还有一个值得说明的地方: 如果坚信自己的 tex 文件代码正确, 但是编译总是出现错误, 可以将自动生成的附属文件全部删除, 然后重新编译。在某些极端错误情况下, 会产生这样的特例。错误信息被附属文件记录下来, 没有随着再次编译自动修正, 导致编译过程产生虚假错误。

一般情况下, 第一次编译正确, 随后几次编译错误, 说明代码是错误的。第一次编译错误, 随后几次编译正确, 说明代码是正确的。一般编译两次足以判断代码是否正确, 但是特殊情况下也有可能需要编译三、四次才会得到正确的显示结果。

tex 文件的基本要点

一般的 tex 分发版本都自带具有语法高亮功能的编辑器, 比如 texlive 的编辑器。还有一些分发版本添加了一些额外的编辑功能, 例如基础代码补全、自动环境加载等。甚至于一些分发版本还内置了一些公式形式的图形界面。不推荐初学者依赖这些高级功能, 因为养成良好的编程习惯比快速获得正确结果重要。

从这个最简单的例子可以看出 tex 文件的几个基本特点:

1. 命令在 utf8 编码环境下用反斜杠引导。

如果在网上看到用斜杠引导命令的 tex 文件例子, 有可能该文档使用的是其他编码。现今 utf8 编码已经成为各个操作系统中的常用编码, 一般不存在操作系统不支持 utf8 编码的特殊情况。

2. 第一行一般是 `\documentclass{类名}`, 除非加载特别的包外, 一个 tex 文件只能有一个 `\documentclass`。tex 的基础语法和默认设置文档是 classes, 该文档介绍了默认的常用类和基础语法功能。

```
texdoc classes
```

不建议初学者一开始就阅读该文档，做完基本练习后再阅读该文档更合适。

3. `\documentclass`语句和 `document` 环境之间是导言部分。一般在导言中加载包并设置包的参数。但是涉及到中文时一般需要将设置参数放到中文环境后，所以也在 `document` 环境之内。
4. `tex` 文件中`\begin{环境名}`和`\end{环境名}`构成环境。必须有且只有一个 `document` 环境。

```
\begin{document}
...
\end{document}
```

`\begin`和`\end`语句必须配对，配对原则是：先 `begin` 的后 `end`。

```
\begin{document} %先begin
\begin{equation}
f(x)
\end{equation}
\end{document} %后end
```

养成良好的编程习惯，先将配对写好再填写内容，这样不容易代码出错。配对错误发生时编译过程给出的错误代码行号信息往往是最后一个无法配对的地方，而不是配对错误发生的地方。大型文档时，往往需要在提示的错误代码行号位置往前查找错误，而不是提示的错误代码行号处。最容易出现编译错误的地方是环境名拼写错误。尤其是对初学者而言，这个错误非常常见，仔细阅读错误信息是可以看出来的。这种错误一般会提示没有定义。

5. 回车不分行，分行需要空白行，多个空白行等价于一个空白行（请自行练习）。利用这个特点，每行一句话，结合 `vim` 编辑器，可以非常快速调整语句顺序。

input 和 include

如果写大型文件，比如书或论文，可以采用`\input`或`\include`方式，将内容放在几个文件中，而不是一个大的 `tex` 文件。`input` 和 `include` 的用法是一样的。

```
\documentclass[12pt, a4paper]{article}
\begin{document}
\input{b.tex}
%或者
\include{b} %tex文件可以不用加扩展名
\end{document}
```

扩展名随意，比如 `b.txt` 亦可，但是 `txt` 文件时必须加扩展名，`tex` 文件时只需要文件名。

```
\input{b.txt}
```

`b.tex` 里的内容为

```
Hello!
New day!
```

`\input`和`\include`的区别在于，`\input`方式加载内容被视为纯插入，编译后不会产生文件 `b` 的附属文件。`\include`方式加载内容被视为包含进来，编译后会产生文件 `b` 的相应附属文件。如果编译过程需要 `b` 的附属文件，就必须采用`\include`方式加载内容。

1.2 加载 xeCJK 包且使用 xelatex 编译

xeCJK 包的中文环境代码如下

```
\documentclass[12pt,a4paper]{article}
\usepackage{fontspec} %使用Times New Roman字体，测试不需要
\usepackage{xeCJK} %加载包
\setCJKmainfont{SimSun}[AutoFakeBold] %默认中文字体并允许加粗
\setmainfont{Times New Roman} %默认英文字体，不设置则为tex默认，自动支持加粗

\setCJKfamilyfont{heiti}{Simhei} %{自己命名}{电脑已有字体名}
\newcommand{\heiti}{\CJKfamily{heiti}} %命令名是自己起的，可以不一样
\setCJKfamilyfont{cuhei}{Simhei}[AutoFakeBold] %设置中文字体加粗
\newcommand{\cuhei}{\CJKfamily{heiti}} %必须时可以加粗的中文字体才有效

\begin{document}
中文

{\CJKfamily{heiti}中文} %使用字体

{\heiti 中文} %自己定义命令获得更简洁的使用方式

{\bfseries 宋体加粗}
{\bfseries\cuhei 黑体加粗} %使用bfseries顺序不影响结果
{\bfseries\heiti 黑体不加粗}

{\bfseries\heiti 第1章}
{\bfseries\cuhei 第1章} %不加粗时两者一样

abcdefg fflammaa

汉字\thesection 汉字 %命令后要加一个空格，pdflatex不需要空格
\end{document}
```

`\setCJKmainfont`命令只能写在导言里，不能写在 `document` 环境内。xeCJK 的字体设置语句为

```
\setCJKmainfont{字体名}{字体性质}[字体性质] %对应tex的rmfamily的中文字体
\setCJKmonofont{字体名}{字体性质}[字体性质] %对应tex的ttfamily的中文字体
\setCJKsansfont{字体名}{字体性质}[字体性质] %对应tex的sfamily的中文字体
\setCJKfamilyfont!{自定义字体名}{电脑存在的字体名}[字体性质]
```

从这里可以看出 tex 字体的 family 和 shape 的不同。`\bfseries`和`\itshape`本身不是某种字体，而是字体的性质。`\setCJKfamilyfont`设置的是字体。加粗[AutoFakeBold]和斜体[AutoFakeSlant]是字体性质。

xelatex 如果不设置字体性质为[AutoFakeBold]，默认中文字体是没有加粗的，加粗命令是无效的。斜体[AutoFakeSlant]也是如此。字体本身是否支持加粗或斜体，linux 下可以用下面的命令查看。

```
fc-list %显示全部字体名和性质
```



```
fc-list |grep 指定字体名
```

定义字体性质的方括号也可以写到前面，写到后面更清楚

```
\setCJKmainfont[AutoFakeBold]{SimSun}
```

可以用下面的语句将`\bf`、`\bfseries`和`\textbf`命令设置为对中文黑体而不是加粗，`\it`、`\itshape`和`\textit`设置成楷体。

```
\setCJKmainfont{SimSun}[BoldFont=SimHei, ItalicFont=AR PL KaitiM GB]
```

天津大学研究生论文摘要二字要求宋体加粗，而不是黑体，所以上面将粗体设置为黑体就不合适了。如果同时需要宋体加粗和黑体加粗，如天津大学本科论文封面题目是黑体加粗，摘要二字是宋体加粗，题目要求黑体，就必须对分别设置黑体和可加粗黑体，这样可以对字母、数字加粗同时黑体不加粗。

`[AutoFakeBold]`这个参数可以写到包引用那里，对所有字体生效。

```
\usepackage[AutoFakeBold]{xeCJK}
```

不推荐这样的方式，逐一定义虽然麻烦，但是更具灵活性。

pdf_lat_ex 编译出来的 pdf 格式版本为 PDF document, version 1.5，而 xel_ate_x 编译出来的 pdf 格式版本为 PDF document, version 1.5 (zip deflate encoded)，不如 pdf_lat_ex 编译出来的 pdf 兼容性好。可以参考第17页的 pdf 格式版本转换语句将 pdf 格式版本号降低到 PDF document, version 1.4。这个问题主要出现在天大打印卷子的地方，pdf 阅读器的版本太低，pdf 文件如果存在分层就会无法正确显示。

更普遍的大型汉化包是 ct_ex，集成了很多功能，经常写中文文章的可以自行学习。

```
\usepackage{ctex}
```

笔者没用过，ct_ex 重新定义了许多命令，修改了很多默认设置，linux 下可能会出现字体匹配错误。如果经常写中文文章和书籍，倒是值得学习一下。

加载 xeCJK 包使用 xel_ate_x 编译时，汉字和数字间的距离比较好看，但是命令后要加一个空格。加载 CJKut_f8 包使用 pdf_lat_ex 编译时，不需要这个空格。笔者经常使用的是 CJKut_f8 包，写习惯了，这个细节总是出错。

1.3 使用 latex 编译

lat_ex 是最初始的编译方式，例如下面最简单的例子

```
\documentclass[12pt, a4paper]{article} %默认是10pt
%这里是导言部分，加载包，进行一些设置
\begin{document} %开始文档
Hello! %不会断行，自动添加一个英文空格
New day!
\end{document}
```

texlive 中 texworks 里选择 lat_ex 编译方式即可编译 a.tex 文件。在命令行下可以使用下面的语句进行 lat_ex 方式的编译。

```
latex a.tex %扩展名可以省略
```

latex 方式编译获得的是 dvi 格式的文件，用 xdvi 命令可以查看 dvi 文件。texlive 自带的 dvi 阅读器为 xdvi，texlive 的 texworks 里可以自动显示编译结果。命令行下的语句为

```
xdvi a.dvi
```

现今最常用的通用文件格式是 pdf 格式，可以使用下面的命令将 dvi 文件转换为 pdf 文件。

```
xdvipdfmx a.dvi
```

latex 编译方式时插入图片的格式只能是 eps。

latex 编译方式看起来有些过时了，但是我们还是需要学习一下，这样可以更好地编译（理解编译）pstricks 包画出的图形。一些功能包，例如 chemfig，虽然支持 latex 编译，但是必须加载包的特殊选项才能在编译后正确地将 dvi 文件转换为 pdf 文件，且只有 pdf 文件才能正确显示结果。坚持这种编译方式必须要阅读包的文档，常见的大型功能包的手册（文档）都会给出不同编译方式的要点说明。

1.4 参考文献

参考文献有两种方式，下面分别介绍。

1.4.1 natbib 包和 bibtex 编译方式

```
\documentclass[12pt,a4paper]{article}

\usepackage{CJKutf8} %解决汉字copy问题
\usepackage[T1]{fontenc} %解决田等copy问题
\usepackage{xcolor}

\usepackage[super,square,comma,sort&compress]{natbib}

\usepackage[unicode,colorlinks,linkcolor=blue]{hyperref}%超链接

\begin{document}
\begin{CJK}{UTF8}{gkai}

第一个参考文献\cite{fluid}第二个参考文献\cite{ol}。

\clearpage
第一个参考文献\cite{fluid}

博士论文\cite{wang}

\bibliographystyle{unsrt} %参考文献格式，扩展名为bst
\bibliography{a}

\end{CJK}
\end{document}
```


参考文献内容写在 bib 文件中，这里采用 a.tex 同名文件 a.bib，bib 文件可以不与 tex 文件同名。

```
@book{fluid,
author={朗道},
title={流体力学},
publisher={高等教育出版社},
address={北京},
edition={第三版},
year={1980},
}

@article{ol,
author={M. Bertero and C. De Mol and G. A. Viano}, %作者用and分隔
title={Restoration of optical objects using regularization},
journal={Optics Letters},
volume={13},
issue={2},
pages={51--53}, %起始页码，也可以是单页码
year={1978},
}

@phdthesis{wang,
author={王某某},
title={博士论文},
school={天津大学},
address={天津},
year={1980},
}
```

参考文献的编译分四步，命令行下 bibtex 编译时不加扩展名，图形界面选择 bibtex 编译即可。

```
pdflatex a.tex
bibtex a %此处是tex文件名，不是bib文件名
pdflatex a.tex
pdflatex a.tex
```

也可以在\documentclass之前加\write18{}语句一步编译到位，

```
\immediate\write18{bibtex \jobname}
```

此时需要 pdflatex 编译三次，因为这句相当于 pdflatex 编译时，自动地用 bibtex 语句编译了一次，后面两次 pdflatex 编译不能省略。这个方法无法在 biber 编译时使用，因为默认配置只允许调用有限几个编译命令，其中最常用的是 bibtex 和 makeindex 两个命令。

投稿时，如果需要网络编译的话，各个杂志社的情况可能不同，一般需要 a.tex 和 a.bib 文件，如果有问题，可以试试上传 a.bbl 文件。所需图片也必须上传。

大型杂志社都给出了自己的参考文献格式 bst 文件，制作 bst 文件的命令为

```
latex makebst
```

天津大学硕士和博士论文的 bst 文件可以选择 zharticle.bst，书和硕士、博士论文都需要给

出地址，不能没有地址项，地址项也不能空白。这个文件不在 texlive 的 bst 文件目录下，可以找到该文件，将其复制到 tex 文件所在目录，也可以将其复制到 bst 所在目录，然后运行 texhash 命令。

```
texhash
```

texhash 命令可以更新整个 texlive 目录的索引，自行安装了新的包或格式文件，texhash 一下即可使用。

bst 文件还可以放在 texmf-local 目录下相应的位置，最近几年，可移植安装选项时，texlive 将安装文件目录和 texmf-local 分开了，方便更新版本时不影响 texmf-local 目录下的内容。windows 下是可移植安装，linux 下，自己从 iso 安装，可以选择为可移植安装。如果 apt-get 等方式安装，默认是不可移植的，texmf-local 目录与安装目录不在同一个地方。

1.4.2 biblatex 包和 biber 或 bibtex 编译方式

biblatex 是新的参考文献包，，可以指定用 biber 编译，也可以指定用 bibtex 编译。

```
\documentclass[12pt,a4paper]{article}

\usepackage{CJKutf8} %解决汉字copy问题
\usepackage[T1]{fontenc} %解决田等copy问题
\usepackage{xcolor}

%使用biblatex代替natbib style=numeric,sorting=none,
\usepackage[backend=biber]{biblatex} %也可以选择bibtex编译

\addbibresource{a.bib} %添加bib文件

\usepackage[unicode,colorlinks,linkcolor=blue]{hyperref}%超链接

\begin{document}
\begin{CJK}{UTF8}{gkai}

第一个参考文献\cite{fluid}第二个参考文献\cite{ol}。

\clearpage
第一个参考文献\cite{fluid}

\printbibliography[title=参考文献] %汉化很容易

\end{CJK}
\end{document}
```

参考文献的编译分四步，命令行下 biber 或 bibtex 编译时不加扩展名，图形界面选择 biber 编译即可。如果 backend=bibtex，第二步则是 bibtex 编译。

```
pdflatex a.tex
biber a %这里是tex文件名
pdflatex a.tex
pdflatex a.tex
```

biblatex 能够支持更复杂的参考文献格式，但是 natbib 目前投稿更常用。

1.5 索引 index 的编译

使用 makeidx 包用来自动获得索引，例子如下

```
\documentclass[12pt]{article}

\usepackage{setspace}

\usepackage{makeidx}
\makeindex %必须有，自动生成idx文件

\usepackage[unicode,colorlinks,linkcolor=blue]{hyperref}%超链接

\begin{document}
\begin{spacing}{1.25}

laser\index{laser} is useful. %设置索引

\clearpage %分页
second place\index{laser} is useful. %再次设置同名索引
\printindex %生成的索引有两个页码

\end{spacing}
\end{document}
```

编译时分三步编译，

1. pdflatex 编译 a.tex 文件

```
pdflatex a.tex
```

编译后自动生成 a.idx 文件，此时生成的 pdf 文件没有索引内容

2. makeindex 编译 a.idx 文件

```
makeindex a.idx
```

编译后生成 a.ilg 和 a.ind 文件

3. pdflatex 再次编译 a.tex 文件，最好编译两次

```
pdflatex a.tex
```

获得有 index 的 pdf 文件

在\documentclass前加下面的语句可以一次到位编译。

```
\immediate\write18{makeindex \jobname.idx}
```

\write18只适合有限命令，最常见的就是 bibtex 和 makeindex。因为 makeindex 也编译符号表，所以这个语句还是非常有用的。只不过这个语句没有在 windows 下测试成功过。

很多时候需要将索引再分类，比如第四章将所有 tikz 的子库和相关功能包放在一起显示，这时只需要写为\index{主索引例如tikz!主索引下的分类索引例如tikz子库} 即可。

可以通过文件设定 index 的格式，此时需要加载 imakeidx 包，

```
\usepackage{imakeidx}
```

默认是两栏，字母，可以在\makeindex语句处修改，比如改为三栏，

```
\makeindex[columns=3, title=Alphabetical Index]
```

还可以在\makeindex语句里加载格式文件 myindex.ist，该文件放在 tex 文件所在目录即可。

```
\makeindex[options=-s myindex.ist]
```

下面是一个格式文件的例子，并不好看

```
headings_flag 1 %插入分组title，下面设定title格式
heading_prefix "\n\\centering\\large\\sffamily\\bfseries\\noindent\\textbf{"heading_
suffix "}"\\par\\nopagebreak\n"

item_0 "\n \\item \\small "

delim_0 " \\hfill " %页码右对齐
delim_1 " \\hfill "
delim_2 " \\hfill "
```

这个文件修改的时候要注意的是，各部分是用\分隔的，\large是命令，所以一般情况下不能随便删除第一个\。上面的例子比较难看，下面这个好点

```
headings_flag 1

heading_prefix "\\par\\large \\textbf{"
heading_suffix "}" \\*~\\\\\\*

item_0 "\n \\item \\small "

delim_0 " \\hfill "
delim_1 " \\hfill "
delim_2 " \\hfill "
```

1.6 符号表 nomencl 的编译

使用 nomencl 包生成符号表

```
\documentclass[12pt]{article}

\usepackage{nomencl}
\makenomenclature

\usepackage[unicode,colorlinks,linkcolor=blue]{hyperref}

\begin{document}
\[
I=I_0\exp(-\alpha l)
```

```
\]
\nomenclature{$\alpha$}{absorption}
\nomenclature{$l$}{sample length}
\printnomenclature
\end{document}
```

分三步编译

1. pdflatex 编译 tex 文件

```
pdflatex a.tex
```

编译后获得 a.nlo 文件

2. 使用 makeindex 编译 a.nlo 文件，手册里的语法是

```
makeindex a.nlo -s nomencl.ist -o a.nls
```

获得 a.ilg 和 a.nls 文件。pdflatex 编译 tex 文件，最好编译两次。

```
pdflatex a.tex
```

编译后获得有符号表的 pdf 文件。

如果希望一次到位，可以在 `\documentclass` 前加下面的语句

```
\immediate\write18{makeindex \jobname.nlo -s nomencl.ist -o \jobname.nls}
```

然后只用 pdflatex 编译即可。

有一种特殊情况，就是单独弄一个符号表文件，这时需要一行文字，哪怕是一个空格，编译成功后，再将这行文字删除就好了。完全的符号表语句，无法正确生成索引文件，所以生成的 pdf 是 0 字节的无法打开的。因为默认符号表是文章或书里的内容的符号表。我因为需要给学生一个单独的符号表，发现没有任何文字内容时的确存在编译问题。如果 copy 已经生成好的符号表，修改内容，只要索引文件正确引导一次也可以独立生成符号表文件。这不算 bug，因为符号表独立出来，仅仅是授课时的特殊需要。

有时前面的符号项比较长，导致描述部分不能左对齐，用下面的语句调整符号和描述之间的间距，形成对齐的两栏样式。

```
\printnomenclature[4cm]
```

上面的例子得到的符号表是混排的，还可以分组，一般情况将字母和希腊字母分开，比如说在希腊字母前加标识 [G]，希腊字母就会与一般的英文字母分开排序。

```
\nomenclature[G]{$\alpha$}{absorption}
```

排序时按分组标志的字母顺序排序，例子中希腊字母加 G 阅读方便而已，如果有多个分组，比如分组名为 F，调换希腊字母组的顺序，可以将 G 改为顺序在 F 前的英文字母。

需要手动排布顺序，可以使用编号。

```
\nomenclature[01]{$\alpha$}{absorption}
\nomenclature[02]{$l$}{sample length}
```

两位数时需要补 0，否则排序不正确。这类似于电脑目录文件名显示时的排序规则。

分组手动排序，[分组, 编号]

```
\nomenclature[P,1]{\alpha}{absorption}
\nomenclature[S,1]{l}{sample length}
\nomenclature[S,2]{x}{position}
```

给分组加小标题略微麻烦一点，需要 ifthen 包，并自己定义标题

```
\usepackage{ifthen}
\renewcommand{\nomgroup}[1]{%
  \item[\bfseries
  \ifthenelse{\equal{#1}{P}}{Physics constants}{%
  \ifthenelse{\equal{#1}{S}}{Other symbols}{}}%
  ]}
```

还有一种办法语句类似，使用 etoolbox 包

```
\usepackage{etoolbox}
\renewcommand\nomgroup[1]{%
  \item[\bfseries
  \ifstrequal{#1}{P}{Physics constants}{%1
  \ifstrequal{#1}{N}{Number sets}{%2
  \ifstrequal{#1}{O}{Other symbols}{}}}%配对3个右花括号
  ]}
```

汉化方式为下面的语句

```
\renewcommand{\nomname}{术语表}
```

一般情况下符号表的间距都太大，可以用下面的命令调整间距，写到导言里即可

```
\setlength\nomitemsep{-5pt}
```

更复杂的修改样式的方法可以参看手册。

1.7 专业术语表 glossaries 的编译

使用 glossaries 包可以产生术语表。超链接包必须放在 glossaries 包之前，否则术语表无法超链接。

```
\documentclass[12pt]{article}

\usepackage[unicode,colorlinks,linkcolor=blue]{hyperref}%超链接

\usepackage{glossaries}
\makeglossaries
\loadglsentries{k.txt} %加载entry文件

\begin{document}

laser is useful.

\clearpage
\gls{YAG} and \gls{Ti} are both lasers.
```

```
\printglossary

\end{document}
```

术语文件 k.txt 里的内容为

```
\newglossaryentry{YAG}{
name={YAG laser},
sort={laser},
description={an example}}

\newglossaryentry{Ti}{
name={Ti:sapphire laser},
sort={laser},
description={an example}}
```

术语表项加了 sort，会按 sort 排布术语表，相同 sort 的放在一起。sort 不是必须的，但是 description 是必须的。

分三步编译

1. pdflatex 编译 tex 文件

```
pdflatex a.tex
```

编译后获得 ist 和 glo 文件

2. makeglossaries 编译 glo 文件

```
makeglossaries g.glo
```

编译后获得 glg 和 gls 文件

3. pdflatex 编译 tex 文件，最好编译两次。

```
pdflatex a.tex
```

编译后获得有术语表 glossary 的 pdf 文件。

术语表如果不打印，可以作为词语或语句替换使用。比如某文件反复出现的某个词，翻译时拿不准怎么翻译的某个词，或者开介绍信写通知时，只在术语表内修改内容，加载不同文件，不修改主体 tex 文件的格式，就可以修改内容，等等。此时术语表项可以没有 sort 项，但是不能没有 description 项，描述置空即可。

```
\newglossaryentry{com}{
name={张三},
description={}}

\newglossaryentry{date}{
name={2022年2月16日},
description={}}
```

1.8 画图包 pstricks 的编译

下面的代码是 pstricks 体系下 pst-optic 包的一个透镜的例子，

```

\documentclass{standalone}
\standaloneconfig{border=5pt}

\usepackage{xcolor}
\usepackage{pst-optic} %光学包，透镜成像等

\begin{document}

\begin{pspicture}[showgrid=true](-5,-2.2)(7,4)
\rput(1.5,1.5){%
\lens[lensType=DVG,lensGlass=true,lensWidth=0.5,rayColor=red,
focus=-2,AB=2,spotAi=270,spotBi=90]}
\end{pspicture}
\end{document}

```

pstricks 画图包有多种编译方式，最初始的编译方式是分三步编译

1. latex 编译 a.tex 文件，得到 dvi 文件

```
latex a.tex
```

2. 将 dvi 文件转换为 ps 文件

```
dvips a.dvi
```

3. 将 ps 文件转换为 pdf 文件

```
ps2pdf a.ps
```

最简单的是用 xelatex 直接编译，可以一步到位生成 pdf 文件。

```
xelatex a.tex
```

利用 pst2pdf 可以获得 pdf 文件，还可以获得其他格式的文件。

```
pst2pdf a.tex -c -p
```

编译后自动生成的 images 文件夹中包含了 pdf 图片和 png 图片。

这门课选讲的画图包是 tikz，但是不能否认，pstricks 有着更丰富的专业包，而且 pstricks 在三维立体图形方面优于 tikz。如果真做对比的话，tikz 语法更接近初始的 tex 语法规则，对空格不敏感，而 pstricks 必须严格书写代码，如果没有空格的地方有空格，就会报错或无法正确显示。化学分子式包 chemfig 是基于 tikz 的，所以更推荐使用 tikz 包。

1.9 文件格式转换

texlive 自带了一些文件格式转换命令。下面这些常用转换语句不会降低清晰度，如果原图是矢量高清的，结果也是矢量高清的。

dvi 转 pdf

```
xdvipdfmx name.dvi
```


ps 转换为 eps, -B 必须有, 这样 ps 图片的空白会被截去, 最后只有图片部分, 即获得一个有尺寸的 tight bounding box。-l 扩展 box 使得边缘有 1pt 的留白。

```
ps2eps -B -l name.ps
```

-R 是旋转, 加号是顺治针转 90 度, 减号是逆时针转 90 度, 上标符号^是转 180 度。

```
ps2eps -B -l -R + name.ps
```

如果有同名 eps 文件存在, 该命令默认是不覆盖的, 可以选择删除同名 eps 文件, 或者使用强制选项-f。

```
ps2eps -B -l -f name.ps
```

eps 转换为 pdf

```
epstopdf name.eps
```

ps 也可以直接转换为 pdf, 之所以选择 ps 到 eps 到 pdf, 是因为很多情况下 ps 直接转换为 pdf 没有 tight bounding box, 导致 pdf 图片有非常大的空白边缘。这就是 ps2eps 命令的-B 选项的作用。但是有的 ps 文件本身就有尺寸设置, 这种情况就可以直接转换到 pdf 了。

```
ps2pdf a.ps
```

adobe 的 reader 有快照工具, 可以选择局部内容并打印到文件, 即打印为 ps 文件。这种方式获得的 ps 文件经过上面的转换步骤, 最终获得的 pdf 文件清晰度不会降低, 而且可以获得边缘空白最少的图片。因为图片尺寸是自动去除空白后计算出来的 tight bounding box 的尺寸, 而不是靠鼠标选取的。高版本的 adobe reader 可以截取局部直接到 pdf 和 png, 据说也不会降低清晰度。其他 pdf 阅读器, 好像只有福昕的阅读器可以快照打印到 ps 文件。

pdf 文件到 png 文件的格式转换, linux 下是 convert 命令。convert 命令可以很方便地转换各种图片格式。网页图片格式 webp 比较特殊, 转换命令是 cwebp 和 dwebp。

xelatex 编译时, pdf 的格式版本号为 PDF document, version 1.5 (zip deflate encoded), 低版本的 adobe reader 无法正确打开, 因此可以用下面的命令将 pdf 的格式版本号转换到 PDF document, version 1.4,

```
pdftops a.pdf //得到ps文件
ps2pdf a.ps //得到pdf的格式版本号为1.4的pdf文件
```

这样可以提高 pdf 文件的兼容性, 以免因为各种问题打不开 pdf 文件。pdflatex 编译没有这样的问题, 直接编译为 PDF document, version 1.5 格式。

1.10 texmf-local 的使用

目前的 texlive 安装文件目录构架, 可移植安装时 (Windows 下安装等价于可移植安装, 除了快捷方式和注册表为, 整个文件系统在一个目录下), 在 texlive 目录下生成两个文件夹, 2022 文件夹下放置 2022 版的文件, 同时还会生成一个名为 texmf-local 的文件夹, 下面是一系里空目录。这个文件夹里可以放置个人的常用配置。

比如可以将修改好的研究生论文参考文献 bst 格式文件放置在texlive/texmf-local/bibtex/bst/local目录下使用。

当配置文件放置在 texmf-local 目录下相应的位置后, 必须使用下面的命令重新建立索引后

才会生效。

```
texhash
```

第二章 排版

排版这章涉及的内容比较多且杂乱。索引、术语表和符号表的内容放在编译讲述。texlive 自带了一份简明手册，中文版的为 lshort-zh-cn.pdf，感谢 ctex 小组持续不断地翻译更新该手册。查阅手册的命令为 texdoc。

```
texdoc lshort-zh
```

L^AT_EX 排版充分体现了 L^AT_EX 的特点：做简单的事情很复杂，做复杂的事情很简单。

2.1 基础设置

这一小节主要内容是基本的排版功能，但是脚注会单独列出。最基本的知识可以参考手册 classes。

```
texdoc classes
```

建议先做一些练习，然后再看 classes 的手册。

默认的几个常用类为 article、report、book、letter。除了 book 的默认选项是 twoside（区分奇偶页）外，其余默认选项都是 oneside，不区分奇偶页。letter 类比较特殊，后面会单独讲用法。

article 类没有 chapter（章），book 和 report 都包含 part、chapter、section、subsection、subsubsection。用法如下

```
\part{飞秒激光器原理} %都可以有part  
\chapter{激光器原理} %article没有chapter  
\section{激光器简介}  
\subsection{激光器的历史}  
\subsubsection{激光器的诞生}
```

不建议修改层数（深度），下面的语句可以修改小节的层数，

```
\setcounter{secnumdepth}{2} %默认是3
```

1 是\section有效，2 是\subsection有效，3 是\subsubsection有效，默认是 3。

2.1.1 字号

类的选项里可以设置三种字号：10pt、11pt 和 12pt，默认是 10pt。这个字号是指一套九种字号，从小到大分别是：\tiny、\scriptsize、\footnotesize、\small、\normalsize、\large、\Large、\LARGE、\huge和\Huge。12pt 意思是\normalsize字号为 12pt，相当于小四字号。

内置字号的默认大小如下表所示，不建议修改默认字号大小，建议采用自定义字号。

tiny	5pt	6pt	6pt
scriptsize	7pt	8pt	8pt
footnotesize	8pt	9pt	10pt
small	9pt	10pt	10.95pt
normalsize	10pt	10.95pt	12pt
large	12pt	12pt	14.4pt
Large	14.4pt	14.4pt	17.28pt
LARGE	17.28pt	17.28pt	20.74pt
huge	20.74pt	20.74pt	24.88pt
Huge	24.88pt	24.88pt	24.88pt

Latex 的字号设置同时包括字体大小和绝对行距`\baselineskip`，其中`\normalsize`、`\small`、`\footnotesize`这三个命令还包括设置`\abovedisplayskip`（总是等于`\belowdisplayskip`）、`\abovedisplayshortskip`、`\belowdisplayshortskip`和 lists 相关参数，具体设置在 classes 手册里。

内置字号可以用花括号括起来，仅仅对花括号里的内容生效，还是很方便的。`{\tiny tiny}`得到的结果是 tiny，其余字体不变。

anyfontsize 包

只有添加 anyfontsize 包后，设置任意字号的语句`\fontsize`才真正起作用。

```
\usepackage{anyfontsize}
```

任意字号的设置语句为

```
\fontsize{size}{baselineskip}\selectfont
```

字号的设置语句是字号 size 和绝对行距 baselineskip 一起设置的，后面的`\selectfont`必须存在才能生效。

```
\fontsize{18pt}{24pt}\selectfont
```

`\fontsize`语句并不改变九种内置字号的大小，内置字号也是同时设置字号和绝对行距的，因此可以使用`\normalsize`修改回默认字号。

可以定义新命令，比如定义五号字和小五号字：

```
\newcommand{\wuhao}{\fontsize{10.5pt}{12pt}\selectfont}
\newcommand{\xiaowu}{\fontsize{9pt}{11pt}\selectfont}
```

使用如同`\small`，可以放在花括号局域环境内使用。

```
{\xiaowu 小五}小四
```

小五小四

小四 是 12pt，四号接近 large，但是小三和三号都没有对应的内置字号。

按汉语的字号进行设置，是否设置合适的`\baselineskip`是需要根据设计理念处理的。如天津大学的论文要求正文是小四、固定 20pt 间距，此时最好设置`\baselineskip`为 20pt。其余字号，则几乎不扩张行距，方便格式语句设置上下间距。不过如果设置了与字号对应的`\baselineskip`也无所谓，只是记住这些都是联动的。

2.1.2 行间距

设置绝对行距

```
\baselineskip=34pt %必须放在document环境之内
```

因为默认字号也是同时设置 `baselineskip` 的，所以使用默认字号`\normalsize`语句可以很方便地把上面语句的设置修改回来。

设置段间距使用设置长度命令`\setlength`，段间距是`\parskip`

```
\setlength{\parskip}{20pt}
```

这里需要注意的是，中间空白一行是分段，强制分行符不是分段，只是分行，因此不受该语句的影响。

Latex 的长度单位很丰富，最常用的是 `cm`，`mm`，`pt`，`em`。

最方便的是使用 `setspace` 包

```
\usepackage{setspace}
```

使用方式是

```
\begin{spacing}{1.25} %类似word的设置，1.25倍行距，随字体大小变化
内容
\end{spacing}
```

表格内容视为文字，所以 `spacing` 环境对表格行距也会生效，用 `spacing` 环境调整表格行距有些时候更方便美观。

2.1.3 字体

内置六种字体，默认字体是直体。

- 字体设置的最简单的但是是无条件的形式为，花括号构成局部有效，花括号内支持强制回车和空白行两种分段方式，支持多行、多段文字。

```
{\rm text} {\it text} {\tt text} {\sf text} {\sl
text} {\sc text}
```

`text text text text text TEXT`

- 少量文字可以使用下面的规范形式

```
\textrm{text} \textit{text} \texttt{text}
\textsf{text} \textsl{text} \textsc{text}
```

`text text text text text TEXT`

`\textbf{}`这样的形式支持强制换行符`\\`，不支持空白行分段。这样的规范形式没有顺序问题。

```
\textbf{\textit{text}}
\textit{\textbf{text}}
```

`text text`

无条件形式存在顺序问题，例如下面的情况，用简单形式一定会失效。

```
{\bf\it text} {\it\bf text}
```

`text text`

- 字体样式声明形式为`\rmfamily`、`\itshape`、`\ttfamily`、`\sffamily`、`\slshape`、`\scshape`以及粗体`\bfseries`，这个形式经常在设定样式的语句中使用。因为这个形式没有顺序问题，大段文字且中间有空行时也可以使用。

```
{\itshape\bfseries aa
bb
}
{\bfseries\itshape aa
bb
}
```

aa
bb aa
bb

- 没有粗体样式的字体不能加粗。例如`\ttfamily`和`\scshape`字体。这个需要看字体的说明，Windows 下可以用 word 或 wps 试试看，linux 下可以执行下面的命令查看安装字体的样式说明。

```
fc-list
```

默认字体中用的最多的还是`\rmfamily`、`\itshape`、`\ttfamily`这三个字体以及前两个字体的加粗。

笔者记得某手册说过，`\rm`和`\bf`这样的命令，处于废弃仍可使用状态，但是笔者还是忍不住经常用。latex2e 手册说的是，大部分情况，`\rmfamily`这样的形式更好，小部分情况仍然需要`\bf`这种无条件形式。目前 vim 和 listings 的语法高亮，都支持`\rmfamily`这样的形式，帮助校验是否拼写错误。

最后，经常用到带圈文字和带方框的数字，命令分别是`\textcircled{1}`和`\boxed{1}`，调整数字的字体大小可以更美观。`\boxed{}`命令需要 amsmath 包。如果不修改字体大小，外框直接加在文字外面。

```
11\boxed{1}11
11\textcircled{1}11
```

11 $\boxed{1}$ 11 11 $\textcircled{1}$ 11

缩放里面的文字、微调文字位置，可以达到和谐的效果。

更多基础字体可以参考下面的手册

```
texdoc psnfss2e
```

上面的圆圈文字，使用下面的包更美观，大小优美，但是使用比较麻烦

```
\usepackage{pifont}
```

命令为

```
\ding{175} %需要查表，175是圆圈4
```

附带了一些特别的字符，包括飞机、剪刀、各种星号和箭头。`\boxed`命令效果比`\textcircled`好得多，所以方框里文字可以自己调节出来需要的效果。圆圈里的文字，使用 pifont 表里的效果更好。

2.1.4 缩进

默认英文小节后第一段首行不缩进，因此专门有一个包修正这一点。

```
\usepackage{indentfirst}
```

设置首行缩进量，可以写到导言部分，`\setlength`是设置长度的常用语句

```
\setlength{\parindent}{2em} %两个标准字号的汉字
```

如果使用 CJKutf8 包，还可以使用下面的中文缩进语句

```
\CJKindent %必须写到CJK环境之内，xelatex不能使用
```

悬挂缩进

```
\hangafter=1 %第一行后悬挂缩进，默认为1，可以不写
\hangindent=2em %设置缩进量
\noindent %首行无缩进，必须直接放在文字前
```

首行无缩进要紧跟文字，先设置悬挂缩进，后设置首行无缩进。这三句对当前段有效。悬挂缩进也可以首行有缩进量。上面的例子中，如果悬挂缩进量和首行缩进量相等，就等于所有行全部缩进了这个量。

2.1.5 横向空白和纵向空白

默认定义了 5 种正值空白和 3 种负值空白，加上`\`，相当于定义了 9 种横向间距，这 9 种横向间距都是随字号变化的。6 种正向间距从小到大为

- `\thinspace`（可简写为`\,`），存在对应的负值间距`\negthinspace`（可简写为`\!`）
- `\medspace`（可简写为`\:`），存在对应的负值间距`\negmedspace`
- `\thickspace`（可简写为`\;`），存在对应的负值间距`\negthickspace`
- 反斜杠空格
- `\quad`
- `\qquad`

最精确的横向空白对齐是使用``方式，花括号里的内容占据空间，但是不显示出来。

```
\noindent aabgYxMixaabgM\\
aabg\phantom{YxMix}aabgM
```

```
aabgYxMixaabgM
aabg      aabgM
```

横向空白语句为`\hspace{1cm}`，纵向空白语句为`\vspace{1cm}`。使用`\hskip1cm`和`\vskip1cm`也可以，这种方式后面文字前要有空格。

强制换行符为两个反斜杠，`\\`。可以在强制换行符后面加长度参数额外添加纵向空白，例如`\\[3mm]`，在多行公式、表格和 ppt 中用来手动调整行间距。

`\hfill`和`\vfill`表示横向填充空白和纵向填充空白。填充满一行、一页或某种 box 的横向或竖向。

```
\hfill 前面自动填充空白\\
题目说明\hfill （5分）
```

前面自动填充空白
(5 分)

题目说明

纵向填充空白请自行练习。

2.1.6 对齐

默认为两边对齐。

1. 少量文字的一行内居中

```
\centerline{少量文字}
```

Latex 的居中是绝对居中，缩进多少都不改变居中文字的位置。

2. 大段文字的居中使用 `center` 环境，这种方式上下有附加间距。

```
\begin{center}
内容
\end{center}
内容 %有空行会缩进，无空行不缩进。
```

3. 居中设置对下面都生效，或者对花括号内局部有效，没有附加间距，也常用于图表环境中，令图表居中。

```
{\centering 大量文字}
```

使用这种形式居中，可以使用强制换行符或空白行。最后必须有强制换行符，否则最后一行不居中。

```
{\centering aa\\bb

dd\\}
aa %永远缩进
```

如果不喜欢后面的缩进，下面不缩进的内容必须写到这里面

```
{\centering aa\\bb\\

不居中不缩进} %这里开始不缩进
不换行 %与上面内容连接
```

4. 左对齐和右对齐

```
{\flushleft 文字，支持换行\\} %左对齐
{\flushright 文字，支持换行\\} %右对齐
```

也可以采用环境形式

```
\begin{flushleft} %或flushright
文字
\end{flushleft}
```

5. 左对齐和右对齐

```
{\raggedright 文字，支持换行\\} %右参差不齐是左对齐
```



```
{\raggedleft 文字, 支持换行\\} %左参差不齐是右对齐
```

左对齐是忽略缩进量的

6. 局部右对齐, 对齐点是不缩进的文字起始位置

```
\llap{abc右对齐}  
\llap{右对齐}
```

7. 局部左对齐, 对齐点是不缩进的文字起始位置

```
\rlap{左对齐abc}  
\rlap{左对齐}
```

8. 局部居中对齐, 对齐点是不缩进的文字起始位置

```
\clap{abc居中}  
\clap{居中}
```

三个局部对齐语句在格式设置代码中非常有用。

9. 默认页面是上下分散对齐的, 如果存在大量图表和长公式时, 一页能放的内容不多, 分散太开并不好看, 可以设置为顶部对齐, 下面留一些空白。

```
\raggedbottom %顶部对齐, 写到document之前  
\flushbottom %上下分散对齐
```

10. 垂直居中需要加载包 midpage

```
\usepackage{midpage}
```

```
\begin{midpage}  
垂直居中内容  
\end{midpage}
```

2.1.7 分页

强制分页可以用两种方式

```
\clearpage %推荐  
\newpage
```

无论哪种方式, 如果后面没有内容, 都不会产生新的一页。

2.1.8 边注

最好先设置边注的宽度, 不能超过右边距

```
\setlength{\marginparwidth}{4em} %使用em对中文更好一些
```

还可以修改边注与文字间的间距, 这个适合宽边注的书的排版。一般情况可以不修改。

```
\setlength{\marginparsep}{3mm} %默认10pt
```

文字旁边可以加入边注, 只需要`\marginpar{边注是自动换行的}`。边注是自动跟随文字的, 可以边注是自动换行的



插图和公式，equation 公式环境（带编号的公式）也可以。插入图与文字等效，跟随文字效果比较好。`\marginpar{\includegraphics[width=1cm]{fig/tju.pdf}}`

插入公式会略微往下一点。`\marginpar{\[f(x)\]}`这里例子显示的效果不是因为上面的图导致放不下，不插入图片，公式的位置也靠下。插入图、表和公式这些情况都只适合宽边注的排版，有些书将大部分插图都放在边注里面了，效果也很好。

边注的时候采用行间公式文字跟随效果最好，因为行间公式在 \LaTeX 里被视为文字。

`\marginpar{\centering$f(x)$}`。

对于书且使用默认的 twoside 选项，边注会按奇偶页自动变换位置。奇数页在右边，偶数页在左边。article 类，边注默认在右边，可以使用下面的语句改到左边

```
\reversemarginpar
```

这本书采用 geometry 包进行页面设计，选择使用装订线，边注从左边距开始，装订线宽度 从左边距部分保留空白。

修改回到右边的语句为

```
\normalmarginpar
```

有一个专门的包，marginnote，

```
\usepackage{marginnote}
```

2.1.9 图文混排

不规则分栏或者图文混排可以使用 minipage 环境。

```
\begin{minipage}[b]{3cm} %b表示底部对齐
\begin{figure}[H]
\centering
\includegraphics[width=3cm]{fig/tju.pdf}
\caption{天大校徽}
\end{figure}
\end{minipage}
\begin{minipage}[b]{3cm}
\begin{table}[H]
\centering
\caption{简单表格}
\begin{tabular}{l l l}\toprule
a & b & c\\
\midrule
1 & 2 & 3\\
4 & 5 & 6\\
\bottomrule
\end{tabular}
\end{table}
\end{minipage}
\begin{minipage}[b]{6cm}
使用figure和table环境，用minipage环境做出不等分的分栏效果时，也需要 float 包。
\end{minipage}
```

三个`minipage`在一行内，后面可以紧跟随文字，后面的文字可以自动换行。



图 2-1: 天大校徽

表 2-1 简单表格

a	b	c
1	2	3
4	5	6

使用 `figure` 和 `table` 环境，用 `minipage` 环境做出不等分的分栏效果时，也需要 `float` 包。

三个 `minipage` 在一

行内，后面可以紧跟随文字，后面的文字可以自动换行。

上面的例子可以看出，`minipage` 环境是等价为文字的。

后面讲的 `parbox` 也可以做出图文混排效果，但是不能使用 `figure` 和 `table` 环境。推荐使用 `minipage` 环境。

当然，如上例所示，`minipage` 环境也可以并排放置几个 `figure` 环境（带 `caption`）的图片，实现多图任意排版功能。还可以使用 `subcaption` 包实现一行中多个带 `caption` 的 `figure` 环境。

2.1.10 常用类的选项

基础类的默认选项如下所示：

article	letterpaper,10pt,oneside,onecolumn,final
report	letterpaper,10pt,oneside,onecolumn,final,openany
book	letterpaper,10pt,twoside,onecolumn,final,openright

最常用的类选项除了字号外，就是纸张大小。

纸张大小	高度	宽度
a4paper	297mm	210mm
a5paper	210mm	148mm
b5paper	250mm	176mm
letterpaper	11in	8.5in
legalpaper	14in	8.5in
executivepaper	10.5in	7.25in

默认是 `letterpaper`，比较小，一般会修改为 `a4paper`。

```
\documentclass[a4paper,12pt]{article}
```

文章默认是单栏 `onecolumn`，但是很多文章都是双栏格式，

```
\documentclass[twocolumn]{article}
```

`book` 类会生成独立的标题页，如果希望文章有独立的标题页，可以设置为

```
\documentclass[titlepage]{article}
```

默认为纵向，横向选项为 `landscape`

```
\documentclass[landscape]{article}
```

默认公式居中、编号右对齐，可以用 `fleqn` 设置公式左对齐，`fleqn` 选项时公式编号还是右对齐的。做数学手册时或者自己做常用数学公式笔记时比较推荐公式左对齐，方便查阅。

```
\documentclass[fleqn]{article}
```

可以用 `leqno` 选项设置公式编号左对齐，公式仍然居中，这样的书也曾经见到过。

```
\documentclass[leqno]{article}
```

默认 book 类是 `twoside`，区分奇偶页，采用下面的语句修改为 `oneside`，

```
\documentclass[oneside]{book}
```

默认 book 类是 `openright`，从右边页开始（即章首页从奇数页开始），也可以修正为 `openany`，这是很多中文书的版式，不存在空白偶数页。

```
\documentclass[openany]{book}
```

2.1.11 划线

一条 `textwidth` 宽度的横线

单线

```
\hrule
```

双线

```
\vskip1mm\hrule\vskip1pt\hrule\vskip1mm
```

前后调整纵向间距后就不太紧密了

单线

双线

前后调整纵向间距后就不太紧密了

`\hrule` 与文字间距太近，需要 `\vskip` 语句调整间距。

`\rule[高度]{长度}{宽度}`，可以画横线和竖线，方括号里第一个参数高度是调整横线高度或竖线起始位置高度的，省略时是 `0pt`。

```
文字\rule[3pt]{2cm}{1pt} %横线
```

```
\rule[1pt]{1pt}{2cm} %竖线
```

文字

文字的下划线语句为

```
\underline{abch} \underline{gfd}
```

```
\underline{中文}
```

abch gfd 中文

如上，默认的下划线高度参差不齐，并不美观。最简单的包为 `umoline`，可以添加下划线、中间线和上划线命令。

```
\usepackage{umoline}
```

更为复杂一点的包为 `ulem`，包含单下划线、双下划线、波浪线、虚下划线、点下划线等多种线型。

```
\usepackage{ulem}
```

2.1.12 条目

LaTeX 自带三个条目环境，分别是 `itemize`，`enumerate` 和 `description`。`itemize` 默认是圆点开始第一层，可以嵌套四层。

设置前导符的方式如下，i 是第一层，ii 是第二层，iii 是第三层，iv 是第四层。

```
\renewcommand{\labelitemi}{\tiny$\blacksquare$}
\renewcommand{\labelitemii}{$\bullet$}
\renewcommand{\labelitemiii}{\small$\bigstar$}
```

默认间距很宽，可以改变 `itemsep`，写到环境里面对当前环境有效。

```
\setlength\itemsep{-5pt}
```

因为本书已经设置了第一层的间距，所以下面的代码就只修改了第二、三层的间距。

```
\begin{itemize}
\item 第一条
\item 第二条
\begin{itemize}
\setlength\itemsep{-5pt}
\item 第a条
\item 第b条
\begin{itemize}
\setlength\itemsep{-5pt}
\item 第i条
\item 第ii条
\end{itemize}
\end{itemize}
\end{itemize}
```

- 第一条
- 第二条
 - 第 a 条
 - 第 b 条
 - ★ 第 i 条
 - ★ 第 ii 条

`itemize` 环境可以使用任意的先导，采用加方括号形式即可。

```
\begin{itemize}
\item[aa] 第一条
\item[bbb] 第二条
\item 第三条
\end{itemize}
```

- aa 第一条
- bbb 第二条
- 第三条

使用 `enumitem` 包可以方便设置 `item` 项，后面的设置采用了该包的语句。

```
\usepackage{enumitem}
```

本书修改了第一层的间距。

```
\setenumerate[1]{itemsep=0pt,partopsep=0pt,parsep=\parskip,topsep=5pt}
\setitemize[1]{itemsep=0pt,partopsep=0pt,parsep=\parskip,topsep=5pt}
\setdescription{itemsep=0pt,partopsep=0pt,parsep=\parskip,topsep=5pt}
```

enumitem 包有大量的设置语句帮助设置 itemize、enumerate 和 description 环境的格式，比基本设置语句方便得多。

enumerate 环境是自动编号的条目。

```
\begin{enumerate}
\item 第一条
\item 第二条
\item 第三条
\end{enumerate}
```

1. 第一条
2. 第二条
3. 第三条

换成其他方式编号语句如下

```
\begin{enumerate}[label=\alph*.]
\item 第一条
\item 第二条
\item 第三条
\end{enumerate}
```

- a. 第一条
- b. 第二条
- c. 第三条

```
\begin{enumerate}[label=\roman*.]
\item 第一条
\item 第二条
\item 第三条
\end{enumerate}
```

- i. 第一条
- ii. 第二条
- iii. 第三条

description 则用于定义

```
\begin{description}
\item[italy] 意大利 %英文的定义是粗体，中文需要额外设置
\item[自然数] 1,2,3,...
\item[有理数] 整数和分数
\item[实数] 实数的定义为...
\item[特别长的名词情况] 与itemize环境对齐方式不同
\end{description}
```

italy 意大利

自然数 1,2,3,...

有理数 整数和分数

实数 实数的定义为...

特别长的名词情况 与 itemize 环境对齐方式不同

设置中文黑体样式

```
\begin{description}[font=\cuhei]
\item[italy] 意大利
\item[自然数] 1,2,3,...
\item[有理数] 整数和分数
\item[特别长的名词情况] 与itemize环境对齐方式不同
\end{description}
```

italy 意大利

自然数 1,2,3,...

有理数 整数和分数

特别长的名词情况 与 itemize 环境对齐方式不同

除了上面的三个环境外，verse 和 quote、quotation 三个环境也可以有 item 项，此时没有前导符号。一般情况下这三个环境使用强制换行符而不是 item 项。

2.2 页面设置包 geometry

默认纸张大小是 letterpaper，默认页面布局如下：默认的文字宽度 textwidth，10pt 是 345pt，11pt 是 360pt，12pt 是 390pt。默认的文字高度 textheight，11pt 是 38 baselineskip，12pt 是 36 baselineskip。默认的左右留白比较宽，用基本语句修改页面布局比较麻烦，一般采用 geometry 包。

页面布局

页面设置包为 geometry，在加载包时就可以设置一些常用参数

```
\usepackage[a4paper,bindingoffset=1cm,top=2cm,bottom=2cm,left=2cm,right=2cm]{geometry}
```

a4paper 是纸张大小，geometry 支持更多的纸张大小。纸张大小只能导言中设定一次。bindingoffset 设置装订预留宽度，然后设置上下左右边距。如果设定了装订预留宽度，实际的左右留白是左右边距加上装订预留宽度（书是分奇偶页的，oneside 就是左边距相加）。

包的选项参数相当于默认的页面设置。我们可以在文章中重新设置页面参数，比如上下左右边距。

```
\newgeometry{left=5cm,right=5cm,top=5cm,bottom=5cm}
\savegeometry{L2} %L2是自定义的名字，用于重复使用页面设置
```

\newgeometry{} 语句后的页面会改变为新的布局。同时还会清除一些格式选项，比如行间距、边注宽度等。

重新回到导言的布局只需要下面的命令

```
\restoregeometry
```

再次使用 L2 布局只需要再次加载即可

```
\loadgeometry{L2}
```

横排

横排可以在两个地方设置，

```
\documentclass[landscape,12pt]{article}
```

或者

```
\usepackage[landscape,paper=a4paper,...]{geometry}
```

如果仅仅需要某一页或几页局部横排，不能通过`\newgeometry{}`修改为横排，所以需要`lscape`包。

```
\usepackage{lscape}
```

使用 landscape 环境

```
\begin{landscape}
```

需要横排的内容

```
\end{landscape}
```

新的一行 %环境外的文字会自动分页到下一页竖排

此时文字图表内容横排，页眉页脚的位置不变，即布局不变。

文章内改变纸张大小

一些包括图纸的书很需要这个功能，必须使用`pdfpages`包，`geometry`包不能在文章中局部改变`papersize`。

```
\usepackage{pdfpages}
```

用下面的语句改变`papersize`，`papersize`不能用`\newgeometry`语句在文档中间修改。

```
\pdfpagewidth=40cm
```

```
\pdfpageheight=30cm
```

此外，还必须修改`layoutsizes`（文字、装订线宽度、上下左右边距（包括页眉页脚部分）和文字宽度的总和），否则页码和文字都不能自动扩展或收缩到新纸张大小。默认就是`layout`大小和纸张大小相同。比较复杂的页面布局，`layout`尺寸可以小于纸张尺寸，这种情况比较少见。

```
\newgeometry{layoutwidth=40cm,layoutheight=30cm,left=2cm,right=2cm,top=2cm,bottom=2cm}
```

这样可以获得一个页面布局正确的结果，页码和文字都在适当的位置。

再次回到原来的纸张大小时，除了将`papersize`重置外，还必须修改`layout`。恢复到默认设置、加载已经保存的设置或者重新设置都可以重置`layout`。

```
\pdfpagewidth=21cm\pdfpageheight=29.7cm %纸张大小改回到A4
```

```
\restoregeometry %回到默认设置
```

%如果需要也可以设置新布局

```
\newgeometry{layoutwidth=21cm,layoutheight=29.7cm,left=2cm,right=2cm,top=2cm,bottom=2cm}
```

这样书中某页是大图纸情况，可以加载任意尺寸的大图和说明文字，页码可以自动编码且在布局规定位置显示，比如说底部居中。

从这个例子可以看出`papersize`和`layoutsizes`的关系。如果不修改`papersize`，只修改`layoutsizes`，比如扩大`layoutsizes`，内容将按`layoutsizes`排布，但是显示按`papersize`显示，内容会被切掉。

如果只修改`papersize`，比如扩大`papersize`，`layoutsizes`保持不变，则文字内容会按`layoutsizes`排布，只占据pdf页面的一部分空间，包括页码都是按原来的`layoutsizes`居中，实际显示效果页码的位置会很偏。

有一种特殊的情况可以不修改`layoutsizes`，只修改`papersize`，就是不存在页眉和页脚时。可

以用`\parbox{大的宽度}{长文字}`来做到文字横向排布的扩展。大的图纸也可以放在大的 `papersize` 里面,但是不能添加 `caption`,因为 `caption` 跟页码(页眉页脚)一样,是按原来的 `layoutsiz` 默认居中的,也是按原来的 `layoutsiz` 换行的。当然,如果使用`\caption{\parbox{大的宽度}{长文字}}`也可以手动逐个地解决问题。

总之,对于放置大图纸、大表格, `papersize` 是一定需要修改的。

最后还需要特别指明的是,局部修改 `papersize` 可以做到单页横排效果,但是与上面介绍的局部横排的显示效果不同,此时页眉页脚布局是横排的。

2.3 颜色包 xcolor

默认只有七种颜色, `black`、`white`、`red`、`blue`、`cyan`、`green`、`yellow`。加载 `xcolor` 包默认颜色扩展到 19 种。

```
\usepackage{xcolor}
```

`xcolor` 包允许用冒号调颜色深浅

```
\pagecolor{yellow!5} %设置背景色为浅黄, 数值范围0到100
```

也允许用多重冒号混色,颜色设置语句可以放在花括号局部生效

```
{\color{blue!70!black}blue}^^I
```

blue

`wave` 参数用可见光波长定义颜色,数值范围是 363–814

```
{\color[wave]{633}633nm laser}
```

633nm laser

默认模式是 `rgb`,但是内置了很多颜色名,需要加载包时设置选项,比如说

```
\usepackage[svgnames]{xcolor}
```

`svgnames` 内置了许多颜色名,比如 `DarkGreen` 这样的。手册后面有各个参数情况下的调色板,对照看选一个自己喜欢的组合即可。

有两个常用命令,一个是给颜色起名,这个命令在画图时比较有用,如果多个区域都准备设置为红色,那就定义红色的颜色别名,需要调整颜色时,比如红色变为绿色,只需要调整颜色别名指向的颜色就行。

```
\colorlet{region-a}{red}
{\color{region-a} colerlet}
```

colerlet

另一个是自定义颜色,

```
\definecolor{bar}{rgb}{0.5,0.3,0.3}
{\color{bar} define}
```

define

大段文字情况下,用上面的加花括号的方式局部改变文字颜色很方便。小段文字,使用`\textcolor{颜色名}{文字}`语句更方便。

```
\textcolor{red}{\textcolor is red\\强制换行符}默认黑色
```

textcolor is red
强制换行符默认黑色

`\textcolor`命令支持强制换行符，但是不支持空白行分段。大段文字更改颜色，用上面说的花括号构成局域环境方式。

使用参数用下面的例子

```
\textcolor[wave]{633}{\textcolor is red}
```

textcolor is red

2.4 盒子 box

这一小节的内容需要 `xcolor` 包。

mbox 和 fbox

最简单的 box 是 `mbox` 和 `fbox`，`mbox` 没有框，`fbox` 有文本框

```
\mbox{text}和\fbox{text}
```

text和 text

colorbox

`fcolorbox` 是可以定义框颜色和背景色的 box

```
\fcolorbox{框颜色}{背景色}{文字}
```

```
\fcolorbox{red}{gray!20}{text}
```

text

不加前面的 `f` 就是 `colorbox`，没有边框只有背景色

```
\colorbox{yellow}{text}
```

text

使用颜色模式时，

```
\colorbox[rgb]{1,0,0}{text}
\fcolorbox[rgb]{1,0,0}[rgb]{0,1,0}{text}
```

text text

rotatebox

可以将 box 旋转任意角度，如将 `text` 旋转 30 度

```
\rotatebox{30}{text}
```

text

比较下面的结果

```
\colorbox{yellow}{\rotatebox{30}{text}} %先旋转后涂色
\rotatebox{30}{\colorbox{yellow}{text}} %先涂色后旋转
```

text text

box 里的换行

box 不能自动换行，也不支持强制换行，需要换行必须使用 vbox 或 parbox，

first \vbox{abc\\abcdefghijklm}

abc
first abcdefghi

改为下对齐

```
first \vtop{abc\\abcdefghijklm}
```

first abc
 abcdefghi

parbox 的语句为 `\parbox{宽度}{内容}`

```
\parbox{\textwidth}{large auto large auto large auto large auto large auto large auto
large auto large auto large auto}
```

对上面的 box 涂色, 先 parbox 后涂色

```
\colorbox{yellow}{\parbox{\textwidth}{large auto large auto large auto large auto
large auto large auto large auto large auto large auto}}
```

large auto large auto large auto large auto large auto large auto large auto large auto large auto large auto

parbox 一共有四个参数，因为高度可以自动按内容扩展，所以高度可以省略。parbox 的宽度必须设置。另外两个参数是控制对齐的，可选择 b 或 t 或 c。

`\parbox`[外部对齐位置][高度][内部文字位置]{宽度}{内容}

```
first \parbox[t][1cm][b]{6cm}{内外对齐方式可调} outside
```

first	outside
-------	---------

内外对齐方式可调

填表时可以用 `parbox` 换行。

makebox 和 framebox

mbox 和 fbox 的复杂版本是 makebox 和 framebox，常用于版面设计

`\makebox[0.7\textwidth][r]{指导教师: }\[2mm]`

```
\makebox[0.7\textwidth][r]{日期: }
```

指导教师:

日期:

使用 `framebox` 可以使得前面的 `makebox` 看得更清楚

`\framebox[0.7\textwidth][r]{指导教师: }\[2mm]`

`\framebox[0.7\textwidth][r]{日期: }`

	指导教师:
	日期:

上面各种 box 其实有细微的差别,

- `\parbox`和`\makebox`没有内外间距, 两个 box 并排放, 不存在任何间距。
- `\framebox`和`\colorbox`有内间距。文字和框之间上下左右都存在间距。`\colorbox`填充的是`\framebox`边框内的空间。下面的例子放大后能看出细节差别。

```
\framebox[0.4\textwidth][r]{指导教师}\framebox[0.4\textwidth][l]{日期}

\makebox[0.4\textwidth][r]{指导教师}\makebox[0.4\textwidth][l]{日期}

\parbox{2em}{日期}\parbox{2em}{日期}\fbox{日期}\fcolorbox{black}{yellow}{日期}
\colorbox{yellow}{日期}
```

指导教师	日期
------	----

指导教师日期

日期日期 日期 日期 日期

`\parbox`本意是精确图文等混配, 存在间距反而会导致对齐上的麻烦。`\makebox`是为了更方便地做 box 内部左右对齐方式的调整。`\framebox`这样带框的情况, 没有间距不美观。

两个`\framebox`写到两行, 不会分段 (与文字规则相同), 但是中间会产生 0.35ex 的间距, 这是英文换行后的字母之间 (单词之间) 的间距。所以上面的例子都是连着写的。

raisebox

升降 box 的位置, 可以用于图文混排的上下位置微调、下划线情况下行间公式的上下位置微调。

```
内容\raisebox{2mm}{上升文字}内容\raisebox{-2mm}{上升文字}内容
```

内容^{上升文字}内容_{上升文字}内容

framed

无论是 vbox、parbox、makebox 等, 都无法内容分页, box 是一个整体, 如果放不下的话会整个移到下一页, 都放不下就显示不全。需要 box 自动分页可以使用 framed 包。framed 包非常适合多页大型表格的填写。

```
\usepackage{framed}
```

放在 framed 环境内的内容很长也会自动分页, 并且每页都自动添加边框

```
\setlength\FrameSep{2mm} %设置内容距离边框的间距, 必须放环境外
\begin{framed}
长内容
\end{framed}
```

长内容

`framed` 环境的边框线宽为`\FrameRule`。加一条横线时，因为横线默认位置是文字起始位置，默认长度是文字宽度，因此需要横线水平位置左移`\FrameSep`加上`\parindent`，线长度等于`\textwidth`加上 2 倍的`\FrameSep`。横线的高度位置一般需要略微调整一下比较好看。

```
\begin{framed}
长内容

\vskip-5pt\hskip-\FrameSep\hskip-\parindent \rule{\textwidth+2\FrameSep}{\fboxrule}

\vskip-5pt

也可以是直接画两次长度分别为文字宽度和2倍\lstinline!\FrameSep!的横线。
如果上面的长度相加无法正确传递的话。

\hskip-\FrameSep\hskip-\parindent \hskip-\parindent
\rule{\textwidth}{\fboxrule}\rule{2\FrameSep}{\fboxrule}

长内容
\end{framed}
```

长内容

也可以是直接画两次长度分别为文字宽度和 2 倍`\FrameSep`的横线。如果上面的长度相加无法正确传递的话。

长内容

但是如果重新设置了`\FrameSep`，此时增加的`\FrameSep`长度会以修改`\textwidth`的长度为代价，整体框宽度不改变。因此并不需要修改上面的画横向的语句。

```
\setlength\FrameSep{5mm}
\begin{framed}
长内容

\vskip-5pt\hskip-\FrameSep\hskip-\parindent \rule{\textwidth+2\FrameSep}{\fboxrule}

长内容
\end{framed}
```

长内容

长内容

也可以在 `framed` 环境外定义一个新长度变量，设置新长度变量等于`\textwidth`。`framed` 外框的宽度其实就等于 `framed` 外的`\textwidth`，进入 `framed` 环境内`\textwidth`的值会被修改。

离开 framed 环境后，`\textwidth` 的值会被再次修改回来。下面的例子可以看出 `\textwidth` 的值的变化。第一条线的长度为 framed 环境内的 `\textwidth`，第二条线实际是 framed 环境外的 `\textwidth`，与 framed 外框宽度一样。

```
\newlength{\mynew}
\setlength{\mynew}{\textwidth}
\noindent framed外起始点
\begin{framed}
aa

\vskip-5pt\hskip-\FrameSep\hskip-\parindent \rule{\textwidth}{0.5pt}

\vskip-5pt\hskip-\FrameSep\hskip-\parindent \rule{\mynew}{0.5pt}

aa
\end{framed}
\noindent framed外起始点
```

framed 外起始点

aa
aa

framed 外起始点

天津大学本科生的开题报告的样式，用 framed 包来做非常合适。分页表格包 longtable 不能单元格自动分页。

2.5 插图

2.5.1 基本用法

不添加任何包就可以插图，但是一般都需要添加 graphicx 来实现图片大小等基本功能。

```
\usepackage{graphicx}
```

单纯的 `\includegraphics[参数]{图片名}` 是文本，figure 构成独立的图片环境。一般情况都应该指定图片大小，使用宽度或高度都可以，另一个按比例变化。但是除非特殊情况，不要同时指定宽或高，否则会扭曲图片。

```
\begin{figure}[ht!] %ht!表示当前位置或top
\centering %一般都加这句使得图片居中
\includegraphics[width=4cm]{fig/tju.pdf}
\caption{天大校徽}
\end{figure}
```

`\begin{figure}[位置参数]` 推荐使用 `[ht!]`，表示图片放在当前位置或下一页 top 位置，`[h!]` 太严格了，做不到会自动转成 `[ht!]`。b 表示底部，p 表示单独成页。这两个参数偶尔也很有用。

图片自动编号，如果仅仅给出 caption，不加编号，可以加星，即 `\caption*{}`。

缩放图片还可以使用 scale 参数



图 2-2: 天大校徽

```
\includegraphics[scale=0.3]{fig/tju.pdf}
```

旋转角度可以使用 `angle` 参数

```
\includegraphics[scale=0.3, angle=30]{fig/tju.pdf}
```



对图片加框使用 `fbox`, 因为`\includegraphics[参数]{图片名}`是文本, 可以放到文本框内

```
\fbox{\includegraphics[width=3cm]{fig/tju.pdf}}
```



加一个正框, 里面图片旋转, 需要 `framebox`, `framebox` 同时设置宽高的时候用圆括号。

```
\framebox(3cm,3cm){\includegraphics[width=3cm, angle=30]{fig/tju.pdf}}
```



关于图片名有几点需要特别说明。

1. latex 编译方式只能使用 eps 格式的图片。
2. pdflatex 和 xelatex 编译方式可以使用 jpg、png 和 pdf 格式的图片, 可以不加扩展名 (linux 下), 但是不能使用 eps 和 ps 格式的图片。

3. 图片名有点时, 如果出现编译错误 (与版本相关), 使用花括号将. 扩展名前的部分括起来。可以不加扩展名。

```
\includegraphics[width=4cm]{fig/{t.j.u}.pdf}
```

4. 图片名有空格时, 使用双引号将. 扩展名前的部分括起来。

```
\includegraphics[height=4cm]{"t j u".pdf} %此处设定图片高度
```

其中第 3 条, 估计用双引号也可以, texlive 2021 版没有这个编译错误了, 所以也无法调试, 最早也没有这个错误, 可能是某版本的短暂 bug。推荐不省略扩展名, 可以准确分辨出同名不同格式的图片并正确加载。

2.5.2 设置 caption 形式

设置 figurename, 加在导言或文件开头, 对所有图片都生效, 这里加在里面, 只对当前图片生效。

```
\begin{figure}[ht!]
\renewcommand{\figurename}{Fig.} %加在figure环境内只对当前图片有效
\captionsetup{labelsep=space} %使用空格取代冒号
\centering
\includegraphics[width=4cm]{fig/tju.pdf}
\caption{天大校徽}
\end{figure}
```



Fig. 2-3 天大校徽

更复杂的 caption 结构设计, 可以参看手册

```
texdoc caption
```

默认 caption 间距就挺好, 修改用下面的语句, 负号更容易看出区别。

```
\abovecaptionskip=-5pt %直接设置
\belowcaptionskip=-5pt
```

或者

```
\addtolength\abovecaptionskip{-5pt} %基础上加減
\addtolength\belowcaptionskip{-5pt}
```

双 caption 可以使用 bicaption 包。

```
\usepackage{bicaption}
```


设置中英文 caption，对于已经设置好的中文 caption 格式，只使用第二个语句设置 second 就行了。这两句不能放在 `figure` 环境里使用，必须放在外面。

```
\captionsetup[figure][bi-first]{name=图} %放在外面
\captionsetup[figure][bi-second]{name=Figure}
\begin{figure}[ht!]
\centering
\includegraphics[width=4cm]{fig/tju.pdf}
\bicaption{天大校徽}{The school badge of Tianjin University}
\end{figure}
```



图 2-4: 天大校徽

Figure 2-4: The school badge of Tianjin University

标签放在 `bicaption` 的任何一个 caption 内容里（花括号内）都可以正确引用图的序号。

2.5.3 子图

子图有三个包，`subfigure` 和 `subfig` 是一个作者写的，`subfig` 据说更新一点，但是手册是一年的。比较新的是 `subcaption`。这三个包不能同时使用。

`subfigure`

加载子图包 `subfigure`。

```
\usepackage{subfigure}
```

使用框架是

```
\begin{figure}[ht!]
\centering
\setcounter{subfigure}{0} %多次使用subfigure时重新编号
\subfigure[a is first]{\includegraphics[width=3cm]{fig/tju.pdf}}\hspace{5mm}
\subfigure[标签加载后面\label{f2}]{\includegraphics[width=3cm]{fig/tju.pdf}}\hspace{5mm}
\subfigure[]{\includegraphics[width=3cm]{fig/tju.pdf}}\hspace{5mm}
\subfigure[]{\includegraphics[width=3cm]{fig/tju.pdf}}
\caption{天大校徽。 \label{f-total}}
\end{figure}
```

引用图使用图`\ref{f2}`。对于图来说，`\label{标签名}`要加在 `caption` 内容里，即花括号内，`subfigure` 的 `caption` 内容的位置是方括号内，所以标签要加在方括号里面，这样可以直接引用



图 2-5: tju

子图序号。

图的 caption 和子图的 caption 可以同时加标签，名字不同不会互相影响的。比如图`\ref{f-total}`。

subfig

加载 subfig 包

```
\usepackage{subfig}
```

子图结构如下

```
\begin{figure}
\centering
\subfloat[one]{\label{sub-1}}
\includegraphics[width=5cm]{fig/tju.pdf}
\subfloat[two]{\label{sub-2}}
\includegraphics[width=5cm]{fig/tju.pdf}
\caption{tju}
\end{figure}
```

subfig 不需要重新设置子图编号，每次会重新开始，这点比 subfigure 方便一些。

subfigure 包和 subfig 包测试的时候不能同时加载，否则会报错。

默认子图是 abcd，修改是用下面的语句，比如修改为小写罗马数字，

```
\renewcommand{\thesubfigure}{\roman{subfigure}}
```

subcaption

加载 subcaption 包

```
\usepackage{subcaption}
```

子图结构如下

```
\begin{figure}[ht!]
\centering
```

```
\subcaptionbox{first figure.\label{sub-1}}
{\includegraphics[width=2cm]{../tju.pdf}}
\hspace{1cm}
\subcaptionbox{second figure.\label{sub-2}}
{\includegraphics[width=5cm]{../tju.pdf}}
\caption{two figures.\label{try-1}}
\end{figure}
```

这个包的 box 还可以有一个尺寸，但是推荐在\includegraphics里设置图片大小。图片大小超过 box 的设置值，默认是溢出 box。并排放两张图，这两张图会重叠，每张图都是以自己的 box 左下角为起点的，这就是 box 和图片的关系。这个包可以不用 parbox 和 minipage，非常方便的并排放两个 figure 独立环境，默认是 caption 第一行在同一行，图片底部水平对齐。

2.6 表格

表格的基础环境是 tabular，独立环境是 table，tabular 的手册是 array。

texdoc array

2.6.1 基本功能

表格基础包是 array，虽然最基本的表格功能不需要加载 array 包，但是除非做一个极简表格，否则 array 包还是要加载的。

```
\usepackage{array} %加载后才可以使⤿用一些功能
```

表格的基本环境是 tabular，tabular 环境是文本。表格的独立环境是 table，table 里可以有 caption，一般表格的 caption 在表格上面。

tabular 的参数，l 表示左对齐，r 表示右对齐，c 表示居中对齐，p{长度} 表示设置这一列的宽度（必须有单位），内容超出自动换行。m{长度} 类似，但是 m 的结果是上下居中对齐，p 是顶端对齐，b 也可以，表示底部对齐。一般情况下推荐用 m。有多少列就需要写多少个参数。

表格内容，一行内的单元格由 & 分开，每行用强制换行符\\分开。

```
\begin{table}[ht!] %当前位置
\caption{这是一个简单表格，caption默认居中。}
\centering
\begin{tabular}{l r c w{c}{12em} m{2.5cm} \hline %一条横线
name &value &description &功能 &说明\\
\hline
苹果 &1元 &水果 &对健康生活很有好处 &对健康生活很有好处\\
\hline %一条横线
\end{tabular}
\end{table}
```

表 2-2 这是一个简单表格，caption 默认居中。

name	value	description	功能	说明
苹果	1 元	水果	对健康生活很有好处	对健康生活很有好处

`p{宽度}`定义单元格宽度，且上下位置是顶部，`m{宽度}`定义单元格宽度，且上下位置是居中，`b{宽度}`定义单元格宽度，且上下位置是底部，`w{对齐方式}{宽度}`定义单元格宽度，`c` 是左右位置居中。

列方向如果需要线，可以在 `tabular` 的参数表相应位置加竖线。横线还需要`\hline`语句。

```
\begin{tabular}{|l|l|l|l|}\hline
a&b&c\\ \hline
e&f&g\\ \hline
h&i&j\\
\hline
\end{tabular}
```

a	b	c
e	f	g
h	i	j

同时改变横线和竖线的线宽可以用下面的语句，如果写到 `table` 环境内只对当前 `table` 有效。

```
\setlength{\arrayrulewidth}{5pt}
\begin{tabular}{|l|}
\hline A \\ \hline
\end{tabular}
```

A

可以通过下面的设置拉长列宽和行高，如果写到 `table` 环境内只对当前 `table` 有效。

```
\setlength{\extrarowheight}{10pt} %行高
\setlength{\tabcolsep}{15pt} %列宽
\begin{tabular}{|l|l|l|l|}\hline
a&b&cccc\\ \hline
e&f&g\\ \hline
h&i&j\\
\hline
\end{tabular}
```

a	b	cccc
e	f	g
h	i	j

`>{前语句}`对齐方式`<{后语句}`设置某列前置语句和后置语句，下面的例子前面加第，后面加日，表格内只需要输入数字，重复的文字不需要再输入。波浪号表示不换行。

```
\begin{tabular}{>{第~}c<{~日} l}\toprule
1 &看书\\
2 &计算\\
3 &总结\\
\bottomrule
\end{tabular}
```

第 1 日	看书
第 2 日	计算
第 3 日	总结

2.6.2 三线表

科技论文中常用的是三线表，使用三线表包 booktabs

```
\usepackage{booktabs} %三线表
```

三线表的三条线分别是顶部粗线`\toprule`、中间细线`\midrule`和底部粗线`\bottomrule`。直接在后面加方括号设置宽度，可以修改线宽。默认设置就很漂亮了，下面的例子仅仅为了表示可以这么做。

```
\centering
\begin{tabular}{l r c m{2.5cm}}\toprule[2pt]
name &value &description &功能\\ \midrule
苹果 &1元 &水果 &对健康生活很有好处\\
荔枝 &15元 &水果 &上火\\
\bottomrule
\end{tabular}
```

name	value	description	功能
苹果	1 元	水果	对健康生活很有好处
荔枝	15 元	水果	上火

booktabs 包还定义了特殊的线，线长为默认表格宽度。

```
\specialrule{线宽}{上间距}{下间距} %tabular环境中使用
```

可以用`\cmidrule`指定其实划中间的细线

```
\cmidrule(l{左边缩短距离}r{右边缩短距离}){n1-n2}\cmidrule(l{左边缩短距离}r{右边缩短距离}){m1-m2}
```

从第 `n1` 栏到第 `n2` 栏划细线，默认有缩短的距离，一般不够用，所以自己需要设置一下，否则细线有可能会连起来。

```
\begin{tabular}{l c c c c c}\toprule
name &A &B &C &D &E\\
\cmidrule(l{5pt}r{5pt}){2-3}\cmidrule(l{5pt}r{5pt}){4-6}
x &1 &2 &3 &4 &5\\
\bottomrule
\end{tabular}
```

name	A	B	C	D	E
x	1	2	3	4	5

2.6.3 表格颜色

使用 colortbl包设置表格颜色。

```
\usepackage{colortbl}
```

或者在 xcolor 包加 table 选项

```
\usepackage[svgnames,table]{xcolor} %推荐这种，更多功能
```

然后，可以设置线的颜色

```
\arrayrulecolor{blue} %放在tabular环境内只对当前表格有效
```

可以设置行的背景色，而且是双色相间的

```
%rowcolors[是否加横线]{起始行}{奇数行颜色}{偶数行颜色}
\rowcolors[\hline]{1}{cyan!30}{yellow!40} %第一行有颜色
\begin{tabular}{l c r l}
goods & price & quantity & total\\
apple & 0 & 5 & 5\\
banana& 1 & 3 & 3\\
pear & 1 & 3 & 3\\
orange& 2 & 2 & 4\\
\rowcolor{red} %仅下面一行换色
apple & 0 & 5 & 5\\
banana& 1 & 3 & 3\\
\end{tabular}
```

goods	price	quantity	total
apple	0	5	5
banana	1	3	3
pear	1	3	3
orange	2	2	4
apple	0	5	5
banana	1	3	3

使用可以\columncolor{颜色}设置列颜色，这个可以放到列的前置语句里，对整列生效。还可以使用\cellcolor{颜色}设置某个格子的背景色，放在单元格内容前。

```
\begin{tabular}{>{\columncolor{cyan!50}}l >{\columncolor{green!50}}c r l}
goods & price & quantity & total\\
apple & 0 & 5 & 5\\
banana& 1 & 3 & 3\\
pear & \cellcolor{yellow}1 & 3 & 3\\
orange& 2 & 2 & 4\\
\rowcolor{red} %仅下面一行换色
apple & 0 & 5 & 5\\
banana& 1 & 3 & 3\\
\end{tabular}
```

goods	price	quantity	total
apple	0	5	5
banana	1	3	3
pear	1	3	3
orange	2	2	4
apple	0	5	5
banana	1	3	3

2.6.4 单元格内换行 makecell 包

表格的单元格内不支持换行。使用 makecell 包可以很容易对单元格断行，并且设置对齐方式

```
\usepackage{makecell}
```

在单元格内使用语句

```
\makecell[对齐方式]{第一行内容\\第二行内容}
```

默认的与表格其他单元格竖直方向对齐方式是上下居中方式。如果双重指定对齐方式，比如外部底部对齐，内部左对齐，如下面第二行的例子，第二个左对齐需要花括号括起来。也可以加p{宽度}这样的对齐方式，但是必须用双重花括号括起来。

```
\begin{tabular}{l l l}\toprule
name & description & price\\ \midrule
apple & A is good & \makecell[l]{1 in summer good\\2 in winter}\\ \hline
apple & A is good & \makecell[b{l}]{1 in summer good\\2 in winter}\\ \hline
apple & A is good & \makecell[b{p{2cm}}]{1 in summer good\\2 in winter}\\
\bottomrule
\end{tabular}
```

name	description	price
apple	A is good	1 in summer good 2 in winter
		1 in summer good
apple	A is good	2 in winter
		1 in summer good
apple	A is good	2 in winter

如果单元格内是大的图片、有机分子式时，对齐方式是同行对齐方式，对齐基准线是图片的底边位置。

```
\begin{tabular}{cl}
\includegraphics[width=2cm]{fig/tju.pdf} & 这是一段文字\\
\includegraphics[width=3cm]{fig/tju.pdf} & 这是一段文字\\
\end{tabular}
```



这是一段文字



这是一段文字

如果希望图文竖直方向居中对齐，则需要将图片用`\makecell{}`括起来。

```
\begin{tabular}{c1}
\makecell[c]{\includegraphics[width=2cm]{fig/tju.pdf}} & 这是一段文字\\
\makecell[c]{\includegraphics[width=3cm]{fig/tju.pdf}} & 这是一段文字\\
\end{tabular}
```



这是一段文字



这是一段文字

还有一种方式，该列格式设置为`m{宽度}`，但是该方式只能获得单元格竖直居中对齐，无法在该列水平居中对齐。这种方式必须手动设置宽度，列间距不够优美。

2.6.5 同行合并列

在同一行中合并 n 列的语句为`\multicolumn{n}{对齐方式}{合并后单元格内容}`。例子中 Item 这里相当于合并了两列，与下面居中对齐。`\cmidrule{第n列-第m列}`是三线表包中从第 n 列到第 m 列划一条细横线的语句。不使用三线表包时，`\cline{第n列-第m列}`用法相同。

```
\begin{tabular}{l l r}\toprule
\multicolumn{2}{c}{Item} & \cmidrule{1-2} %1--2列划线
name & description & price\\ \midrule
A & A is good & 1\\
\bottomrule
\end{tabular}
```


Item		
name	description	price
A	A is good	1

2.6.6 同列合并行 multicol 包

同列合并行需要 multirow 包。

```
\usepackage{multirow}
```

下面的例子，相当于合并了 3 行，默认对齐方式是竖直居中

```
\begin{tabular}{l l l}\toprule
序号 & 原子 & 分类\\ \midrule
1 & Li & \multirow{3}{5cm}{第一主族}\\
2 & Na & \\
3 & K & \\ \midrule
4 & Be & \multirow[b]{3}{5cm}{第二主族} \\ %改为底部对齐
5 & Mg & \\
6 & Ca & \\ \bottomrule
\end{tabular}
```

序号	原子	分类
1	Li	第一主族
2	Na	
3	K	
4	Be	第二主族
5	Mg	
6	Ca	

2.6.7 表格 tabular 环境的嵌套

单元格换行、合并行、合并列虽然是获得复杂表格结构的常用语句，但是有些情况，用表格嵌套更方便。多个表格嵌套时，表格列格式声明前可以添加并排表格环境的对齐参数，下面的例子是顶部对齐。

```
\begin{tabular}{ll}\toprule
\begin{tabular}{t}{ll}
左边~I&子表格1\\
左边~I&子表格1\\
左边~I&子表格1\\
\end{tabular}
&\begin{tabular}{t}{lp{2cm}}
右边~I&子表格顶部对齐\\
右边~I&子表格顶部对齐\\
\end{tabular}
\end{tabular}
```

```
\end{tabular}\\
\bottomrule
\end{tabular}
```

左边	子表格 1	右边	子表格顶部
左边	子表格 1		对齐
左边	子表格 1	右边	子表格顶部
			对齐

绘制复杂表格，合理地拆解表格结构非常重要，这个需要经验和灵感。

2.6.8 表格脚注

表格的脚注希望显示在表格下面，而不是页面底部，需要加载包 `threeparttable`。

```
\usepackage{booktabs}
\usepackage{threeparttable} %表格内脚注
```

代码如下

```
\begin{table}[h]
\begin{center}
\begin{threeparttable}
\caption{表头脚注\tnote{\dag}}
\begin{tabular}{l l l}\toprule
name & description & price\\ \midrule
apple\tnote{1} & A is good & 1\tnote{a}\\
\bottomrule
\end{tabular}

\begin{tablenotes}
\footnotesize
\item [\dag] 表头说明
\item[1] 水果
\item[a] 单位：元
\end{tablenotes}
\end{threeparttable}
\end{center}
\end{table}
```

表 2-3 表头脚注[†]

name	description	price
apple ¹	A is good	1 ^a

[†] 表头说明
¹ 水果
^a 单位：元

这种方式可以满足大部分表格脚注的要求，但是没有超链接。因为脚注跟随表格，所以超

链接用处也不是很大。

2.6.9 跨页表格 longtable

使用 longtable 包可以很方便的使得表格跨页。

```
\usepackage{longtable}
```

longtable 环境为

```
\begin{longtable}{l}
\caption{环境内可以加caption}\\
\endhead %每页都有caption, 否则只有第一页有
表格内容
\end{longtable}
```

caption 放在前面, 每页都会重复 caption。如果希望后面的 caption 不同, 比如添加 (continue), 则需要写成下面的样式

```
\caption{caption的内容}\\
\endfirsthead %第一页caption是上面花括号里的
\caption{caption的内容(continue)}\\
%这里还可以放每页重复的表格行
\endhead %其他页的caption是上面花括号里的
```

longtable 环境里面不能添加\centering, 有些表格设置语句也不能添加。比如要使得列间距扩大, 要写为下面的形式

```
\begin{longtable}{@{\extracolsep{10pt}} m{1cm}| m{14cm} }
```

longtable 可以使得表格跨页, 但是并不能使得单元格内容跨页。

可以定义\NewLine并使用它手动分页。

```
\newcommand\NewLine{\setlength\parfillskip{0pt}\tabularnewline}
```

threeparttablex 包是将 threeparttable 与 longtable 功能结合的拓展包。

2.6.10 表格单元格内的公式环境

表格内的单元格内容也可以是带编号的公式环境, 该列的形式必须是m{宽度}, 不能是 lcr, 且公式环境本身必须用花括号括起来。这是因为独立的公式环境默认是公式居中对齐、编号右对齐的, 不设置单元格宽度, 就破坏了公式环境的默认格式, 无法正确显示。另外, 公式与上下文的默认间距比较大, 可以用\setlength{\abovedisplayskip}{距离}和\setlength{\belowdisplayskip}{距离}语句局域调整。这两个语句对 equation 环境有效, 但是对表格单元格内的 equation 环境无效, 可以用 align 环境替换 equation 环境写表格内的单行编号公式。

```
\begin{table}
\setlength{\abovedisplayskip}{-12pt}
\setlength{\belowdisplayskip}{-12pt}
\begin{tabular}{l m{4cm} l}
\hline
公式 &{
```

```

\begin{align}
f(x)
\end{align}} &这是一段文字，m是否居中\\
\hline
多行公式 &{
\begin{align}
&f(x)=\sin x\\
&g(x)=\cos y
\end{align}
} &这是一段文字\\
\hline
\end{tabular}
\end{table}

```

2.6.11 参考书和一些有用的包

表格有本集大成的书，Typesetting tables with L^AT_EX，作者是 Herbert Voss，2011 年。基本常用用法都罗列出来了。

一些可能有用的包如下：

- datatool 包，从数据文件导入数据方式做表
- tabularx 包，设置表格宽度
- widetable 包，与 tabularx 包接近，根据表格宽度自动计算列间距。
- diagbox 包，画斜线表头的。slashbox 已经不再 texlive 里了。

2.7 超链接包 hyperref

做复杂的事情很简单，这点在排版这一章中体现在超链接包 hyperref 上，该包尽量放在其他包后面，但是 glossaries 包例外。加载包的语句为

```
\usepackage[unicode,colorlinks,linkcolor=blue]{hyperref}
```

中文环境推荐 CJKutf8 包，而不是 CJK 包，这样自动生成的 pdf 文件书签中文显示不会出现乱码。

包的参数最常用的是 unicode：设置字符集；colorlink：链接带颜色，参考文献、索引等有一套默认的标准颜色设置；linkcolor=blue：设置 label 的引用链接为蓝色，因为默认是红色，所以如果觉得红色太醒目，设置为蓝色比较好看。

默认自动包含目录标题超链接（也可以修改为页码超链接）、index、参考文献、label 的引用、网址、文件等超链接，并且自动生成 pdf 文件书签。比较常用的包选项设置为：

pagebackref=false/true	参考文献带页码超链接，默认无
hyperindex=true/false	默认索引超链接
hyperfootnotes=true/false	默认脚注超链接
linktoc=section/page/all/none	目录默认小节标签超链接
linktocpage=false/true	目录仅页码超链接，默认是标题
colorlinks=false/true	超链接显示不同颜色
linkcolor=red	设置文字超链接颜色，默认红色
citecolor=green	参考颜色超链接颜色，默认绿色
filecolor=cyan	网址和文件超链接颜色，默认青色
runcolor=filecolor	运行超链接（视频等），默认同文件
urlcolor=magenta	网址超链接颜色为品红
hidelinks	隐藏颜色和边框
bookmarks	pdf 自动书签
bookmarksnumbered	pdf 书签带章节号

除了默认外，图表编号、公式编号和需要引用的地方都需要通过语句`\label{标记名}`设置标记，然后通过`\ref{标记名}`引用，也可以通过`\pageref{标记名}`引用页码，公式一般用`\eqref{标记名}`引用，在数学公式一章中还会介绍。标记名不能数字开头，不带点，个人喜欢使用连字符，英文冒号、英文感叹号都可以，方便分类。

```
\label{eq-fft} \label{eq:fft}
```

pdf_latex 编译时不能是中文，xelatex 编译是标记名可以使用中文。
下面介绍常用的例子。

章节的标记和引用

不在任何图表公式环境内的标记视为对章节的标记，所以标记章节时放在标题花括号内外都可以。其他标记都必须放在环境内，否则会误标记到章节。

```
\section{第一节}\label{sec-1}
\subsection{第二节}\label{sec-2}

第\ref{sec-1}节在\pageref{sec-1}页
第\ref{sec-2}节在\pageref{sec-2}页
```

图表的标记和引用

图表的 label 要添加到图表环境里或者 caption 里面，引用时使用`\ref{标记名}`获得图表的编号，使用`\autoref{标记名}`获得带 caption name 的结果。

```
\begin{figure}
\caption{aaa\label{fig-1}}
\end{figure}

\begin{table}\label{table-1}
...
\end{table}
```

Figure~\ref{fig-1}在\pageref{fig-1}页，
\autoref{table-1}在\pageref{table-1}页。

如果不修改图表的 figurename 和 tablename，这样引用的结果没有问题。如果修改了，就需要重新定义自动引用名。

```
\renewcommand{\figureautorefname}{图}
\captionsetup{labelsep=doublespace}
\begin{figure}[ht!]
\renewcommand{\figurename}{图}
\centering
\includegraphics[width=4cm]{fig/tju.pdf}
\caption{天大校徽\label{tju-logo}}
\end{figure}
使用\lstinline!\autoref{tju-logo}!得到\autoref{tju-logo}，使用\lstinline!图\ref{tju-
logo}!得到图\ref{tju-logo}。
```



图 2-6 天大校徽

使用\autoref{tju-logo}得到图 2-6，使用图\ref{tju-logo}得到图2-6。

引用对自动编号的图、表、公式、章和节的编号、脚注等都有效，对无编号的可以\pageref得到正确页码，使用\ref不会报错，但是得不到正确结果。

网页、文件和运行程序

网页有两种形式，一种是直接显示网址，点击网址文字可以跳转到默认浏览器打开链接。

```
\url{www.baidu.com}
```

还有一种是设置点击标识，可以是文字或图片，点击标志可以跳转到默认浏览器打开链接。

```
\href{www.baidu.com}{百度}
\href{www.tju.edu.cn}{\includegraphics[width=1cm]{fig/tju.pdf}}
```

也可以跳转到文件，比如 pdf 文件，注意最好给出全路径文件名，这样不容易出错。pdflatex 编译时全路径文件名不能是中文，否则使用 xelatex 编译。

```
\href{全路径文件名}{链接标识名}
```

还可以使用 run 来播放视频，下面的例子全部是仅仅 linux 下测试有效。run 语句表示运行 mplayer 命令播放文件，windows 系统应该需要给出播放器的命令行命令。run 语句是否可行与 pdf 阅读器相关。即使 linux 下，不同 pdf 阅读器的代码测试结果也是不同的，xpdf 阅读器

结果全部正确，qpdfview 则只能用 mplayer 打开视频，acroread 则全部不行。所以 run 方式是否可行深度依赖于系统和 pdf 阅读器。

```
\href{run:mplayer 全路径文件名}{视频}
```

也可以使用 run 来运行程序

```
\href{run:全路径可运行程序名}{程序}
```

其他命令也可以用 run 打开文件，或者给出全路径文件名，或者是当前目录下的文件名。

```
\href{run:xloadimage 当前目录下的文件名}{打开}\\
\href{run:xloadimage 全路径文件名}{open}
```

pdf 文件信息

利用下面的代码可以输出 pdf 文件信息，包括 title 和 author 等。

```
\hypersetup{
  pdftitle = {The documents title},
  pdfauthor = {name}
}
```

2.8 文章 article 类

文章类默认可以添加题目、作者、时间，用\thanks{脚注内容}加载题目里的脚注。

```
\title{You Have Only One Life}
\author{作者1 \and 作者2\thanks{email}}
\date{} %取消日期
\maketitle %显示题目
```

默认是有日期的，也即不写\date语句，会采取默认结果

```
\date{\today} %每次编译时改变
```

指定日期要填写日期

```
\date{2021年10月1日}
```

默认的题目等字号都有些太大，仅仅是自己写文章用，可以手动直接修改

```
\title{\large\bf You Have Only One Life\vskip-0.5em}
\author{\small 作者1 \and \small 作者2\thanks{email}}
\date{\vskip-0.5em\small 2021年10月1日}
```

从例子可以看出，这是一个很繁琐的工作。注意：调整纵向间距的语句一定要放在题目里和日期里，不要放在作者里，因为 maketitle 的原始代码里作者是以表格的方式构成的。这也是为什么作者的字体大小的改变需要逐项都修正的原因，作者很多时就非常麻烦了。杂志社做模板时一般都是重新定义 maketitle 命令。自己做模板直接按照 maketitle 命令的定义方式做效果更方便一些。具体可以参照 classes 文档。

```
texdoc classes
```

文章类的默认有摘要环境

```
\begin{abstract}
Abstract text ...
\end{abstract}
```

默认自动添加居中单列一行的 Abstract，汉化方式为

```
\renewcommand{\abstractname}{摘要}
```

文章默认是单栏，但是我们看到的很多文献是双栏的，在加载类的时候使用双栏选项。

```
\documentclass[12pt,twocolumn]{article}
```

这样文章的题目、作者和日期，也即 maketitle 的内容，是单栏，其余都是双栏。如果希望摘要仍然是双栏，可以有两种做法，简单的就是加载 abstract 包。

```
\usepackage{abstract} %摘要包
```

使用时很简单

```
\twocolumn[
\maketitle
\begin{onecolabstract}
Abstract text ...
\end{onecolabstract}
]
\saythanks %双栏格式显示标题脚注
```

这么做 thanks 脚注有时显示不对，可以使用下面的语句

```
\twocolumn[
\begin{@twocolumnfalse}
\maketitle
\begin{abstract}
Abstract text ...
\end{abstract}
\end{@twocolumnfalse}
]
\saythanks
```

这样显示无误，脚注也是双栏的。

默认双栏按内容填充，并不好看，添加 balance 包，可以使得内容不满时，双栏底部是对齐的。

```
\usepackage{balance} %双栏内容不满的页面底部对齐
```

索引默认是双栏的，但是单栏的书添加 balance 后并不会自动 balance，需要使用下面的命令。

```
\balance
\printindex
```


2.9 参考文献

参考文献分三个部分：数据文件（bib 文件），样式和命令。参考文献目前有两套体系，一套是 natlib 包，用 bibtex 命令编译，参考文献的格式使用 bst 文件。这是目前杂志社常用的方式，texlive 里内置了许多杂志社的 bst 格式文件。另一套是 biblatex 包，用 biber 命令编译，这是比较新的参考文献格式方法。参考文献本身的条目，都可以采用 bib 格式，但是 biblatex 和 biber 这种方法还可以采用其他格式。我个人习惯于 bib 格式，因为它相当简洁，大部分时候足够用。下面主要以 bib 格式来讲述参考文献的引用。

2.9.1 bib 格式的文献条目

bib 文件格式是最常见的参考文献条目格式之一，很多杂志社都提供文献的 bib 格式方便大家引用。下面是文章类的例子。

```
@article{huminglie-zgjg-2021, %文献的自定义引用名
title={光纤激光器泵浦的飞秒光学参量振荡器研究进展}, %文章题目
author={胡明列 and 王珏 and 范锦涛}, %作者用and分开
journal={中国激光}, %杂志名
volume={48}, %卷号
issue={19}, %期号
pages={1901001}, %页码，有页码范围的使用55--59
year={2021}, %年，最后的逗号可以省略
}
```

@article 是文章，参考文献的类别，后面的都是域。bib 常见类参见下表。

表 2-4 bib 文件的常见类

类	@ 类名
公开出版的图书	@book
硕士论文	@mastersthesis
博士论文	@phdthesis
会议论文集	@proceedings
会议论文集中的一篇	@inproceedings
会议论文	@conference（等价于 @inproceedings）
无出版商或作者的图书	@booklet
书籍的一部分章节	@inbook
书籍中带独立标题的章节	@incollection
技术文档	@manual
其他	@misc
教育，商业机构的技术报告	@techreport
未出版的论文，图书	@unpublished

每一个参考文献类别的域分为必要域和可选域，总结如下。

- @article
必要域 author, title, journal, year
可选域 volume, number, pages, month, note
- @book

必要域 author/editor, title, publisher, year

可选域 volume/number, series, address, edition, month, note

■ @mastersthesis

必要域 author, title, school, year

可选域 type, address, month, note

■ @phdthesis

必要域 author, title, year, school

可选域 address, month, keywords, note

■ @proceedings

必要域 title, year

可选域 editor, volume/number, series, address, month, organization, publisher, note

■ @conference

必要域 author, title, booktitle, year

可选域 editor, volume/number, series, pages, address, month, organization, publisher, note

■ @inproceedings

必要域 author, title, booktitle, year

可选域 editor, volume/number, series, pages, address, month, organization, publisher, note

■ @booklet

必要域 title

可选域 author, howpublished, address, month, year, note

■ @inbook

必要域 author/editor, title, chapter and/or pages, publisher, year

可选域 volume/number, series, type, address, edition, month, note

■ @incollection

必要域 author, title, booktitle, publisher, year

可选域 editor, volume/number, series, type, chapter, pages, address, edition, month, note

■ @manual

必要域 title

可选域 author, organization, address, edition, month, year, note

■ @misc

必要域 none

可选域 author, title, howpublished, month, year, note

■ @techreport

必要域 author, title, institution, year

可选域 type, number, address, month, note

■ @unpublished

必要域 author, title, note

可选域 month, year

zharticle.bst

天津大学的本科生和研究生论文的 bst 文件，可以采用 texlive 版本自带的 zharticle.bst 文件，这是 Xu Yuan 从 seuthesis.bst 修订的，除了博士论文和硕士论文的标识要求不同以外，其余皆可满足天津大学的要求。可以自行删除 [D]:[Master's Thesis] 和 [D]:[PhD Thesis] 后面的:[Master's Thesis] 和:[PhD Thesis]，天津大学只要求标识为 [D]。zharticle.bst 文件并不在 bst 默认搜索目录下，而是在 texlive/2021/texmf-dist/doc/latex/seuthesis/zharticle/下，可以复制该文件到自己的 tex 文件目录下使用。如果希望添加到系统里，可以复制该文件到系统默认搜索目录下，并且执行命令 texhash，重建索引即可。但是仅仅使用一两次的 bst 不推荐这样的方法。

texhash

缺乏必要域会报错，但是如果缺乏可选域也会获得错误结果，这其实取决于 bst 文件的规定。按 zharticle.bst 的规定，书、硕士和博士论文，都必须有地址，否则会输出不正确的结果。如果天津大学论文的参考文献格式要求没有地址项，则还需要自行修改 zharticle.bst 文件。这个规定是考虑到很多国际出版社是存在不同地区公司的，而大学会有不同城市的校区，因此显示地址是合理的要求。

doi 域

上面列出的 bib 域里是没有 doi 的，但是在.bib 文件添加 doi 域并不会报错，只是不会显示，比如使用 unsrt 格式时，bib 里的 doi 域不会显示出来。但是目前很多杂志社的 bst 支持 doi，比如使用 elsarticle-num 时，会显示出来 doi 号且自动为超链接格式，可以自动打开网页，即从 doi 网站进入到该文献所在的网页。因此推荐在 bib 文件中都添加 doi 项。下面介绍的 biblatex 包是支持 doi 项的。

2.9.2 natlib 包 +bibtex 编译

使用 natlib 包的语句为

```
\usepackage[super,square,comma,sort&compress]{natbib}
```

super 表示文献为上标格式，square 表示引用编号自动添加方括号，compress 表示自动压缩，即 [5-8]，而不是 [5,6,7,8]。

在文章里使用 cite 语句，还有最常见的使用 citenum 语句引用参考文献的序号。

飞秒光纤激光器还可以作为飞秒光学参量振荡器的泵浦源\cite{huminglie-zgjg-2021}和双光子荧光显微镜的光源。参见参考文献\citenum{huminglie-zgjg-2021}

文章后面的文献列表将按照加载的格式文件排列。

```
\bibliographystyle{zharticle} %使用zharticle.bst作为格式文件
```

向杂志社投稿文章时一般会使用杂志社的类，而不是 article，这时有两种情况。杂志社的类包含参考文献格式，就不用`\bibliographystyle`语句了。如果杂志社的类不包含参考文献格式，则需要该语句。有的大杂志社如 iee 或 acs，有不只一种参考文献格式，加载与投稿杂志匹配的格式。大部分的 bst 文件在 texmf-dist/bibtex/bst 目录下。下面是几个例子。

```
\bibliographystyle{IEEEtranSN} %ieee
\bibliographystyle{IEEEtran} %ieee
\bibliographystyle{elsarticle-num} %elsevier
```

然后指定自己写的 bib 文件，

```
\bibliography{a} %bib文件名是a.bib
\bibliography{bib文件名} %可以是其他文件名
```

编译方式：分四步编译，pdflatex 命令编译 tex 文件 3 次，bibtex 命令编译 bib 文件 1 次。

```
pdflatex a.tex
bibtex a %编译的是主tex的文件名
pdflatex a.tex
pdflatex a.tex
```

可以写多个 bib 文件，任意起名，比如 a1.bib 和 a2.bib，加载时用逗号分割

```
\bibliography{a1,a2}
```

编译时还是上面的步骤，bibtex 编译的是 a，不带扩展名

可以通过下面的语句自己编写 bst 文件，执行完该语句后，通过选择可以自动建立 bst 文件，但是总有不能满足要求的地方需要修改 bst 文件。

```
latex makebst
```

bst 格式文件虽然比较复杂，修改也比较麻烦，但是 natbib+bibtex 方式仍然是最简洁、冗余文件最少的编译方法。

投稿的时候，杂志社的模板（杂志社自己定义的类）都是包含许多包的加载的，基本不会遇到 natbib 包选项与 style 匹配的问题。如果是自己写书，使用定义好的 style，就需要注意在加载 natbib 包时，选项与选用的 style 匹配，选项不能全部删除，有些选项必须加载。natbib 的选项与 style 冲突时，会以 natbib 选项为主，比如格式不是 super，定义了 super，会以 super 方式显示。

2.9.3 natbib 包情况下分章参考文献

chapterbib 包

分章参考文献需要加载 chapterbib 包，chapterbib 包与 biblatex 包不匹配，必须使用 natbib 包。

```
\usepackage[super,square,comma,sort&compress]{natbib}
\usepackage[sectionbib]{chapterbib} %参考文献为不编号小节
```

文件结构必须是每章单独一个 tex 文件，比如原始文件为 a.tex，第一章文件为 c1.tex，第二章为 c2.tex。在 a.tex 里以`\include`形式加载 c1.tex 和 c2.tex。

```
\include{c1.tex} %不写扩展名也可以，推荐写全名
```

```
\include{c2.tex}
```

`\input{文件名}`和`\include{文件名}`最大的不同是，后者编译时对所有加载的文件都单独建立辅助文件，这样就可以对 `c1.tex` 和 `c2.tex` 进行 `bibtex` 编译。`c1.tex` 和 `c2.tex` 除了章节内容和引用外，最后都包含各自的同名 `bib` 文件。

章节内容

```
\bibliographystyle{elsarticle-num}
\bibliography{c1} %与c1.tex对应的bib文件
```

章节内容

```
\bibliographystyle{elsarticle-num}
\bibliography{c2} %与c2.tex对应的bib文件
```

编译时按下面的顺序编译

```
pdflatex a.tex %编译主文件
bibtex c1 %编译第一章参考文献
bibtex c2 %编译第二章参考文献
pdflatex a.tex %第二次编译主文件
pdflatex a.tex %第三次编译主文件
```

最后生成的 `pdf` 文件，参考文献直接放在章的最后，不单独分页，参考文献本身没有小节编号，目录不显示分章的参考文献。这也是分章参考文献最常见的格式。`book` 类的默认`\bibname`是 `Bibliography`，如果希望修改名字，可以加到 `a.tex`，也可以加到任何一章中，即 `c1.tex` 和 `c2.tex` 都可以。`\bibname`是对全局有效的，只需要添加一次就可以修改名称，比如修改成 `References` 或者汉化。

```
\renewcommand{\bibname}{References}
\renewcommand{\bibname}{参考文献}
```

`article` 类时是`\refname`

```
\renewcommand{\refname}{参考文献}
```

如果去掉包的 `sectionbib` 选项，

```
\usepackage{chapterbib}
```

参考文献就会以不编号的章的格式显示，默认 `book` 类的设置使得分章的参考文献从奇数页开始，按章的格式显示。这样会添加许多空白页，所以推荐加载 `sectionbib` 选项。

还可以选择双重参考文献，使用下面的语句

```
\usepackage[duplicate,sectionbib]{chapterbib}
```

然后在主文件 `a.tex` 末尾添加

```
\backmatter
\bibliographystyle{elsarticle-num}
\bibliography{s1,s2}
```

文章最后会重复一遍分章参考文献，但是这种情况并不常见。

bibunits 包

分章参考文献也可以加载 bibunits 包。

```
\usepackage[super,square,comma,sort&compress]{natbib}
\usepackage[sectionbib]{bibunits} %
```

bibunits 可以使用一个 tex 文件 a.tex 和一个 bib 文件 ref.bib, tex 和 bib 文件不需要强制同名, 不用`\include`语句方式。不需要加载 natbib 包也可以获得正确结果, 但是不加载 natbib 包参考文献没有超链接, 所以推荐加载。

下面的例子是按书格式的结构,

```
\chapter{laser}
\begin{bibunit}[plain]
aaa\cite{fluid}
\putbib[ref] %ref是bib文件, 一个bib文件就行
\end{bibunit}

\chapter{math}
\begin{bibunit}[plain] %可以选择不同的参考文献样式
aaa\cite{fs-1}
\putbib[ref]
\end{bibunit}
```

编译的时候使用下面的语句

```
pdflatex a.tex %编译主tex文件一次
bibtex bu1 %bu1文件是自动生成的
bibtex bu2 %bu2文件是自动生成的, 有几个环境就有几个文件
pdflatex a.tex %第二次编译主tex文件
pdflatex a.tex %第三次编译主tex文件
```

写书的时候, 还是更推荐 chapterbib 包使用`\include`语句方式。

2.9.4 biblatex 包 +biber 编译

使用 biblatex 包, 且使用 biber编译

```
\usepackage[backend=biber]{biblatex} %指定biber编译方式
\addbibresource{a.bib} %指定bib文件
```

biblatex 的格式文件后缀为.bbx, 使用方法为添加选项 style=bbx 文件名

```
\usepackage[backend=biber,style=numeric]{biblatex}
\usepackage[backend=biber,style=authoryear]{biblatex}
\usepackage[backend=biber,style=authortitle]{biblatex}
```

文章中 cite 的引用方式相同。biblatex 可以在 numeric 方式下用`\citeauthor{label}`引用作者。但是 biblatex 没有定义`\citenum`命令, 可以自己定义, 但是没有超链接。

```
\DeclareCiteCommand{\citenum}{\printfield{labelnumber}}{\}
```

biblatex 方式比较容易汉化

```
\printbibliography[title=参考文献] %输出参考文献列表
```

编译方式：分四步编译，pdflatex 命令编译 tex 文件 3 次，biber 命令编译 bib 文件 1 次。

```
pdflatex a.tex
```

编译后获得 a.aux, a.log, a.out, a.run.xml 和 a.pdf 文件，还获得 a.bcf 文件，查看 a.run.xml 文件可以看到该过程所需的文件。此时生成的 a.pdf 文件不能正确显示引用。

```
biber a
```

编译后获得 a.bbl 和 a.blg 文件

```
pdflatex a.tex
pdflatex a.tex
```

编译后获得正确引用的 a.pdf 文件。因为有的时候比较复杂的文件需要编译两次才能正确显示，所以这里建议一般情况都编译两次。

可以使用 sorting 选项，并输出两种不同的排序方式的参考文献列表

```
\usepackage[sorting=nyt]{biblatex} %默认nyt方式排序
...
\printbibliography
\newrefcontext[sorting=ydnt] %重新设置排序方式为ydnt
\printbibliography %再次输出参考文献
```

可以在 bib 文件里添加 doi 选项，且选择是否打印 doi

```
\usepackage[sorting=nyt,doi=true]{biblatex} %false则不显示
```

还可以用\footfullcite代替\cite，会在脚注上显示参考文献列表。这个功能在 beamer 中很有用。

2.9.5 biblatex 包 +bibtex 编译

使用 biblatex 包，但是使用 bibtex 编译

```
\usepackage[backend=bibtex]{biblatex}
\addbibresource{a.bib} %指定bib文件
```

文章中 cite 的引用方式相同 bibtex 编译。

biblatex 方式比较容易汉化

```
\printbibliography[title=参考文献] %输出参考文献列表
```

编译方式：分四步编译，pdflatex 命令编译 tex 文件 3 次，bibtex 命令编译 bib 文件 1 次。

```
pdflatex a.tex
```

编译后获得 a.aux, a.log, a.out, a.run.xml 和 a.pdf 文件，还获得 a-blx.bib 文件（没有 a.bcf 文件），查看 a.run.xml 文件可以看到该过程所需的文件。此时生成的 a.pdf 文件不能正确显示引用。

```
bibtex a %第二步用bibtex编译
```

编译后获得 a.bbl 和 a.blg 文件

```
pdflatex a.tex
pdflatex a.tex
```

最后 pdflatex 编译两次，获得正确引用结果。

只要是使用 biblatex 包，格式文件都是 bbx，bibtex 方式编译也是如此。

虽然最后的 pdf 文件都是一样的效果，但是推荐使用 natlib 包 +bibtex 编译或者 biblatex 包 +biber 编译方式，不建议使用 biblatex 包 +bibtex 编译方式。

2.9.6 biblatex 的常见功能介绍

biblatex 不需要额外的包就能够支持很多复杂功能。

分章参考文献

使用 refsection 选项，每一次 `\chapter` 语句都会有参考文献。也可以每个 part、小节或小小节等开始参考文献。

```
\usepackage[backend=biber,refsection=chapter]{biblatex}
\addbibresource{c.bib}
\begin{document}
\mainmatter
\chapter{}
...
%按无编号section格式开始参考文献
\printbibliography[title=参考文献,heading=subbibliography]
\chapter{}
...
%按无编号章格式开始参考文献
\printbibliography[title=参考文献]
\backmatter
\end{document}
```

也可以使用 refsection 环境方式，此时 biblatex 的选项中不能再添加 refsection 选项。

```
\usepackage[backend=biber]{biblatex}
\addbibresource{c.bib}
\begin{document}
\mainmatter
\begin{refsection} %可以在章开始前
\chapter{}
...
\printbibliography[title=参考文献,heading=subbibliography]
\end{refsection}

\chapter{}
\begin{refsection} %也可以在章开始后
...
\printbibliography[title=参考文献]
\end{refsection}
```



```
\backmatter
\end{document}
```

也可以在末尾集中分章节打印，或者章后打印一次，末尾集中分章节再打印一次，相当于 copy 了一次。

```
\usepackage[backend=biber]{biblatex}
\addbibresource{c.bib}
\begin{document}
\mainmatter
\begin{refsection} %可以在章开始前
\chapter{}
...
\printbibliography[title=参考文献]
\end{refsection}

\chapter{}
\begin{refsection} %也可以在章开始后
...
\printbibliography[title=参考文献]
\end{refsection}
\backmatter
\printbibheading[title=参考文献] %默认heading是Bibliography
\printbibliography[section=1,heading=subbibliography]
\printbibliography[title=第二章,section=2,heading=subbibliography]
\end{document}
```

refsection 环境可以使用每章独立的 bib 文件。导言部分添加多个 bib 文件

```
\addbibresource{s1.bib}
\addbibresource{s2.bib}
```

也可以使用下面的语句表示这是部分 bib 文件

```
\addsectionbib{s1.bib}
\addsectionbib{s2.bib}
```

refsection 环境添加参数

```
\begin{refsection}[s1.bib]
...
\end{refsection}
```

注意 biber 命令编译的时候不要编译 bib 文件名 s1 和 s2，编译主文件名，如果是 a.tex，编译过程如下

```
pdflatex a.tex
biber a
pdflatex a.tex
pdflatex a.tex
```

2.10 版面设计 titlesec、titles 和 titletoc

一共三个包，titlesec、titles 和 titletoc，三个包是一个手册，使用下面的语句调出共同的手册文件

```
texdoc titlesec
```

页眉页脚的简单设置已经包括在上面的手册里了，更复杂的设置，还有一个单独的手册

```
texdoc titles
```

用 titlesec 等包排版时，至少需要编译两次才可以正确显示。编译一次正确，再编译错误，表示有代码错误。第一次编译有错误，但是再次多次编译正确，说明代码正确。曾经遇到过错误严重到必须删除自动生成的文件才能通过编译的特殊情况。

etoc 包可以设计复杂样式的目录，一般科技书籍不会采用太绚丽的目录。

2.10.1 book 类的结构

article 类没有目录、页眉，只有 title 的样式和页脚。但是书有一套默认的样式。必须知道默认设置是什么，才有可能在合适的地方覆盖默认设置，从而得到希望的样式。

book 类默认是 twoside，即区分奇偶页。如果上一章内容在奇数页结束，会在随后的偶数页产生无内容的空白页，每一章都会从奇数页开始。titlesec 包的选项 clearempty，将无内容的空白偶数页的页眉页脚清除，相当于自动添加了\cleardoublepage。加载包时，因为 titlesec 需要加参数，一般写成下面的分开形式

```
\documentclass{book} %默认区分奇偶页，即twoside
\usepackage[clearempty]{titlesec} %偶数页无内容自动空白页清楚页眉页脚
\usepackage{titles,tiletoc}
```

这是因为在默认样式中，章的起始页是 plain 样式，无页眉，页码底部居中。而偶数页没有页脚，页码在页眉的左边。这样空白偶数页清空页眉页脚后，页眉部分比较对称，比较美观。

一般国内的本科生和研究生论文格式都要求章的起始页有页眉页脚，要求偶数空白页仍然保持页眉页脚，这样从对称的角度比较美观，这时不需要这个选项。

将 book 类设置为不区分奇偶页，

```
\documentclass[oneside]{book}
```

加载 ifthen 包，设置局域某几页为不区分奇偶页，然后重置为区分奇偶页。

```
\usepackage{ifthen}
\setboolean{@twoside}{false} %设置为oneside
\setboolean{@twoside}{true} %重新设置为twoside
```

book 类的基本结构是三大部分，\frontmatter、\mainmatter和\backmatter。

```
\frontmatter
扉页
版权页
\chapter{Preface} %无section，章无编号，自动添加到目录，有默认目录样式
前言内容
\clearpage
符号表 %自动添加章名，有时放目录后，但是必须在mainmatter前，不自动添加到目录
```

```

\tableofcontents %自动生成目录

\mainmatter %主体部分，章节都有编号
\chapter{Introduction}
\section{}
\subsection{}
\subsubsection{}
内容
\section*{} %加星号表示该节不编号

\chapter{多个章节}
...

\appendix %附录一定在mainmatter里，章的编号是ABCD编号，公式也按ABCD起始编号
\chapter{Math} %附录部分可以有小节，只有mainmatter部分允许节存在
\section{场论}
\chapter{Quantum mechanics}

\backmatter %参考文献、索引等，参考文献没有节，应该放在backmatter里
\chapter{致谢} %无section，默认是不编号的章，自动添加到目录
\clearpage

```

从分类角度，上面的结果是因为 tex 将章分为了三类

- 由`\chapter`命令生成，放在`\mainmatter`里，章下允许有节、小节、小小节，章节都有编号。这类自动生成目录，目录样式属于有编号的章。
- 由`\chapter`命令生成，放在`\frontmatter`和`\backmatter`里，章下不允许存在节。这类自动生成目录，目录样式属于无编号的章。
- 凡是自动生成的，如目录、参考文献、符号表、术语表、索引，无论它们放在什么部分，都不会自动添加到目录里，可能的命令是`\chapter*`。这些在 book 类是没有节的章，在 article 类是节。它们的目录样式属于无编号的章。它们应该放在`\frontmatter`或`\backmatter`里。只要它们放在`\frontmatter`或`\backmatter`里，就很容易手动添加到目录。然而，它们也被允许放在`\mainmatter`里。例如天津大学要求参考文献在附录之前，这个顺序是放在了允许节的章之间，手动添加目录时比较麻烦，后面会专门讲述。

`\frontmatter`和`\backmatter`里都不能添加节命令`\section{...}`，然而它们之间是有些细微差别的。可以在`\frontmatter`之后立刻写内容，例如做书的扉页和版权说明页。但是不能在无章命令`\chapter{...}`的情况下直接在`\backmatter`之后写文字内容。因为这些直接的内容会视为前面章或节里的内容，即使添加`\clearpage`分页，内容显示上可能没有问题，但是对例如目录页码这样的自动索引和链接的扰乱仍然存在。

可以没有`\frontmatter`，但是不能省略`\backmatter`，否则格式排版会有显示不正确的地方。如果没有`\backmatter`，在内容最后使用`\clearpage`语句也可以解决这个问题。article 类如果设置页眉页脚时也经常会有编译问题，内容最后也需要添加`\clearpage`语句。article 类默认是没有页眉页脚的，所以单纯的 article 类是不需要最后`\clearpage`的。article 类添加目录时，内容最后也要`\clearpage`一下。

2.10.2 页眉页脚的样式

book 类内置了默认的页眉页脚。只有理解了默认的设置，才能够比较熟练地修改为自己想要的结果。需要先了解一下内置的页眉页脚样式。内置样式有四个

empty	无页眉，无页脚	用于扉页和版权说明等
plain	无页眉，页脚居中显示页码	章起始页默认
headings	无页脚，页眉分奇偶，位置不同设定不同	默认页眉页脚
myheadings	可能已废弃	

默认页眉页脚样式

查看默认设置需要是全英文文件，因为如上所示，对中文无法进行大小写转换。所以出现编译错误、无法正确显示中文的\chaptertitle。 \frontmatter和\backmatter也是 headings，只是没有\chaptertitle和节相关内容。

\frontmatter

- \frontmatter里的页码为小写罗马字母，从第一页开始编号。
- \frontmatter里，没有章编号\thechapter，这里也没有节的相关内容。这部分只存在\chaptertitle和页码\thepage。
- 章的首页永远内置为 plain 样式，无页眉，页码底部居中。
- 章第二页开始，设置为 headings 样式，此时 headings 的具体样式为，偶数页左页眉为页码，右页眉为大写章名\Alph{\chaptertitle}。奇数页左页眉空，右页眉为页码。
- 最开始的几页，如果没有 chapter，直接写内容，是相同的 headings 样式，此时不存在\chaptertitle，奇数页右页眉显示页码，偶数页左页眉显示页码。
- 目录部分首页仍然是 plain 样式，第二页开始仍然是 headings 样式。此时的 headings 样式有所改变，这时\chaptertitle是\contentsname，转换为大写是 CONTENTS。区分奇偶页，偶数页页码在左，奇数页页码在右，相对的位置奇偶页都显示\contentsname。

\mainmatter里页眉页脚的默认结果为

- 页码为阿拉伯数字，重新从 1 开始编号。
- 章首页永远内置为 plain 样式。
- 章的第二页开始页眉页脚按 headings 样式。此时 headings 样式为，偶数页左页眉是页码，右页眉依次是大写章 title、章编号、大写章名，即\Alph{\chaptertitlename}~\thechapter~~\Alph{\chaptertitle}。奇数页左页眉是节编号、节名，即\thesection.~~\Alph{\sectiontitle}，右页眉为页码。
- 附录部分的章自动重新开始编号，章编号的样式是英文大写字母 ABCD... 编号。页眉页脚样式不变。 \chaptertitlename此时变为\appendixname。

\backmatter里页眉页脚的默认结果为

- 页码为阿拉伯数字，与前面编号连续。
- \backmatter里没有章编号\thechapter，这里也没有节的相关内容。这部分只存在\chaptertitle和页码\thepage。
- 章首页永远内置为 plain 样式。
- 从章的第二页开始，偶数页左页眉是页码、右页眉是大写的章名 (\chaptertitle)。奇数页右页眉是页码，左页眉因为没有节，所以空白。

参考文献的页眉页脚样式

首页 plain，第二页开始，区分奇偶页，偶数页页码在左，奇数页页码在右，相对的位置奇偶页都显示\bibname。

跟目录逻辑相同，默认按类别分了三个样式：\frontmatter和\backmatter里自己生成的章属于同一个样式，\mainmatter里自己生成的章（包括附录）属于同一个样式，自动生成的章（目录、参考文献等）属于同一个样式。

页眉页脚样式设置语句

页眉页脚设置语句有两个，一个是重置内置样式命令\renewpagestyle{内置样式名}{样式内容}，一个是自定义样式命令\newpagestyle{自定义样式名}{样式内容}。样式内容如下：

```
\renewpagestyle{plain}[前置命令]{ %前置命令可以设置字号等
\sethead[左页眉][中页眉][右页眉] %方括号为偶数页页眉
{左页眉}{中页眉}{右页眉} %花括号奇数页页眉
\headrule %页眉横线
\setfoot[左页脚][中页脚][右页脚]{左页脚}{中页脚}{右页脚} %同上分左右
}
```

article 类为 oneside，只需要花括号即可，方括号里面的设置可以省略。如果方括号里存在其他设置，oneside 时也不起作用。

titlesec 包作者的思想是尽量不动原始语句，所以页眉页脚都不支持换行。需要换行时使用\parbox或\ vbox。

设置页眉页脚时最经常使用的命令如下：

- \headrule：页眉横线。分在哪个位置结果都一样。
- \chaptertitlename、\chaptername和\chaptertitle，还是做个表格比较清楚。

	\chaptertitlename	\chaptername	\chaptertitle
\frontmatter	Chapter	Chapter	当前章名
目录	Chapter	Chapter	\contentsname
\mainmatter	Chapter	Chapter	当前章名
附录	\appendixname	Chapter	当前章名
\backmatter	\appendixname	Chapter	当前章名
参考文献（放\backmatter里）	\appendixname	Chapter	\bibname
索引（放\backmatter里）	\appendixname	Chapter	\indexname

因为\backmatter里的页眉一般不显示\chaptertitlename，所以附录那里修改过一次后，没有再修改。这个结果说明了一件事情：默认的 headings 样式肯定使用了 if 判断语句，根据上面的三个类别原则进行了页眉页脚设置。如果修改页眉页脚样式，从代码易懂原则出发，不同位置设定不同的样式更方便。

- \thechapter：当前章的编号。
使用\zhnumber{\thechapter}可以汉化章编号。
- \thesection：当前节的编号。
- \sectiontitle：当前小节名。
- \thepage：页码。
- \Alph{英文}：将英文文字转换为大写。

- ~: 留英文空白间距, 前后内容不断行, 辅助保证代码有效性。

headrule 格式调整

默认 headrule 是 0.4pt, 如果只是修改线宽很容易, 注意下面的语句写到页眉页脚设置语句里面, 替代\headrule。

```
\setheadrule{1pt}
```

如果需要修改样式, 略微麻烦一些, \setheadrule{线宽}本质上是修改的\makeheadrule命令, 所以修改样式直接用下面的重置命令语句替代\headrule (不要再加载这个命令), 必须写到页眉页脚的设置语句里。例如改成不一样宽度的双线, 下面一条线是红色, 直接将下面的代码替代\headrule命令。

```
\renewcommand{\makeheadrule}{
\makebox[0pt][l]{\rule[-0.3\baselineskip]{\linewidth}{1pt}}
\color{red}
\rule[-0.45\baselineskip]{\linewidth}{0.4pt}}
```

更多页眉页脚的复杂设置, 可以参考手册

```
texdoc titleps
```

这个手册里的很多东西, 自己写代码估计也能做出来。

其他常见书籍页眉页脚样式

经常遇到的样式是前几页没有页眉和页码, 也没有章, 放置扉页。扉页包括题目和作者名, 以及标注第 n 版, 第一版一般不标注。有的书扉页还包含作者单位和出版社名。扉页不是封面, 封面有彩图背景, 扉页是黑白的, 且两者内容并不完全一致。后面的版权页也不会添加页眉和页码。有些书是一系列书中的一本, 还会有系列书名等内容。这时需要采用的样式基本都是 empty, 空白样式, 无页眉、页脚、页码。使用的语句为

```
\thispagestyle{empty} %设置当前页样式, 此处为empty
```

上面的语句只对当前页有效, 多页空白可以使用

```
\thispagestyle{empty}
\pagestyle{empty} %设置当前页以及后面所有页为empty
```

一旦使用\pagestyle命令, 将清空除了章首页 (plain 样式) 以外的其他默认样式。

前言 (Preface)、不同版本的前言等, 都可以采用章的形式。如果前面有符号表, 符号表的位置随意, 有的书放在目录后, 但是一定要在\frontmatter部分, 即\mainmatter之前。章的编号, 是从\mainmatter后面的 chapter 开始的。

\frontmatter部分的页码编号, 笔者认为应该从 Preface 开始编号, 扉页和版权页不编号。设置 empty 是不显示页码, 不是页码不编号。默认从 pdf 的第一页开始编号, 如果前面有 empty 样式的内容, Preface 页的编号就不是 1, 个人觉得有些怪异。 \frontmatter部分的页码默认样式是小写罗马字母, 大部分书都是大写罗马字母。

下面的语句重新设置页码编号和页码样式,

```
\pagenumbering{Roman} %默认是roman, 小写罗马数字
```



```
\setcounter{page}{1} %从下面这页开始前面的编号
```

从 Preface 开始设置页眉页脚，书的页码一般放在页眉，所以章首页采用 empty 比较合适，重置 plain 样式为 empty。章首页一律是 plain 样式，所以修改章首页的页眉页脚，必须重置 plain 样式。然后为\frontmatter部分设计页眉页脚样式，比如加粗页码、页码前后加圆点等。

```
\renewpagestyle{plain}{% %用empty样式覆盖默认的plain样式
```

```
\newpagestyle{pre}{%设定pre样式，pre是自己起的样式名字
```

```
\sethead[\bfseries\thepage] [] [\chaptertitle]
```

```
{\chaptertitle}{\bfseries\thepage}
```

```
\headrule
```

```
\setfoot [] [] [] {} {} {}
```

```
}
```

```
\pagestyle{pre} %除了章首页外的页眉页脚采用pre样式
```

中文书喜欢采用突出页码的设计，例如可以在页码两边添加横线或点，`$$-\textbf{\thepage}~$-$`。

\mainmatter和\backmatter部分，修改章首页的页眉页脚也必须重置 plain 样式。

2.10.3 章节名称的样式

章节名称的样式主要是两个命令，\titleformat和\titlespacing，前者定义样式，后者定义间距。下面以章为例说明语句用法。

```
\titleformat{章/节/小节/小小节}[内置样式]{前导设置语句，一般定义字体大小、字体样式、是否居中等，对后面全部生效}{手册称为label，章名前的内容，比如第几章，\chaptertitlename的样式}{章名前的内容与章名之间的间距}{章名前的前导设置语句，可以改变字体大小，仅对章名有效，章名自动加载无需任何命令}{后续语句，可省略{多行语句必须加花括号}}
```

上下左右间距为

```
\titlespacing*{章/节/小节/小小节} {左间距}{上间距}{下间距}{右间距，可省略花括号}
```

手册 25 页给出了 Standard classes 的默认结果，这个结果间距比较大，章节名格式不符合中文书籍的一般情况。

```
\titleformat{\chapter}[display]{\normalfont\huge\bfseries}{\chaptertitlename\ \thechapter}{20pt}{\Huge}
```

```
\titleformat{\section}{\normalfont\Large\bfseries}{\thesection}{1em}{}{}
```

```
\titleformat{\subsection}{\normalfont\large\bfseries}{\thesubsection}{1em}{}{}
```

```
\titleformat{\subsubsection}{\normalfont\normalsize\bfseries}{\thesubsubsection}{1em}{}{}
```

```
\titleformat{\paragraph}[runin]{\normalfont\normalsize\bfseries}{\theparagraph}{1em}{}{}
```

```
\titleformat{\subparagraph}[runin]{\normalfont\normalsize\bfseries}{\thesubparagraph}{1em}{}{}
```

```
\titlespacing*{\chapter} {0pt}{50pt}{40pt}
```

```
\titlespacing*{\section} {0pt}{3.5ex plus 1ex minus .2ex}{2.3ex plus .2ex}
```

```
\titlespacing*{\subsection} {0pt}{3.25ex plus 1ex minus .2ex}{1.5ex plus .2ex}
```

```
\titlespacing*{\subsubsection} {0pt}{3.25ex plus 1ex minus .2ex}{1.5ex plus .2ex}
```

```
\titlespacing*{\paragraph} {0pt}{3.25ex plus 1ex minus .2ex}{1em}
\titlespacing*{\subparagraph} {\parindent}{3.25ex plus 1ex minus .2ex}{1em}
```

内置样式为

- display: 章默认, 章编号单独一行
- hang: 节默认, 左对齐, 一般中文书的样式, 但是需要设置位置居中
- ruin: 段默认, 对应`\paragraphi{title}`和`\subparagraphi{title}`.
- block: 方便使用图片工具
- leftmargin: 章名放在左边注
- rightmargin: 章名放在右边注
- drop:
- wrap: 类似于 drop
- frame: 类似于 display, 但是有边框

复杂的章节样式设计比较难掌握, 需要多练习。但是一般毕业论文要求很容易, 都是要求 hang 样式。

```
\titleformat{\chapter}[hang]{\xiaosan\heiti\filcenter}{第~\thechapter~章}{1em}{}
```

要求汉化章节编号时为

```
\titleformat{\chapter}[hang]{\xiaosan\heiti\filcenter}{第\zhnumber{\thechapter}章}{1em}{}
```

没有章节编号情况下, 间距可以设为 0em, 这里按天大摘要的要求换了一下字体字号, 目录和致谢等, 就把上面的例子对应的花括号里的内容删除就行。

```
\titleformat{\chapter}[hang]{\xiaoer\bfseries\filcenter}{}{0em}{}
```

2.10.4 目录样式

目录默认有一套目录条目的对齐规则, 使用下面的命令自动生成

```
\tableofcontents
```

`\contentsname`的样式, 本质上是`\frontmatter`里无编号的章的样式, 所以用`\titleformat{chapter}`修改。默认的`\contentsname`是 Contents, 汉化时需要重新定义。

```
\renewcommand{\contentsname}{目录}
```

自动添加的目录条和需要手动添加的目录条

自动添加到目录里的章节目录条有

- `\frontmatter`里的`\chapter`
- `\mainmatter`里的`\chapter`
- `\mainmatter`里的`\section`、`\subsection`, 目录一般三层。
- `\mainmatter`里`\appendix`后的`\chapter`
- `\backmatter`里的`\chapter`

上面默认添加到目录的条目，加星方式取消编号，自动取消目录条，例如`\chapter*{}`或`\section*{}`。

需要手动添加的目录条有

- 目录本身的目录条（一般不添加）
- 参考文献的目录条
- 索引的目录条
- 符号表和术语表的目录条
- `\mainmatter`里的`\section*{}`和`\subsection*{}`

目录样式的设定语法

目录样式的语句是`\titlecontents`，放在自动起作用的章节命令之前。需要手动添加章节目录的情况，`\titlecontents`命令放在`\addcontentsline`命令之前即可。即可以在生成参考文献、索引等命令之后，也可以在前面。

```
\titlecontents{章或节}[章节名的缩进间距]{前置代码，一直到页码都有效}{有编号章节样式，页码
样式前生效{间距}，这里可以补充字号字体样式，仅对章节名有效}{无编号章节样式}{页码样式}[后置
代码]
```

这个语句写在起作用的章节之前，对后面所有章节的目录格式都生效。因为可以同时定义无编号章节目录条样式，默认的样式，可以全书只定义一次目录条样式。一般中文书和毕业论文都需要定义多次。

可以在前置代码里设置字体样式，前置代码里的设置会影响后面的点和页码。个人认为点和页码字体大小完全一致比较好看。

目录样式设计理念为章节名起始点对齐和章节编号起始点对齐，当小节号从 1.1 变为 1.13 时，压缩的是小节号与小节名之间的间距。这两个对齐位置用两个间距来设置：

- 第一个是方括号里的间距，设置章名的缩进距离。
- 第二个是花括号里的间距，设置章编号到章名的间距。

章节目录条的缩进距离等于方括号里的间距减去花括号里的间距。

上面设定的这两个位置是绝对的。两者相等，目录条不缩进，一般是带编号的章的样式。如果花括号里的间距大于方括号里的间距，目录条将超出左边界。如果花括号里的间距太小，则会发生文字重叠。

目录样式的默认设置是

```
\titlecontents{章或节}[章节名的缩进间距]{章编号、章名和页码都字体加粗}{\contentslabel{章
节编号起始点到章节名起始点的间距}}{\hspace{-章名的缩进间距}}{\titlerule*[0.5pc]{.}\
contentspage}[后置代码]
```

%章没有点，节不加粗

%无编号章名不缩进，所以章名需要回退方括号里的章名的缩进量。

%节名不回退，节名不用修改缩进量，无论是否有编号节名都对齐

`\contentslabel{间距}`是默认的章节目录样式，章节名前只有章节的编号，没有`\chaptertitlename`。无编号章节名默认是无缩进的。但是如果修改了有编号章节目录条的章名缩进量，需要重新设置无编号章节目录条，以确保章名回退量正确，这样章名才能无缩进。一般书籍，无编号节的节名都是同距离缩进的，因此无编号节的样式没有上面的`\hspace`语句。

关于间距设置使用的单位，还有一些需要说明的细节。如果间距是em这样的单位，方括号里的间距1em，对应的是此处正文的字号对应的1em。而花括号里的间距，如果前面修改了字体大小或者加粗，1em就变为修改字体大小（加粗）后对应的1em。所以只有不改变字号和字体情况下，使用em为单位计算起来才比较简单。如果修改字号字体，使用 pt 为单位并精确计算间距比较准确，但是计算量比较多。这个问题可以用下面的方式解决

```
\titlecontents{chapter}[6em]{}{\contentslabel[\bfseries\Large第\zhnumber{\thecontentslabel}章]{6em}\bfseries\Large}{\hspace{-6em}\bfseries\Large}{\titlerule*[0.3pc]{\small.}\contentspage[(\thecontentspage)]}[]
```

这里使用了字体字号设置语句作用范围是局域的特点。

- 方括号里的1em间距是此处正文字体字号对应的1em。
- 章编号的字号字体设置写到\contentslabel[重新设定章名前的目录样式]{间距}设置命令的方括号里，仅对章编号起作用（作用域是方括号内），不对后面的花括号里的间距起作用。这保证了两个间距的1em是同一个长度，都是对应正文的字体字号的结果。
- 章名的字号字体设置，写到花括号间距之后，作用范围是章名部分。
- 无编号的章名，章名不缩进，先设置回退的间距，1em仍然是按正文为基础的，与前面的间距同基准。
- 无编号的章名的字号字体设置，写到回退距离语句之后，不影响设置回退时1em单位的基准。

这样，减少了许多计算量，但是小节号的位置，仍然需要根据设计理念手动计算获得目录条的缩进量。

```
\titlecontents{section}[6em]{}{\bfseries\Large\contentslabel{2em}}{\bfseries\Large}{\titlerule*[0.3pc]{\small.}\contentspage[(\thecontentspage)]}[]
```

此处，花括号里的2em是加粗加大后为基准的结果。

以上这两个例子，仅仅是为了说明间距单位的基准是怎么变化的，不是一个美观的设计。

一般 subsection 的目录，目录条的缩进量等于章名的缩进量，title 的缩进量在此基础上再添加，两个间距的差值为章名缩进量，这个设计比较美观整齐，此处的计算量比较简单。

很少有书的目录到 subsubsection 的。所以从对齐的角度，只有 section 目录条的缩进量需要计算和优化。

目录编号汉化

如果需要对章编号汉化，采用下面的形式

```
\titlecontents{chapter}[缩进量]{}{\contentslabel[修改的汉化的章节编号部分的样式]{间隔}}{\hspace{-缩进量}}{\titlerule*[0.5pc]{.}\contentspage}
```

比如说修改为第一章，如下所示，同时给出节的样式方便看对齐方式。

```
\titlecontents{chapter}[6em]{}{\contentslabel[\bfseries\Large第\zhnumber{\thecontentslabel}章]{6em}\bfseries\Large}{\hspace{-6em}}{\titlerule*[0.3pc]{.}\contentspage}
\titlecontents{section}[6em]{}{\bfseries\large\contentslabel{2.0em}}{}{\titlerule*[0.3pc]{.}\contentspage}
```

章名和节名都是缩进6em，因此章节名和节名严格左对齐。节编号的缩进距离是2em（`\bfseries\large`后的文字为基准）。这个是目测的结果，此处应该精算到 pt 才最优美。

目录处的章节编号是`\contentslabel`，章节编号的计算器`\thechapter`和`\thesection`等在目录处为 0。所以不能直接调用正文里内置的各种计数器的值，包括后面的页码也是如此。

`\zhnumber{数字}`是自动汉化数字为中文数字一三四五，这个命令在 `zhnumber` 包里，使用 `pdflatex` 编译时需要加参数设置编码为 UTF8。

```
\usepackage[encoding=UTF8]{zhnumber} %不加这个参数并没有测试出来差别
```

修改页码样式

`\titlerule*[0.3pc]{.}`写在页码样式里，写在引用页码前，章节名和页码之间添加点作为连接线，方括号里的0.3pc是点的间距（点的疏密），点本身的大小可以用字体字号调节。也可以是其他符号，用点的情况是最多的。

修改页码样式的方法与修改章节样式类似，`\contentspage[修改后的页码样式]`，默认是直接引用页码。比如给页码加圆括号，可以写为`\contentspage[(\thecontentspage)]`。

手动添加目录

手动添加目录的语句为`\addcontentsline`，比如添加无编号节到目录：

```
\section*{练习} %加星不编号
\addcontentsline{toc}{section}{练习}
```

手动添加目录语句必须在`\chapter*{}`或`\section*{}`语句之后、文字内容之前，pdf 书签自动链接到题目处。凡是添加到目录的，有超链接包，生成的 pdf 都自动添加书签。样式目录放在手动添加命令之前，`section` 的一般不需要修改样式。

但是添加自动生成的章的目录条，诸如目录、参考文献、符号表、索引等的目录条，比较麻烦。笔者猜测，这些采用了`\chapter*`命令，随后就是自动生成的内容。如果`\addcontentsline`在生成语句之前，目录页码会比实际首页页码小 1，如果`\addcontentsline`在生成语句之后，则目录页码会显示这部分最后一页的页码。

添加目录、符号表、参考文献、索引等的目录条，可以使用下面的虚拟节语句，下面是以索引为例

```
\phantomsection %虚拟节
%样式语句
\titlecontents{chapter}[0em]{\bfseries}{\titlerule*[0.5pc]{.}\contentspage}
\addcontentsline{toc}{chapter}{\indexname} %页码自动生成
%输出索引反而放在后面，自动生成语句在添加目录条语句之后
\printindex
```

因为在`\frontmatter`和`\backmatter`里不允许存在节，所以在这两部分里所有自动生成的章，可以用这种虚拟节的方式添加目录，自动获得正确页码。符号表一般在`\frontmatter`里，参考文献一般在`\backmatter`，自然可以用这样的方式添加目录条。

如果参考文献在`\mainmatter`里，`\phantomsection`方式不适用，就需要额外做页码引用。比如按天大的要求，参考文献放在附录前，如果有附录，参考文献就必须放在`\mainmatter`里，因为附录在`\mainmatter`里，这种方式就不能使用了。

此时需要自建 label，利用超链接的`\pageref`命令获取正确的首页页码。

```
\renewcommand{\bibname}{参考文献\label{page-bib}} %放在最前面也可以

\titlecontents{chapter}[0em]{}{}{}{\titlerule*[0.3pc]{.}\contentspage[\pageref{page-bib}}]{}

\bibliographystyle{tju} %符合天大要求的样式文件
\bibliography{paper} %先自动生成文献列表

\addcontentsline{toc}{chapter}{\bibname} %添加目录语句放在自动生成语句后面

%需要重新定义后面的目录样式
```

注意这两种方法，自动生成语句和添加目录语句的顺序是不同的。

在生成命令之前建立新的计数器，令当前页码`\thepage`加 1，`\contentspage`显示新计算器数值，原理是一样的，估计也可以，没有测试。

自动生成目录条后还会自动生成 pdf 书签。其实添加目录自己的目录条没有意义，对目录而言，仅仅添加 pdf 书签、不添加目录条，比较实际。

仅添加 pdf 书签不添加目录条

如果添加了 `hyperref` 包，目录条里的章节，都会自动生成 pdf 书签（bookmark）。目录的目录条没有任何实际意义，如果不添加目录的目录条，只添加 pdf 书签，下面有两种方法。

1. 同时使用下面两个包

```
\usepackage{hyperref}
\usepackage{bookmark}
```

添加下面的语句，生成目录和目录的 pdf 标签，但是目录本身不包含该目录条。

```
\hypersetup{next-anchor=toc}
\tableofcontents
\bookmark[dest=\HyperDestNameFilter{toc},level=chapter]{\contentsname}
```

2024 版，上面语句做出来的 pdf 书签的点击效果与添加目录的目录条目自动获得的 pdf 书签的点击效果一样好。

2. 笔者后来还想到一种更简单的方式，不需要 bookmark 包，直接在设定目录章样式的语句中添加 pdf 书签，添加到前面比添加到后置语句的点击效果更好。

```
\titleformat{\chapter}[hang]{\pdfbookmark{目录}{\contentsname}\Large\bfseries\filcenter}{}{0em}{}[]
```

添加书签的语句的语法为

```
\pdfbookmark[层级，默认为0]{pdf 书签显示名}{锚点}
```

目录这里的锚点是`\contentsname`。

对于其他不显示目录条的章节，不需要这么麻烦，直接在`\section*`后面添加 pdf 书签即可。只有自动生成的章才会出现各种问题，因为此时不可能在建立章的语句之后立刻添加 pdf

书签语句。章的建立和内容添加是一起完成的，在自动生成语句后添加任何东西，本质上都是在自动生成的内容之后添加的。

2.10.5 扉页设计

扉页设计推荐使用前面介绍过的各种 box 完成，可以获得比较精确的位置设定。本科生论文，题目、姓名、学院、指导教师等都推荐采用无边框的表格形式，这样可以做到整齐美观。研究生论文，还需要有边框表格。

独创性声明页的签名部分推荐使用 makebox 语句，前面有例子。

仅仅是空白签名，比较简单，前面的例子就可以。考虑到现在电子签名的情况也比较多，下面是一个电子签名的情况

```
\def\techersign{\includegraphics[width=0.2\textwidth]{fig/tju-text.pdf}}
%分离到其他文件
\noindent
\makebox[0.8\textwidth][r]{\parbox[][6em][b]{9em}{论文指导教师签名：}}\makebox[
0.2\textwidth][r]{\parbox[][6em][b]{0.2\textwidth}{\flushright \techersign}}\hspace{1em}
\makebox[\textwidth][r]{年\hspace{8mm}月\hspace{8mm}日}
```

论文指导教师签名：天津大学
年 月 日

用\parbox将此处高度固定下来，加载的电子签名高了也不影响版式。电子签名采用右对齐，不超过\textwidth即可。仅仅使用\parbox难以做到完美的对齐效果，所以需要\makebox自带的对齐功能。

论文扉页也做了相同的处理，题目的\parbox可以设置上下居中对齐，其他的地方底部或顶部对齐。这样题目超过一行，会自动在原位置上对齐，不会只向下走。采用 box 设计方案，后面的内容位置不会变化，都是固定位置。

2.10.6 其他常用书籍排版

分章目录

有些人写的合集每章都会有目录，可以使用下面几个语句：

```
\startcontents %开始目录
\printcontents{前缀}{开始层}[目录深度]{代码}
\stopcontents %停止目录，某些小节可以不显示在目录
\resumecontents %重新添加，不常用
```

一般开始层为 1，从小节开始，目录深度 1 层显示 section 目录，2 层显示 subsection。显示 subsection 时需要定义 subsection 样式。章名是中文时，会报错，可以使用 xeCJK+xelatex 方式。

下面是一个例子，目录做在章的样式内，目录在章标题和章内容之间，目录放在两条横线中间。注意最后方括号里的多行语句需要花括号括起来。在章的样式外定义目录格式即可，定

义全局有效，写到前面后面都行。

```
\titleformat{\chapter}[hang]{\huge\bf}{\chaptertitlename--\thechapter}{20pt}{\normalsize\rm\vspace{2pt}\titlerule\vspace{8pt}}
\startcontents
\printcontents{}{1}{2}{\vspace{6pt}\titlerule}
```

可以同时使用\tableofcontents和章内目录，目录样式全书可以是统一的，比如默认设置就比较适合。下面的例子是为了测试不一样的章内目录效果设置的。

```
\titlecontents{chapter}[2.5em]{\vspace{1em}}{\contentslabel{2.5em}}{\titlerule*[0.5pc]{.}\contentspage[(\thecontentspage)]}[]
\titlecontents{section}[2.5em]{\contentslabel{2.5em}}{\titlerule*[0.5pc]{.}\contentspage[(\thecontentspage)]}[]
\titlecontents{subsection}[5.7em]{\contentslabel{3em}}{\titlerule*[0.5pc]{.}\contentspage[(\thecontentspage)]}[]
```

上面这个例子中，使用\contentspage[(\thecontentspage)]给页码加圆括号，不能直接加圆括号。 \titlerule*[0.5pc]{.}定义小节标题到页码之间的圆点样式。 \stopcontents语句加在某个小节前，这一章的后面小节不会在目录中显示，不会影响到其他章的小节目录。

如果希望前面的总目录和后面的章内目录样式不一样，那么需要设置\printcontents的参数为 1，

```
\printcontents{1}{1}{2}
```

然后定义 lsection 和 lsubsection 的样式

```
\titlecontents*{lsection}...
\titlecontents*{lsubsection}...
```

章内目录仍然是默认标题超链接的，这点很方便。

也可以放在章样式外，但是这样会显示在章标题之前，twoside 时会额外增加两页，并且需要特别定义页眉页脚样式。一般带章目录的书籍并不采用这样的样式。

紧凑目录

有些书的小节目录不断行，只需要加星， \titlecontents*，此时目录样式一般会设置成下面的样式，

```
\titlecontents{chapter}[2.5em]{\vspace{1em}}{\contentslabel{2.5em}}{\titlerule*[0.5pc]{.}\contentspage[(\thecontentspage)]}[]
\titlecontents*{section}[2.5em]{\thecontentslabel\hspace{0.5em}}{\ \contentspage[(\thecontentspage),]}{\hspace{3em}}
```

上面的例子设置小节号前无缩进，小节号与小节标题间距 0.5em，小节标题与小节页码空一格，小节页码加圆括号并跟随逗号，最后跟随 3em 的空白，这个空白包含了小节号占据的空间，所以逗号与小节号之间的空白比 3em 小。

如果希望再加一层，包含小小节，不能完全照搬上面的例子，需要略做修改，小小节号所占空间用小小节号前横向空白定义，前面缩进仍然是 0em，而后面的空白不需要再定义横向空白 3em，否则会重复定义。

```
\titlecontents*{subsection}[0em]{}{\hspace{3em}\thecontentslabel\hspace{0.5em}}{}{\ \
contentspage[(\thecontentspage),]}
```

例子是用逗号分隔，也可以用章节号 § 前引，或者点号 • 前引等等，这样最后不会有一个逗号显得比较突兀。上面例子的逗号可能改成点比较好看。

双栏目录

双栏目录略微有些复杂，同时它也能够完成目录这页在目录中的显示和正确添加 pdf 书签。`\contentsname`，即汉语目录二字，英文 Contents，其实是`\frontmatter`里的章名，所以目录二字的格式（双栏时一般需要在页面居中位置）需要用`\titleformat`调整。同时，使用`\tableofcontents`命令时，会自动加载`\contentsname`，若以如果把`\tableofcontents`放到多栏环境 multicols 里，就需要把自动加载的`\contentsname`去除，并且上移将留下的空白也去除。假设已经在前面设置好了目录的目录样式和章样式，章的`\titlespacing*`按默认。需要以下语句实现双栏目录。

```
\chapter{目录} %用章名添加汉语目录二字
\titlespacing*{\chapter}{0pt}{0pt}{-40pt minus 1.0\baselineskip}{}
\renewcommand{\contentsname}{}

%设置mainmatter部分的章节目录样式的语句可以放在这里，也可以放在mainmatter语句之后。

\begin{multicols}{2}
\tableofcontents
\end{multicols}
```

在`\titlespacing`语句里，在单栏居中位置正确显示目录章名后，首先设置章名的上间距为 0pt，下间距为`-40pt minus 1.0\baselineskip`，其中`-40pt`是默认章间距样式中章名下面的间距，`\baselineskip`是章名文字占用的竖直长度。两者加起来是章名整体所占用的竖直长度，负号表示向上移动。接着将`\contentsname`设置为空，不显示，最后的结果就是从目录项开始双栏内容，如果各层目录字号相同，双栏显示结果左右整齐划一。`-40pt`是默认章名下面的间距，如果在前面设置了`\titlespacing`，就需要修正为修改后的结果，或者在前面设置时增加一个新长度变量以实现自动调整。

单栏时不能使用`\chapter{目录}`语句，因为`\tableofcontents`内置自动加载`\chapter{目录}`，这样相当于起了两章。因为 multicols 环境内可以一页内另起一章，所以上面的设置才是有效的，而且自动加载 pdf 书签。

2.10.7 中文书的常用公式图表编号样式

中文书籍的图表的 caption 一般都需要汉化，为了方便修改默认样式，可以加载 caption 包。

```
\usepackage{caption}
```

设置语句为：

```
\captionsetup[figure]{name=图,labelsep=space,font=small,labelfont=small}
\renewcommand{\thefigure}{\arabic{chapter}-\arabic{figure}}

\captionsetup[table]{name=表,labelsep=space,font=small,labelfont=small}
```

```
\renewcommand{\thetable}{\thechapter-\arabic{table}}

\renewcommand{\theequation}{\thechapter-\arabic{equation}}
```

上面图表的编号设置中，`\arabic{chapter}`在正文中的效果同`\thechapter`，但是如果有附录，附录默认按 ABCD 编号，就必须是`\thechapter`，这样到了附录，自动是 (A-1)

如果觉得空一个英文空格太小，可以在导言（不能在 document 环境之后）添加语句

```
\DeclareCaptionLabelSeparator{doublespace}{\ \ }
```

然后将参数修改为 `labelsep=doublespace`。

同理，设置字体大小为五号字

```
\DeclareCaptionFont{wuhao}{\fontsize{10.5pt}{12pt}\selectfont}
\captionsetup[figure]{name=图,labelsep=doublespace,font=wuhao,labelfont=wuhao}
\captionsetup[table]{name=表,labelsep=doublespace,font=wuhao,labelfont=wuhao}
```

上面的例子中的字体设置都可以用统一定义新字号完成，但是涉及代码完整性，书中大部分例子都是展开的形式。

2.10.8 天津大学论文的格式要求

本科生和研究生要求略有不同，研究生略微复杂一点，下面先按研究生的格式介绍。

格式需要的包

```
\usepackage{xCJK}
\setCJKmainfont[AutoFakeBold,ItalicFont=AR PL KaitiM GB]{SimSun} %宋体可加粗
\setmainfont{Times New Roman}
\setCJKfamilyfont{heiti}{SimHei} %黑体不可加粗
\newcommand{\heiti}{\CJKfamily{heiti}}
\setCJKfamilyfont{cuhei}{SimHei}[AutoFakeBold] %黑体可加粗
\newcommand{\cuhei}{\CJKfamily{cuhei}}

\usepackage{anyfontsize}
\usepackage{zhnumber}
```

这里设置了两个黑体，可以不可以加粗，一个可以加粗。本科生毕业论文扉页的题目是黑体加粗。而章题目是黑体，这里数字加粗比较好看，所以采用上面定义的不可加粗黑体，就可以做到只加粗数字。

```
\usepackage{titlesec} %无需clearpage
\usepackage{titles,titlotoc}

\usepackage[indentfirst]
\usepackage{setspace}
```

```
% 图表
\usepackage{caption} %下面的定义必须写在外面
\DeclareCaptionLabelSeparator{doublespace}{\ \ }
```



```
\DeclareCaptionFont{wuhao}{\fontsize{10.5pt}{12pt}\selectfont}
```

天津大学论文字号要求

反复使用的字号，最好定义一下，新要求中页码与页眉字号是不相同的。

```
\newcommand{\xiaowu}{\fontsize{9pt}{11pt}\selectfont} %页码
\newcommand{\wuhao}{\fontsize{10.5pt}{12pt}\selectfont} %页眉
\newcommand{\xiaosi}{\fontsize{12pt}{20pt}\selectfont} %正文
\newcommand{\sihao}{\fontsize{14pt}{16pt}\selectfont} %节、小节标题
\newcommand{\xiaosan}{\fontsize{15pt}{17pt}\selectfont} %章标题
\newcommand{\sanhao}{\fontsize{16pt}{18pt}\selectfont}
\newcommand{\xiaoer}{\fontsize{18pt}{20pt}\selectfont}
\newcommand{\erhao}{\fontsize{22pt}{24pt}\selectfont} %摘要、题目
```

除了小四的`\baselineskip`定义为 20pt，其余都是接近无行距留白的设置。这样做有好处也有坏处，看个人设计观点。也可以扩大各个字号的`\baselineskip`。

天津大学研究生论文标题的字号设置为：

- 扉页题目要求二号黑体，摘要要求二号加粗宋体

```
\titleformat{\chapter}[hang]{\erhao\bf\filcenter}{0em}{}
\titlespacing*{\chapter}{0pt}{12pt}{34pt}
```

摘要部分可以略微调整一下间距

- 目录要求与正文设置相同，但是没有章号部分。

```
\renewcommand{\contentsname}{目录}
\titleformat{\chapter}[hang]{\xiaosan\heiti\filcenter}{0em}{}{}
```

- 章名要求小三黑体，这里设置为数字加粗

```
\titleformat{\chapter}[hang]{\xiaosan\bf\heiti\filcenter}{第~\thechapter~章}{1em}{}{}
```

- 节和小节要求四号黑体，这里设置为数字不加粗

```
\titleformat{\section}{\sihao\heiti}{\thesection}{0.5em}{}{ }
\titleformat{\subsection}{\sihao\heiti}{\thesubsection}{0.5em}{}{ }
```

- 小小节要求小四黑体，相当于`\normalsize`

```
\titleformat{\subsubsection}{\normalsize\heiti}{\thesubsubsection}{0.5em}{}{ }
```

- 如果有附录，需要单独定义章样式

```
\titleformat{\chapter}[hang]{\xiaosan\bf\heiti\filcenter}{\chaptertitlename~\thechapter}{1em}{}{ }
```

这里将章号和章名之间的间距设置为 1em 了，论文模板显示的效果小于 1em，感觉不好看。节号和节名等的间距没有说明，这里设置为 0.5em。 `\backmatter` 里的章默认没有 `\chaptertitlename` 和 `\thechapter`，所以不需要再额外定义。

间距没有给固定标准，给的是一个范围，下面的语句设置正文处的间距

```
\titlespacing*{\chapter}{0pt}{12pt}{24pt}
\titlespacing*{\section}{0pt}{18pt}{10pt}
\titlespacing*{\subsection}{0pt}{18pt}{8pt}
\titlespacing*{\subsubsection}{0pt}{12pt}{4pt}
```

基础框架

采用页内顶部对齐

```
\raggedbottom %写到document之前
\begin{document}
```

\frontmatter前可以定义字号和图表样式，汉化

```
%字号语句在前面，省略
% 设置图、表、公式编号格式
\captionsetup[figure]{name=图,labelsep=doublespace,font=wuhao, labelfont=wuhao}
\renewcommand{\thefigure}{\arabic{chapter}-\arabic{figure}}

\captionsetup[table]{name=表, labelsep=doublespace,font=wuhao, labelfont=wuhao}
\renewcommand{\thetable}{\arabic{chapter}-\arabic{table}}

\renewcommand{\theequation}{\arabic{chapter}-\arabic{equation}}

%汉化
\renewcommand{\contentsname}{目录}
\renewcommand{\bibname}{参考文献\label{page-bib}} %自动获取参考文献首页页码
\renewcommand{\appendixname}{附录}
```

定义主体、附录和最后的页眉页脚样式

```
% 设置主体页眉页脚样式
\newpagestyle{main}[\wuhao]{
\sethead[] [天津大学硕士学位论文] []
}{第~\thechapter~章\hspace{1em}\chaptertitle}{ %花括号奇数页
\headrule
\setfoot[] [\xiaowu\thepage] [] {}{\xiaowu\thepage}{}
}
%设置附录页眉页脚样式
\newpagestyle{fulu}[\wuhao]{
\sethead[] [天津大学硕士学位论文] []
}{\chaptertitlename~\thechapter\hspace{1em}\chaptertitle}{ %花括号奇数页
\headrule
\setfoot[] [\xiaowu\thepage] [] {}{\xiaowu\thepage}{}
}
%设置参考文献、发表论文、致谢等页眉页脚样式
\newpagestyle{back}[\wuhao]{
\sethead[] [天津大学硕士学位论文] []
}{\chaptertitle}{ %花括号奇数页
\headrule
```

```
\setfoot[][\xiaowu\thepage]{}{\xiaowu\thepage}{}
```

\frontmatter部分，扉页和独创性声明为 empty 样式。摘要和目录为 plain 样式，从摘要页开始编号，页码为大写罗马数字。

```
\frontmatter
\setboolean{@twoside}{false} %前两页不区分奇偶页
\thispagestyle{empty}
设计扉页
\clearpage
\thispagestyle{empty}
独创性声明
\clearpage
\setboolean{@twoside}{true}

%摘要部分
%摘要格式，无章号
\titleformat{\chapter}[hang]{\erhao\bf\filcenter}{}{0em}{}{}
\titlespacing*{\chapter}{0pt}{12pt}{34pt}
%页码
\pagenumbering{Roman} %默认是roman，小写罗马数字
\setcounter{page}{1}
%页眉页脚设置，首页自动是plain
\pagestyle{plain}
\xiaosi %保险一些，设置20pt间距的小四字号

\chapter*{摘要} %不添加到目录
中文摘要

\noindent{\sihao\bf 关键词: }
\chapter*{ABSTRACT} %不添加到目录
English

\noindent{\sihao\bf KEYWORDS: }

%目录部分
%目录章名样式
\titleformat{\chapter}[hang]{\xiaosan\heiti\filcenter}{}{0em}{}{}

%设置目录样式
\titlecontents{\chapter}[4em]{}{\contentslabel[第~\thecontentslabel~章]{4em}}{\hspace{-4em}}{\titlerule*[0.3pc]{.}\contentspage}[]
\titlecontents{section}[4em]{}{\contentslabel{2.0em}}{}{\titlerule*[0.3pc]{.}\contentspage}
\titlecontents{subsection}[7em]{}{\contentslabel{3em}}{}{\titlerule*[0.3pc]{.}\contentspage}

\tableofcontents %生成目录
```

\mainmatter部分，开始正文

```
\mainmatter
%设置章节样式，这里重复copy一下
\titleformat{\chapter}[hang]{\xiaosan\bf\heiti\filcenter}{第~\thechapter~章}{1em}{}
\titleformat{\section}{\sihao\heiti}{\thesection}{0.5em}{}
\titleformat{\subsection}{\sihao\heiti}{\thesubsection}{0.5em}{}
\titleformat{\subsubsection}{\normalsize\heiti}{\thesubsubsection}{0.5em}{}

\titlespacing*{\chapter}{0pt}{12pt}{24pt}
\titlespacing*{\section}{0pt}{18pt}{10pt}
\titlespacing*{\subsection}{0pt}{18pt}{8pt}
\titlespacing*{\subsubsection}{0pt}{12pt}{4pt}

%重置plain设置必须写在mainmatter之后
\renewpagestyle{plain}[\wuhao]{
\sethead[] [天津大学硕士学位论文] []
}{第~\thechapter~章\hspace{1em}\chaptertitle}{} %花括号奇数页
\headrule
\setfoot[] [\xiaowu\thepage] []{}{\xiaowu\thepage}{}
}
\pagestyle{main} %使用先前的定义

\chapter{绪论}
开始主体章节内容
```

天大要求参考文献在附录之前，额外设置比较多。如果没有附录，可以把参考文献放在\backmatter之后。参考文献是默认无编号的章。所以此处的\chaptertitle是\bibname。

```
%参考文献
%重置plain样式，
\renewpagestyle{plain}[\wuhao]{
\sethead[] [天津大学硕士学位论文] []
}{\chaptertitle}{} %花括号奇数页
\headrule
\setfoot[] [\xiaowu\thepage] []{}{\xiaowu\thepage}{}
}
\pagestyle{back} %使用backmatter部分的页眉页脚样式

\bibliographystyle{tju} %天大的参考文献样式添加到合适位置
\bibliography{k} %bib文件名

%参考文献目录样式
\titlecontents{chapter}[0em]{}{}{}{\titlerule*[0.3pc]{.}\contentspage[\thecontentspage
]{\pageref{page-bib}}}]

%参考文献需要手动添加到目录，且参考文献的目录样式按无编号章不缩进
\addcontentsline{toc}{chapter}{\bibname}
```

附录必须在\backmatter之前，附录是有编号的章，此处\chaptertitlename是\appendixname.

```
%附录
\appendix
%修改附录的目录样式，同时定义无编号章不缩进
\titlecontents{chapter}[4em]{}{\contentslabel[\appendixname~\thecontentslabel]{4em}}{\hspace{-4em}}{\titlerule*[0.3pc]{.}\contentspage[\thecontentspage]}
%修改附录的章样式，这里章编号也设置了加粗
\titleformat{chapter}[hang]{\xiaosan\bf\heiti\filcenter}{\chaptertitlename~\thechapter}{1em}{}

%重置章首页的plain样式
\renewpagestyle{plain}[\wuhao]{
\sethead[] [天津大学硕士学位论文] []
{}{\chaptertitlename~\thechapter\hspace{1em}\chaptertitle}{}%花括号奇数页
\headrule
\setfoot[] [\xiaowu\thepage] [] {\xiaowu\thepage}{}
}
\pagestyle{fulu} %其余页加载前面设置的附录页眉页脚样式

\chapter{Math} %附录一般是数学公式等
附录内容
```

\backmatter里要求页眉为\chaptertitle。没有附录，参考文献放在这里，设置会少一些。

```
\backmatter %重置plain设置必须写在backmatter之后
%这里是无编号章，附录的目录样式里已经设置为无缩进
%修改页眉页脚样式
\renewpagestyle{plain}[\wuhao]{
\sethead[] [\chaptertitle] []
{}{\chaptertitle}{} %花括号奇数页
\headrule
\setfoot[] [\xiaowu\thepage] [] {\xiaowu\thepage}{}
}
\pagestyle{back} %加载前面的页眉页脚定义

%没有附录，可以在这里加载参考文献
%参考文献的目录样式和后面致谢的不同，需要设定两次

\chapter{发表论文和参与科研情况的说明}
发表论文等情况

\chapter{致谢}
致谢
\clearpage
```

可以将设置语句用\def\mycommand{}定义，然后需要是调用。

本科生论文的一些说明

本科生论文的一些要求与硕士不同，本科生是有模板的。

- 本科生是 oneside，奇偶页的页眉页脚相同，更简单一些。
- 本科生论文目录给的模板样式，假设的是单一附录不编号，按硕士的要求应该编号。
- 本科生论文的章样式，数字要汉化。
- 因为摘要、目录、附录和致谢的章名样式是两个汉字间要加一个汉字的空白，因为目录里的附录和致谢也有中间的空白，所以仍然可以直接调用`\chaptertitle`。在`\chapter`命令那里加中间的空白。

```
\chapter{致\hspace{1em}谢}
```

个人感觉，目录这里没有空格比较好。如果附录没有编号，附录在参考文献后比较好看。但是如果附录有编号，附录在参考文献前比较好看。

最后，默认`\titlerule`和页码之间是有间距的，这样 1 位页码和 2 位、3 位页码可以保证点线右端是对齐的。天大本科论文目录样式没有留白，用下面的语句可以修正到天大模板显示的结果：

```
\contentsmargin{0pt}
```

2.11 页眉页脚包 fancyhdr

fancyhdr是一个比较简单的页眉页脚包，据说国外杂志社模板用的就是这个包。相对 titlesec 来说，它定义了奇数页和偶数页的页眉页脚语句，并且可以换行。titlesec 的页眉页脚不能换行，多行需要 parbox 这样的语句。

```
\usepackage{fancyhdr}
```

加载包后，下面是一个简写命令的例子

```
\lhead{左页眉} %强制换行左对齐
\chead{中页眉\\可以换行} %强制换行居中对齐
\rhead{右页眉} %强制换行右对齐
\lfoot{左页脚}
\cfoot{中页脚}
\rfoot{右页脚}
\pagestyle{fancy}
```

奇偶页不同仍然可以用方括号和花括号分别定义

```
\lhead[左页眉]{左页眉}
```

推荐自定义形式，比如给样式起名 main，

```
\fancypagestyle{main}{
\lhead{学校\\天津大学}
\chead{天津大学}
\rhead{\thepage}
\lfoot{}
\cfoot{}}
```

```
\rfoot{}
}
\thispagestyle{main} %当前页格式
\pagestyle{main} %其他页格式
```

article 的首页和 book 的章首页，仍然是默认 plain 样式，对于书来说，如果需要调整章首页的样式，则需要修改 plain 样式。非章首页时，仅此一页，可以直接使用 `\thispagestyle` 命令。

fancyhdr 包的内容很丰富，支持换行，fancyhdr 的效果 titlesec 也都可以做出来，只是代码不够简洁罢了，titlesec 的页眉页脚内容如果换行，需要自己做 box。

2.12 脚注 footnote

脚注可以说是相当复杂的一个东西，因为有太多特例需要额外语句才能完成。脚注默认是章内自动连续编号的，加载 hyperref 包后脚注自动超链接。最简单的语句为

```
内容\footnote{脚注内容}。
```

在小节标题使用脚注需要添加语句 `\protect`,

```
\section{小节内加脚注\protect\footnote{小节内脚注}}
```

希望脚注按页编号，而不是章内连续编号，可以使用下面两个包，一个是 footnpag，另一个是 footmisc，后者功能更多一些。

```
\usepackage{footnpag}
\usepackage[perpage]{footmisc}
```

footnpag 包的功能比较单一，footmisc 包功能略多一些。我喜欢整齐一点的效果

```
\usepackage[hang,perpage]{footmisc} %悬挂缩进，编号左边与文字左边框对齐
\setlength\footnotemargin{1em} %设置编号和脚注内容之间的间距
```

这样的脚注无法重复使用。那么希望重复使用脚注，就必须手动添加编号。

文章第二个脚注 `\footnote` {利用现有的包编号重启为1}

```
\addtocounter{footnote}{1} %脚注计数器加1
\newcounter{first} %新计数器
\setcounter{first}{\thefootnote} %设置当前脚注编号到新计数器
```

```
第三个脚注\footnotemark[\thefootnote]
\footnotetext[\thefootnote]{脚注内容}
```

第四个脚注 `\footnote` {编号重启为3}

重复使用全文第三个脚注，当前页第二个脚注 `\footnotemark[\value{first}]` %读取新计数器的值

也可以这么复用 `\footnotemark[\thefirst]` %thefirst是新计数器当前的值

`\footnotemark` 和 `\footnotetext` 的问题是打破了脚注的自动编号。超链接对这两个命令产生的脚注无效，所以无法使用超链接来实现自动编号。一个略微复杂一点的例子，如果有一个脚注需要重复，可以新定义一个计数器来实现自动编号。一页有两个脚注需要重复，则需要新定

义两个计数器。

在表格里加脚注的例子，放到表格那一小节了，参见第50页。

2.13 信 letter 类

现在虽然都 email 了，但是一般 cover letter 还是以信的格式来书写，所以简单介绍一下书信的格式。

```
\documentclass{letter} %书信专门的类

%结束问候语等移动左侧，默认居中
\longindentation=0pt

%写在document之前的部分是自己的，经常用不用改
\signature{自己的名字}
\address{自己的地址\\Tianjin University} %日期是自动的

\begin{document}
\begin{letter}{对方地址}
\opening{Dear Mr. Smith,} %开头敬称
内容主体
\closing{Best regards,} %结束时的问候语
\end{letter}
\end{document}
```

信的日期是编译日期，自动加载。

2.14 背景水印

2.14.1 background

使用 background 包可以很方便添加水印

```
\usepackage{background}
```

默认的选项是缩放 scale=10，位置页面中间，透明度 opacity=0.5，angle=60 度。这些可以在包选项里修改，也可以在语句内修改。如果不同页添加不同水印，语句内修改比较合适。

```
\backgroundsetup{contents={\includegraphics[width=1cm]{tju.pdf}},angle=0}
```

设置语句写到\begin{document}之前，是设置所有页的水印，否则是设置当前页及以后页的水印。

background 包虽然可以非常方便的设置水印，将水印移到任意位置，对奇偶页设置不同的水印等等，但是如果做官方信纸这样的需要多重图像的，没有测试成功，此时可以使用 eso-pic 包。

2.14.2 eso-pic

eso-pic 包设置水印的语句相对麻烦，但是它可以设置多个图，下面的例子是做天津大学信笺用的。

```
\documentclass{article}
```



```

\usepackage[left=2.5cm,right=2.5cm,top=4.7cm,bottom=2.5cm]{geometry}
\usepackage{fancyhdr}

\usepackage{graphicx}
\usepackage{eso-pic}

%put从左下角算坐标，默认cm，可以put多张图
\newcommand\BackgroundPic{
\put(2.3, 24.5){
\parbox[t]{\textwidth}{
\includegraphics[keepaspectratio,width=7cm]{tju-5.pdf}%
}}
\put(14.5, -0){
\parbox[t]{\textwidth}{
\includegraphics[keepaspectratio,width=7cm]{tju-7.pdf}
}
}
}

%不加*号，每页都添加logo，*号对当前页加logo
\AddToShipoutPicture{\setlength{\unitlength}{1cm}\BackgroundPic}

\begin{document}
%写到这里方便改写为中文的页眉页脚
\lfoot{92 Weijin Road, Nankai District\\Tianjin 300072, China}
\cfoot{Tel: +86 22 27404204}
\pagestyle{fancy}

内容
\clearpage
内容
\end{document}

```

2.15 文章内分栏

2.15.1 多栏 multicol 包

使用 multicol 包。

```
\usepackage{multicol}
```

分栏环境是 multicol，注意末尾有 s。

```

\setlength\columnsep{15mm} %必须放在multicol环境外
\begin{multicol}{3}
multicol 宏包能够在一页之中切换单栏和多栏，也能处理跨页的分栏，
且各栏的高度分布平衡。只有multicol环境中的内容才是多栏的。

```

```
\end{multicols}
```

后面还是单栏，所以multicol包很方便。下面看看加入表格和图片的效果。

multicol 宏包能够在一页中能处理跨页的分栏，且各栏的高度分布平衡。只有 multicols 环境中的内容才是多栏的。

后面还是单栏，所以 multicols 包很方便。下面看看加入表格和图片的效果。

```
\begin{multicols}{3}
\includegraphics[width=3cm]{fig/tju.pdf}
```

```
\begin{tabular}{l l l}\toprule
a & b & c\\
\midrule
1 & 2 & 3\\
4 & 5 & 6\\
\bottomrule
\end{tabular}
```

如果不使用figure和table环境，图表是比较容易做出效果的。

```
\end{multicols}
```



a	b	c
1	2	3
4	5	6

如果不使用 figure 和 table 环

境，图表是比较容易做出效果的。

如果使用 figure 和 table 环境，需要添加 float 包

```
\usepackage{float}
```

图表这样的浮动体就可以在 multicols 环境内使用，可以添加 caption。

```
\begin{multicols}{3}
\begin{figure}[H]
\includegraphics[width=3cm]{fig/tju.pdf}
\caption{天大校徽}
\end{figure}
\begin{table}[H]
\caption{简单表格}
\begin{tabular}{l l l}\toprule
a & b & c\\
\midrule
1 & 2 & 3\\
4 & 5 & 6\\
\bottomrule
\end{tabular}
\end{table}
\end{multicols}
```

注意图表位置用[H]，这里图表没有使用居中，caption是自动居中的，所以可以看清楚样式设计。
`\end{multicols}`




表 2-5 简单表格

a	b	c
1	2	3
4	5	6

注意图表位置用 [H]，这里图表没有使用居中，caption 是自动居中的，所以可以看清楚样式设计。

图 2-7 天大校徽

多栏环境内跨多栏（单栏）放置一张宽图，可以使用figure*的环境，表也是一样加星即可。

```
\begin{figure*}
\includegraphics[]{}
\caption{}
\end{figure*}
```

2.15.2 交互两栏 paracol

paracol 包可以很方便在两栏之间互换输入，

```
\usepackage{paracol}
```

设置两栏比例

```
\columnratio{0.7}
```

设置两栏背景色，可以分别设置，下面的语句设置的是右栏的背景色

```
\backgroundcolor{c[1]}{yellow!30}
```

两栏，交互使用

```
\begin{paracol}{2}
\columncolor{blue} %左边文字颜色
一段文字放在左边
\switchcolumn
\columncolor{red} %右边文字颜色
另一段文字
\switchcolumn
回到左边栏
\switchcolumn
回到右边
\end{paracol}
```

一段文字放在左边
回到左边栏

另一段文字
回到右边

2.15.3 更复杂的多栏版面 flowfram

flowfram 包用于非常规的杂志排版和报纸排版。

```
\usepackage{flowfram}
```

其手册则是 ffuserguide, flowfram.pdf 里是这个包的源代码, 不是用户手册。

```
texdoc ffuserguide
```

flow frame, 文字是自动填充的, 仅仅看着类似多栏, 其实是一个 flow frame 填满后自动流到下一个 flowframe。

```
\newflowframe[页码表]{frame的宽度}{frame的高度}{起点x位置}{起点y位置}[label]
```

static frame, 固定的 frame, 指定位置指定内容。

```
\newstaticframe[页码表]{frame的宽度}{frame的高度}{起点x位置}{起点y位置}[label]
```

使用 static frame 时,

```
\begin{staticcontents*}{static frame label}
一般是图表
\end{staticcontents*}
```

每一个\newstaticframe按标签对应一个staticcontents*环境。设置 static frame 的页相当于特殊位置指定排版了 box, 所以都需要特殊指定相应的 flow frame。所以很多情况下图片会放在上角或下角, 而不是中间, 因为这样 flow frame 的计算量会小一点。最后就得到图文任意混排的效果, 但是版面越复杂, 宽高和位置的计算量越多。

dynamic frame 跟 static frame 在使用上差不多。

```
\newdynamicframe[页码表]{frame的宽度}{frame的高度}{起点x位置}{起点y位置}[label]
```

对应的内容是 dynamiccontents* 环境里的内容。

```
\begin{dynamiccontents*}{label}
hcontentsi
\end{dynamiccontents*}
```

还可以设置 id, 但是 label 更方便一些。

科学期刊的排版量不大, 一般固定 2、3 栏, 合并栏做图片, 无非是合并 1、2 栏、合并 2、3 栏、全部合并 3 栏。图片放底边或右上角, 减少排版量。报纸的排版量还是相当大的, 每次版面都需要特别设计, box 的数目也多。理论上都可以做到, 感兴趣的可以试着自己排一个报纸的版面出来测试能力。

第三章 数学公式

3.1 行间公式

使用下面的代码可以获得下面的结果。

行间公式环境为一对美元符号，必须配对， $f(x)=\exp(x)$ ，行间公式一般采用压缩格式，比如 $\int_a^b f(x)dx$ ，公式里的空格不会起作用，比如 $\sum_{i=1}^N f_i g_i$ 。公式插入后行间距不会立刻自动调整，直到不能放下后才会扩展行间距。如果希望采用舒展模式，可以使用 $\displaystyle \int_a^b f(x)dx$ 。这样会使得行间距变得不可接受。如果希望舒展的公式形式，同时不撑开行间距，则利用行间公式是文本的特点，使用整体缩放方式获得希望的效果，比如 $\text{\resizebox{0.7\width}{0.7\height}{\displaystyle \int_a^b f(x)dx}}$ 。该命令可以设定宽度和高度比例，也可以使用感叹号获得按比例缩放，比如 $\text{\resizebox{0.6\width}{!}{\displaystyle \int_a^b f(x)dx}}$ 。这样可以保持行间距不变、公式形式优美的结果。当然遇到 $1 \times n$ 矩阵这样的行间公式，行间距总会拉长的， \LaTeX 仅仅能做到尽量优美。

行间公式环境为一对美元符号，必须配对， $f(x) = \exp(x)$ ，行间公式一般采用压缩格式，比如 $\int_a^b f(x)dx$ ，公式里的空格不会起作用，比如 $\sum_i^N f_i g_i$ 。公式插入后行间距不会立刻自动调整，直到不能放下后才会扩展行间距。如果希望采用舒展模式，可以使用 $\int_a^b f(x)dx$ 。这样会使得行间距变得不可接受。如果希望舒展的公式形式，同时不撑开行间距，则利用行间公式是文本的特点，使用整体缩放方式获得希望的效果，比如 $\int_a^b f(x)dx$ 。该命令可以设定宽度和高度比例，也可以使用感叹号获得按比例缩放，比如 $\int_a^b f(x)dx$ 。这样可以保持行间距不变、公式形式优美的结果。当然遇到 $1 \times n$ 矩阵这样的行间公式，行间距总会拉长的， \LaTeX 仅仅能做到尽量优美。

3.2 独立公式环境

不用加载任何包就可以使用独立公式环境 `equation`。`equation` 环境是带编号的公式环境，默认格式为公式居中，公式编号右对齐。

```
\documentclass{article}
\begin{document}
\begin{equation}
f(x)=\sin x
\end{equation}
\end{document}
```

$$f(x) = \sin x \quad (3-1)$$

如果希望公式编号左对齐，可以在 `documentclass` 的选项里添加 `leqno`，结果是公式居中，公式编号左对齐（无缩进）。

```
\documentclass[leqno]{article}
```

如果希望公式左对齐（可设置缩进量），公式编号右对齐，下面的语句设置公式缩进量为无缩进

```
\documentclass[fleqn]{article} %公式左对齐，编号右对齐
\setlength{\mathindent}{0pt} %公式无缩进
```

不编号的公式环境为

```
\[
f(x)=\cos x
\]
```

$$f(x) = \cos x$$

原来一直喜欢这种简写方式，现在变得更推荐下面的加星方式，即采用`equation*`环境，取消和添加编号更方便一些。

3.3 amsmath 包的基本用法

默认情况只能满足公式的基本功能，一般都会加载 amsmath 包，ams 是美国数学学会的缩写。

```
\usepackage{amsmath} %美国数学学会数学包
```

3.3.1 自定义函数

amsmath 包预先定义了一些常用数学函数，如三角函数、反三角函数、指数函数、对数函数等。这些函数名在公式里是直体`\rm`，变量默认是`\it`。如果某函数不存在，可以使用下面的语句，大部分情况它们的显示效果是相同的。

```
\[
\text{sgn}(x)\{\rm sgn}(x)\mathrm{sgn}(x)
\]
```

$$\mathrm{sgn}(x)\mathrm{sgn}(x)\mathrm{sgn}(x)$$

结果虽然是等价的，但是这三种方式还是有细微差别。`\text{}`方式，如果修改了文本的字体，就无法保证在公式中得到的结果是直体。`\rm`方式在 tex 中是有效的，它的问题在于部分杂志社排版时可能是将 tex 公式转其他公式软件，`\rm`方式不兼容（`\text{}`一般也兼容）。因此`\mathrm{}`对于函数名来说是最准确和保险的。

上面的例子显示不出来自定义函数和`\mathrm{}`方式的区别，如果不采用圆括号时，定义好的函数名前后各有一个`\thinspace`，自定义函数真正等价的代码是`\,\mathrm{sgn}\,`。这是因为公式环境内任何空格都是无效的，所以内置函数名前后各加了一个`\thinspace`。第一个间隙是为了隔开前面的函数或变量，后一个间隙是为了隔开后面的变量或函数。定义好的函数名前面是括号时，或变量放在括号里时，相应的`\thinspace`会被压缩掉。因为括号是很明确的间隔符号，取代了`\thinspace`的功能。

可以采用两种方式自定义函数名，第一个是定义数学操作符

```
\documentclass{article}
\usepackage{amsmath}
\DeclareMathOperator{\sgn}{sgn} %自定义sgn函数，必须在document之前
\begin{document}
\[
\sgn(x) %如同内置函数一样使用自定义函数
```

```
\]
\end{document}
```

这样不利于截取部分内容做交流。自己写书或做长笔记时，如果有大量的未预先定义函数需要频繁复用，推荐采用这样的方式，数学系推荐这样的方式。

直接在公式里定义

```
\[
\sin x\operatorname{abg}x\cos x
\]
```

$$\sin x \operatorname{abg} x \cos x$$

代码直接 copy 就可以使用。`\operatorname`命令有点长，不容易记忆，可以在.vimrc 里定义宏，方便使用。

公式中的`\text{文字}`会直接显示文字，包括汉语

```
\[
\text{激光光强}\propto|\mathcal{E}|^2
\]
```

$$\text{激光光强} \propto |\mathcal{E}|^2$$

如果公式文字太多，数学公式或字符很少，可以在公式环境内使用 `text` 嵌套行间公式，如

```
\text{the best estimate of $\mu$ is $\bar{x}$}
```

$$\text{the best estimate of } \mu \text{ is } \bar{x}$$

`\bar`是短横，`\overline`是长横。变量上加横向推荐`\bar`，其余时候应该用`\overline`。

3.3.2 amsmath 包的公式环境

加载 amsmath 包后，就可以使用以下公式环境。

单行公式环境

`equation*` 环境，星号表示无编号。

```
\begin{equation*}
f(x)=\ln x
\end{equation*}
```

$$f(x) = \ln x$$

这个是与上面的简写形式等价的，但是如果采用这样的形式，比较容易恢复编号。

多行可对齐公式环境

`aligned` 环境用于一个公式编号的多行公式，`&` 用于设置对齐位置，`&&` 用于多个对齐位置的设定，会自动添加一点空白。`hspace` 用来设置横向空白。

```
\begin{equation}
g(x)=
\begin{aligned}
&2&+\sin x& \&x<0\\
&\exp(-x)&+\cos(x)& \&x\geq 0
\end{aligned}
\end{equation}
```

```
\end{equation}
```

$$g(x) = \begin{array}{ll} 2 & + \sin x \quad x < 0 \\ \exp(-x) & + \cos(x) \quad x \geq 0 \end{array} \quad (3-2)$$

aligned 环境可以设置公式编号的位置，默认是居中，用[b]改为底部。

```
\begin{equation}
\begin{aligned}[b]
f&=g(x)\backslash
&=\sin x
\end{aligned}
\end{equation}
```

$$\begin{aligned} f &= g(x) \\ &= \sin x \end{aligned} \quad (3-3)$$

关于 aligned 环境，比较这两种写法，第一种是两个 aligned 环境，可以加 box，用一个 & 符号设置为 x 前对齐。

```
\[
f(x)=
\boxed{\begin{aligned}
&ax\backslash
&bx^2
\end{aligned}}\hspace{5mm}
\begin{aligned}
0<&x<1\backslash
-1<&x<0
\end{aligned}
\]
```

$$f(x) = \boxed{\begin{array}{ll} ax & 0 < x < 1 \\ bx^2 & -1 < x < 0 \end{array}}$$

第二种是一个 aligned 环境， x 前对齐需要两个 &，第一个表示这是另起一列，第二个是对齐位置

```
\[
f(x)=
\begin{aligned}
&ax\hspace{5mm} & 0<x<1\backslash
&bx^2 & -1<x<0
\end{aligned}
\]
```

$$f(x) = \begin{array}{ll} ax & 0 < x < 1 \\ bx^2 & -1 < x < 0 \end{array}$$

split 环境使用方法接近 aligned 环境，tbtags 选项对 aligned 环境无效，仅对 split 有效。

```
\usepackage[tbtags]{amsmath} %公式编号在最后一行
\begin{equation}
\begin{split}
a&=bb\backslash
&=f(x)g(x)
\end{split}
\end{equation}
```


下面的情况使用单 & 号，每列公式都添加了一点空白分割开，每个单元公式等号处对齐。

```
\[
\begin{aligned}
x&=y & a&=b & i&=j\\
x_1&=y_1 & a_1&=b_1 & i_1&=j_1\\
x_1+1&=y_1+1 & a_1+1&=b_1+1 & i_1+1&=j_1+1
\end{aligned}
\]
```

$$\begin{array}{lll} x = y & a = b & i = j \\ x_1 = y_1 & a_1 = b_1 & i_1 = j_1 \\ x_1 + 1 = y_1 + 1 & a_1 + 1 = b_1 + 1 & i_1 + 1 = j_1 + 1 \end{array}$$

align 环境用于多行公式多个编号情况

```
\begin{align}
f(x)&=a_0+a_1x\label{eq-a}\\
g(x)&=\exp(x)\label{eq-b}
\end{align}
```

$$f(x) = a_0 + a_1x \quad (3-4)$$

$$g(x) = \exp(x) \quad (3-5)$$

这里我们顺便学习一下公式引用。align 环境里，label 要加在换行符之前，引用公式时有两种方式，用 ref 引用没有括号，`\ref{eq-a}`，得到 3-4，用 eqref 引用自动添加括号，`\eqref{eq-b}`，得到 (3-5)。

align* 环境用于多行公式无编号情况

```
\begin{align*}
f(x)&=a_0+a_1x\\
g(x)&=\exp(x)
\end{align*}
```

$$f(x) = a_0 + a_1x$$

$$g(x) = \exp(x)$$

还有一种情况，使用 align 环境，多行公式每个都有编号且对齐，但是局部有特定行不加编号。比如最常见的情况，某行是省略符，可以在这行的强制换行符 `\` 前添加语句 `\notag`，表示这行不再编号。使用 `\nonumber` 也可以，但是手册是 `\notag`。

```
\begin{align}
a&=f(x)\\
b&=g(x)\\
&\vdots\notag\\
h&=z(x)
\end{align}
```

$$a = f(x) \quad (3-6)$$

$$b = g(x) \quad (3-7)$$

$$\vdots$$

$$h = z(x) \quad (3-8)$$

还有一种情况，就是中间插入一句文字，文字下面的公式仍然需要对齐，使用 `\intertext{文字}`

```
\begin{align}
f_1&=x+y\\
f_2&=xy
\end{align}
```

```
\intertext{文字} %文字无缩进
f_3&=\sin x\cos y %仍然与上面对齐
\end{align}
```

$$f_1 = x + y \quad (3-9)$$

$$f_2 = xy \quad (3-10)$$

文字

$$f_3 = \sin x \cos y \quad (3-11)$$

多列公式对齐, align 环境是均分对齐, 效果并不好。alignat 环境是手动留白, 相对效果比较好, 但是需要手动设置列数目、手动调整间距。flalign 环境则是全宽充满式的。下面的例子可以比较一下。

- aligned 环境, 个人感觉, aligned 效果最优美, 但是 aligned 环境是不能代替 align 环境的。

```
\[
\begin{aligned}
&\sin x \quad \cos x \quad \tan x \\
&x \quad x \quad x
\end{aligned}
\]
```

$$\begin{array}{ccc} \sin x & \cos x & \tan x \\ x & x & x \end{array}$$

- align 环境, 空白均分, 很多时候自动间隔有点太大了

```
\begin{align}
&\sin x \quad \cos x \quad \tan x \\
&x \quad x \quad x
\end{align}
```

$$\begin{array}{ccc} \sin x & \cos x & \tan x \\ x & x & x \end{array} \quad (3-12)$$

$$\begin{array}{ccc} x & x & x \end{array} \quad (3-13)$$

- alignat 环境, 必须指定多少列, 需要手动设置间隔, 推荐使用 alignat 代替 align 环境。个人喜欢 \hspace{5mm}, 但是 \quad 和 \qquad 也挺好。

```
\begin{alignat}{3}
&\sin x \quad \cos x \quad \tan x \\
&x \quad x \quad x
\end{alignat}
```

$$\sin x \quad \cos x \quad \tan x \quad (3-14)$$

$$x \quad x \quad x \quad (3-15)$$

`\quad`或`\hspace`的位置，需要根据具体的公式形式确定，不见得是第一行公式。

- `flalign` 环境，均分且左右充满，并不美观，但是可以做出对比性的两栏效果。

```
\begin{flalign}
&\sin x \quad \cos x \quad \tan x \\
&x \quad x \quad x \\
\end{flalign}
```

$$\sin x \quad \cos x \quad \tan x \quad (3-16)$$

$$x \quad x \quad x \quad (3-17)$$

多行自动对齐公式环境

`gather`和`gather*`环境，不能设置对齐位置，多行带编号或完全不带编号的公式自动居中对齐

```
\begin{gather}
f \\
\sin x \cos x \tan x \\
\end{gather}
```

$$f \quad (3-18)$$

$$\sin x \cos x \tan x \quad (3-19)$$

`multline`环境，第一行左对齐，最后一行右对齐，中间的居中对齐，这个环境是单一编号的，且编号默认在最后一行。但是如果是 `leqno` 选项，编号在左侧，则在第一行。

```
\begin{multline}
f \\
\sin x \cos x \\
f \\
g \\
\end{multline}
```

$$f$$

$$\sin x \cos x$$

$$f$$

$$g \quad (3-20)$$

子公式环境

`subequations` 环境用于子公式编号情况

```
\begin{subequations}\label{sub-a}
\begin{equation}
f(x)=ax^2
\end{equation}
\begin{equation}\label{sub-b}
g(x)=\exp(x)
\end{equation}
\end{subequations}
```

$$f(x) = ax^2 \quad (3-21a)$$

$$g(x) = \exp(x) \quad (3-21b)$$

子公式分别设置 label, 引用时自动加编号, `\eqref{sub-a}`, 获得(3-21), `\eqref{sub-b}`, 获得(3-21b)。`\ref{公式label}`不加括号, 只引用公式编号。

subequations 环境里使用 align 环境更好, 能够对齐公式。

```
\begin{subequations}
\begin{align}
f(x)&=\sin x+\cos x\label{sub-aa}\\
g(x)&=\exp(\mathrm{i}x)-\exp(-\mathrm{i}x)\label{sub-bb}
\end{align}
\end{subequations}
```

$$f(x) = \sin x + \cos x \quad (3-22a)$$

$$g(x) = \exp(ix) - \exp(-ix) \quad (3-22b)$$

3.3.3 一些格式调整

多行公式分页

以上多行公式环境, 可以在 document 之前添加下面的语句, 尽量避免但是允许公式跨页

```
\allowdisplaybreaks[1] %2,3,4表示增加纵容
```

设定运行跨页情况下, 可以使用`*强制此处不跨页`。仅仅强制这两行之间不跨页, 这个多行公式本身仍然是允许跨页的, 会自动下调一行公式完成跨页。有时公式分行是因为写不下, 此时不跨页比较好。设定距离的参数写到星号后面, 例如`*[2mm]`。这个用法只在公式环境有效, 文章内无效。

公式编号复用

写书的时候, 某个重要公式多次出现, 可以通过`\tag{\ref{公式label}}`重复使用第一次的编号。即手动设置公式编号为`\ref{公式label}`获得的编号。只要是前面设置完 label 的公式编号都可以在后面多次重复使用。

```
\begin{equation}
f(x)=a_0+a_1x\tag{\ref{eq-a}}
\end{equation}
```

$$f(x) = a_0 + a_1x \quad (3-4)$$

下面的例子都将以简洁的无编号公式环境给出, 方便快速练习。

间距调整

调整公式与上下文字之间的间距

展现上下文字间距,

```
\begin{equation}
\setlength{\abovedisplayskip}{5pt}
\setlength{\belowdisplayskip}{5pt}
f(x)=\sin x
\end{equation}
```

展现上下文字间距,

展现上下文字间距,

$$f(x) = \sin x \quad (3-23)$$

展现上下文字间距,

多行公式之间的间距由`\jot`决定, 默认是多增加 3pt,, 如果是 aligned 环境, 放在 equation 里即可

```
\begin{equation}
\setlength{\jot}{10pt}
\begin{aligned}
f(x)&=\sin x\\
f(x)&=\sin x
\end{aligned}
\end{equation}
```

$$f(x) = \sin x \quad (3-24)$$

$$f(x) = \sin x$$

如果是 align 环境, 局域调整需要加花括号括起来才不影响后面

```
{
\setlength{\jot}{10pt}
\begin{align}
f(x)&=\sin x\\
f(x)&=\sin x
\end{align}
}
```

$$f(x) = \sin x \quad (3-25)$$

$$f(x) = \sin x \quad (3-26)$$

当多行公式为分式时, 默认的 3pt 太小, 不足以很好的区分每行, 这时这个语句就非常有用。但是如果多行公式, 有分式也有单行公式, 设置单一值会造成有些竖直间距太宽, 则使用`\\[距离]`手动设定更优美。

缩放

使用`\resizebox`缩放公式且不缩放编号, 必须加载 graphicx 这个包。

```
\begin{equation}
\resizebox{1.5\width}{!}{$f(x)=\sin x$}
\end{equation}
```

$$f(x) = \sin x \quad (3-27)$$

公式内容和编号一起缩放

```
\begin{equation}\huge
f(x)=\sin x
\end{equation}
```

$$f(x) = \sin x \quad (3-28)$$

缩放到任意字体大小，加载 anyfontsize 包，与文字大小设置相同。

字号调整只能放在 equation 环境中，不能用于 align 环境中，此时采用下面的方式

```
\begin{huge}
align环境里的公式
\end{huge}
```

或者放在花括号内局域生效

```
{设置字号语句
align环境里的公式
}
```

编号样式

默认 article 类的公式编号是(公式编号)，book 类的公式编号是(章编号.公式编号)。用下面的语句设置公式带小节编号

```
\numberwithin{equation}{section}
```

自定义公式编号，article 类时带小节编号，且改点为横线

```
\renewcommand{\theequation}{\thesection--\arabic{equation}}
```

为子公式修改样式，需要放在子公式环境内部

```
\renewcommand{\theequation}{\theparentequation(\alph{equation})}
```

章号后面直接连公式编号

```
\renewcommand{\theequation}{\thechapter--\arabic{equation}}
```

3.4 分式和 displaystyle

`\frac{分子}{分母}` 构成分式，

```
\[
f(x)=\frac{1}{1+x^2}
\]
```

$$f(x) = \frac{1}{1+x^2}$$

多重分式只需要合理嵌套即可

```
\[
f(x)=\frac{1}{\frac{1}{1+x^2}}
\]
```

$$f(x) = \frac{1}{\frac{1}{1+x^2}}$$

这样的公式下面的公式太小，可以使用 `displaystyle`

```
\[
f(x)=\frac{1}{\displaystyle\frac{1}{1+x^2}}
\]
```

$$f(x) = \frac{1}{\frac{1}{1+x^2}}$$

因为这种情况非常多，因此内置了简写方式 `\dfrac`，等价于 `\displaystyle\frac`

```
\[
f(x)=\frac{1}{\dfrac{1}{1+x^2}}
\]
```

$$f(x) = \frac{1}{\frac{1}{1+x^2}}$$

即使采用 `\dfrac` 也不够漂亮，手册推荐使用连续分式语句 `\cfrac`，比较如下

```
\[
\cfrac[c]{1}{\sqrt{2}+
\cfrac{1}{\sqrt{2}+
\cfrac{1}{\sqrt{2}+\dotsb
}}}
\times
\dfrac{1}{\sqrt{2}+
\dfrac{1}{\sqrt{2}+
\dfrac{1}{\sqrt{2}+\dotsb
}}}
\]
```

$$\frac{1}{\sqrt{2} + \frac{1}{\sqrt{2} + \frac{1}{\sqrt{2} + \dots}}} \times \frac{1}{\sqrt{2} + \frac{1}{\sqrt{2} + \frac{1}{\sqrt{2} + \dots}}}$$

`\cfrac` 可能更优美，但是显然也更占空间。

`\cfrac` 可以指定对齐方式，默认居中，`\cfrac[l]{ }{ }` 分子在左边，`\cfrac[r]{ }{ }` 分子在右边。对齐方式只对分子有效。

反之，如果在独立的公式环境中采用行间公式的压缩形式，可以使用 `\tfrac`

```
\[
f(x)=\tfrac{1}{2}(\sin x+\cos x)
\]
```

$$f(x) = \frac{1}{2}(\sin x + \cos x)$$

式中的求和、积分，如果在分子分母上，默认为行间公式形式，如果觉得舒展开好看，也需要加 `\displaystyle`

$$\frac{\sum_{i=0}^{i=N} x^i}{x}$$

```
\[
\frac{\displaystyle\sum_{i=0}^{i=N} x^i}{x}
\]
```

二项式为`\linline!\binom!`,

`\linline!\displaystyle!`的二项式为`\linline!\dbinom!`, 行间

```
\begin{vcode}
```

```
\[
2^k-\binom{k}{1}2^{k-1}+\binom{k}{2}2^{k-2}
\]
```

```
\end{vcode}
```

二项式的底层代码和分式`\linline!\frac!`的一样, 只是设置分式的

二项式为`\binom`, `\displaystyle`的二项式为`\dbinom`, 行间公式形式的二项式为`\tbinom`

```
\[
2^k-\binom{k}{1}2^{k-1}+\binom{k}{2}2^{k-2}
\]
```

$$2^k - \binom{k}{1}2^{k-1} + \binom{k}{2}2^{k-2}$$

分式不显示横线。

二项式的底层代码和分式`\frac`的一样, 只是设置分式的横线为`0pt`, 导致分式不显示横线。

3.5 开根号的细节

L^AT_EX 的公式排版很注意细节, 下面的例子, 第二个写法可以使得三个开根号的横线完全在同一水平线上。下面的结果需要放大后才能看出效果。

```
\[
\sqrt{x}\sqrt{y}\sqrt{z}
\]
\[
\sqrt{x}\sqrt{\smash[b]{y}}\sqrt{z}
\]
```

$$\sqrt{x}\sqrt{y}\sqrt{z}$$

$$\sqrt{x}\sqrt{y}\sqrt{z}$$

顺便练习一下开根号, 前面方括号里定义次数

```
\[
\sqrt[n]{x}
\]
```

$$\sqrt[n]{x}$$

如果只定义次数, 位置可能不好看

```
\[
\sqrt[n]{\frac{a}{x}}
\]
```

$$\sqrt[n]{\frac{a}{x}}$$

好看的样子是调整左右和上下位置


```
\[
\sqrt[\leftroot{-2}\uproot{12}n]{\frac{a}{x}}
\]
```

$$\sqrt[n]{\frac{a}{x}}$$

3.6 括号

默认了五种括号大小，除了普通的括号外，从小往大依次是 big, Big, bigg, Bigg，不需要配对

```
\[
\Bigg(\bigg(\Big(\big(\frac{1}{x}\Big)
\]
```

$$\left(\left(\left(\frac{1}{x}\right)\right)\right)$$

big Big bigg Bigg 还可以灵活地应用到其他符号，如除号或绝对值符号等

```
\[
\Bigg|\frac{x}{y}\Bigg/\frac{a}{b}\Bigg|
\]
```

$$\left|\frac{x}{y} \Bigg/ \frac{a}{b}\right|$$

使用\left(\right)可以自动改变括号大小，注意 left 和 right 必须配对使用

```
\[
\left(\frac{1}{x}\right)
\]
```

$$\left(\frac{1}{x}\right)$$

括号可以多重配对

```
\[
\left(1+\left(\frac{1}{x}\right)+f(x)\right)
\]
```

$$\left(1 + \left(\frac{1}{x}\right) + f(x)\right)$$

方括号、花括号、绝对值和范数等，都支持\left和\right配对方式，

```
\[
\left[\frac{1}{x}\right]
\left\{\frac{1}{x}\right\}
\left|\frac{1}{x}\right|
\left\|\frac{1}{x}\right\|
\]
```

$$\left[\frac{1}{x}\right] \left\{\frac{1}{x}\right\} \left|\frac{1}{x}\right| \left\|\frac{1}{x}\right\|$$

left(和 right) 中间不能换行，要换行需要使用\right.和\left.辅助配对。

```
\[
\begin{aligned}
xxx \left(\int_t yyy\right. \\
\left. zzz \dots \right)
\end{aligned}
\]
```

$$xxx \left(\int_t yyy \right. \\ \left. zzz \dots \right)$$

这种方式得到的左右括号大小并不一致，可以采用方式配对。

```
\[
\begin{aligned}
xxx \left(\int_t yyy \right. \\
\left. \phantom{\int_t} zzz \dots \right)
\end{aligned}
\]
```

$$xxx \left(\int_t yyy \right. \\ zzz \dots \left. \right)$$

从上面的例子也可以看出`\left.`对齐的地方是`\left`的位置。

自动配对括号有时也并不完美，采用指定括号大小方式更好看，例如只有求和下标这样的纵向不对称的公式结构，指定方式较美观。此时纵向不全部覆盖里面的内容，括号在感官上反而纵向更对称更优美。分式不对称情况需要加括号时，可以换种形式表达相同的公式，比如指数形式。这里就不举例了。

cases 和单边括号

最常用的单边括号是左花括号，内置了 `cases` 环境，`cases` 环境左边是已经设置好对齐的，不需要额外对齐符

```
\[
f(x)=
\begin{cases}
ax&x>0\\
bx^2 &x<0
\end{cases}
\]
```

$$f(x) = \begin{cases} ax & x > 0 \\ bx^2 & x < 0 \end{cases}$$

`cases` 环境只支持一个`&`对齐，所以下面的情况只能使用``方式对齐 x

```
\[
f(x)=
\begin{cases}
ax\quad & \phantom{-}0<x<1\\
bx^2 & -1<x<0
\end{cases}
\]
```

$$f(x) = \begin{cases} ax & 0 < x < 1 \\ bx^2 & -1 < x < 0 \end{cases}$$

遇到实在不好对齐的情况，可以使用单边括号配对的方式，

```
\[
f(x,y)=\left\{
\begin{aligned}
&a \quad -1<x<1, \quad y>3 \\
&b \quad y\leq 3
\end{aligned}
\right.
\]
```

$$f(x,y) = \left\{ \begin{array}{l} a \quad -1 < x < 1, y > 3 \\ b \quad y \leq 3 \end{array} \right.$$

右单边括号

```
\[
\left.
\begin{aligned}
&ax\\
&bx^2
\end{aligned}
\right\} \text{单边大括号}
\]
```

$$\left. \begin{array}{l} ax \\ bx^2 \end{array} \right\} \text{单边大括号}$$

mathtools 包定义的 rcases 环境，右单边大括号，参见第118页。

3.7 积分

积分用`\int_{下标}^{上标}`表示，上下标如果只有一个数字或字母，花括号可以省去。养成良好编程习惯，最好不省去花括号

```
\[
\int_{a}^b f(x)\mathrm{d}x=120
\]
```

$$\int_a^b f(x)dx = 120$$

默认的积分有四重，分别是 int, iint, iiint, iiiint

```
\[
\int f(x)\mathrm{d}x
\hspace{5mm}
\iint_S f(x,y)\mathrm{d}x\mathrm{d}y
\hspace{5mm}
\iiint_V f(x,y,z)\mathrm{d}x\mathrm{d}y\mathrm{d}z
\hspace{5mm}
\iiint_{\Omega} f(x,y,z,t)\mathrm{d}x\mathrm{d}y\mathrm{d}z\mathrm{d}t
\]
```

$$\int f(x)dx \quad \iint_S f(x,y)dxdy \quad \iiint_V f(x,y,z)dxdydz \quad \iiint_{\Omega} f(x,y,z,t)dxdydzdt$$

多重积分的写法为

```
\[
\idotsint_{\Omega} f(x_1,\dots,x_k)
\]
```

$$\int \cdots \int_{\Omega} f(x_1, \dots, x_k)$$

环积分

环积分需要加载 esint 包

```
\usepackage{esint} %环积分包
```

积分命令前加 o 即可

```
\[
\oint f(x,y)\mathrm{d}l
\]
```

$$\oint f(x,y)dl$$

二重环积分

```
\[
\oiint f(x,y)\mathrm{d}S
\]
```

$$\oiint f(x,y)dS$$

bigints 包，超大积分号。公式太大了或者换行时你想让积分号包含所有行，可以使用这个包。

```
\usepackage{bigints}
```

默认的积分上下脚标写到侧面，如果希望写到下面，使用下面的语句

```
\[
\int\limits_{A}^{B} f(x)\mathrm{d}x
\]
```

$$\int_A^B f(x)dx$$

如果一篇文章中希望积分脚标都使用上下形式，可以加载 amsmath 包的选项

```
\usepackage[intlimits]{amsmath}
```

注意如果加载了 esint 包，必须也添加相同选项，否则会使得 amsmath 的 [intlimits] 选项失效

```
\usepackage[intlimits]{amsmath}
\usepackage[intlimits]{esint}
```

3.8 矢量

我习惯用黑体表示矢量或矩阵或张量，加载 bm 包，bm 包是标准 L^AT_EX 包，不是 ams 体系。bm 包的黑体保持斜体，有些书的公式黑体是正体的，使用的是 `\mathbf{}`。

```
\usepackage{bm} %黑体且斜体
```

叉乘，这里可以比较一下 `\bm{}` 和 `\mathbf{}` 效果

```
\[
\nabla\times\bm{A}\hspace{1cm}\nabla\times\mathbf{A}
\]
```

$$\nabla \times \bm{A} \quad \nabla \times \mathbf{A}$$

点乘

```
\[
\nabla\cdot\bm{A}
\]
```

$$\nabla \cdot \bm{A}$$

梯度

```
\[
\bm{E}=\nabla\phi(\bm{r})
\]
```

$$\boldsymbol{E} = \nabla\phi(\boldsymbol{r})$$

梯度算符，这里学习偏微分符号

```
\[
\nabla=\bm{i}\frac{\partial}{\partial x}+\bm{j}\frac{\partial}{\partial y}+\bm{k}\frac{\partial}{\partial z}
\]
```

$$\nabla = \boldsymbol{i} \frac{\partial}{\partial x} + \boldsymbol{j} \frac{\partial}{\partial y} + \boldsymbol{k} \frac{\partial}{\partial z}$$

时间微分，内置定义到四重，分别是 dot, ddot, dddot, dddddot

```
\[
\dot{\bm{E}}\hspace{3mm}\dddot{x}
\]
```

$$\dot{\boldsymbol{E}} \quad \ddot{x}$$

有些书习惯用上箭头表示矢量

```
\[
\vec{E}\vec{\nabla}
\]
```

$$\vec{E}\vec{\nabla}$$

专门的箭头包为

```
\usepackage[d]{esvect} %箭头包，d是默认，箭头样式从a到h
```

3.9 矩阵

矩阵环境是圆括号 pmatrix, 方括号 bmatrix, 花括号 Bmatrix, 绝对值 vmatrix, 模 Vmatrix, 和小矩阵 smallmatrix (不带括号)。矩阵的行用换行符\\分隔，矩阵的列用表格单元格符号 & 分隔

```
\[
\begin{pmatrix}
a & b \\
c & d
\end{pmatrix}
\]
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

小矩阵适合用于行间公式

```
\[
(
\begin{smallmatrix}
a & b\\
c & d\\
\end{smallmatrix}
)
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

跨越数列的点号，`\dots`是位置偏下的三个横点，`\cdots`是位置居中的三个横点，`\vdots`是三个竖点，`\ddots`是对角线方向的三个点。

```
\[
\begin{pmatrix}
a & b & c & \cdots & z\\
b & c & d & \cdots & a\\
\vdots & \vdots & \vdots & \ddots & \vdots\\
z & a & b & \cdots & y\\
\end{pmatrix}
```

$$\begin{pmatrix} a & b & c & \cdots & z \\ b & c & d & \cdots & a \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z & a & b & \cdots & y \end{pmatrix}$$

有时会使用`\hdotsfor{列数}`获得贯穿所有列的横向点，这个不知道哪个包冲突，这本书里编译不出来效果。

```
\[
\begin{pmatrix}
a & b & \cdots & d\\
a & b & \cdots & d\\
\hdotsfor{4}\\
a & b & \cdots & d\\
\end{pmatrix}
```

不用加括号的情况用 `array` 环境，本质就是公式里的表格，必须指定列数和列的对齐方式

```
\[
\begin{array}{cc}
a & b\\
c & d\\
\end{array}
```

$$\begin{array}{cc} a & b \\ c & d \end{array}$$

经常遇到需要修改矩阵内的行距的情况，采用下面的语句

```
\[
\begingroup
\renewcommand*{\arraystretch}{1.7}
\begin{pmatrix}
\dfrac{1}{2} & \dfrac{1}{2} \\
\dfrac{1}{2} & \dfrac{1}{2}
\end{pmatrix}
\endgroup %结束扩张行距, 不影响下面
\begin{pmatrix} %不改变时行距太小
\dfrac{1}{2} & \dfrac{1}{2} \\
\dfrac{1}{2} & \dfrac{1}{2}
\end{pmatrix}
\]
```

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

`\renewcommand*{\arraystretch}{1.7}`只能放在矩阵环境外, 因此需要添加`\begingroup`和`\endgroup`, 使得行距调整局域有效, 不影响后面的矩阵, 方便矩阵连乘情况。`\begingroup`和`\endgroup`模块只对单行公式有效, 不能用于多行情况。矩阵环境视为单行内容, 不是多行公式。这个模块代码太啰嗦, 其实一对花括号代表局域环境就行了, 而且花括号方式适合任何情况, 单行公式、多行公式、多个公式环境等。单独一个矩阵需要调节行距不需要`\begingroup`和`\endgroup`模块, `\renewcommand*{\arraystretch}{倍数}`这个语句在公式环境内是局域在单行的, 换行就会失效。这个语句只适规则分式情况, 因为它是上下对称地扩张间距。

如果矩阵内不是所有的行距都需要调整, 特殊行需要调整下面的行距时, 直接使用`\\[8mm]`方式即可。很多情况下手动单行调整更美观。

`nicematrix` 包提供了更多的矩阵形式, 且手册一直在更新。

```
\usepackage{nicematrix}
```

3.10 狄拉克符号

左矢`\langle`和右矢`\rangle`, `\hat`是算子符号

```
\[
\langle i | \hat{H} | j \rangle = H_{ij}
\]
```

$$\langle i | \hat{H} | j \rangle = H_{ij}$$

正交性

```
\[
\langle i | j \rangle = \delta_{ij}
\]
```

$$\langle i | j \rangle = \delta_{ij}$$

完备性

```
\[
\sum_i | i \rangle \langle i | = \hat{I}
\]
```

$$\sum_i | i \rangle \langle i | = \hat{I}$$

左右矢可以用`\left`和`\right`配对

```
\[
\left\langle i \left| \frac{\partial}{\partial x} \right| j \right\rangle
\]
```

$$\left\langle i \left| \frac{\partial}{\partial x} \right| j \right\rangle$$

3.11 复杂上下标

公式里的上下标不支持换行，换行可使用`\substack`

```
\[
\sum_{\substack{0 < i < m \\ 0 < j < n}} x_{ij}
\]
```

$$\sum_{\substack{0 < i < m \\ 0 < j < n}} x_{ij}$$

如果还需要设置对齐方式，可使用 `subarray` 环境，比如设置左对齐

```
\[
\sum_{\begin{subarray}{l} i \\ 0 < j < n \end{subarray}} x_{ij}
\]
```

$$\sum_{\substack{i \\ 0 < j < n}} x_{ij}$$

设置为居中对齐

```
\[
\sum_{\begin{subarray}{c} i \\ 0 < j < n \end{subarray}} x_{ij}
\]
```

$$\sum_{\substack{i \\ 0 < j < n}} x_{ij}$$

求和号加撇不能直接加，需要是细小间隔或花括号

```
\[
\sum \thinspace 'x_i
\sum {'x_i
\]
```

$$\sum 'x_i \sum 'x_i$$

这两种方式都不好，更美观且正确的方式为

```
\[
\sum \nolimits 'x_i
\]
```

$$\sum' x_i$$

或者

```
\[
\sideset{}{'} \sum_i x_i
\]
```

$$\sum_i' x_i$$

`\sideset`可以在符号前上下和后上下都加入脚标

```
\[
\sideset{_a^b}{_c^d}\prod_i^N x_i
\]
```

$$\prod_{i=a}^b \prod_c^d x_i$$

加撇的时候可以省去一个上标号

```
\[
\sideset{_a^b}{_c'}\sum_i^N x_i
\]
```

$$\sum_{i=a}^b \sum_c' x_i$$

看上去好像这么复杂的脚标不常见，下面是一个复变函数里的积分的例子

```
\[
\sideset{}{C}\int\limits_a^b f(z)\mathrm{d}z
\]
```

$$\int_C^b f(z) dz$$

3.12 一些比较常见的特殊形式

等号上有文字，使用`\overset`

```
\[
f(x)\overset{\text{def}}{=}x^2-1
\]
```

$$f(x) \stackrel{\text{def}}{=} x^2 - 1$$

等号下有文字，使用`\underset`

```
\[
f(x)\underset{\text{def}}{=}x^2-1
\]
```

$$f(x) \underset{\text{def}}{=} x^2 - 1$$

这两个命令对解释公式定义非常有用，比如

```
\[
\text{等压: } \hspace{5mm} \overset{[\text{焓的变化}]}{\Delta H} = \overset{[\text{内能变化}]}{\Delta U} + \overset{[\text{对外界做功}]}{P \cdot \Delta V} = \overset{[\text{从外界获得热量}]}{\Delta Q}
\]
```

$$\text{等压:} \quad \overset{[\text{焓的变化}]}{\Delta H} = \overset{[\text{内能变化}]}{\Delta U} + \overset{[\text{对外界做功}]}{P \cdot \Delta V} = \overset{[\text{从外界获得热量}]}{\Delta Q}$$

公式上面横向大花括号，使用`\overbrace`

```
\[
\overbrace{a+b+\cdots+z}^n
\]
```

$$\overbrace{a+b+\cdots+z}^n$$

公式下面横向大花括号，使用`\underbrace`

```
\[
\underbrace{a+b+\cdots+z}_n
\]
```

$$\underbrace{a + b + \cdots + z}_n$$

横向大花括号也经常用来解释公式。

箭头上下有文字

```
\[
A\xleftarrow[belowbelow]{above}B
\]
```

$$A \xleftarrow[\text{belowbelow}]{\text{above}} B$$

```
\[
A\xrightarrow[below]{above}B
\]
```

$$A \xrightarrow[\text{below}]{\text{above}} B$$

右箭头太常见了，所以有一个简写，

```
\[
\lim_{x\to 0}
\]
```

$$\lim_{x \rightarrow 0}$$

3.13 结构型公式

需要加载 amscd 包

```
\usepackage{amscd}
```

amscd 定义好了向左、向右、向上、向下的箭头结构。上下箭头不换行，否则会在一条线上。联用就得到更复杂的公式结构。

```
左上角 @>向右箭头上方的文字>向右箭头下方的文字> 右上角\\
@A{b}A{aa}A @V{b}V{cc}V\\
左下角 @<向左箭头上方的文字<向左箭头下方的文字< 右下角
```

```
\[
\begin{CD}
X @>above>j> T \\
@A{b}A{aa}A @V{b}V{cc}V \\
B @<ii<below< D
\end{CD}
\]
```

$$\begin{array}{ccc} X & \xrightarrow[\text{j}]{\text{above}} & T \\ b \uparrow \text{aa} & & b \downarrow \text{cc} \\ B & \xleftarrow[\text{below}]{\text{ii}} & D \end{array}$$

3.14 标注重点

使用{\color{颜色}换色的部分公式}

```
\[
{\color{red}\sin x}{\color{blue}\cos x}
\]
```

$$\sin x \cos x$$

boxed 只能用于 equation, aligned, 不能用于其他多行公式

```
\begin{equation}
\boxed{f(x)=\sin x}\cos x
\end{equation}
```

$$\boxed{f(x) = \sin x} \cos x \quad (3-29)$$

加载 empheq 包, 可以给公式加框

```
\usepackage{empheq}
```

```
\begin{empheq}[box=\fbox]{equation}
f(x)=\sin x
\end{empheq}
\begin{empheq}[box=\fbox]{align}
f(x)&=\sin x\\
g(x)&=\cos x
\end{empheq}
```

$$\boxed{f(x) = \sin x} \quad (3-30)$$

$$\boxed{f(x) = \sin x} \quad (3-31)$$

$$\boxed{g(x) = \cos x} \quad (3-32)$$

或加底色

```
\begin{empheq}[innerbox=\colorbox{yellow!60}]{equation}
f(x)=\sin x
\end{empheq}
```

$$\boxed{f(x) = \sin x} \quad (3-33)$$

3.15 数学符号

数学符号

笔者平时尽量用 binary 符号, 少打字, 主要是基本够用。特殊情况下还是要加载其他包的。最常用的符号不需要额外加载包, ams 有自己的数学符号, 比基础符号多一些, 需要加载 amssymb 包

```
\usepackage{amssymb}
```

另一些不常用但是有用的, 比如线段上面长矢量箭头, λ 上部加横线, 在下面的包

```
\usepackage{MnSymbol} %不推荐mdsymbol, adobe打开符号全部错乱
```

如果希望使用 mathabx 包里的 \widebar 功能, 又想使用 MnSymbol 包里的 \middlebar 功能, 推荐加载 MnSymbol 包, 并使用下面的语句自定义 \widebar

```
\newcommand{\widebar}[2][4]{%
第2个方括号里的参数调整bar的长短, 4与widebar原始大概相同
}{\mkern#1mu\overline{\mkern-#1mu#2}}
\[
\widebar{A}\widebar{B}
\]
```

$$\overline{A}\overline{B}$$

oplotsymb1

一些新的不常用的特殊符号需要加载 oplotsymb1 包

```
\usepackage{oplotsymb1}
```

```
\[
\trianglepadot\ \trianglepacross
\]
```

\triangle \otimes

L^AT_EX 的数学符号非常强大，常用的手册是下面这两个

```
texdoc maths-symbols %常用符号
texdoc symbols %符号大全
```

symbols 手册笔者也没有研究过，总之非常丰富。符号这块笔者是用多少学多少，用多了也就不需要查笔记了。比如 oplotsymb1 包是因为有学生问才去找的，符号大全里也有介绍。

常用字体 amsfonts 包

ams 可以使用 \mathcal 字体，这个字体可以视为现代花体，与下面古老的花体相比较，这个字体的字母很容易辨认

```
\[
\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}
\]
```

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

amsfonts 是最常见的数学符号包

```
\usepackage{amsfonts}
```

加载后可以使用 \mathfrak{}

```
\[
\mathfrak{ABCDEFGHIJKLMNOPQRSTUVWXYZ}
\]
```

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

\mathfrak 这个字体有小写

```
\[
\mathfrak{abcdefghijklmnopqrstuvwxyz}
\]
```

a b c d e f g h i j k l m n o p q r s t u v w x y z

使用 \mathbb{} 得到

```
\[
\mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ}
\]
```

ABCDEFGHIJKLMNOPQRSTUVWXYZ

mathrsfs

加载 mathrsfs 包可以得到旧花写，很多老书或老文献里会出现

```
\usepackage{mathrsfs}
```

使用 `\mathscr{}` 得到旧花写，有些字母很漂亮，但是个别字母难以辨认出是什么。现代书籍和文献中已经不常使用了。

```
\[
\mathscr{ABCDEFGHIJKLMNOPQRSTUVWXYZ}
\]
```

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

mathabx

```
\usepackage{mathabx} %与MnSymbols冲突
```

加载 mathabx 包，还解决了 `\bar{A}` 太短、`\overline{A}` 太长还会与下一个 `\overline{X}` 连接在一起的问题，定义了 `\widebar{A}`。

单位

国际单位制包

```
\usepackage{siunitx}
```

微米的 μ 是斜体，因此需要加载 upgreek 包换成直体

```
\usepackage{upgreek}
```

单位包的确很好用，大部分常见单位都定义了简写形式。

投稿时不要轻易使用 `siunitx` 包，有些杂志社的编辑器可能不支持。采用杂志社给出模板的默认包比较合适，加载杂志社的类后尝试一下命令，如果编译出错就是没有加载这个包。如果杂志社给出的模板有某些包，直接用就行了，一般不会出错。投稿时尽量不要自己加包。

3.16 mathtools

加载时建议分开写

```
\usepackage{amsmath}
\usepackage{mathtools}
```

mathtools 包定义了一些常用的格式细节调整。

rcases 环境,

```
\[
\begin{rcases}
f+ax\\
bx
\end{rcases}
\]
```

$$\left. \begin{array}{l} f+ax \\ bx \end{array} \right\}$$

矩阵单元对齐位置的细调

```
\[
\begin{pmatrix*}[r]
a & -b \\
-c & d
\end{pmatrix*}
\]
```

$$\begin{pmatrix} a & -b \\ -c & d \end{pmatrix}$$

手册有许多细节美感的公式例子, 比如\vdots与等号居中对齐

```
\begin{align*}
a&=b\\
&\vdots\\
&=c\\
&\shortvdots\\
&=d
\end{align*}
```

$$\begin{array}{c} a = b \\ \vdots \\ = c \\ \vdots \\ = d \end{array}$$

数学系的可以好好研究一下这个包, 大部分常见的结果 amsmath 就足够了。

3.17 定理

新定理命令是标准 L^AT_EX 语句, 不需要额外的包, 可以很方便的自定义新的定理环境, 方括号里是可选项, 默认编号不带小节。

```
\newtheorem{新定理环境名}{新环境显示名称}[section]
```

\newtheorem写在 document 环境之前或之后都可以。定理环境全部需要自定义, 没有默认定义好的。

```
\newtheorem{homework}{作业}[section]
\begin{homework}[曲线积分]
作业内容
\end{homework}
```

作业 3.17.1 (曲线积分) 作业内容

可以规定某些环境一起编号, 下面的例子, joke 环境会单独编号, 不会与 homework 同时编号, 但是 amuse 环境与 joke 环境混合在一起编号。

```
\newtheorem{新定理环境名}[一起编号的环境名]{新环境显示名称}[section]
```

设定合在一起编号时，后面方括号里设定的带小节号语句推荐省略，编号样式会自动与一起编号的环境相同。

```
\newtheorem{joke}{笑话}[section]
\newtheorem{amuse}[joke]{开心的事} %不需要再规定section
\begin{joke}
笑话
\end{joke}
\begin{amuse}[现代]\hfill\\ %这里不能直接换行
开心一刻
\end{amuse}
```

笑话 3.17.1 笑话

开心的事 3.17.2 (现代)

开心一刻

使用 ntheorem 包可以方便设定 theorem 的格式。

```
\usepackage[选项]{ntheorem}
```

这个包需要说明的比较多，略微写一下。

- 是否加载预先定义的环境，默认是不加载 (noconfig)，加载 [standard]，获得以下几种环境
 - Theorems: Theorem, Lemma, Proposition, Corollary, Satz, Korollar
 - Definitions: Definition
 - Examples: Example, Beispiel
 - Remarks: Anmerkung, Bemerkung, Remark
 - Proofs: Proof, Beweis

这个包是德国人写的，所以有些环境是相同意义的德语单词

- framed 选项，需要同时加载 framed 包。可以使用 \newframedtheorem 命令，该命令定义的环境，整体自动加框。
- thmmarks 选项，最后结束添加小的方框表示结束。对 standard 定义的环境有效。对新环境，需要指定类才生效。

```
\usepackage[standard,thmmarks,framed]{ntheorem} %同时加载
\theoremclass{Theorem} %指定新环境继承哪个standard环境的设置，不加没有thmmarks
\newframedtheorem{mynew}{笑话}
```

- thref 选项，额外的索引性质
- amsmath 选项，与 amsmath 包兼容
- hyperref 选项，与 hyperref 包兼容

推荐 standard 和 thmmarks 选项。

这个包使用默认 style 比较麻烦，因为加载默认 style 只对新定义的环境有效，standard 就修改为默认的 plain。其实这不影响使用。

```
\usepackage[standard]{ntheorem}
```

```
\theoremclass{Theorem} %thmmarks时必须在指定style之前，否则style无效
\theoremstyle{break} %自动换一行
\newtheorem{mynew}{笑话} %对新定义的环境起作用
\renewtheorem{Theorem}{定理} %推荐不加载standard，反正都要重新定义
```

如果修改结束符号，不是采用默认的 thmmarks，就不需要\theoremclass{}这句了。手册里单独修改样式的语句，也只对新定义的环境或重新定义的环境有效。常用修改语句如下

```
\theoremheaderfont{\normalfont\bfseries} %默认定理、笑话这样的header直体加粗
\theorembodyfont{\itshape} %定理内容斜体
\theoremseparator{} %无分隔符，这个指内容之前的符号
\theoremprskip{\topsep} %上下距离，item的间距
\theorempostskip{\topsep}
\theoremindent0cm %默认无缩进
\theoremnumbering{arabic} %编号为阿拉伯数字
\theoremsymbol{$\bullet$} %默认没有符号，加thmmarks时为小方框，这句修改为实心圆点
\theoremprwork{\bigskip\hrule} %开始环境前的前置语句，这里是纵向留白加横线
\theorempostwork{\hrule\bigskip} %结束环境前的后置语句，这里定理后加横线纵向留白
```

3.18 网页公式

mathjax 支持网页上直接写 L^AT_EX 公式，下面是一个网络引擎的例子，公式带编号且自动编号，支持 amscd。如果自己做网站，可以将 mathjax 库下载到本地，链接方式改为本地。

```
<!DOCTYPE HTML>
<html>
<body>

<script type="text/javascript" id="MathJax-script" async src="https://cdn.bootcss.com/mathjax/2.7.1/latest.js?config=TeX-AMS-MML_HTMLorMML">
</script>

<script type="text/x-mathjax-config">
  MathJax.Hub.Config({ TeX: { equationNumbers: { autoNumber: "AMS" } } });
</script>

<script type="text/x-mathjax-config">
  MathJax.Hub.Config({extensions: ["tex2jax.js"],jax: ["input/TeX", "output/HTML-CSS"],
  tex2jax: {inlineMath: [ ['$','$'], ["\\(", "\\)"] ],displayMath: [ ['$$','$$'], ["\\[",
  "\\]"] ],processEscapes: true},"HTML-CSS": { availableFonts: ["TeX"] }, });
</script>

\begin{equation}
f(x)
\end{equation}

in text, \(\sin x\), \(\int f_n(x)\mathrm{d}x\)
```



```

\[ \cos x \]

\begin{align}
f(x)\\
g(x)
\end{align}

$\int_a f(x)\mathrm{d}x$

\[\require{AMScd}
\begin{CD}
A @>j>> T\\
@AA{aa}A @VV{cc}V\\
B @= D
\end{CD}
\]

</body>
</html>

```

写这本书时这个国外的网络引擎是可用的，有时打开会很慢。网络引擎也有国内地址的，但是不同版本号支持的强度不同，有的版本支持 `amscd`，有的不支持，需要实际测试一下。

建网站的时候把 `mathjax` 下载到本地，链接到本地目录即可。

第四章 画图包 tikz

L^AT_EX 体系有两大矢量画图包体系，pstricks 和 tikz。之所以称为体系，是因为基于这两个包衍生出了许多有用的包或库，能够实现许多设计好的功能。还有一些矢量画图包相对比较小众，比如 asymptote。而网上 L^AT_EX 画图例子也多是基于这两个包。

tikz 和 pstricks 比较来说，tikz 继承了 L^AT_EX 忽略空格的特点，语句的鲁棒性更强。语句末尾按 C 语言风格加分号，语法上更规范易懂。但是 tikz 没有定义面这个概念，所以目前在三维上不如 pstricks。比起 pstricks，tikz 画图有一个很大的好处：跟 L^AT_EX 一样，大部分时候对空格不敏感，这极大地减少了代码纠错的压力。

包的丰富程度上，pstricks 因为发展较早，所以更占优，尤其是 pstricks 有一个化学实验包 pst-labo，做得非常好。但是基于 tikz 的平面化学分子式包 chemfig 功能强大，给 tikz 体系加分不少。两个体系笔者都没有见到化学分子的三维立体球棍模型。

这章主要学习 tikz 画图体系，tikz 可以直接 pdf_lat_ex 编译生成 pdf，大部分情况下对 xelatex 的支持也很好。三维公式、数据画图包 pgfplots 讲课的时候分开了，参考书仍然放在 tikz 这一章，因为它们是一个体系的。化学分子包 chemfig 单独为一章。

pgf 是 tikz 的底层，所以 tikz 和 pgf 是一个文档，下面的命令调出的是同一个文件。

```
texdoc tikz
texdoc pgf
```

加载 tikz 包的语句为

```
\usepackage{tikz}
```

加载 tikz 库的语句为

```
\usetikzlibrary{tikz库}
```

tikz 画图环境为 tikzpicture，这个环境视同文字，很方便图文混排。


```
直线视为行间文字
\begin{tikzpicture}
\draw (0,0)--(1,0);
\end{tikzpicture}
```

直线视为行间文字

也可以采用简写形式

```
文字\tikz{\draw (0,0)--(1,0);\draw (0,0)--(1,1);}文字
```

文字



文字

tikz 画图语句用分号结尾，如果缺失分号会编译报错。

直接添加到 figure 环境内可以构成独立的图片。

```
\begin{figure}[ht!]
\begin{tikzpicture}
\draw (0,0)--(1,0);
\end{tikzpicture}
\caption{tikz画出的图。}
\end{figure}
```

除了极少数简单图形和动画外，一般情况推荐使用 standalone 类。使用这个类，每一个 tikzpicture 构成 pdf 的一页，大小跟随图片大小变化。笔者一般一个 tex 文件一幅图，生成的 pdf 文件本身是带 tight bounding box，可以作为图片直接使用。文件结构参见第197页。

4.1 直线

4.1.1 坐标

tikz 画图基于坐标点，默认单位是厘米。支持自定义长度单位。主要需要掌握的坐标点的形式如下

- 二维笛卡尔坐标(x,y)
- 三维笛卡尔坐标(x,y,z)，tikz 的三维本质上是三维图的二维投影效果，可以直接使用三维坐标，已经内置好了三个坐标轴在二维图中的方向。
- 极坐标(角度:径矢长度)
- 椭圆极坐标(角度:x轴半径 and y轴半径)，椭圆极坐标配合 calc 库，可以很方便实现圆规画椭圆弧。
- node 坐标，在 node 里会重点介绍。tikz 最灵活最实用的坐标形式。

默认长度是 cm，矢量高清图缩放不影响清晰度，所以一般不建议修改默认长度。但是发现图大了或小了，可以使用下面的方式缩放

```
\begin{tikzpicture}[x=5mm,y=5mm]
画图语句
\end{tikzpicture}
```

或者

```
\begin{tikzpicture}[scale=0.5]
画图语句
\end{tikzpicture}
```

scale 方式是整体缩放，这两种缩放方式都只缩放图片，不缩放其中的文字。

tikz 支持随时修改长度单位，支持只修改一个长度单位。在\draw的参数里面修改，对该语句所有坐标适用。还支持直接在坐标里修改。

```
\begin{tikzpicture}[very thick]
\draw (0,0)--(1,0); %画线语句
\draw [x=2cm,y=2cm,blue] (0,-0.3)--(1,-0.3); %改单位长度
\draw [red] (0,-0.6)--(1,-0.6); %上面的修改不影响这句
\draw [x=2cm,cyan] (0,-0.3)--(1,-0.3); %改单位长度
\draw (0,-0.9)--(2cm,-0.9); %直接在坐标里改
\end{tikzpicture}
```



第一次修改，纵坐标是-0.3，实际画图是在-0.6。第二次修改，只修改横坐标，实际画了2cm，纵坐标不修改，所以在0.3的位置画线。第三次修改是直接在坐标值里修改，该方式对两种极坐标也适用。

生成的图片大小是相对结果，不是绝对结果。所以下面的画图语句结果是一样的，都得到1cm 的水平线。

```
\begin{tikzpicture}
\draw (0,0)--(1,0);
\end{tikzpicture}
```

```
\begin{tikzpicture}
\draw (10,0)--(11,0);
\end{tikzpicture}
```

```
\begin{tikzpicture}
\draw (0,0)--(0:1);
\end{tikzpicture}
```

划线有两个简省语句，-|表示先横后竖直线连接两个坐标点，|-表示先竖后横直线连接两个坐标点。

```
\begin{tikzpicture}
\draw (0,0)-|(1,1);
\draw (2,0)|-(3,1);
\end{tikzpicture}
```



可以用内置函数（不需要加载任何库）计算坐标的值，此时坐标的值需要花括号括起来。

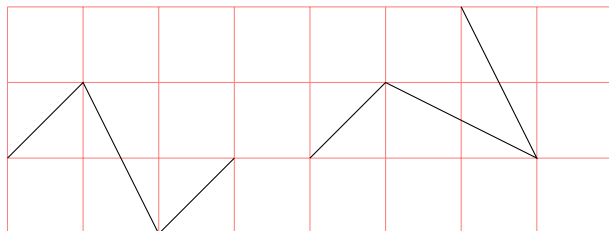
```
\begin{tikzpicture}
\draw (0,0)--({2*sqrt(2)*cos(30)},{2*sqrt(2)*sin(30)});
\end{tikzpicture}
```

很多时候，加号非常好用。单加号表示当前坐标与起点坐标相加，双加号表示当前坐标与前面的坐标相加。

```
\begin{tikzpicture}
\draw (0,0)---+(1,0)---+(1,1)---+(0,1)--cycle;
\draw [color=brown] (2,0)---++(1,0)---++(0,1)---++(-1,0)--cycle;
\end{tikzpicture}
```

单加号和双加号虽然可以混用，但是混用时需要留意坐标如何相加的。如下面的例子，单加号和双加号会互相影响，增加计算坐标时的困难度。

```
\begin{tikzpicture}
\draw [step=1,red!50] (0,-1) grid (8,2);
\draw (0,0)---+(1,1)---++(2,-1)---+(1,1);
\draw (4,0)---++(1,1)---+(2,-1)---++(1,1);
\end{tikzpicture}
```



坐标还有一种方式是上下左右，双加号和单加号意思如前面，默认单位是厘米，可以修改单位。

```
\begin{tikzpicture}
\fill [red] (0,0) circle (2pt);
\draw (0,0)--++(up:1)--++(left:1)---++(down:1)---++(right:1);
\end{tikzpicture}
```



可以用 `turn` 修饰坐标，表示转多少角度，走多远距离。`turn` 修饰的坐标如果是公式计算出来的，需要额外添加花括号。只有一个花括号会报错。

```
\begin{tikzpicture}
\draw [thick] (0,0)--(1,0)--([turn]30:2)--([turn]{asin(-0.5)}:1);
\end{tikzpicture}
```



额外的花括号也可以这么加

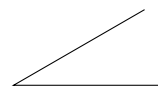
```
([turn]{asin(-0.5)}:1)
```

上下左右相当于固定角度的 `turn` 语句。

给坐标点起名

坐标点可以先用 `\coordinate` 语句定义，然后调用名字使用，这样可以起名后多次复用。

```
\begin{tikzpicture}
\coordinate (A) at (2,0);
\coordinate (O) at (0,0);
\coordinate (B) at (30:2);
\coordinate (D) at (-30:2);
\draw (A)--(O)--(B);
\end{tikzpicture}
```



自动获得路径上的点的坐标

可以在划线过程中的任意中间坐标处设置坐标点名称，写到画线语句 `--` 或 `edge`、`to` 后面。

```
\draw (0,0) coordinate (a) -- coordinate (m) (1,0) coordinate (b) --(1,1) coordinate (c);
```

`m` 点坐标默认是线段的中点，可以使用 `pos` 参数调节

```
\draw (0,0) coordinate (a) -- coordinate [pos=0.8] (m) (1,0) coordinate (b) --(1,1) coordinate (c);
```

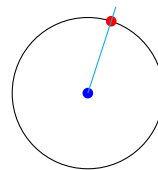
`m` 点坐标在线段 80% 长度处。

`coordinate` 和 `node` 可以混用，即对一个坐标点同时起名和在其附近指定位置添加 `node`。

```
\draw (0,0) coordinate (a) node (l) [left,draw] {left text} -- coordinate [pos=0.7] (m) (2,0) coordinate (b) node (r) [right, draw] {$x_2$};
```

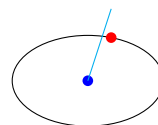
画圆（椭圆）不能沿圆周按 `[pos=数值]` 定义坐标，但是画弧可以，因为画弧是从起点画到终点，有明确的路径和弧线长度，所以可以从 0 到 360 度画弧获得圆周上的位置坐标。

```
\begin{tikzpicture}
\draw (1,0) arc [start angle=0, end angle=360,
radius=1] coordinate [pos=0.2] (a);
\fill [red] (a) circle (2pt);
\fill [blue] (0,0) circle (2pt);
\draw [cyan] (0,0)--(360*0.2:1.2); %与角度的关系
\end{tikzpicture}
```



设定[pos]的数值，坐标点位置是严格按曲线长度计算的，画圆长度比例等价于角度比例，画椭圆就不再等价于角度比例了。

```
\begin{tikzpicture}
\draw (1,0) arc [start angle=0, end angle=360, x
radius=1, y radius=0.6] coordinate [pos=0.2] (a);
\fill [red] (a) circle (2pt);
\fill [blue] (0,0) circle (2pt);
\draw [cyan] (0,0)--(360*0.2:1); %关系不一致
\end{tikzpicture}
```



如果只希望提取坐标，可以使用\path代替\draw，手册中的解释，\draw是\path[draw]的简写。

```
\begin{tikzpicture}
\path (0,0) -- coordinate [pos=0.3] (a) (1,0);
\fill [] (0,0) circle (2pt);
\fill [] (1,0) circle (2pt);
\fill [red] (a) circle (2pt);
\end{tikzpicture}
```



4.1.2 线参数

线的参数有颜色、线宽、线型、圆角大小等。

内置线宽为 ultra thin, very thin, thin, semithick, thick, very thick, ultra thick。thin 的效果等价于\hrule。自定义线宽的方式是line width=2pt。

内置线型为 dashed, dotted, dash dot 和 dash dot dot，可以用 loosely 和 densely 修饰线型。

默认是黑色的直的细线，无圆角。

划线讲究一笔画下来，中间不要停顿。最后加 cycle 表示回到起点的闭合线。

```
\begin{tikzpicture}
\draw [color=blue, line width=2pt, rounded corners=5pt]
(0,0)--(1,0)--(1,1)--(0,1)--cycle;
\end{tikzpicture}
```



颜色、线宽、线型等参数是画线语句--的基本参数，都不可以在一个\draw语句内改变，如果需要修改，需要使用 edge 替代--，使用的时候有一些细微差别。本质上--是一笔画，edge 不是。但是圆角参数是可以的，圆角参数修饰的是--，所以放在其前面。这个功能在画腔体的时候挺有用的。

```
\begin{tikzpicture}
\draw [color=blue, line width=2pt, rounded corners=5pt] (0,0)--(3,0)[rounded
corners=0pt]--(3,1)[rounded corners=15pt]--(0,1)[rounded corners=0pt]--cycle;
\end{tikzpicture}
```



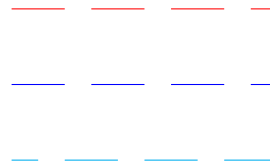
自定义 dash 线型

虚线样式可以自定义，on 表示划线，off 表示留白。

```
\begin{tikzpicture}
\draw [dash pattern=on 2pt off 4pt on 4pt off 4pt] (0,0)--(6,0);
\end{tikzpicture}
```

可以使用 phase 来移动线型，默认 phase 是 0pt，但是语句不一样，不设置 phase 时需要加 pattern。

```
\begin{tikzpicture}
\draw [red] [dash pattern=on 20pt off 10pt] (0,0) --
++(3.5,0);
\draw [blue] [dash=on 20pt off 10pt phase 0pt] (0,-1)
-- ++(3.5,0);
\draw [cyan] [dash=on 20pt off 10pt phase 10pt] (0,-2)
-- ++(3.5,0);
\end{tikzpicture}
```



画双色虚线可以使用 postaction，需要准确计算两色交替位置。

```
\begin{tikzpicture}
\draw[postaction={draw,red,dash pattern= on 3pt off 5pt,dash
phase=4pt,thick}]
[blue,dash pattern= on 3pt off 5pt,thick] (0,0) rectangle (1,1);
\end{tikzpicture}
```



计算量的确比较大，但是如果计算准确，其实可以画多色线。下面这个例子是四色的效果。

```
\begin{tikzpicture}
\draw[postaction={draw,red,dash pattern= on 0.5cm off 1.5cm,dash
phase=0cm,thick}] [postaction={draw,green,dash pattern= on 0.5cm off
1.5cm,dash phase=0.5cm,thick}] [postaction={draw,yellow,dash pattern= on
0.5cm off 1.5cm,dash phase=1cm,thick}] [blue,dash pattern= on 0.5cm off
1.5cm,dash phase=1.5cm,thick] (0,0) rectangle (1,1);
\end{tikzpicture}
```



double 线型

double 线性可以画出双线的效果，可以定义两种不同的颜色，默认是中间白两边黑，即 draw =black, double=white。可以定义双线的间距 double distance，整个双线的宽度等于两倍的 line

width 加上 double distance。double distance between line centers这个参数，意思是上面和下面 4pt 红线的中线的间距为 5pt，蓝线就只剩下 1pt 了。

```
\begin{tikzpicture}
\draw [double] (0,0)--(2,0);
\draw [draw=red, double=blue, line width=4pt, double distance=2pt] (3,0)--+(2,0);
\draw [draw=red, double=blue, line width=4pt, double distance between line
centers=5pt] (5.2,0)--+(2,0);
\end{tikzpicture}
```



4.1.3 箭头和 arrows.meta 库

划线语句的一个非常有用的参数是箭头参数。基本箭头不需要添加额外的库。

```
\begin{tikzpicture}
\draw [->] (0,0)--+(1,0);
\draw [<-] (0,-0.5)--+(1,0);
\draw [<->] (0,-1)--+(1,0);
\end{tikzpicture}
```



默认箭头太难看了，推荐使用 latex 或 Latex 箭头。

```
\begin{tikzpicture}
\draw [-latex] (0,0)--(1,0);
\draw [latex-] [red] (0,-0.5)--+(1,0);
\draw [Latex-Latex] [blue] (0,-1)--+(1,0);
\end{tikzpicture}
```



这样比较麻烦，加载 arrows.meta 库，加载 tikz 库的命令是\usetikzlibrary{库名}。

```
\usetikzlibrary{arrows.meta}
```

使用>=箭头形状来重新定义默认箭头，例如设置默认箭头大于号和小于号等于 Latex 形式，Latex 可以添加参数，小写的 latex 不能。双箭头用两个大于号或小于号，多个箭头用多个大于号或小于号。sep 调整多个箭头的间距。加点可以多个箭头是断续的。

```
\begin{tikzpicture}[>={Latex[width=4pt,length=10pt]}]
\draw [{<<[sep=2mm]}->>] (0,0)--(3,0);
\draw [{<<[sep=2mm]}.<[sep=2mm]}->.>>] (4,0)--+(4,0);
\end{tikzpicture}
```



或者使用 scale width 和 scale length

```
\begin{tikzpicture}[>={Latex[scale width=1.5,scale
length=2]}]
\draw [<->] (0,0)--(2,0);
\end{tikzpicture}
```



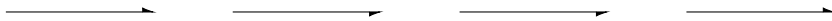
使用 reversed 可以反转箭头方向

```
\begin{tikzpicture}[>={Latex[reversed]}]
\draw [<->] (0,0)--(2,0);
\end{tikzpicture}
```



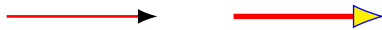
可以只画一半

```
\begin{tikzpicture}
\draw [-{Latex[harpoon]}] (0,0)--(2,0);
\draw [-{Latex[harpoon,swap]}] (3,0)--++(2,0);
\draw [-{Latex[right]}] (6,0)--++(2,0);
\draw [-{Latex[left]}] (9,0)--++(2,0);
\end{tikzpicture}
```



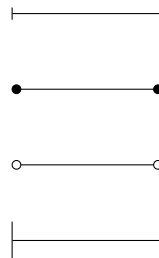
可以设置线与箭头不同色，还可以设置不同的箭头填充颜色和箭头线宽（默认跟随线的宽度）

```
\begin{tikzpicture}
\draw [-{Latex[color=black]}] [red, line width=1pt] (0,0)--(2,0);
\draw [-{Latex[color=blue,fill=yellow, line width=0.5pt]}] [red, line width=2pt]
(3,0)--++(2,0);
\end{tikzpicture}
```



arrows.meta 库包含了许多箭头类型，比如直线和弯勾，用 scope 环境可以很方便更换箭头类型。参见 tikz 手册第五部分的 16.5 小节。

```
\begin{tikzpicture}[>=Bar]
\draw [<->] (0,0)--(2,0);
\draw [Circle-Circle] (0,-1)--++(2,0);
\begin{scope}[>={Circle[open]}]
\draw [<->] (0,-2)--++(2,0);
\end{scope}
\draw [{Bar[width=5mm]}-{Bar[width=5mm]}]
(0,-3)--++(2,0);
\end{tikzpicture}
```



可以多种箭头联用，且对每个箭头都定义不同的参数，每种箭头支持的参数可以查阅手册，上面的例子都是普遍的参数。

```
\begin{tikzpicture}
\draw [-{Latex[red, sep=1mm] . Stealth[blue, sep=1mm] . Stealth[yellow, sep=1mm] .
Circle[red]}] (0,0)--(2,0);
\end{tikzpicture}
```



还有一种特殊的用法，下面的语句得到一个向上的箭头，单边也行。

```
\begin{tikzpicture}[>=latex]
\draw [<->] (0,0);
\end{tikzpicture}
```

4.2 矩形、圆、椭圆和弧

tikz 内置了画矩形、圆、椭圆和弧的命令。

4.2.1 矩形

画矩形的语句有两种，第一种是给出一对对角的坐标值。

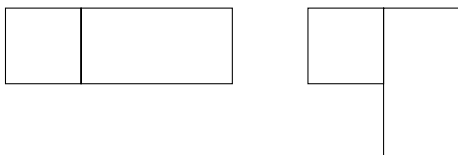
```
\begin{tikzpicture}
\draw [green] (x1,y1) rectangle (x2,y2);
\end{tikzpicture}
```

另一种是利用加号，给出一个角的坐标值和用双加号获得的另一个角的坐标值，即长宽方向的延展长度。

```
\begin{tikzpicture}
\draw [green] (x1,y1) rectangle ++(x2,y2);
\end{tikzpicture}
```

如果只画一个矩形，单加号和双加号没有区别。但是如果一个语句连着画两个矩形，单加号和双加号的区别就很大。

```
\begin{tikzpicture}
\draw (0,0) rectangle ++(1,1) rectangle ++(2,-1);
\draw (4,0) rectangle +(1,1) rectangle +(2,-1);
\end{tikzpicture}
```



凡是闭合曲线都可以填充颜色，默认边框颜色为黑色，填充为白色，所以只给出 fill 的颜色还是会有边框颜色。

```
\begin{tikzpicture}
\draw [fill=red] (0,0) rectangle ++(1,1);
\draw [draw=black,fill=red] (2,0) rectangle ++(1,1);
\end{tikzpicture}
```



如果不希望有边框颜色，可以采用 `\fill` 语句，或将边框设置为白色，但是这两种方式效果不同。`\fill` 语句填充范围是带边框中线围成的区域，而 `[draw=white]` 方式填充范围是不带边框的内部区域。这个例子也可以看出边框宽度是从中线两边扩展的。

```
\begin{tikzpicture}
\draw [draw=gray!30,fill=yellow,line width=3pt] (0,0) rectangle ++(1,1);
\draw [draw=white,fill=yellow,line width=3pt] (1.2,0) rectangle ++(1,1);
\fill [fill=yellow,line width=3pt] (2.4,0) rectangle ++(1,1);
\draw [very thin] (0,0) rectangle ++(1,1);
\draw [very thin] (1.2,0) rectangle ++(1,1);
\end{tikzpicture}
```



任何闭合曲线都可以填充，即使不闭合的曲线也可以填充，不闭合曲线填充时边框会少一个边。

```
\begin{tikzpicture}
\draw [fill=yellow,line width=2pt] (0,0)-| ++(1,1)--cycle;
\draw [fill=yellow,line width=2pt] (2,0)-| ++(1,1);
\end{tikzpicture}
```



4.2.2 圆和圆弧

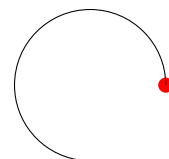
画圆的语句需要给出圆心坐标和半径。有一种简写方式，`[radius=0.5]` 可以用 `(0.5)` 代替。

```
\begin{tikzpicture}
\draw (0,0) circle [radius=0.5];
\draw (2,0) circle (0.5);
\end{tikzpicture}
```



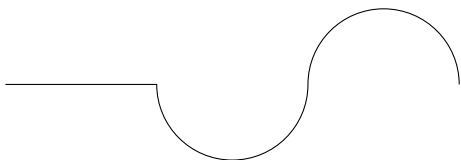
圆弧需要给出起点坐标、半径和起始角度。角度取值范围是 $[-360, 360]$ 区间。

```
\begin{tikzpicture}
\fill [red] (0,0) circle (0.1);
\draw (0,0) arc [start angle=0, end angle=260, radius=1];
\end{tikzpicture}
```



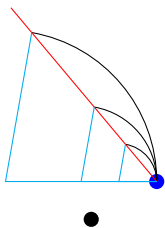
这好像与我们习惯的思维方式不同，圆规画弧是圆心坐标、半径和起始角度。从起点坐标开始画图其实是数控运动的一般情况，这样可以做到直线和弧线的连接。两个弧线语句可以连续使用，第一个弧线画完后，坐标值就移到终点，第二个弧线以第一个弧线终点开始继续画弧。

```
\begin{tikzpicture}
\draw (0,0)--(2,0) arc [start angle=180, end angle=360, radius=1] arc [start
angle=180, end angle=0, radius=1];
\end{tikzpicture}
```



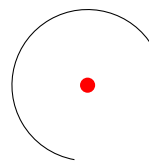
上面只是一个简单的例子，很容易确定圆心，那么从起点坐标画弧的圆心到底在什么位置呢？可以练习一下下面这个例子。起点坐标和起始角度都相同情况下，半径不同，圆心也不同。始点坐标和圆心都在一条直线上。

```
\begin{tikzpicture}
\fill (0,0) circle (0.1);
\fill [blue] (30:1) circle (0.1);
\draw (30:1) arc [start angle=0, end angle=80, radius=0.5];
\draw (30:1) arc [start angle=0, end angle=80, radius=1];
\draw (30:1) arc [start angle=0, end angle=80, radius=2];
\draw [red] (30:1)--++(130:3);
\draw [cyan] (30:1)--++(180:0.5)--++(80:0.5);
\draw [cyan] (30:1)--++(180:1)--++(80:1);
\draw [cyan] (30:1)--++(180:2)--++(80:2);
\end{tikzpicture}
```



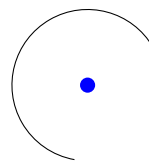
很多时候我们希望像圆规那样画图，可以采用 shift 参数。shift 表示平移，坐标需要用花括号括起来。

```
\begin{tikzpicture}
\fill [red] (1,0) circle (0.1);
\draw [shift={(30:1)}] (1,0) arc [start angle=30, end
angle=260, radius=1];
\end{tikzpicture}
```



如果希望从圆规画圆弧的角度画弧，使用 calc 子库是最方便的，它可以让两个坐标相加，相当于平移

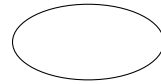
```
\begin{tikzpicture}
\fill [blue] (1,0) circle (0.1);
\draw ($(1,0)+(30:1)$) arc [start angle=30, end
angle=260, radius=1];
\end{tikzpicture}
```



4.2.3 椭圆和椭圆弧

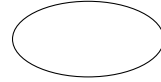
画椭圆的语句需要给出圆心坐标、x 半径和 y 半径。

```
\begin{tikzpicture}
\draw (0,0) ellipse [x radius=1, y radius=0.5];
\end{tikzpicture}
```



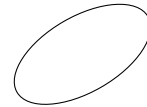
有一种简写方式, `[x radius=1, y radius=0.5]` 可以用 `(1 and 0.5)` 代替。

```
\begin{tikzpicture}
\draw (0,0) ellipse (1 and 0.5);
\end{tikzpicture}
```



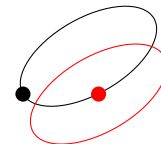
基本语句画的都是正向椭圆, 通过旋转来改变椭圆方向。rotate 表示围绕原点 (0,0) 旋转, 后面的参数是角度。

```
\begin{tikzpicture}
\draw [rotate=30] (0,0) ellipse (1 and 0.5);
\end{tikzpicture}
```



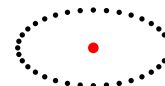
当希望围绕椭圆圆心旋转时, 需要 `rotate around={角度:(旋转基准点坐标)}`

```
\begin{tikzpicture}
\fill (0,0) circle (0.1);
\draw [rotate=30] (1,0) ellipse (1 and 0.5);
\fill [red] (1,0) circle (0.1);
\draw [red,rotate around={30:(1,0)}] (1,0) ellipse (1 and 0.5);
\end{tikzpicture}
```



使用椭圆极坐标, (角度:x半径 and y半径), 下面是一个例子。

```
\begin{tikzpicture}
\fill [red] (0,0) circle (2pt);
\foreach \x in {0,10,...,350}
\fill (\x:1 and 0.5) circle [radius=1pt];
\end{tikzpicture}
```



椭圆弧也是 arc, 需要给出起点坐标、起始角度、x 半径和 y 半径。

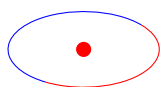
```
\begin{tikzpicture}
\fill [red] (0,0) circle (0.1);
\draw (0,0) arc [start angle=40, end angle=240, x radius=1, y
radius=0.5];
\end{tikzpicture}
```



可以通过平移来获得围绕圆心画椭圆。

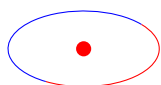
```
\begin{tikzpicture}
\fill [red] (2,0) circle (0.1);
\draw [blue] [shift={({1+cos(40)},{0.5*sin(40)})}] (1,0) arc [start angle=40, end
angle=240, x radius=1, y radius=0.5];
\draw [red] [shift={({1+cos(-120)},{0.5*sin(-120)})}] (1,0) arc [start angle=-120,
end angle=40, x radius=1, y radius=0.5];
\end{tikzpicture}
```

```
\end{tikzpicture}
```



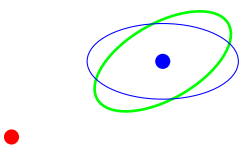
如果采用椭圆极坐标，结合 calc 库，就不需要这么麻烦地计算平移量了。

```
\begin{tikzpicture}
\fill [red] (2,0) circle (0.1);
\draw [blue] ($(2,0)+(40:1 and 0.5)$) arc [start angle=40, end angle=240, x
radius=1, y radius=0.5];
\draw [red] ($(2,0)+(-120:1 and 0.5)$) arc [start angle=-120, end angle=40, x
radius=1, y radius=0.5];
\end{tikzpicture}
```



scope 环境也可以用来做平移和旋转，注意顺序，先画图，然后写到后面的平移先操作，最后围绕椭圆心（此时已经平移到了 (2,1) 点）旋转 30 度。使用 xshift 和 yshift 时必须加单位。使用 shift=(x,y) 时必须加花括号，不加单位默认是厘米，加单位则为指定单位。

```
\begin{tikzpicture}
\fill [red] (0,0) circle (0.1);
\begin{scope}[color=green, line width=1pt, rotate around={30:(2,1)},
xshift=2cm,yshift=1cm]
\draw (0,0) ellipse (1 and 0.5);
\end{scope}
\fill [blue] (2,1) circle (0.1);
\draw [blue] (2,1) ellipse (1 and 0.5);
\end{tikzpicture}
```



所有的弧线都可以加箭头。

```
\begin{tikzpicture}
\draw [-latex] (0,0) arc [start angle=30, end
angle=120, radius=1];
\end{tikzpicture}
```



4.3 曲线

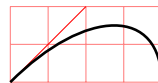
4.3.1 任意曲线 controls 方式

语句 controls 的设计理念很不错，但是实际并不好用。controls 的用法如下所示，controls 后面的第一个坐标与前面的坐标（即该段曲线的起点坐标）构成该段曲线起点处的切线，第二

个坐标与后面的坐标（即该段曲线的终点坐标）构成该段曲线终点处的切线。所以使用 `controls` 语句需要计算构成切线的坐标值。

```
\begin{tikzpicture}
\draw [step=0.5,red!50] (0,0) grid (2,1);

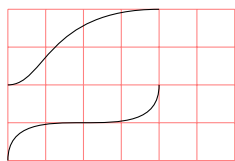
\draw [red] (0,0)--(1,1);
\draw [red] (2,1)--(2,0);
\draw [line width=1pt] (0,0) .. controls (1,1) and (2,1) .. (2,0);
\end{tikzpicture}
```



或者用 `left`、`right`、`up`、`down` 方式，固定四个方向。

```
\begin{tikzpicture}
\draw [step=0.5,red!50] (0,-1) grid (3,1);

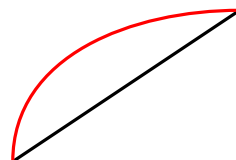
\draw (0,0) .. controls +(right:0.5) and +(left:1.5) .. (2,1);
\draw (0,-1) .. controls +(up:1) and +(down:1) .. (2,0);
\end{tikzpicture}
```



4.3.2 任意曲线 `bend` 方式

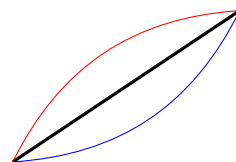
`bend` 方式定义的是出角和入角，即切向方向的角度。这比 `controls` 语句简单多了。

```
\begin{tikzpicture}[very thick]
\draw (0,0) to (3,2);
\draw [red] (0,0) to [out=90,in=180] (3,2);
\end{tikzpicture}
```



也可以 `bend` 左右，但是没有上下。

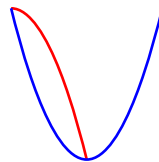
```
\begin{tikzpicture}
\draw [very thick] (0,0) to (3,2);
\draw [red] (0,0) to [bend left] (3,2);
\draw [blue] (0,0) to [bend right] (3,2);
\end{tikzpicture}
```



4.3.3 抛物线

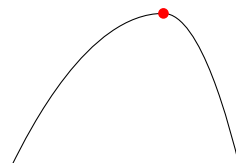
比较常用的曲线形状是抛物线，默认是起点弯曲的（二次求导为 0 的点），可以修改为终点弯曲。


```
\begin{tikzpicture}
\draw [red,line width=1pt] (-1,2) parabola (0,0);
\draw [blue,line width=1pt] (-1,2) parabola [bend at end] (0,0)
parabola (1,2);
\end{tikzpicture}
```



或者在设定的中间点弯曲，中间点两边的抛物线可以不对称。

```
\begin{tikzpicture}
\draw (0,0) parabola bend (2,2) (3,0);
\fill [red] (2,2) circle (2pt);
\end{tikzpicture}
```



4.3.4 正弦余弦曲线

正弦余弦曲线不是很好用，偶尔需要单独画一小段时使用。

```
\begin{tikzpicture}
\draw (0,0)--(1.57,1);
\draw [color=red,thick] (0,0) sin (1.57,1);
\end{tikzpicture}
```



完整的一条曲线很麻烦，下面有更容易的方法。

```
\begin{tikzpicture}
\draw [x=1.57cm,y=1cm] (0,0) sin (1,1) cos (2,0) sin (3,-1) cos (4,0);
\end{tikzpicture}
```

4.3.5 plot 语句

plot 用来函数画图。最简单的是将一系列点连成线。

```
\begin{tikzpicture}
\draw plot coordinates {(0,0) (1,0) (1,1) (2,1)};
\end{tikzpicture}
```



函数画图，画图最常用的参数有，

- 设定画图区间，[domain=画图区间初值:画图区间终值]，
- [smooth]，设定平滑曲线，这个参数推荐加，否则就是逐点画图，如果希望显示平滑的曲线，就必须加大取样点。
- [samples=取样点数目]。默认 25 个取样点，当加了 smooth 参数后，仍然不足，就需要增大取样点。
- [variable=\t]，默认变量是\textit{x}。有时自己命令一个变量名代码更容易懂。
- [samples at={分立值, 逗号分隔}]，这种时候使用\foreach语句就可以。

使用函数时或表达式太长时都需要花括号括起来。图形的坐标不一定是 $(\textit{x}, f(\textit{x}))$ ，还可以是 $(g(\textit{x}), f(\textit{x}))$ 。三角函数时， r 表示弧度。下面的例子中红色线代码定义了\textit{t}做变量。

```
\begin{tikzpicture}
\draw [line width=1pt,color=blue] [domain=3:5,smooth, samples=100] plot
(\x,{(\x-4)*(\x-4)});
\draw [line width=1pt,color=red] [domain=-2.5:2.5,smooth, samples=100,variable=\t]
plot (\t,{exp(-\t*\t)*cos(16*\t r)});
\end{tikzpicture}
```



plot语句中，\draw语句的参数都可以使用，所以可以填充。非闭合曲线，填充是按终点与起点相连围成的区域进行填充，因为是一个画图语句，所以此时填充满足奇偶原则。

tikz 内置了一些常用函数，还内置了随机数。rand 表示 $[-1, +1]$ 区间的伪随机数，rnd 为 $[0, 1]$ 区间。代码有变动，再编译会产生新的随机数。

```
\begin{tikzpicture}
\draw plot [only marks, mark=ball, mark size=1.5, domain=-1:1, samples=20] (\x,rand
*1.2);
\end{tikzpicture}
```

4.4 一些功能

4.4.1 标识角度 angles 库

因为我们经常会遇到画两条直线之间夹角的情况，所以有一个专门的库 angles 做这个事情。

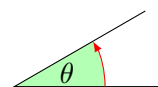
```
\usetikzlibrary{angles}
```

如果给角度加文字注释，还需要 quotes 库。

```
\usetikzlibrary{quotes}
```

文字注释需要双引号括起来。

```
\begin{tikzpicture}
\coordinate (A) at (2,0);
\coordinate (B) at (0,0);
\coordinate (C) at (30:2);
\draw (A)--(B)--(C) pic [draw=red,-latex,fill=green!30,angle
radius=12mm,"$\theta$"] {angle=A--B--C};
\end{tikzpicture}
```



还可以在角度标签后面用花括号加参数，比如换个颜色，换个字体大小。

```
\draw (A)--(B)--(C) pic [draw=red,-latex,fill=green!30,angle radius=12mm, "$\theta$"{
blue,font=\tiny}] {angle=A--B--C};
```

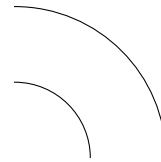
或者用 shift 换个位置，当不采用默认单位厘米时，需要指定单位。shift 参数也可以分开写为 {xshift=7mm, yshift=2mm}

```
\draw (A)--(B)--(C) pic [draw=red,-latex,fill=green!30,angle radius=8mm,"$\theta$"{
shift={({6mm,1.5mm})}] {angle=A--B--C};
```

4.4.2 剪切 clip

clip 语句可以设置一个闭合曲线框，超出该区域的部分会被剪切掉。下面的例子，最后得到的图形大小是剪切框的大小，而不是圆的大小。

```
\begin{tikzpicture}
\clip (0,0) rectangle (2,2);
\draw (0,0) circle (1);
\draw (0,0) circle (2);
\end{tikzpicture}
```

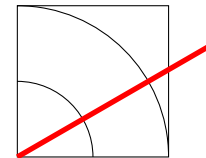


如果加参数 [draw] 会显示这个剪切框。

```
\clip [draw] (0,0) rectangle (2,2);
```

将剪切放在 scope 里，只在 scope 里做剪切，不影响下面的结果，所以红色线不会被剪切。

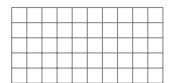
```
\begin{tikzpicture}
\begin{scope}
\clip [draw] (0,0) rectangle (2,2);
\draw (0,0) circle (1);
\draw (0,0) circle (2);
\end{scope}
\draw [red,line width=2pt] (0,0)--(30:3);
\end{tikzpicture}
```



4.4.3 网格 grid

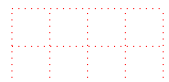
打网格的语句是 grid, help lines 是默认的辅助线 style 名，默认设置是 very thin 的灰色线。

```
\begin{tikzpicture}
\draw [step=0.2, help lines] (0,0) grid (2,1);
\end{tikzpicture}
```



可以直接设置颜色和线型等参数。

```
\begin{tikzpicture}
\draw [step=0.5, red, dotted] (0,0) grid (2,1);
\end{tikzpicture}
```



4.5 循环 foreach

foreach 语句用来重复画图，变量可以自己任意命名，下个例子是 `\x`。取值范围写在花括号里。如果是等间距的一系列值，可以省略中间结果，写三个值就行，间距是第二个值减去第一个值，语法是 {初值, 第二个值, ..., 终值}。终值不需要精确计算，超过后不会画图。如果是不等间距的值则需要全部列举。列举和等间距可以混用，按语法规则来就行。注意 foreach 语句取值

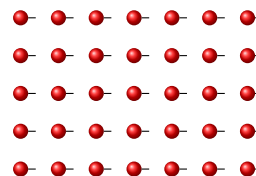
范围后面没有分号，因为这句不是画图语句，没有完结。下面的画图语句，单行语句可以省略花括号，多行语句必须加花括号。

```
\begin{tikzpicture}
\foreach \x in {0,1,...,6}
{
\fill [fill=red] (\x,0) circle (0.3);
\fill [fill=blue] (\x,0) circle (0.1);
}
\end{tikzpicture}
```



可以是多个变量。

```
\begin{tikzpicture}
\foreach \x in {0,0.5,...,3}
\foreach \y in {0,0.5,...,2}
{
\shade [ball color=red] (\x,\y) circle (0.1);
\draw (\x+0.1,\y)--(\x+0.2,\y);
}
\end{tikzpicture}
```



也可以是变量配对。

```
\begin{tikzpicture}
\foreach \a/\c in {20/red,40/blue,60/brown}
\draw [\c] (0,0)--(\a:1);
\end{tikzpicture}
```



foreach 语句在画晶体结构这样重复性高的情况非常有用。

4.6 渐变色和混色

4.6.1 渐变 shade

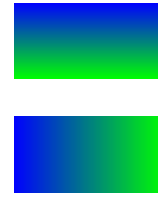
\shade 用来做渐变色效果。默认渐变色为黑白，默认渐变方向是上下渐变。

```
\begin{tikzpicture}
\shade (0,0) circle (1);
\end{tikzpicture}
```



可以设置为上下渐变色或左右渐变色。

```
\begin{tikzpicture}
\shade [top color=blue,bottom color=green] (0,0)
rectangle (2,1);
\shade [left color=blue,right color=green] (0,-1.5)
rectangle ++(2,1);
\end{tikzpicture}
```



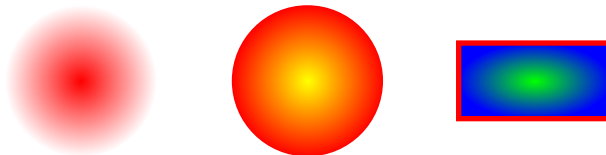
可以定义中间颜色，但是顺序不能改变。

```
\begin{tikzpicture}
\shade [top color=white,bottom color=white,middle
color=red] (0,0) rectangle (2,0.4);
\end{tikzpicture}
```



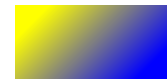
或者按径向渐变，如果没有设置 outer color，默认是白色。

```
\begin{tikzpicture}
\shade [shading=radial, inner color=red] (0,0) circle (1);
\shade [shading=radial, inner color=yellow, outer color=red] (3,0) circle (1);
\shade [inner color=green, outer color=blue, draw=red, line width=2pt] (5,-0.5)
rectangle ++(2,1);
\end{tikzpicture}
```



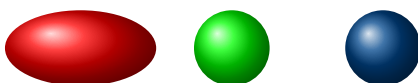
还可以设置角度使得渐变色方向倾斜

```
\begin{tikzpicture}
\shade [shading=axis, left color=yellow, right color=blue, shading
angle=45] (0,-2) rectangle ++(2,1);
\end{tikzpicture}
```



tikz 的 ball color 非常漂亮，有三维立体感。

```
\begin{tikzpicture}
\shade [ball color=red] (0,0) ellipse (1 and 0.5);
\shade [ball color=green] (2,0) circle (0.5);
\definecolor{tjublue}{RGB}{0,70,140}
\shade [ball color=tjublue] (4,0) circle (0.5);
\end{tikzpicture}
```



`\pgfdeclareverticalshading` 语句可以自定义 `\shading` 的方式，例如彩虹色。两边的颜色需要设定一个范围是同一种颜色，否则立刻就会渐变，反而看不出来设定的颜色了。

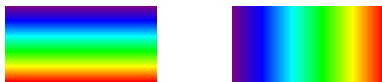
%设置彩虹色，rainbow是自己起的名字，里面的数值可变

```
\pgfdeclareverticalshading{rainbow}{100bp}{color(0bp)=(red); color(25bp)=(red);
color(35bp)=(yellow); color(45bp)=(green); color(55bp)=(cyan); color(65bp)=(blue);
color(75bp)=(violet); color(100bp)=(violet)}
```

%使用彩虹色

```
\begin{tikzpicture}
\shade[shading=rainbow] (0,0) rectangle (2,1);

\shade[shading=rainbow,shading angle=90] (3,0) rectangle +(2,1);
\end{tikzpicture}
```



以上都不需要额外的库。shadings 库定义了颜色轮，颜色轮的代码很复杂。

```
\usetikzlibrary{shadings}
```

定义了三种颜色轮，color wheel, color wheel white center, color wheel black center，颜色环是通过奇偶混色原则做成的。

```
\begin{tikzpicture}
\shade [shading=color wheel] (0,0) circle (1);
\shade [shading=color wheel white center] (3,0) circle (1);
\shade [shading=color wheel,even odd rule] (6,0) circle (1) (6,0) circle (0.5);
\end{tikzpicture}
```

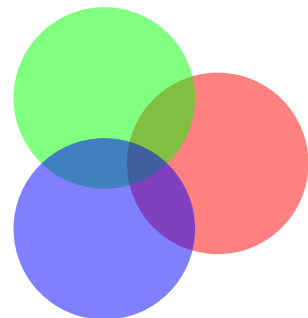


颜色轮的代码计算量很大，如果想做多周期色彩环，可以用 pgfplots 包的函数画图方式做。

4.6.2 透明 transparent

通过 opacity 参数设置填充色的透明度。

```
\begin{tikzpicture}[opacity=0.5]
\fill [red] (0:1cm) circle (1.2);
\fill [green] (120:1cm) circle (1.2);
\fill [blue] (-120:1cm) circle (1.2);
\end{tikzpicture}
```



透明的效果是浅色，但是浅色不等于透明，画光路图时可以用透明来显示透明、分束器的效果。

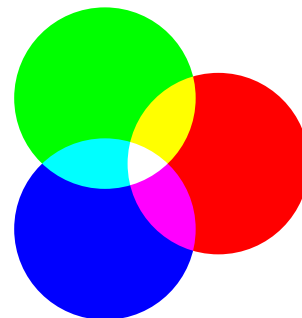
```
\begin{tikzpicture}
\draw [red,line width=2pt] (0,0)--(2.5,0);
\fill [cyan!25] (1,0) ellipse (0.08 and 0.4);
\fill [cyan,opacity=0.3] (2,0) ellipse (0.08 and 0.4);
\end{tikzpicture}
```



4.6.3 光学混色 blend

光学三原色为红绿蓝，三色混合是白色，定义 `blend group=screen` 可以获得光学三原色的混色效果。

```
\begin{tikzpicture}[blend group=screen]
\fill [red] (0:1cm) circle (1.2);
\fill [green] (120:1cm) circle (1.2);
\fill [blue] (-120:1cm) circle (1.2);
\end{tikzpicture}
```



`blend group` 参数还有多个参数，其他效果可以自行练习或参考手册 23.3 节。

4.6.4 奇偶原则

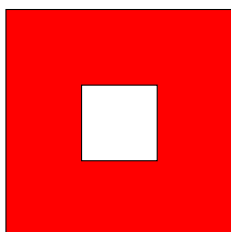
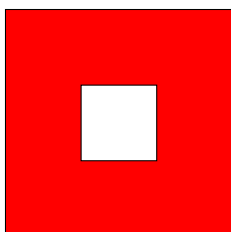
tikz 的画图原则是覆盖，先画的在下面，后画的在上面。如果两个图案写在一句内，可以定义奇偶原则。奇偶原则：同向的（比如都是逆时针或顺时针）图案，偶数为无填充，奇数为有填充。

```
\begin{tikzpicture}
\fill [fill=red,even odd rule] (0,0) circle (1) (1,0) circle (1)
(0.5,0) circle (0.2);
\end{tikzpicture}
```



顺时针图形和逆时针图形默认是 `nonzero rule`，就是一个是顺指针一个是逆时针，重叠部分是无填充。圆和矩形是同向的，所以需要加奇偶原则才能起作用。

```
\begin{tikzpicture}
\fill [draw=black,fill=red] (0,0)--(0,1)--(1,1)--(1,0)--cycle
(-1,-1)---+(3,0)---+(0,3)---+(-3,0)--cycle;
\fill [draw=black,fill=red] (4,0)---+(1,0)---+(0,1)---+(-1,0)--cycle
(3,-1)---+(0,3)---+(3,0)---+(0,-3)--cycle;
\end{tikzpicture}
```



4.7 分层

tikz 画图默认是先画的在下面，后画的在上面。tikz 可以很方便定义层和层顺序，打破原来的覆盖原则。

```
\pgfdeclarelayer{层名称A}
\pgfdeclarelayer{层名称B}
\pgfsetlayers{层顺序, 从底部到上面的顺序}
\begin{pgfonlayer}{层名称A}
环境内画图语句的结果在A层
\end{pgfonlayer}
```

下面的例子定义了五层，规定层顺序为 ABCDE，A 在最下面，E 在最上面，虽然先画的是 E 的蓝色线，但是最后呈现的是 E 的蓝色线。

```
\begin{tikzpicture}
\pgfdeclarelayer{A}
\pgfdeclarelayer{B}
\pgfdeclarelayer{C}
\pgfdeclarelayer{D}
\pgfdeclarelayer{E}
\pgfsetlayers{A,B,C,D,E}

\begin{pgfonlayer}{E}
\draw [blue,line width=2pt] (0,0)--(2,0);
\end{pgfonlayer}

\begin{pgfonlayer}{D}
\draw [cyan,line width=2pt] (0,0)--(2,0);
\end{pgfonlayer}

\begin{pgfonlayer}{C}
\draw [green,line width=2pt] (0,0)--(2,0);
\end{pgfonlayer}

\begin{pgfonlayer}{B}
\draw [red,line width=2pt] (0,0)--(2,0);
\end{pgfonlayer}

\begin{pgfonlayer}{A}
\draw [black,line width=2pt] (0,0)--(2,0);
\end{pgfonlayer}
\end{tikzpicture}
```

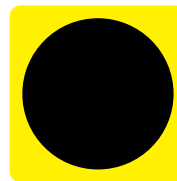
通过自动获取坐标的方式画图很方便，但是必须在画图语句之前先获得坐标。这导致坐标定义顺序和画图层叠顺序之间有可能产生矛盾。设定层和层的顺序可以很好的解决这个问题。

可以加载 backgrounds 库。

```
\usetikzlibrary{backgrounds}
```

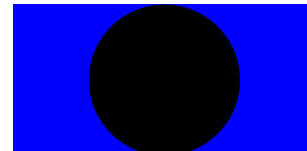

然后使用自动扩展的背景框和颜色, framed 和 show background rectangle, 写哪一个都可以。

```
\begin{tikzpicture}[background rectangle/.style={rounded
corners,fill=yellow},framed,show background rectangle]
\fill (0,0) circle (1);
\end{tikzpicture}
```



以及直接使用定义好的 background layer

```
\begin{tikzpicture}
\fill (0,0) circle (1);
\begin{scope}[on background layer]
\fill[blue] (-2,-1) rectangle ++(4,2);
\end{scope}
\end{tikzpicture}
```



4.8 数学库 math

加载数学库 math。

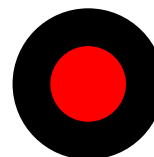
```
\usetikzlibrary{math}

\tikzmath{每条语句结尾都有分号}
```

`\tikzmath{}`里面不能有空行。

数学库 math 可以定义变量, 给变量赋值, 这样很多时候多次重复的长度、半径等画图参数可以通过修改变量赋值一次性修改。

```
\begin{tikzpicture}[>=Latex,x=5mm,y=5mm]
\tikzmath{
\x=1;
\c="red"; %必须加引号
}
\fill (0,0) circle (\x+1);
\fill [\c] (0,0) circle (\x);
\end{tikzpicture}
```

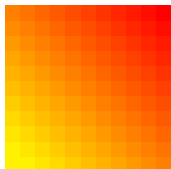


或者使用 `let \c=red;` 也可以, 不用加引号。

math 库有一些规则需要测试才能了解, 例如书上的例子换了下面的两种方法做, 里面不能空行。第一种是直接定义变量, 画图语句放在循环里。

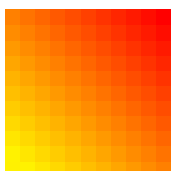
```
\begin{tikzpicture}[>=Latex,x=2mm,y=2mm]
\tikzmath{
int \f,\i,\j; %定义为整数
for \i in {0,...,10} {
for \j in {0,...,10} {
\f=(\i+\j)*5;
{
\fill [red!\f!yellow] (\i,\j) rectangle ++(1, 1);
}
}
}
}
```

```
}; %tikzmath里的画图语句需要用花括号括起来，可以多行语句，可空行，需加分号
}; %循环的结束，不能加空行
}; %循环的结束，不能加空行
} %结束tikzmath，不加分号
\end{tikzpicture}
```



第二种是定义函数，在循环里调用函数，画图语句放在循环里。

```
\begin{tikzpicture}[>=Latex,x=2mm,y=2mm]
\tikzmath{
function a(\i,\j){
return (\i+\j)*5;
}; %这里不能空行
int \f,\i,\j; %定义为整数
for \i in {0,...,10} {
for \j in {0,...,10} {
\f=a(\i,\j);
{
\fill [red!\f!yellow] (\i,\j) rectangle ++(1, 1);
}; %tikzmath里的画图语句需要用花括号括起来，可以多行语句，可空行，需加分号
}; %循环的结束，不能加空行
}; %循环的结束，不能加空行
} %结束tikzmath，不加分号
\end{tikzpicture}
```

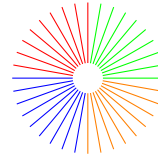


条件语句的用法是

```

\begin{tikzpicture}[>=Latex]
\tikzmath
{
  int \a;
  for \a in {0,10,...,350} {
    if \a>260 then {let \c=orange;} else{
    if \a>170 then {let \c=blue;} else{
    if \a>80 then {let \c=red;} else{
    let \c=green;}; %结束条件
  }; %结束条件
  }; %结束条件
  {
    \draw [\c] (\a:0.2)--(\a:1);
  }; %结束画图
  }; %结束循环
}
\end{tikzpicture}

```



```

\begin{tikzpicture}[>=Latex]
\tikzmath
{
  coordinate \a; %定义变量为坐标变量
  \a=(1,0)+(0,1); %直接坐标变量相加，不需要美元符号，必须有calc库
  {
    \fill (0,0) circle (2pt);
    \fill [red] (\a) circle (2pt);
  };
}
\end{tikzpicture}

```



数学库的功能还有很多，可以定义函数，有循环和判断语句，所以可以像其他语言那样编程画图，一些有规则的复杂图案可以用少量语句完成。

4.9 node

tikz 的 node 不仅仅是文本框，node 非常强大好用。

4.9.1 基本功能

最基本的用法是在某点放置文本框，坐标值默认情况是文本框中心点位置。默认是无边框白色填充。可以设置文本框的边框颜色、填充色、字体等。还可以给 node 起名，然后调用内置的 node 坐标。基本功能这部分，主要是讲述 node 的参数。

node 的位置参数

在指定位置放置 node 的几种常用方法如下

- 不加任何位置参数，node 中心位置在指定坐标点上

```
\begin{tikzpicture}
\node at (0,0) {$x_1$};
\node at (1,0) {$x_2$};
\draw [|-|] (0,-0.4)--++(1,0);
\end{tikzpicture}
```

x_1 x_2
|-----|

- 在指定坐标点的上[above]、下[below]、左[left]、右[right]、左上[above left]、右上[above right]、左下[below left]、右下[below right]放置 node。

```
\begin{tikzpicture}
\node at (0,0) [below right,draw] {$x_1$};
\fill [red] (0,0) circle (2pt);
\end{tikzpicture}
```

x_1

- 定义了八个 anchor 方位，东南西北、东北、西北、东南、西南。这个用法跟上面的用法不同，指定坐标点在 node 的这几个方位。

```
\begin{tikzpicture}
\draw (0,0) node [anchor=north east ,fill=yellow] {(0,0)点在文字框的东北角}
rectangle (2,1) node [anchor=south west, fill=green] {(2,1)点在文字框的西南角};
\end{tikzpicture}
```

(2,1) 点在文字框的西南角

(0,0) 点在文字框的东北角

- 直线画图时直接放在点坐标之后使用，不用加at，

```
\begin{tikzpicture}
\draw (0,0) node [left] {$x_1$} --++(1,0) node
[right] {$x_2$};
\end{tikzpicture}
```

x_1 ——— x_2

从这里可以看出，文本框的文字之外有留白，这个参数是 inner sep，初始值为 0.3333em。下面会讲到。

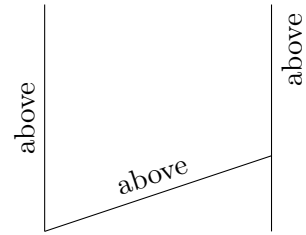
- 沿直线放置 node，node 语句放在直线语句--之后，终点坐标之前。可以用 pos 调整位置，可以联用几个 node，pos 的值可以超出直线范围。如果不指定 pos 的值，默认是两个坐标点的连线中点位置。默认竖直位置在连线中间，为了看清楚，一般会采用直线上或下的位置，以及填充白色。

```
\begin{tikzpicture}
\draw (0,0) -- node [above] {m} node [pos=0.1,above] {s} node [pos=1.2,above]
{x} ++(4,0);
\end{tikzpicture}
```

s m x

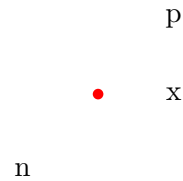
- 斜线时可以使用sloped参数，完全沿斜线方向放置文字，两个参数前后位置不影响结果。

```
\begin{tikzpicture}
\draw (0,0) -- node [above,sloped] {above} (3,1);
\draw (0,0) -- node [sloped,above] {above} (0,3);
\draw (3,0) -- node [pos=0.8,sloped,below] {above}
++(0,3);
\end{tikzpicture}
```



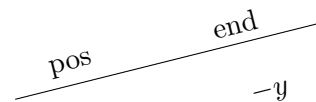
- 可以加平移参数，使得 node 偏离设定位置，xshift和yshift方法必须加单位，[shift={ (坐标) }]则必须加花括号。

```
\begin{tikzpicture}
\node at (0,0) [xshift=1cm] {x}; %必须加单位
\node at (0,0) [xshift=1cm,yshift=1cm] {p};
\node at (0,0) [shift={(-1,-1)}] {n}; %必须加花括号
\fill [red] (0,0) circle (2pt);
\end{tikzpicture}
```



如果采用了sloped参数，xshift是沿线方向，朝向起点移动为负方向，终点为正方向，与pos 结果类似。yshift是沿垂直于直线的方向。平移参数是 node 先放置好完成后再平移。

```
\begin{tikzpicture}
\draw (0,0) -- node [pos=0.2,above,sloped] {pos}
node [xshift=1cm,above,sloped] {end} node
[pos=0.8,yshift=-1cm,above,sloped] {$-y$} (4,1);
\end{tikzpicture}
```

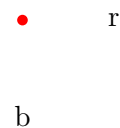


- 可以添加 positioning 子库

```
\usetikzlibrary{positioning}
```

然后使用设置好的平移语句

```
\begin{tikzpicture}
\coordinate (o) at (0,0);
\node at (0,0) [below=1cm of o] {b};
\node at (0,0) [right=1cm of o] {r};
\fill [red] (o) circle (2pt);
\end{tikzpicture}
```



笔者喜欢直接平移。

node 的旋转参数

可以用 rotate 参数旋转文字，旋转是绕 node 的坐标点旋转。

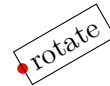
- 没有任何位置参数时，旋转最好理解

```
\begin{tikzpicture}
\fill [red] (0,0) circle (2pt);
\node at (0,0) [draw,rotate=30] {rotate};
\end{tikzpicture}
```



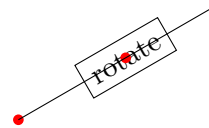
- 有位置参数时，放置好 node 后，绕设定的坐标点旋转

```
\begin{tikzpicture}
\fill [red] (0,0) circle (2pt);
\node at (0,0) [draw,right,rotate=30] {rotate};
\end{tikzpicture}
```



有平移时也一样，但是平移和旋转都是在 node 放置后完成后的操作，它们之间就存在先后顺序的区别了。放在后面的先操作，下面的例子先平移后旋转，30 度在 (0,0) 与 node 中心的连线上

```
\begin{tikzpicture}
\fill [red] (0,0) circle (2pt);
\node (a) at (0,0)
[draw,right,rotate=30,xshift=1cm] {rotate};
\fill [red] (a.center) circle (2pt);
\draw (0,0)--++(30:3);
\end{tikzpicture}
```



下面的例子先旋转后平移

```
\begin{tikzpicture}
\fill [red] (0,0) circle (2pt);
\node (a) at (0,0)
[draw,right,xshift=1cm,rotate=30] {rotate};
\fill [red] (a.west) circle (2pt);
\draw [cyan,dashed] (0,0)--++(1,0);
\end{tikzpicture}
```



- node 的旋转也支持[rotate around={角度:(坐标点)}]，可以绕指定坐标点旋转。

node 的字体参数

默认字体是直体，

- 修改字体的大小、字体设置等使用[node font={字体设置语句}]

```
\begin{tikzpicture}
\node at (0,0) [node font={\bfseries, \itshape,
\Large}] {rotate};
\end{tikzpicture}
```

rotate

- 修改文字颜色使用[text=颜色名]，必须是颜色名，不能是颜色命令。

```
\begin{tikzpicture}
\node at (0,0) [text=blue,node font={\bfseries,
\itshape, \Large}] {rotate};
\end{tikzpicture}
```

rotate

这个参数可以最保险地设置文字颜色。

- 放置行间公式时，颜色参数也起作用。

```
\begin{tikzpicture}
\node at (0,0) [text=red] {$\displaystyle\int_0^a
f(x)dx$};
\end{tikzpicture}
```

$$\int_0^a f(x)dx$$

node 的颜色参数

node 可以设置边框的颜色、填充的颜色、文字的颜色、渐变色等。

- 设置边框的颜色，[draw=颜色]，还可以添加线的参数。[draw]不写颜色，默认是显示边框、颜色为黑色。

```
\begin{tikzpicture}
\node at (0,0) [draw=red,very thick] {rotate};
\end{tikzpicture}
```

rotate

仅仅设置线的参数，没有[draw]参数，不会显示边框，相当于不起作用。node 默认不画边框，只有添加了该参数才会显示边框。所以仅仅设置线参数无效。

- 设置填充的颜色，[fill=颜色]

```
\begin{tikzpicture}
\node at (0,0) [fill=yellow] {rotate};
\end{tikzpicture}
```

rotate

- 设置文字的颜色，[text=颜色]

```
\begin{tikzpicture}
\node at (0,0) [text=blue] {rotate};
\end{tikzpicture}
```

rotate

- 设置全部颜色，边框和文字都改成蓝色。

```
\begin{tikzpicture}
\node at (0,0) [draw, blue] {rotate};
\end{tikzpicture}
```

rotate

[fill,颜色]，填充色和文字色一样，就看不到文字了。

- 关于设置颜色语句的一些说明。

如果放在直线内添加 node，不设置颜色，颜色跟随\draw语句的颜色

```
\begin{tikzpicture}
\draw [red] (0,0)-- node [draw,above] {rotate}
(4,0);
\end{tikzpicture}
```

rotate

如果直接修改颜色，边框和文字都修改为指定颜色。

```
\begin{tikzpicture}
\draw [red] (0,0)-- node [draw,blue,above]
{rotate} (4,0);
\end{tikzpicture}
```



如果希望设定边框、填充和文字都属于不同颜色，就只能逐一指定颜色，不能再直接写一个颜色参数，否则会强制所有颜色都是这个颜色。

```
\begin{tikzpicture}
\draw [red] (0,0)-- node
[draw=blue,fill=yellow,text=cyan,above] {rotate}
(4,0);
\end{tikzpicture}
```



node 的其他参数

■ 倾斜

倾斜和旋转不一样，倾斜参数使得 node 的方框变为平行四边形，而不再是矩形。只设置 `xslant`，上下边框仍然水平，文字也不旋转。只设置 `yslant`，左右边框仍然竖直，文字倾斜。两个都设置，边框两个方向都倾斜，文字也倾斜。

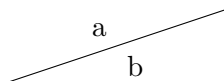
```
\begin{tikzpicture}
\node (a) at (-2,0) [draw,yslant=0.1] {TJUTJU};
\fill [red] (a.0) circle (2pt);
\fill [green] (a.north east) circle (2pt);
\end{tikzpicture}
```



■ 自动和镜像位置

可以设置 `[auto]`，在线（包括 `bend` 那样的曲线情况）中间放置 node 可以自动安排位置。使用 `[auto,swap]` 在相对的位置放置另一个文字，注意这个 `auto` 不能省略。

```
\begin{tikzpicture}
\draw (0,0)-- node [auto] {a} node [auto,swap] {b}
(3,1);
\end{tikzpicture}
```



笔者一般不这么用，还是喜欢自己调整位置。但是不精确布局的时候这个功能的确也很方便。

■ 镜像

可以设置 `[xscale=-1]`、`[yscale=-1]`、`[scale=-1]` 获得镜像效果。

```
\begin{tikzpicture}
\node at (0,0) [xscale=-1] {TJU JY};
\fill [red] (0,0) circle (2pt);
\end{tikzpicture}
```



镜像这个功能跟 node 与坐标的关系相关，比如 node 在坐标点上面，镜像后就变成 node 在坐标点下面，文字头朝下。手册的封面效果就是竖直方向镜像，镜像那个 node 还做了

倾斜和 shading 效果。

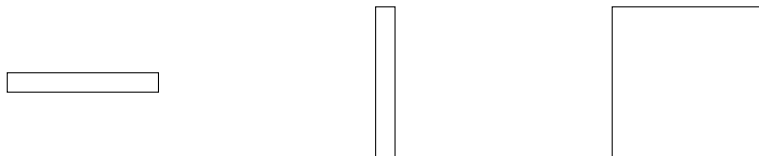
```
\begin{tikzpicture}
\node at (0,0) [yscale=-1,above] {TJU JY};
\fill [red] (0,0) circle (2pt);
\end{tikzpicture}
```



node 的尺寸参数和文字对齐方式

- 对齐方式的设定语句是[align=对齐方式]。一共有两组对齐方式，对应文本框最小宽度的三种对齐方式是左对齐left、右对齐right和居中对齐（默认）center。对应文字内容宽度的三种对齐方式是左对齐flush left（默认）、右对齐flush right和居中对齐flush center。如果不设定对齐方式align，node 里不能强制换行。
- 文本框的最小尺寸有三个参数，文本框的最小宽度minimum width、文本框的最小高度minimum height和文本框的最小尺寸minimum size（相当于宽高的最小值一起设定）。没有文字，文本框也会占据设定的最小空间。

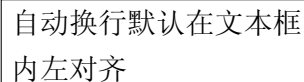
```
\begin{tikzpicture}
\node at (0,0) [draw,minimum width=2cm] {};
\node at (4,0) [draw,minimum height=2cm] {};
\node at (8,0) [draw,minimum size=2cm] {};
\end{tikzpicture}
```



这个尺寸是文本框占据的最小值，文字超出范围，文本框会自动扩大。

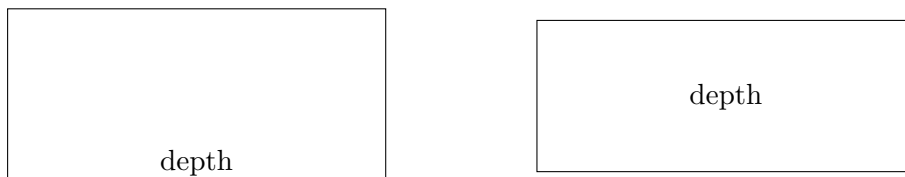
- 文字内容所占空间的大小有三个参数，文字内容的宽度text width、文字内容的高度text height和文字内容的深度text depth。设置了文本框的文字内容宽度后，这个设定会绝对遵守，如果文字内容长度超过设定宽度，文字会自动折行。设不设定文字高度，如果内容放不下，都会自动扩展。

```
\begin{tikzpicture}
\node at (0,0) [draw,text width=10em]
{自动换行默认在文本框内左对齐};
\end{tikzpicture}
```



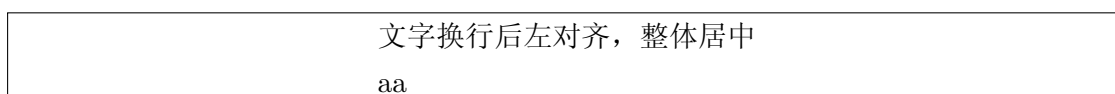
文字内容的深度只在设定了text height后才会生效，仅仅设定文本框的最小高度minimum height时不会生效。

```
\begin{tikzpicture}
\node [draw, minimum width=5cm, text height=2cm, text depth=0.5mm] at (0,0)
{depth};
\node [draw, minimum width=5cm, minimum height=2cm, text depth=0.5mm] at (7,0)
{depth};
\end{tikzpicture}
```



- 两组对齐方式的不同，本质上是两组尺寸设置的意义不同。对比下面的例子就可以看出区别，当设置了`minimum width`，如果文字内容小于设定值，设置对齐方式为左对齐，仅仅是多行文字的左对齐，整个文字内容是居中对齐的。

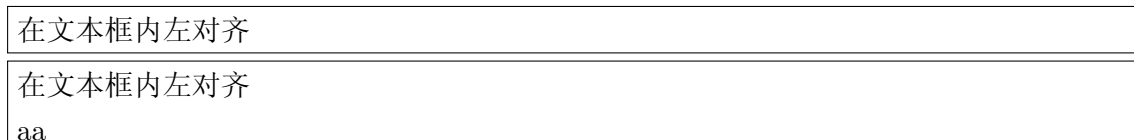
```
\begin{tikzpicture}
\node at (0,0) [draw,minimum width=\textwidth, align=left]
{文字换行后左对齐，整体居中\\aa};
\end{tikzpicture}
```



所以单行文字时，设定左对齐或右对齐，都仍然是居中对齐的效果。在设定最小宽度时，`[align]`设定的是换行后文字的对齐方式，不是文字在 `node` 里的对齐方式。即使因为最小宽度设置过大，看上去似乎有设定文字部分在文本框里的对齐方式的需求，但是这只是一个错觉。设置文本框最小宽度的意义本身就在于文字宽度方向上增加文本框到文字内容之间的间距。

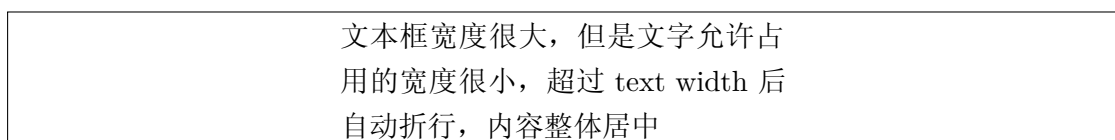
如果设定文字宽度，则单行时，默认是左对齐。多行当然也是默认结果或设定结果。

```
\begin{tikzpicture}
\node at (0,0) [draw,text width=\textwidth] {在文本框内左对齐};
\node at (0,-1) [draw,text width=\textwidth] {在文本框内左对齐\\aa};
\end{tikzpicture}
```



如果同时设定两个宽度，`minimum width`大于`text width`时（`minimum width`更小时不起作用），文字内容超过`text width`设定的宽度就会自动折行，默认是 `flush left`，文字内容左对齐，文字整体居中对齐。

```
\begin{tikzpicture}
\node at (0,0) [draw,minimum width=\textwidth, text width=6cm]
{文本框宽度很大，但是文字允许占用的宽度很小，超过text
width后自动折行，内容整体居中};
\end{tikzpicture}
```



- 文本框的边框和文字内容间的间距`inner sep`

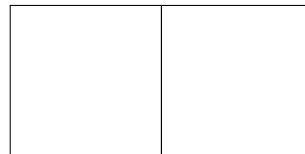
```
\begin{tikzpicture}
\node at (0,0) [draw] {A};
\node at (1,0) [draw,inner sep=2pt] {A};
\node at (2,0) [draw,inner sep=8pt] {A};
\end{tikzpicture}
```



用 node 画点时，直接修改 `inner sep` 获得点的大小变化。

如果没有文字时，设定文本框尺寸和设定文字内容尺寸，效果就差 `inner sep`，将其设为 0，两者才是等价的。

```
\begin{tikzpicture}
\node at (0,0) [draw,minimum size=2cm] {};
\node at (2,0) [draw,inner sep=0pt, text
width=2cm, text height=2cm] {};
\end{tikzpicture}
```



文本框尺寸永远等于文字内容的尺寸加上 `inner sep` 后的大小。没有设定尺寸或者设定尺寸太小，文字放不下都会自动调整。

- 插入图片后的文本框大小是按图片外围方框大小计算的，必须包含图片方框的对角线。如果 node 的形状改为圆形，因为要包含对角线，所以变得很大。

```
\begin{tikzpicture}
\node at (0,0) [fill=yellow,inner sep=0pt]
{\includegraphics[width=2cm]{fig/tju.pdf}};
\node at (4,0) [fill=yellow,inner sep=0pt,circle]
{\includegraphics[width=2cm]{fig/tju.pdf}};
\end{tikzpicture}
```



因为天津大学校徽的 pdf 图片是透明的（无背景），所以可以使用 `clip` 语句切边，制作填充了背景色的图片。

```
\begin{tikzpicture}
\clip (0,0) circle (1);
\node at (0,0) [fill=yellow] {\includegraphics[width=2cm]{fig/tju.pdf}};
\end{tikzpicture}
```



- 利用 `spy` 子库局域放大图片。

```
\usetikzlibrary{spy}
```

```
\begin{tikzpicture}[spy using
outlines={circle,magnification=4,size=2cm,connect
spies}]
\node at (0,0)
{\includegraphics[width=2cm]{fig/tju.pdf}};
\spy [red] on (0.5,0.7) in node at (2.5,0.7);
\end{tikzpicture}
```



也可以是方框，注意下面两个 size 意义不同，值相同是没有边缘。

```
\begin{tikzpicture}[spy using outlines={spy using
overlays={size=3cm},magnification=4,size=2.5cm}]
\node at (0,0) {\includegraphics[width=2cm]{fig/tju.pdf}};
\spy [cyan] on (0.5,0.7) in node at (3,0.7);
\end{tikzpicture}
```



4.9.2 命名 node 并使用 node 名坐标

我们可以命名（标记）node，

```
\node (name) at (x,y) {text};
```

或者在连线后的 node 那里直接起名

```
\begin{tikzpicture}
\draw (0,0)-- node (a) [above,draw] {} (4,0);
\fill [red] (a) circle (2pt);
\end{tikzpicture}
```



node 在连线的中点的上面，node 坐标代表的是 node 的中心点，而不是 node 放置的参考点。放置参考点与 node 位置的关系是很复杂的。连线上的点的坐标提取，一般不建议 node 形式，coordinate 方式更准确。

node 的名称可以被当作坐标使用。node 坐标最常用的有三种形式，分别是

- (name), 画图时，连线的方向是 node 中心决定的，但是起点是该方向上 node 边框交点。因此下面两个 node 连线，线段长度是边框点到边框点的距离，而不是两个 node 中心点之间的距离，这个设定非常实用。

```
\begin{tikzpicture}
\node (a) at (0,0) [draw,inner sep=3mm] {};
\node (b) at (2,0) [draw,inner sep=3mm] {};
\draw (a)--(b);
\end{tikzpicture}
```



- (name.内置坐标), 例如 anchor 和 center。

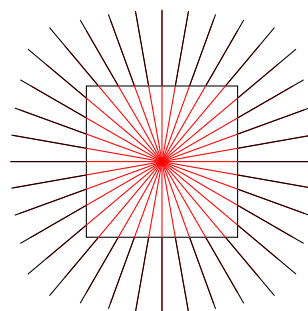
```
\begin{tikzpicture}
\node (a) at (0,0) [draw,inner sep=3mm] {};
\node (b) at (2,0) [draw,inner sep=3mm] {};
\draw (a.south east)--(b.center);
\end{tikzpicture}
```



node 的形状不同, 内置坐标也不同, 内置坐标点还是很丰富的, 具体名称可以查看手册。需要注意的是, name.center、name.mid、name.base 和 name.text 四个坐标是不一样的, 尤其是前三个存在细微差别。

- (name.angle), 角度的取值范围是 $[-360 : 360]$, 表示从 node 的中心画这个角度的线, 与边框相交的点。

```
\begin{tikzpicture}
\node (a) at (0,0) [draw,inner sep=1cm] {};
\foreach \a in {0,10,...,350}
{
\draw [red] (a.center)--++(\a:2);
\draw (a.\a)--++(\a:1);
}
\end{tikzpicture}
```



大部分情况下这种方式很好用, 但是如果纵横比很大或很小, 就会难以判断合适的角度值了, 如下所示

```
\begin{tikzpicture}
\node (a) at (0,0) [draw,minimum width=4cm,inner
sep=0.5mm] {};
\fill [red] (a.5) circle (2pt);
\end{tikzpicture}
```



使用 node 坐标很方便将公式加到图片里, 或者添加说明。这个例子相当于将数据图加载后, 在图上任意位置添加文字、箭头、数学公式等。对于发表文章时图内嵌入漂亮的公式或者其他特殊标记非常有用。

```
\begin{tikzpicture}
\node (tju) at (0,0) {\includegraphics[width=3cm]{fig/tju.pdf}};
\node [below=2mm of tju] {天津大学校徽}; %需要positioning库
\node [left=-1mm of tju] {\rotatebox{90}{天津大学校徽}};
\node at (tju.0) [below,rotate=90] {天津大学校徽};
\node at (tju.north east) [above left] {天津大学校徽};
\end{tikzpicture}
```

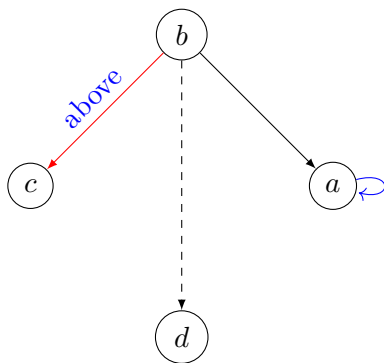


4.9.3 edge 操作和 quotes 子库

edge 操作一般结合 node 用，所以这里放在 node 坐标这一节了。但是 edge 本质上是一种 path 操作，这里的 node 只起到以 node 名为坐标的用处，直接写作用也可以使用 edge。edge 非常适合画复杂的关系图。

```
\begin{tikzpicture}
\node foreach \name/\angle in {a/0,b/90,c/180,d/270}
(\name) at (\angle:2) [circle,draw] {$\name$};

\draw [-latex] (b) edge (a)
edge [red] node [above,sloped,blue] {above} (c)
edge [dashed] (d)
(a) edge [blue,loop right] (a);
\end{tikzpicture}
```



edge 和 qutoes 子库

edge 还可以结合 quotes 库用，也很方便，注意第二个字符串后面还有一个单引号，表示该字符串放在下面。单引号和字符串的双引号之间可以空格也可以不空格，有空格容易识别。字符串样式如前面 quotes 库的例子。如果需要对放在下面的字符串修改样式，需要写成下面例子中的样式。表示放在下面的单引号、表示颜色的 blue 和表示放置位置的 near end 都是字符串"end"的参数，要一起放在花括号内。位置可以用 pos= 数值这样的样式，也很方便。

```
\begin{tikzpicture}
\draw (0,0) edge ["above"{red},"below" ', "start" near start, "end"{' ,blue, near
end}] (6,0);
\end{tikzpicture}
```

start	above
below	end

draw、to 和 edge 比较

edge 和 to 都可以代替\draw画线，也都支持在画线语句后、末端坐标前的位置添加 node。但是它们的使用有许多细节不同，edge 和 to 比\draw有更多功能。to 和 edge 都支持以出角和入角方式画曲线，也都支持 quotes 子库的标签功能。to 也支持上面的例子中的[loop right]这样的曲线。但是 edge 支持随时改变\draw参数，to 只支持改变角度，不支持改变颜色、线型等。这是因为 to 与--一样是一笔画线，从起始点到第二个点、第二个点到第三个点连续画线。而 edge 其实是分支画线，相当于一段一段的画线。edge 是从起始点，不断从起始点到其他点画线，edge 画的不是连续的线，而是树状的。edge 和 to 还有画线语句--可以联用，但是从第一个 edge 开始分支，并影响到第一个 to 或--，连续两个 to 或--时，edge 不影响第二个画线语句。edge 还会影响到\draw的箭头参数，因为 edge 是树状图，所以后面的点都是终点。当它们混用时，细微之处需要多练习才能体会。因为一条语句中，edge 可以开始新的起始点，所以 edge 可以做到沿线改颜色或画箭头很容易。

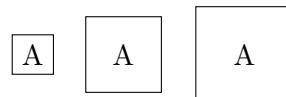
```
\begin{tikzpicture}[thick]
\draw (0,0) edge [red] (1,0) (1,0) edge [blue] (1,1)
(1,1) edge [cyan] (0,1) (0,1) edge [green] (0,0);
\end{tikzpicture}
```



4.9.4 node 的形状和 shapes.geometric

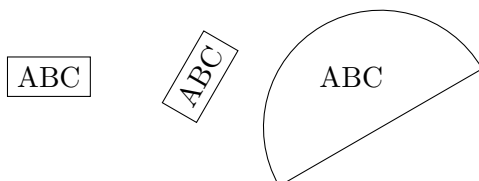
node 的通用参数为 text width, text height (不建议使用), minimum size, minimum width, minimum height, inner sep, inner xsep, inner ysep 等。

```
\begin{tikzpicture}
\node at (0,0) [draw] {A};
\node at (1.2,0) [minimum size=1cm,draw] {A};
\node at (2.8,0) [inner sep=0.5cm,draw] {A};
\end{tikzpicture}
```



node 可以旋转，一些形状可以只旋转边框，不旋转文字。

```
\begin{tikzpicture}
\node at (0,0) [draw] {ABC};
\node at (2,0) [draw,rotate=60] {ABC};
\node at (4,0) [draw,semicircle,shape border uses incircle,shape border rotate=30]
{ABC};
\end{tikzpicture}
```



node 默认边框形状是上面例子中的方框，内置了一些常用形状。不需要加载任何库就可以使用圆形。

```
\begin{tikzpicture}
\node at (0,0) [circle,fill=yellow] {B};
\end{tikzpicture}
```

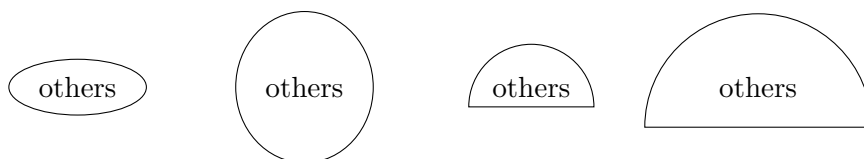
其他边框形状需要加载 shapes.geometric 库。

```
\usetikzlibrary{shapes.geometric}
```

然后可以使用椭圆 ellipse、半圆形 semicircle、菱形 diamond、梯形 trapezium、正多边形 regular polygon、星形 star、等边三角形 isosceles triangle、风筝 kite、飞镖 dart、扇形 circular sector、圆柱 cylinder。

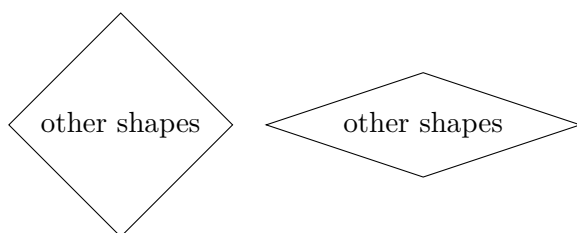
椭圆和半圆比较简单，没有特殊选项，可以用基础选项 minimum width 和 minimum height 改变纵横比。只设置一个有时候会等比例缩放。

```
\begin{tikzpicture}
\node at (0,0) [ellipse,draw] {others};
\node at (3,0) [ellipse,minimum height=2cm,draw] {others};
\node at (6,0) [semicircle,draw] {others};
\node at (9,0) [semicircle,minimum height=1.5cm,draw] {others};
\end{tikzpicture}
```



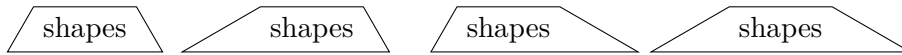
菱形 diamond 有 aspect 参数，用来调整纵横比。

```
\begin{tikzpicture}
\node at (0,0) [diamond,draw] {other shapes};
\node at (4,0) [diamond,aspect=3,draw] {other shapes};
\end{tikzpicture}
```



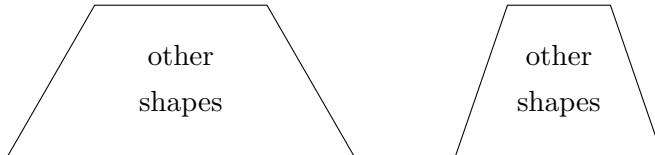
梯形参数比较多，可以设置左右角度和角度。

```
\begin{tikzpicture}
\node at (0,0) [trapezium,draw] {shapes};
\node at (3,0) [trapezium,trapezium left angle=30,draw] {shapes};
\node at (5.6,0) [trapezium,trapezium right angle=30,draw] {shapes};
\node at (9.2,0) [trapezium, trapezium angle=30,draw] {shapes};
\end{tikzpicture}
```

梯形有 `trapezium stretches=false/true` 选项, `false` 时梯形不改变基础比例形状, 相似地缩放。 `true` 时则可以拉伸。

```
\begin{tikzpicture}
\node at (0,0) [trapezium,minimum height=2cm,draw,align=center] {other\\shapes};
\node at (5,0) [trapezium,trapezium stretches=true,minimum
height=2cm,draw,align=center] {other\\shapes};
\end{tikzpicture}
```



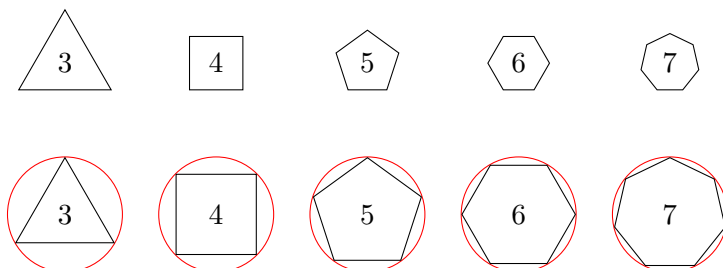
梯形还有 `trapezium stretches body=false/true` 选项, 横向拉伸时保持角度不变。

```
\begin{tikzpicture}
\node at (0,0) [trapezium,trapezium stretches=true,minimum
width=2cm,draw,align=center] {A};
\node at (5,0) [trapezium,trapezium stretches body=true,minimum
width=2cm,draw,align=center] {B};
\end{tikzpicture}
```



正多边形 `regular polygon` 有一个参数, `regular polygon sides= 整数`。没有文字时, 如果设置 `minimum width` 相等, 外接圆半径是相等的。有文字时, 如果文字所占空间比较大, 会自动扩展边框, 否则仍然是相等的。

```
\begin{tikzpicture}
\foreach \n in {3,...,7}
{
\node at (\n*2,0) [regular polygon, regular polygon sides=\n,draw] {\n};
\draw [red] (\n*2,-2) circle (0.76);
\node at (\n*2,-2) [regular polygon, regular polygon sides=\n,draw,minimum
width=1.5cm] {\n};
}
\end{tikzpicture}
```



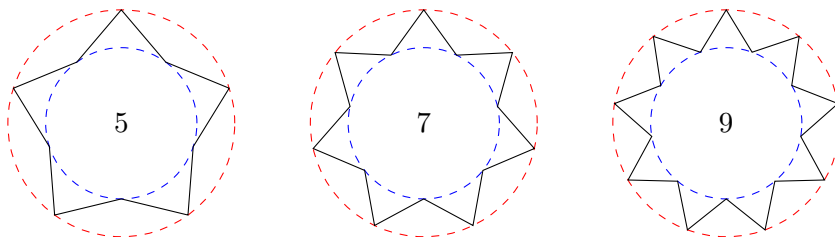
星形 `star` 的参数有 `star points`, `star point height`, `star point ratio`。

```
\begin{tikzpicture}
\foreach \n in {5,7,9}
\node at (\n,0) [star, star points=\n,draw] {\n};
\end{tikzpicture}
```



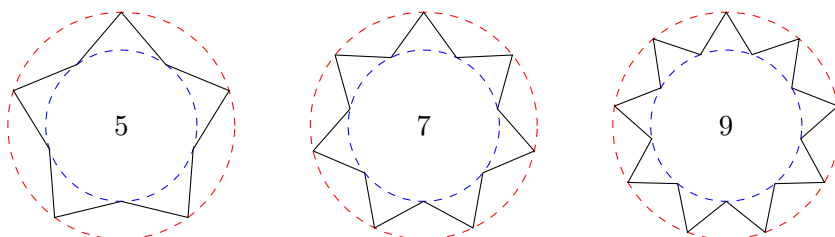
设置最小尺寸和 star point height，星形的外圆直径等于最小尺寸，内圆直径等于外圆半径减去 star point height。

```
\begin{tikzpicture}
\foreach \n in {5,7,9}
{
\draw [red, dashed] (\n*2,0) circle (1.5);
\draw [blue, dashed] (\n*2,0) circle (1);
\node at (\n*2,0) [star, star points=\n, star point height=0.5cm, minimum size=3cm,
draw] {\n};
}
\end{tikzpicture}
```



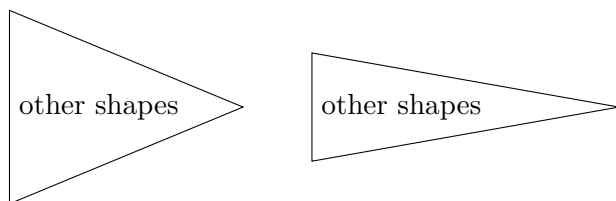
还可以设置 star point ratio，该比例即外圆半径比内圆半径，上面的例子该比例是 1.5。

```
\begin{tikzpicture}
\foreach \n in {5,7,9}
{
\draw [red, dashed] (\n*2,0) circle (1.5);
\draw [blue, dashed] (\n*2,0) circle (1);
\node at (\n*2,0) [star, star points=\n, star point ratio=1.5, minimum size=3cm,
draw] {\n};
}
\end{tikzpicture}
```



等边三角形的参数有顶角 isosceles triangle apex angle= 度数, 是否拉伸 isosceles triangle stretches=false/true,

```
\begin{tikzpicture}
\node at (0,0) [isosceles triangle,draw] {other shapes};
\node at (4,0) [isosceles triangle,isosceles triangle apex angle=20,draw] {other
shapes};
\end{tikzpicture}
```



风筝形 kite 参数有上顶角 kite upper vertex angle 和下顶角 kite lower vertex angle, 顶角 kite vertex angles, 即上下顶角相等情况, 这时类似 diamond。

```
\begin{tikzpicture}
\node at (0,0) [kite,draw] {kite};
\node at (5,0) [kite,kite upper vertex angle=120, kite lower vertex angle=30, draw]
{kite};
\end{tikzpicture}
```



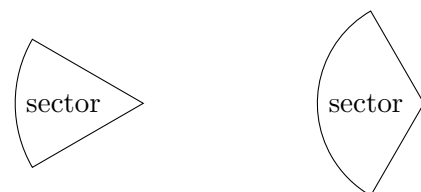
飞镖形 dart 有 dart tip angle 和 dart tail angle,

```
\begin{tikzpicture}
\node at (0,0) [dart,draw] {dart};
\node at (5,0) [dart,dart tip angle=30, dart tail angle=150, draw] {dart};
\end{tikzpicture}
```



扇形 circular sector 参数有 circular sector angle

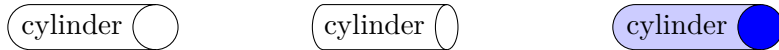
```
\begin{tikzpicture}
\node at (0,0) [circular sector,draw] {sector};
\node at (4,0) [circular sector, circular sector angle=120,draw] {sector};
\end{tikzpicture}
```



圆柱形 cylinder 参数有纵横比 aspect, 填充 cylinder uses custom fill=false/true, cylinder

end fill 和 cylinder body fill。

```
\begin{tikzpicture}
\node at (0,0) [cylinder,draw] {cylinder};
\node at (4,0) [cylinder,aspect=0.5,draw] {cylinder};
\node at (8,0) [cylinder,cylinder uses custom fill=true, cylinder end fill=blue,
cylinder body fill=blue!20,draw] {cylinder};
\end{tikzpicture}
```



圆柱形的 node 其实用处很多，大部分是不放置文字，单独设置minimum width和minimum height，画各种比例的圆柱体。

4.9.5 node 的其他库

node 还内置了许多常用符号，需要加载 shapes.symbols 库

```
\usetikzlibrary{shapes.symbols}
```

这里只介绍比较常用的 signal。

```
\begin{tikzpicture}
\node at (0,0) [signal,draw] {管理};
\node at (2,0) [signal,signal to=west, draw] {管理};
\node at (4,0) [signal,signal to=east,signal from=west, draw] {管理};
\node at (6,0) [signal,signal to=east,signal from=west, signal pointer angle=60,
draw] {管理};
\end{tikzpicture}
```



node 的箭头库 shapes.arrows也比较实用。

```
\usetikzlibrary{shapes.arrows}
```

单箭头 single angle 和双箭头 double angle 用途比较广，参数差不多。

```
\begin{tikzpicture}
\node at (0,0) [single arrow, draw] {淬火};
\node at (2,0) [single arrow, minimum width=1.5cm, minimum height=2cm,draw] {淬火};
\node at (4,0) [single arrow, single arrow head extend=0.5cm, draw] {淬火};
\node at (6,0) [single arrow, single arrow head indent=0.1cm, draw] {淬火};
\node at (8,0) [single arrow, single arrow tip angle=60, draw] {淬火};
\end{tikzpicture}
```

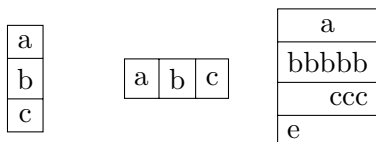


文字分割库。

```
\usetikzlibrary{shapes.multipart}
```

这里介绍一下 `rectangle split`，这个有些时候非常有用。默认是纵向分割的。`rectangle split parts=n` 表示分为 n 个格子，后面的内容必须小于等于 n ，多出来的不会显示。使用 `rectangle split part align={对齐列表}` 可以对每个格子设置不同的对齐方式。这个库表现计算机原理时挺有用。

```
\begin{tikzpicture}
\node at (0,0) [rectangle split, rectangle split parts=3, draw]
{a\nodepart{two}b\nodepart{three}c};
\node at (2,0) [rectangle split, rectangle split parts=3, rectangle split
horizontal, draw] {a\nodepart{two}b\nodepart{three}c};
\node at (4,0) [rectangle split, rectangle split parts=4, rectangle split part
align={center,left,right,left}, draw]
{a\nodepart{two}bbbb\nodepart{three}ccc\nodepart{four}e};
\end{tikzpicture}
```



`callout` 库对理工科的用处不大，需要时参看手册。

```
\usetikzlibrary{shapes.callouts}
```

`misc` 库做 ppt 时可能有些用处。

```
\usetikzlibrary{shapes.misc}
```

4.10 定义样式

4.10.1 修改默认样式 style

`tikz` 的设置语句为 `\tikzset`，设置样式的语句如下

```
\tikzset{样式名/.style={样式}}
```

默认样式名为 `every picture`

```
\tikzset{every picture/.style={line width=5pt}}
```

默认 `node` 样式名为 `every node`，例如将 `node` 都设置为可换行居中对齐，字号为 `small`。

```
\tikzset{
every node/.style={
align=center,
font={\small},
}}
```

4.10.2 自定义样式

例如设置图形的默认颜色和线宽，在 tikzpicture 环境内设置 style，当前环境有效，否则对下面的都生效。定义好的样式可以复用。

```
\definecolor{tjublue}{RGB}{0,70,140}
\tikzset{tju/.style={color=tjublue,line width=2pt}}

\begin{tikzpicture}[tju]
\fill (0,0) circle (0.5);
\end{tikzpicture}

\begin{tikzpicture}
\fill [tju] (0,0) circle (0.5);
\end{tikzpicture}
```

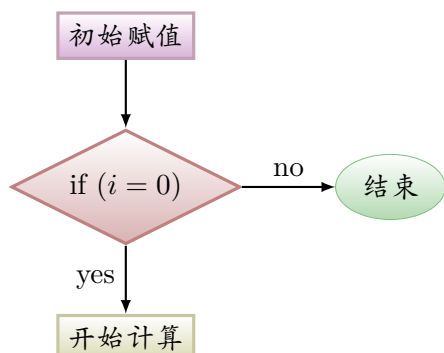


4.10.3 自定义带参数的样式

下面的例子自定义带参数的 node 样式，使用时直接调用自定义名 fang 即可。虽然自定义的 node 形状是方框，但是可以覆盖为其他形状。node 的各种参数仍然可以使用和覆盖。

```
\tikzset{
fang/.style={
rectangle,
minimum size=6mm,
very thick,
draw=#1!50!black!50,
top color=white,bottom color=#1!50!black!30,
font=\itshape,
}}

\begin{tikzpicture}[thick]
\node (a) at (0,0) [fang=magenta] {初始赋值};
\node (b) at (0,-2) [fang=red,diamond,aspect=2,font=\rmfamily] {if $(i=0)$};
\node (c) at (3.5,-2) [fang=green,ellipse,very thin] {结束};
\node (d) at (0,-4) [fang=yellow] {开始计算};
\draw [-latex] (a)--(b);
\draw [-latex] (b)-- node [above] {no} (c);
\draw [-latex] (b)-- node [left] {yes} (d);
\end{tikzpicture}
```



上面的例子还可以带多个参数，需要指定几个参数，调用的时候多个花括号并列方式指定参数。不过上面手册的这个配色很漂亮，一般不需要两个颜色参数来配色。

```
\tikzset{
fang/.style 2 args={
rectangle,
minimum size=6mm,
very thick,
draw=#1!50!black!50,
top color=white,bottom color=#2!50!black!30,
font=\itshape,
}}
\begin{tikzpicture}
\node (a) at (0,0) [fang={red}{blue}] {初始赋值};
\end{tikzpicture}
```

4.11 修饰库

修饰库很丰富。

```
\usetikzlibrary{decorations.shapes,decorations.pathmorphing,decorations.markings}
```

下面举几个理工科常用的例子。使用蛇形线表示光子（声子），snake 只需要 decorations.pathmorphing 库。

```
\begin{tikzpicture}
\draw [decorate,decoration={snake,pre length=15pt, post length=15pt, segment
length=8pt, amplitude=4pt}] [-latex] (0,0)-- node [above=2mm] {$\hbar\omega$} (3,0);
\end{tikzpicture}
```



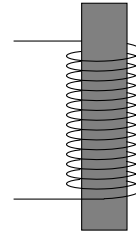
使用 coil 来表示弹簧

```
\begin{tikzpicture}
\node (a) [circle,ball color=green] at (0,0) {};
\node (b) [circle,ball color=green] at (2,0) {};
\draw [decorate,decoration={coil, pre length=2mm, post length=2mm, segment
length=1mm, aspect=0.5}] (a)--(b);
\end{tikzpicture}
```



下面的例子相对麻烦一点，需要自定义虚线的方式达到线圈缠绕的效果

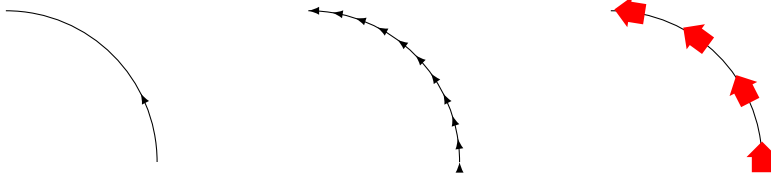
```
\begin{tikzpicture}
\draw [fill=gray] (-0.3,-1.5) rectangle ++(0.6,-3);
\draw[decorate,dashed,dash pattern=on 15mm off 6.05mm,
dash phase=-3mm, decoration={coil,amplitude=5mm,segment
length=1.3mm,aspect=0.2}] (0,-2) -- ++(0,-2.09)
coordinate (a);
\draw (a)--(-1.2,-4.09);
\draw (-1.2,-2)--(-0.3,-2);
\end{tikzpicture}
```



直接画横向的线圈更麻烦一点，需要使用`\clip`将不需要的线圈头剪裁掉，或者精确计算线圈长度。

使用 `markings` 库沿线画箭头，需要 `decorations.markings`

```
\begin{tikzpicture}
\draw (0,0) arc [start angle=0, end angle=90, radius=2];
\draw [decorate,decoration={markings,mark=at position 0.3 with {\arrow{latex}}}]
(0,0) arc [start angle=0, end angle=90, radius=2];
\draw (4,0) arc [start angle=0, end angle=90, radius=2];
\draw [decorate,decoration={markings,mark=between positions 0.0 and 1.0 step 3.14mm
with {\arrow{latex}}}] (4,0) arc [start angle=0, end angle=90, radius=2];
\draw (8,0) arc [start angle=0, end angle=90, radius=2];
\draw [decorate,decoration={markings,mark=between positions 0.0 and 1.0 step 9.42mm
with {\node [single arrow,fill=red, single arrow head extend=2pt, transform
shape]{}}}] (8,0) arc [start angle=0, end angle=90, radius=2];
\end{tikzpicture}
```



加载 `decorations.shapes` 库可以添加其他符号，注意 `signal` 的参数在 `decoration={}` 之外。

```
\begin{tikzpicture}
\draw [decorate,draw,fill=blue!40,decoration={shape backgrounds, shape size=3mm,
shape=signal, shape sep=4mm}, signal from=west, signal to=east] (0,0)--(3,0);
\end{tikzpicture}
```



下面的例子使用了双重 `markings`

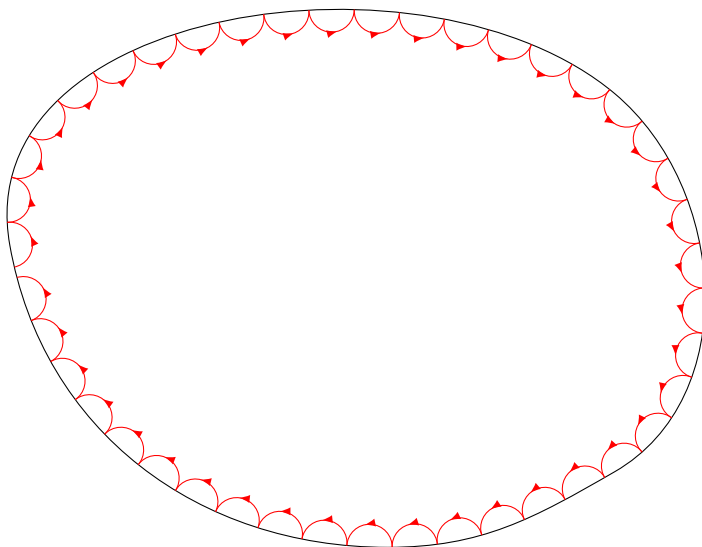
```
\begin{tikzpicture}
\clip [draw] (-4,3) to [out=100,in=140] (4,5) to [out=-40,in=30] (4,0) to
[out=-150,in=0] (1,-1) to [out=180,in=-80] (-4,3);

\tikzset{aaa/.style={decorate,decoration={markings,mark=between positions 0.1 and 1
step 6.5mm with {\color{red}\arrow{latex}}}}}
```



```
\draw [decorate,decoration={markings, mark=between positions 0.0 and 0.98 step 6mm
with {\draw [red] circle [radius=3mm];}}] (-4,3) to [out=100,in=140] (4,5) to
[out=-40,in=30] (4,0) to [out=-150,in=0] (1,-1) to [out=180,in=-80] (-4,3);

\draw [decorate,decoration={markings, mark=between positions 0.0 and 0.98 step 6mm
with {\draw [aaa] circle [radius=3mm];}}] (-4,3) to [out=100,in=140] (4,5) to
[out=-40,in=30] (4,0) to [out=-150,in=0] (1,-1) to [out=180,in=-80] (-4,3);
\end{tikzpicture}
```



另外，文字库 decorations.text 和分形库 decorations.fractals 可以自行查阅手册练习。

4.12 三维坐标

tikz 没有定义面，所以在 3D 画图方便，比 pstricks 的展现效果要弱一些。因为 3D 图的专业软件又远远超过 pstricks，所以 tikz 这个弱点笔者个人觉得不算太大的问题。tikz 的 3D 展现，本质上是 3D 图在二维平面上的投影。

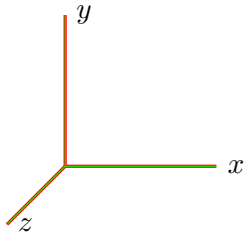
4.12.1 默认三维坐标

tikz 自带三维坐标，也可以自设置坐标方向和单位长度，即设置三维单位矢量在二维平面的投影。下图红色线为默认三维坐标的结果，三个轴长度都是 2，绿色线是自定义单位矢量结果。 x, y 轴方向长度仍为 2， z 轴方向长度为 1，因此测试出来 z 轴默认投影矢量是在 -135 度，单位长度接近 0.5445cm。设定投影矢量时要指定单位。

```
\begin{tikzpicture}
\draw [very thick,red] (0,0,0)--(2,0,0) node [right,text=black] {$x$};
\draw [very thick,red] (0,0,0)--(0,2,0) node [right,text=black] {$y$};
\draw [very thick,red] (0,0,0)--(0,0,2) node [right,text=black] {$z$};

\begin{scope}[x={(1cm,0cm)}, y={(0cm,1cm)}, z={(-135:1.089cm)}]
\draw [green] (0,0,0)--(2,0,0);
\draw [green] (0,0,0)--(0,2,0);
\end{scope}
\end{tikzpicture}
```

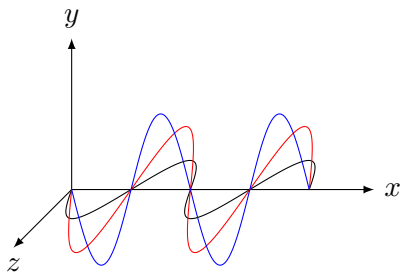
```
\draw [green] (0,0,0)--(0,0,1);
\end{scope}
\end{tikzpicture}
```



三维坐标在二维坐标的展现，是需要旋转两次的，角度关系可以看 tikz-3dplot 包的说明。

内置的三维坐标可以绕轴旋转。下面的例子中黑色线在 xz 平面，红色线是绕 x 轴旋转 30 度的结果，蓝色线是绕 x 轴旋转 90 度。

```
\begin{tikzpicture}
\draw [-latex] (0,0,0)--(4,0,0) node [right] {$x$};
\draw [-latex] (0,0,0)--(0,2,0) node [above] {$y$};
\draw [-latex] (0,0,0)--(0,0,2) node [below] {$z$};
\draw [domain=0:pi,smooth,samples=100,variable=\x] plot (\x,0,{sin(4*\x r)});
\draw [red] [domain=0:pi,smooth,samples=100,variable=\x,rotate around x=30] plot
(\x,0,{sin(4*\x r)});
\draw [blue] [domain=0:pi,smooth,samples=100,variable=\x,rotate around x=90] plot
(\x,0,{sin(4*\x r)});
\end{tikzpicture}
```



如果仅仅在一个固定的三维坐标系下画图，自定义 x, y, z 轴的单位矢量方向这个方法还是更方便一些，下面是个电磁波的例子。

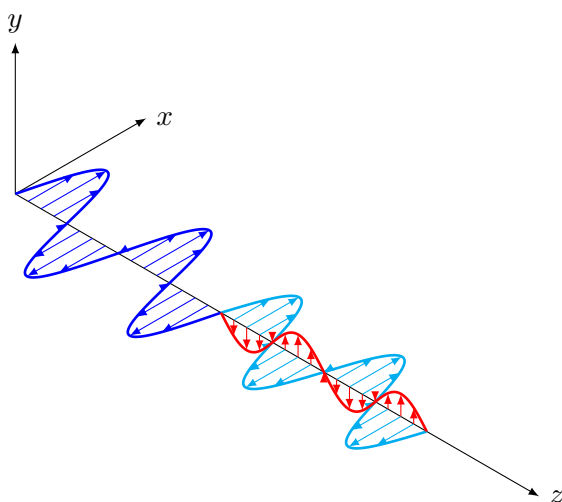
```
\begin{tikzpicture}[x={({30:1cm}}),y={({0cm,1cm}}), z={({-30:1cm}}),>=latex]
\draw [->] (0,0,0)--(2,0,0) node [right] {$x$};
\draw [->] (0,0,0)--(0,2,0) node [above] {$y$};
\draw [->] (0,0,0)--(0,0,8) node [right] {$z$};
% x 偏振
\draw [line width=1pt, color=blue] [domain=0:pi,smooth,samples=100,variable=\z]
plot({sin(4*\z r)},0,\z);
\foreach \z in {0.19625,0.3925,...,3}
\draw [->,blue] (0,0,\z)--({sin(4*\z r)},0,\z);
% 光参量过程
\draw [line width=1pt, color=cyan] [domain=pi:2*pi,smooth,samples=100,variable=\z]
plot({0.8*sin(4*\z r)},0,\z);
```

```

\foreach \z in {3.33625,3.5325,...,6.2}
\draw [->,cyan] (0,0,\z)--({0.8*sin(4*\z r)},0,\z);

\draw [line width=1pt, color=red] [domain=pi:2*pi,smooth,samples=100,variable=\z]
plot(0,{0.3*sin((4*\z+pi) r)},\z);
\foreach \z in {3.33625,3.5325,...,6.2}
\draw [->,red] (0,0,\z)--(0,{0.3*sin((4*\z+pi) r)},\z);
\end{tikzpicture}

```



4.12.2 tikz-3dplot 包

tikz-3dplot是一个包，不是 tikz 的库，需要使用加载包的语句。

```
\usepackage{tikz-3dplot}
```

调用手册有详细的数学公式解释角度关系。

```
texdoc tikz-3dplot
```

利用这个包设置角度关系是最准确的，设置语句有两种方式，一个是两个角度关系的

```
\tdplotsetmaincoords{角度}{角度}
```

使用时将参数加到方括号里，加到 tikzpicture 后面的参数表内对所有画图语句生效，加到 scope 后面的参数表内对 scope 内的语句生效，还可以直接加到 \draw 的参数表内，对当前语句生效。

```
[tdplot_main_coords]
```

另一个是三个角度关系的

```
\tdplotsetrotatedcoords{角度}{角度}{角度}
```

```
[tdplot_rotated_coords]
```

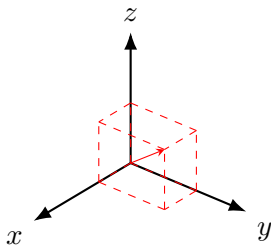
还可以随时可以重新设定坐标轴角度。

可以随时修改主坐标系设置和旋转坐标系设置，如果涉及多次旋转坐标系后结果的展现，这个包还是很方便的。

这个包还设置了三维点的投影点，不需要自己再算各个方向的投影坐标了。画晶体结构的

时候会很方便。

```
\tdplotsetmaincoords{60}{130}
\begin{tikzpicture}[scale=2,tdplot_main_coords,>=Latex]
\coordinate (O) at (0,0,0);
\tdplotsetcoord{P}{.8}{55}{60}
\draw[thick,->] (O,0,0) -- (1,0,0) node[anchor=north east]{$x$};
\draw[thick,->] (O,0,0) -- (0,1,0) node[anchor=north west]{$y$};
\draw[thick,->] (O,0,0) -- (0,0,1) node[anchor=south]{$z$};
\draw[-stealth,color=red] (O) -- (P);
\draw[dashed,color=red] (O) -- (Px);
\draw[dashed,color=red] (O) -- (Py);
\draw[dashed,color=red] (O) -- (Pz);
\draw[dashed,color=red] (Px) -- (Pxy);
\draw[dashed,color=red] (Py) -- (Pxy);
\draw[dashed,color=red] (Px) -- (Pxz);
\draw[dashed,color=red] (Pz) -- (Pxz);
\draw[dashed,color=red] (Py) -- (Pyz);
\draw[dashed,color=red] (Pz) -- (Pyz);
\draw[dashed,color=red] (Pxy) -- (P);
\draw[dashed,color=red] (Pxz) -- (P);
\draw[dashed,color=red] (Pyz) -- (P);
\end{tikzpicture}
```



这个包的手册到 2012，后面就没有再更新了，但是有些功能还是不错的，数学上也很严谨。

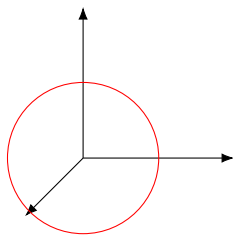
4.12.3 3d 库

3d 库是基于默认三维坐标（伪投影）的结果，3d 库提供了一些三维画图样式。

```
\usetikzlibrary{3d}
```

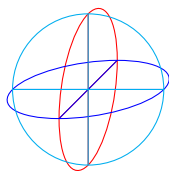
下面的例子与默认三维坐标的第一个例子是相同的。

```
\begin{tikzpicture}[->,>=Latex]
\draw (0,0,0) -- (xyz spherical cs:radius=2);
\draw (0,0,0) -- (xyz spherical cs:radius=2,latitude=90);
\draw (0,0,0) -- (xyz spherical cs:radius=2,longitude=90);
\draw [red] (0,0,0) circle (1);
\end{tikzpicture}
```



3d 库定义了 canvas 和 plane，然后在该 plane 画图得到的是投影效果。比如下图，画圆的语句会得到椭圆。

```
\begin{tikzpicture}
\begin{scope}[canvas is zy plane at x=0,red]
\draw (0,0) circle (1cm);
\draw (-1,0) -- (1,0) (0,-1) -- (0,1);
\end{scope}
\begin{scope}[canvas is zx plane at y=0,blue]
\draw (0,0) circle (1cm);
\draw (-1,0) -- (1,0) (0,-1) -- (0,1);
\end{scope}
\begin{scope}[canvas is xy plane at z=0,cyan]
\draw (0,0) circle (1cm);
\draw (-1,0) -- (1,0) (0,-1) -- (0,1);
\end{scope}
\end{tikzpicture}
```



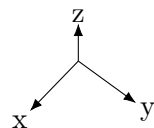
4.12.4 perspective

perspective 是 tikz 的库，加载时语句为

```
\usetikzlibrary{perspective}
```

这个库也是按旋转角度设置三维坐标系，但是角度参数顺序与 tikz-3dplot 包相反。

```
\begin{tikzpicture}[>=Latex,3d view={130}{60}]
\draw[->] (0,0,0) -- (1,0,0) node[pos=1.2]{x};
\draw[->] (0,0,0) -- (0,1,0) node[pos=1.2]{y};
\draw[->] (0,0,0) -- (0,0,1) node[pos=1.2]{z};
\end{tikzpicture}
```



画出来的效果是方向一致，但是 z 轴尺度不同，比 tikz-3dplot 要小。因为这个库是三点透视效果库，手册给出了透视图的例子，具体可看手册。

4.13 坐标计算 calc 库

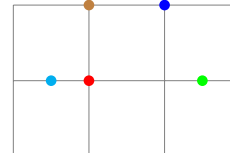
坐标计算，不是坐标值计算，坐标值计算是数学函数计算。加载 calc 库的语句为

```
\usetikzlibrary{calc}
```

对坐标进行计算时，需要加行间公式符号 $\$$ 。基本操作是对整个坐标的 (x,y) 进行加减乘法。

```
\begin{tikzpicture}
\draw [help lines] (0,0) grid (3,2);
\fill [red] (1,1) circle (2pt);
\fill [blue] ( $2*(1,1)$ ) circle (2pt);

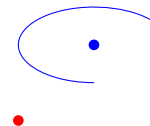
\coordinate (a) at (0.5,1);
\fill [cyan] (a) circle (2pt);
\fill [brown] ( $2*(a)$ ) circle (2pt);
\fill [green] ( $(a)+(2,0)$ ) circle (2pt);
\end{tikzpicture}
```



利用 calc 库和椭圆极坐标，就很容易圆规画弧了。

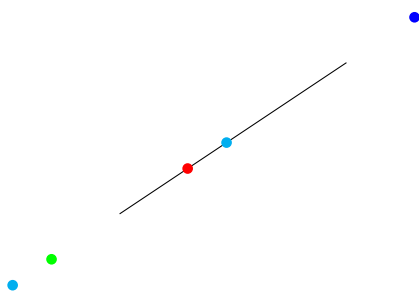
```
\begin{tikzpicture}
\fill [red] (0,0) circle (2pt);
\fill [blue] (1,1) circle (2pt);

\draw [blue] ( $(1,1)+(30:1 and 0.5)$ ) arc [start
angle=30, end angle=270, x radius=1, y radius=0.5];
\end{tikzpicture}
```



使用 $\$(a \text{ 点坐标})! \text{数值}!(b \text{ 点坐标})\$$ 可以实现沿 ab 连线移动的坐标点。该坐标点距离 a 点长度由数值给定。数值可以是比例，表示坐标点距离 a 点的长度为 ab 长度乘以比例，也可以是距离 a 点坐标的长度。比例和长度都可以是负值，表示逆向延展。

```
\begin{tikzpicture}
\draw (0,0)--(3,2);
\fill [red] ( $(0,0)!0.3!(3,2)$ ) circle (2pt);
\fill [blue] ( $(0,0)!1.3!(3,2)$ ) circle (2pt);
\fill [green] ( $(0,0)!-0.3!(3,2)$ ) circle (2pt);
\fill [cyan] ( $(0,0)!1.7cm!(3,2)$ ) circle (2pt);
\fill [cyan] ( $(0,0)!-1.7cm!(3,2)$ ) circle (2pt);
\end{tikzpicture}
```

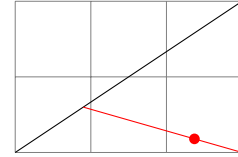


上面的语句可以连用，按从左到有顺序进行，红色点在黑色直线 0.3 长度处的点连线 $(3,0)$ 坐标长度距离黑色线与红色线交点为 0.7 处。

```

\begin{tikzpicture}
\draw [help lines] (0,0) grid (3,2);
\draw (0,0) -- (3,2);
\draw[red] ($(0,0)!.3!(3,2)$) -- (3,0);
\fill[red] ($(0,0)!.3!(3,2)!.7!(3,0)$) circle (2pt);
\end{tikzpicture}

```

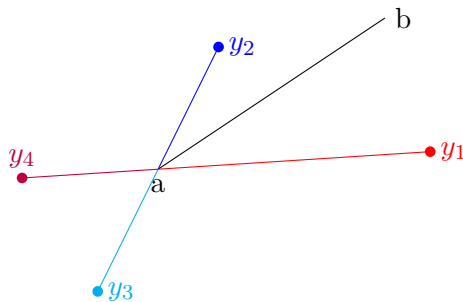


使用 $(a \text{ 点坐标})! \text{ 数值}! \text{ 角度}:(b \text{ 点坐标})\$$ 语句, ab 绕 a 点转动后, 再按数值缩放与 a 点的距离, 获得最后的坐标。角度为负是顺时针, 角度为正是逆时针, 数值为负是反方向。从例子中可以看出, y_2 点和 y_3 点在一条直线上。

```

\begin{tikzpicture}
\coordinate (a) at (0,0);
\coordinate (b) at (3,2);
\draw (a) node [below] {a}--(b) node [right] {b};
\coordinate (y1) at ($(a)!.10!-30:(b)$);
\fill[red] (y1) circle (2pt);
\draw [red] (a)--(y1) node [right] {$y_1$};
\coordinate (y2) at ($(a)!.05!30:(b)$);
\fill[blue] (y2) circle (2pt);
\draw [blue] (a)--(y2) node [right] {$y_2$};
\coordinate (y3) at ($(a)!-0.5!30:(b)$);
\fill[cyan] (y3) circle (2pt);
\draw [cyan] (a)--(y3) node [right] {$y_3$};
\coordinate (y4) at ($(a)!-0.5!-30:(b)$);
\fill[purple] (y4) circle (2pt);
\draw [purple] (a)--(y4) node [above] {$y_4$};
\end{tikzpicture}

```

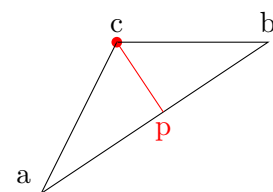


语句 $(a \text{ 点坐标})!(c \text{ 点坐标})!(b \text{ 点坐标})\$$, 得到从 c 点做 ab 连线的垂直投影。

```

\begin{tikzpicture}
\coordinate (a) at (0,0);
\coordinate (b) at (3,2);
\coordinate (c) at (1,2);
\fill[red] (c) circle (2pt);
\draw (a) node [above left] {a}--(b) node [above] {b};
{b}--(c) node [above] {c}--cycle;
\draw [red] (c)--($(a)!(c)!(b)$) node [below] {p};
\end{tikzpicture}

```



语句 $(a \text{ 点坐标})!(c \text{ 点坐标})! \text{ 角度}:(b \text{ 点坐标})$ ，得到从 c 点做 ab 连线的投影 p 点， cp 与 ab 的角度为给定角度。但是带角度时例子测试很不成功，不知道长度按什么规则计算的。

4.14 几何做图

结合 `calc` 库，`tikz` 的几何做图功能很不错。

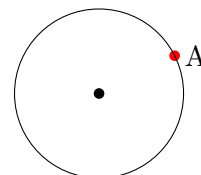
4.14.1 through

加载 `through` 库。

```
\usetikzlibrary{through}
```

`through` 子库用于通过某点画圆，必须在 `node` 的选项里使用，且只能画圆。使用 `through` 时，`inner sep=0pt`，`outer sep=0pt`。

```
\begin{tikzpicture}
\coordinate [label=right:{A}] (a) at (2,1.5);
\fill [red] (a) circle (2pt);
\fill (1,1) circle (2pt);
\node at (1,1) [draw,circle through={(a)}] {};
\end{tikzpicture}
```



4.14.2 相交 intersections

加载 `intersections` 库。

```
\usetikzlibrary{intersections}
```

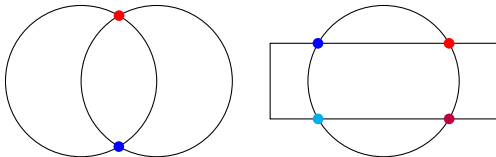
相交库可以得到任意曲线的相交点。比如两个圆有两个相交点，而一个矩形可能与圆有四个交点。

```
\begin{tikzpicture}
\path [draw, name path=a] (0,0) circle (1);
\path [draw, name path=b] (1,0) circle (1);
\path [name intersections={of=a and b}];
\coordinate (c1) at (intersection-1);
\coordinate (c2) at (intersection-2);
\fill [red] (c1) circle (2pt);
\fill [blue] (c2) circle (2pt);

\begin{scope}[xshift=4cm]
\path [draw, name path=a] (0,0) circle (1);
\path [draw, name path=b] (-1.5,-0.5) rectangle ++(3,1);
\path [name intersections={of=a and b}];
\coordinate (c1) at (intersection-1);
\coordinate (c2) at (intersection-2);
\coordinate (c3) at (intersection-3);
\coordinate (c4) at (intersection-4);
\fill [red] (c1) circle (2pt);
\end{scope}
\end{tikzpicture}
```



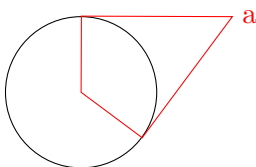
```
\fill [blue] (c2) circle (2pt);
\fill [cyan] (c3) circle (2pt);
\fill [purple] (c4) circle (2pt);
\end{scope}
\end{tikzpicture}
```



4.14.3 相切

tikz 的相切不是库，而是做成了坐标系。tikz 的坐标系有我们前面常用的直角坐标系、极坐标系、三维坐标系、node 坐标系，还有不怎么常用的重心坐标系和相切坐标系。相切坐标系必须加载 calc 库后方能使用。手册上的例子是画一个 node 的圆。相切点也可以多个，用 solution=n 表示第 n 个相切点。

```
\begin{tikzpicture}
\coordinate (a) at (3,2);
\node [circle,draw] (c) at (1,1) [minimum size=2cm] {};
\draw [red] (a) node [right] {a}--(tangent
cs:node=c,point={a},solution=1)--(c.center)--(tangent
cs:node=c,point={a},solution=2)--cycle;
\end{tikzpicture}
```



但是实际上我们可以将 node 画圆语句改为画圆语句，结果一样。这是因为 tikz 允许我们标记图形，下面的语句将 (c) 从标记 node 修改为标记圆，结果是一样的。这样就很方便我们做几何图形了。

```
\draw (c) (1,1) circle (1);
```

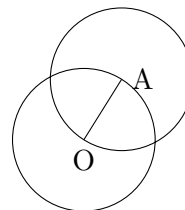
4.14.4 calc 库和几何做图

通过两点画双圆， $\text{veclen}(x,y) = \sqrt{x^2 + y^2}$ 。

```

\begin{tikzpicture}
\coordinate [label=below:{O}] (o) at (0,0);
\coordinate [label=right:{A}] (a) at (0.5,0.8);
\draw (o)--(a);
\draw let
\p1=($(a)-(o)$),
\n1={veclen(\x1,\y1)}
in
(o) circle (\n1)
(a) circle (\n1);
\end{tikzpicture}

```



4.15 matrix 库

加载矩阵库 matrix

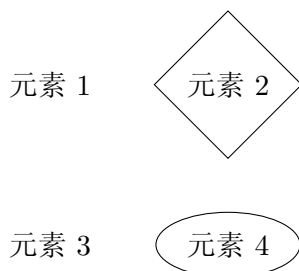
```
\usetikzlibrary{matrix}
```

矩阵库是 node 作为矩阵元素

```

\begin{tikzpicture}
\matrix[column sep=7mm, row sep=7mm]
{
\node (a11) {元素1}; & \node [diamond,draw] (a12) {元素2}; \\
\node (a21) {元素3}; & \node [ellipse,draw] (a22) {元素4}; \\
}; %注意这个分号
\end{tikzpicture}

```

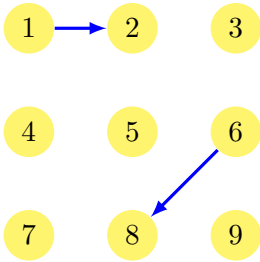


可以定义矩阵名，然后按矩阵元素调用。

```

\begin{tikzpicture}
\matrix (a) [matrix of nodes, nodes={circle,fill=yellow!70,minimum size=2pt}, column
sep=7mm,row sep=7mm]
{
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
}; %注意这个分号
\draw [-latex,very thick,blue] (a-1-1)--(a-1-2);
\draw [-latex,very thick,blue] (a-2-3)--(a-3-2);
\end{tikzpicture}

```



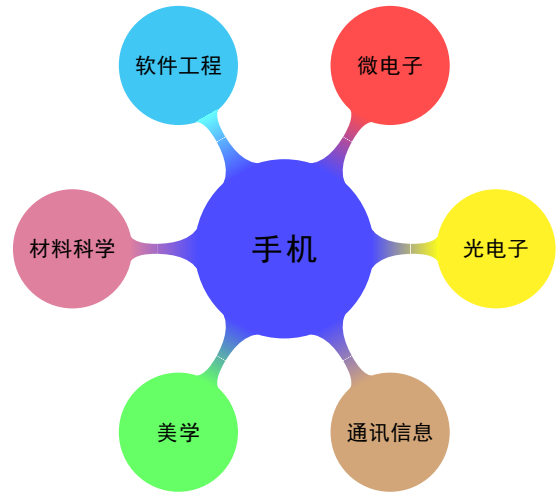
4.16 思维导图

加载思维导图库 mindmap,

```
\usetikzlibrary{mindmap}
```

并不喜欢这样的思维导图，觉得太占地方了，且形式比较单一。

```
\begin{tikzpicture}[small mindmap,concept color=blue!70,every node/.style={node
font={\normalsize,\bf}}]
\node [concept] {\large 手机}
child [grow=0, concept color=yellow!90] {node[concept]{光电子}}
child [grow=60, concept color=red!70] {node[concept]{微电子}}
child [grow=120, concept color=cyan!60] {node[concept]{软件工程}}
child [grow=180, concept color=purple!50] {node[concept]{材料科学}}
child [grow=240, concept color=green!60] {node[concept]{美学}}
child [grow=300, concept color=brown!70] {node[concept]{通讯信息}};
\end{tikzpicture}
```



4.17 电路

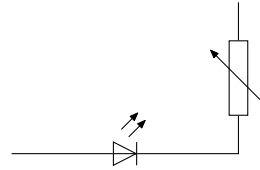
4.17.1 tikz 的电路子库

tikz 的电路需要加载的包比较多，电路必须指定标准，加载的时候以指定标准的方式加载。下面加载时指定 IEC 标准。。

```
\usetikzlibrary{circuits.ee,circuits.ee.IEC,circuits.logic.IEC}
```

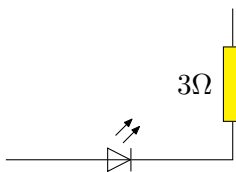
仅仅加载包是不行的，还必须在 tikzpicture 环境的参数里指定标准。电子元件用方括号表示，默认位置在前后两个坐标的中间位置处。

```
\begin{tikzpicture}[circuit ee IEC]
\draw (0,0) to [diode={light emitting}] (3,0) to
[resistor={adjustable}] (3,2);
\end{tikzpicture}
```



电阻可以添加阻值，也可以涂色

```
\begin{tikzpicture}[circuit ee IEC]
\draw (0,0) to [diode={light emitting}] (3,0) to
[resistor={info={3\Omega},fill=yellow}] (3,2);
\end{tikzpicture}
```



如果使用了 xeCJK 包，需要写成上面的形式，但是如果不使用 xeCJK 包，无论是 pdf_latex (CJKutf8) 还是 xel_atex (全英文) 编译情况，都可以使用下面的默认设置。

```
\draw (0,0) to [diode={light emitting}] (3,0) to [resistor={ohm=3 ,fill=yellow}] (3,2)
;
```

原因可能是 xeCJK 无法匹配 amsmath 的 $\mathrm{\Omega}$ ，没有显示。如果写成 info 样式才可以正确显示。

可以使用 circuit declare unit= 语句重置默认单位 ohm，

```
\begin{tikzpicture}[circuit ee IEC,circuit declare unit={ohm}{\text{欧}}]
\draw (0,0) to [diode={light emitting}] (3,0) to [resistor={ohm=3 ,fill=yellow}]
(3,2);
\end{tikzpicture}
```



但是不能重置为 Ω ，因为设置语句本身是 $\mathrm{\Omega}$ ，xeCJK 无法显示 $\mathrm{\Omega}$ 。修改为中文单位要设置为 欧，取代结果是 $\mathrm{\text{欧}}$ 。看上去很麻烦，但是可以通过这种方法将所有单位都改为中文。

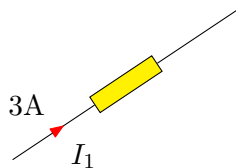
如果使用 info 方式，也可以写为中文的欧姆，但是不如上面的结果紧凑，因为 xeCJK 包中英文混排时会自动扩展英文到两边中文的间距。更细致的调节比较罗嗦，不如上面的设置语句简单方便。

```
\begin{tikzpicture}[circuit ee IEC]
\draw (0,0) to [diode={light emitting}] (3,0) to [resistor={info={3欧},fill=yellow}]
(3,2);
```

```
\end{tikzpicture}
```

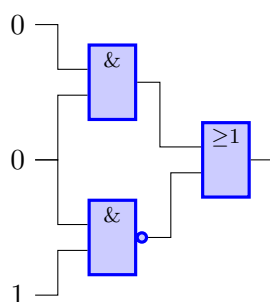
使用 `info` 方式标记器件参数是很方便灵活的。下面是用 `info` 做电流标记的例子，`info` 表示标记在下方，`current direction` 则是电流换个方向。电流箭头的位置使用 `pos=` 方式最方便，可以自动倾斜表示电流方向的三角的角度跟随电线方向。`pos=0.2` 是电阻前后两个坐标连线的 0.2 处，不止起点到电阻的 0.2 处。

```
\begin{tikzpicture}[circuit ee IEC]
\draw (0,0) to [current direction={red,pos=0.2, info={3A}, info'={\$I_1\$}},
resistor={fill=yellow}] (3,2);
\end{tikzpicture}
```



指定逻辑电路标准，然后可以使用内置的逻辑电路符号。

```
\begin{tikzpicture}[circuit logic IEC,every circuit symbol/.style={logic gate IEC
symbol color=black,fill=blue!20,draw=blue,very thick}]
\matrix[column sep=7mm]
{
\node (i0) {0}; & & \\
& \node [and gate] (a1) {} & \\
\node (i1) {0}; & & \node [or gate] (o) {} & \\
& \node [nand gate] (a2) {} & & \\
\node (i2) {1}; & & & \\
};
\draw (i0.east)-- ++(right:3mm) |- (a1.input 1);
\draw (i1.east)-- ++(right:3mm) |- (a1.input 2);
\draw (i1.east)-- ++(right:3mm) |- (a2.input 1);
\draw (i2.east)-- ++(right:3mm) |- (a2.input 2);
\draw (a1.output)-- ++(right:3mm) |- (o.input 1);
\draw (a2.output)-- ++(right:3mm) |- (o.input 2);
\draw (o.output)-- ++(right:3mm);
\end{tikzpicture}
```



更多逻辑电路符号可以加载包

```
\usetikzlibrary{shapes.gates.logic}
```

```
\usetikzlibrary{shapes.gates.logic.US} %美标
\usetikzlibrary{shapes.gates.logic.IEC} %IEC标准
```

4.17.2 基于 tikz 的电路包 circuitikz

除此之外，推荐一个更新活跃的包 circuitikz

```
\usepackage{circuitikz}
```

这个包的手册基本还算容易，下面还是添加一个入门的例子看构思。ctikzset是电路包的 style 设置语句

```
\begin{circuitikz}
\ctikzset{
resistor=european, %设置电路为欧标，欧标是矩形
}

\ctikzset{
resistors/scale=0.7, %设置电阻的大小为默认的0.7倍
}

\node (OA) at (0,0) [op amp, anchor=-] {};
\draw (OA.-) to [short,-*] (-1,0) to [empty photodiode,fill=black] (-3,0) node [ocirc]
{} node [left] {$+U_{\mathrm{b}}$};

\draw (-1,0)--++(0,1) to [resistor,R=$R$] ++(2,0) -| (OA.out) to [short,*-o] ++(1,0);

\draw (OA.+)-| ++(-0.3,-0.3) node [ground] {};
\draw (OA.up) to [short,-o] ++(0,0.5) node [right] {$+$};
\draw (OA.down) to [short,-o] ++(0,-0.5) node [right] {$-$};
\end{circuitikz}
```

上面的电路图思路如下

- 先以 node 的方式放置运算放大器，anchor=-表示坐标(0,0)的位置是运算放大器的 - 端口点。运算放大器的 node 名字为 OA，可以调用的端口是OA.+、OA.-、OA.up、OA.down、OA.out。
- 从OA.-到左边实心极点划线，[short,-*]表示该线段的末端为实心极点，空心极点则使用[short,-o]。调用默认器件，包括极点，都必须使用to语句。
- 继续向左边划线并添加光电二极管接收器，器件默认放置在线段中间。
- 如果不是直接划线，末端添加实心极点和空心极点都需要使用node，极点的器件名是ocirc，默认是空心的，实心的只需要添加参数ocirc,fill=black，后面的花括号不能省略。
- 并排再写一个文字框的node。
- 从左边实心极点连线，然后连接电阻，电阻标识符可以使用R=\$R\$，R表示电阻。也可以使用l=\$R\$，l表示标识。如果想放在下面，只能使用l，l_=\$R\$，下划线表示与默认对称的位置。
- 从电阻到OA.out，可以采用-|划线方式，这里的实心极点放到后面的线段上。
- 后面的线段，起始处加实心极点，末尾处加空心极点，所以是[short,*-o]。

- OA.-接地，接地采用 node 形式，所以可以用-|划线方式。接地的器件名是 ground，后面的花括号不能省略。
- 最后OA.up和OA.down分别加引线和极点，再加正负文字标识。

这里没有给结果图，因为 circuitikz 和前面介绍的 tikz 默认电路子库冲突，同时加载该包和 tikz 的电路子库，会破坏前面的电路子库结果展示，同时 circuitikz 包的结果也会有部分器件无法正确显示，比如接地。

电子领域，还有一个继电器符号包 tikz-relay

```
\usepackage{tikz-relay}
```

展示时序的包 tikz-timing 很漂亮，非常推荐。

```
\usepackage{tikz-timing}
```

交换构架 switching architectures 包 sa-tikz手册更新到 2014 年，如果有需要也可以练习一下。

chart 库，加载时是 tikz 的 circuits.plc.sfc 库，手册是 tikz-sfc。

```
\usetikzlibrary{circuits.plc.sfc}
\textdoc tikz-sfc
```

tikz-ladder 是画 ladder diagram 的库。

```
\usetikzlibrary{circuits.plc.ladder}
```

pinoutikz 是标识芯片管脚的包

```
\usepackage{pinoutikz}
```

4.18 pgfplots

虽然 tikz 手册里有专门的 data visualization 部分，但是目前 pgfplots 包成长得相当不错，小一点的数据还是比较推荐使用的。如果数据文件太大，更推荐基于 c++ 的 gnuplot 软件。

加载 pgfplots 包即可，不用再加载 tikz 包。

```
\usepackage{pgfplots}
```

4.18.1 二维函数画图

设置有点繁琐，简单的函数画图还是很漂亮的，而且坐标轴标识和图例的公式都可以直接得到 L^AT_EX 公式的美丽输出。图例位置默认值是东北角，四个角可以简单的 legend pos= 来设置，图中的例子是随意设置图例位置的语句。因为是 tikzpicture 环境内，所以图上在添加指示箭头、公式或者其他图形都非常方便。

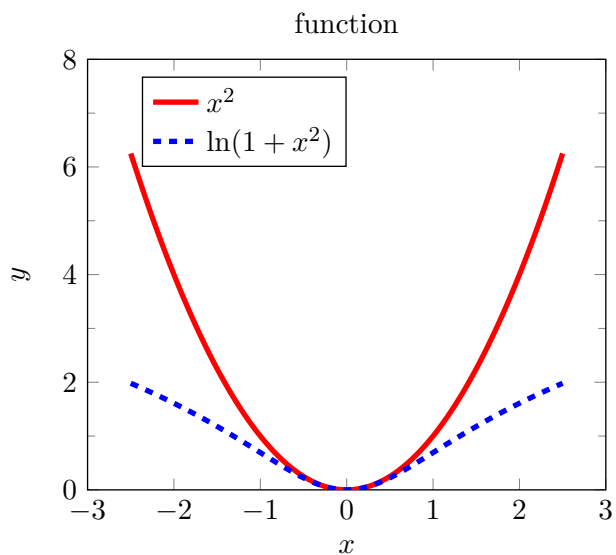
```
\pgfplotsset{
every axis legend/.append style={
at={{(0.5,0.96)}},
anchor=north east,
legend cell align=left,
},
```

```

}
\begin{tikzpicture}
\begin{axis}[
title=function,
xmin=-3,xmax=3,
xlabel={$x$},
ylabel={$y$},
ylabel style={yshift=-10pt},
ymin=0, ymax=8,
minor y tick num=1,
legend entries={$x^2$, $\ln(1+x^2)$},
line width=0.5pt,
]
\addplot [
red,
domain=-2.5:2.5,
samples=51,
smooth,
line width=2pt,
]
{x*x};

\addplot [
blue,
dashed,
domain=-2.5:2.5,
samples=51,
smooth,
line width=2pt,
]
{ln(1+x*x)};
\end{axis}
\end{tikzpicture}

```



4.18.2 二维数据画图

数据画图设置与上面的例子相同，需要修改的仅仅是画图语句。数据文件的分隔符需要特别指明，下面的例子是逗号分隔情况。

```
\addplot [画图参数] table [col sep=comma] {data.txt};
```

多列数据时，有两种方式选取列。一个是 $x=$ 列名， $y=$ 列名，数据文件的列名分隔符必须与数据文件的分隔符一致。还有一个是 $x \text{ index}=$ 列序号， $y \text{ index}=$ 列序号，列序号从 0 开始，遵循了 C 语言的风格，gnuplot 的列序号是从 1 开始的。这个例子需要单独的数据文件才能实现，就不给代码了，请自行练习。

4.18.3 三维函数画图

从函数画三维图很漂亮，

```
\begin{tikzpicture}
\begin{axis}
\addplot3[
surf,
domain=-2:2,
domain y=-2:2,
samples=50
]
{exp(-x^2-y^2)};
\end{axis}
\end{tikzpicture}
```

感觉 pgfplots 配色比 gnuplot 好看，但是画图比较慢。

4.18.4 三维 map 图

这个例子仍然是从函数画图。

```
\begin{tikzpicture}
\begin{axis}[view={0}{90},colorbar]
\addplot3[
surf,shader=interp, %必须写在同一行
domain=-2:2,
domain y=-2:2,
samples=50
]
{exp(-x^2-y^2)*x};
\end{axis}
\end{tikzpicture}
```

4.18.5 从数据画三维图

从数据文件画三维图语句是类似的，但是需要额外注意的是，pgfplots 的数据文件要求与 gnuplot 一样，需要断成块，中间需要一行空行。数据文件要写成，

```

x1, y1, f(x1,y1)
x1, y2, f(x1,y2)
x1, ..., ...
x1, yn, f(x1,yn)
空行
x2, y1, f(x2,y1)
x2, y2, f(x2,y2)
x2, ..., ...
x2, yn, f(x2,yn)
空行
...
```

x 相同的是一组数据块，还是 y 相同是一组数据块，结果是一样的，但是空行隔断是必须有的。

这方面 origin 做得最好，数据不断行也能画，但是 origin 画图也慢，画图最快的是 gnuplot。python 的画图库，自己的数据比较容易，从数据文件画图则比较麻烦。用 python 处理数据，包括给没有断行的数据断行，然后用 gnuplot 画图，个人认为是最优的画图方式。如果需要美化图例或标识，可以 gnuplot 画图到 eps 文件，然后使用 node 完成加载和美化。也可以使用设置参数，gnuplot 直接输出为 tex 文件，然后修改 tex 文件美化。

pgfplots 的画图功能相对来说越来越完善，数据量一般的情况下，pgfplots 画图相对来说比较美观。设置虽然麻烦些，但是做出来的各种效果都还不错。手册比较长，写得还算详细，感兴趣的可以自行练习。

4.19 基于 tikz 和 pgf 的科学包或 tikz 库

4.19.1 高亮数学公式

hf-tikz 包用来高亮数学公式，标记重点步骤。

```
\usepackage{hf-tikz}
```

4.19.2 物理类

光谱

推荐使用 pgf-spectra，这是一个元素光谱包，可以画可见光光谱。

```
\usepackage{pgf-spectra}
```

默认是 NIST 数据，如果希望加载 LSE 数据，可以添加包选项

```
\usepackage[LSE]{pgf-spectra}
```

内置了一些原子谱线数据，texlive 2024 版本可以直接使用。

pgf-spectra 包支持自定义谱线，也可以按 spectra.data.NIST.tex 或者 spectra.data.LSE.tex 的谱线数据格式自行添加自己常用的光谱数据。

费曼图

画费曼图的包 `tikz-feynman`，该包必须 `lualatex` 编译，`latex`、`pdflatex` 或 `xelatex` 编译都无法正确显示。加载语句如下

```
\usepackage{tikz-feynman}
```

经常画费曼图使用该包比较方便，但是偶尔画一下，直接用 `tikz` 画也可以，省得再去学语句了。

还有一个费米图包是 `tikz-feynhand`，是 `tikz-feynman` 的低配版，`tikz-feynhand` 可以直接 `pdflatex` 编译。

```
\usepackage{tikz-feynhand}
```

原子轨道

画玻尔原子轨道的包，基于 `pgf` 开发，还不错。

```
\usepackage{bohr}
```

分子轨道

`tikzorbital` 是画分子轨道的包，不算很漂亮，但是能用。

```
\usepackage{tikzorbital}
```

粒子加速器

粒子加速器包 `tikz-palattice`，内置了一些加速器的常用组件。

```
\usepackage{tikz-palattice}
```

这个包还算漂亮，配色也不错，不是这个领域的功能不好评价。

星球

`tikz-planets` 是一个开始于 2019 年的新包，画星球用的。

```
\usepackage{tikz-planets}
```

4.19.3 量子

`quantikz` 是一个库，画量子电路用的。

```
\usetikzlibrary{quantikz}
```

4.19.4 光路图包 `tikz-optics`

画光路图的包 `tikz-optics`，手册更新到 2017，法语手册。这个包感觉会就此放弃，而且功能上也不是很强。

```
\usepackage{tikz-optics}
```

pstricks 画图体系有一个光路图包，手册一直在更新，

```
\usepackage{pst-optexp}
```

pstricks 画图体系还有一个几何光路图包，手册更新到 2016 年，效果还不错。使用时必须注意，pstricks 的语句是空格敏感的，没有空格的地方要严格按手册不能添加空格。

```
\usepackage{pst-optic}
```

直接用 tikz 就可以画出漂亮的光路图，但是凸透镜笔者一般会偷懒为椭圆。pst-optexp 手册封面的光栅彩图用 tikz 也可以做出。

4.19.5 工程

标尺寸的包 tikz-dimline，因为机械画图软件很多很强大，所以这个包也就不是很有用了。

```
\usepackage{tikz-dimline}
\usepackage{wasysym} %diameter需要
```

轨道示意图是一个 2018 年开始的新包，

```
\usepackage{tikz-track schematic}
```

tikz-network 是一个画网络理论的新包

```
\usepackage{tikz-network}
```

4.19.6 chart

Texlive 里包含的卡诺图包很多，tikz 自己有一个卡诺图的库

```
\usetikzlibrary{karnaugh}
```

tikz-cd 有点类似数学的 cd 库，加载 tikz-cd 有两种方式，包和库都行

```
\usepackage{tikz-cd}
\usetikzlibrary{cd}
```

4.19.7 其他

语言学包 tikz-dependency 非常漂亮

```
\usepackage{tikz-dependency}
```

tikzmark 是 tikz 的库，

```
\usetikzlibrary{tikzmark}
```

tikzrput 包用来放置图或文字

```
\usepackage{tikzrput}
```

4.20 基于 tikz 和 pgf 的其他包

4.20.1 装饰纹包 pgfornament

pgfornament 包设计请柬或者彩笺时比较有用。

```
\usepackage{pgfornament}
```

还有一个汉化的国风花纹包。

```
\usepackage{pgfornament-han}
```

4.20.2 tikz-among-us

AmongUs 是一个游戏，这个新包做得挺漂亮的，虽然对笔者没啥用。

```
\usepackage{tikz-among-us}  
\usepackage{tikz-among-us-fancyhdr}  
\usepackage{tikz-among-us-watermark-eso-pic}
```

4.20.3 tikz-truchet

这是一个画瓷砖图案的包。

```
\usepackage{tikz-truchet}
```

4.20.4 动物包

人物包比较推荐，

```
\usepackage{tikzpeople}
```

鸭子包

```
\usepackage{tikzducks}
```

土拨鼠

```
\usepackage{tikzmarmots}
```

动物的合集包

```
\usepackage{tikzlings}
```

4.20.5 符号

tikzsymbols 包可以画各种符号和图标，

```
\usepackage{marvosym}  
\usepackage{tikzsymbols}
```

4.21 几个光路图的例子和一些小技巧

4.21.1 直线中间添加箭头

沿线画很多箭头，可以用`decorations.markings`库，但是只画一个箭头，这个库太繁琐。下面是几个简便一点的方式。

可以采用 `shapes.geometric` 库中 `isosceles triangle` 和 `dart` 这两种 `node` 形状。这个方法比较简单，但是只能添加相似形状的箭头。

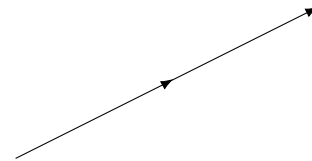
```
\begin{tikzpicture}[>=Latex]
\draw [->] (0,0)-- node [dart,fill,inner sep=0.8pt] {} (2,0);
\draw [->] (3,0)-- node [isosceles triangle,isosceles triangle apex
angle=30,aspect=2,fill,inner sep=1.0pt] {} ++(2,0);
\end{tikzpicture}
```



可以使用 `mark` 来加载中间的箭头或其他符号，需要知道中间箭头的方向。语句也比较啰嗦。

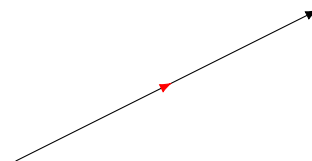
```
\begin{tikzpicture}[>=Latex]
\pgfdeclareplotmark{mid-arrow}{\draw[->] (0,0)--
(0.1,0);}
\draw [->] (0,0)-- coordinate (a) (4,2);

\draw [mark=mid-arrow,mark options={rotate=30}] plot
coordinates {(a)};
\end{tikzpicture}
```

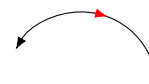


更推荐在 `node` 里嵌套 `tikz` 图片的方式，`\tikz` 画图语句或 `tikzpicture` 环境，图片等价于文字，可以加载到文本框内。此时不需要知道箭头方向，使用 `node` 的 `sloped` 参数即可。`node` 是写在画线语句后面的，所以箭头位置也很容易用 `pos` 参数调整。这个方法可以使用所有的 `tikz` 内置箭头样式和设置参数。

```
\begin{tikzpicture}[>=Latex]
\draw [->] (0,0)-- node [sloped]
{\tikz\draw[-{Latex[red]}] (0,0)--(0.1,0);} (4,2);
\end{tikzpicture}
```



```
\begin{tikzpicture}[>=Latex]
\def\aa{\tikz!\draw [->] (0,0)--++(0.2,0);} %使用定义def
\draw [->] (0,0) arc [start angle=0, end angle=150,
radius=1] node [pos=0.5,sloped,red] {\aa};
\end{tikzpicture}
```



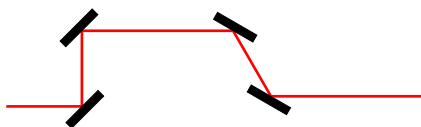
4.21.2 光路图技巧

光路图中，能使用 `node` 的尽量使用 `node`。

反射镜

光路图经常会用到反射镜，使用 node 画反射镜最方便，因为 node 默认的旋转点是 node 语句放置文本框的坐标点，而设置位置左右后 node 语句的坐标点就在 node 边框上。等同于绕该点可以 360 度旋转。需要注意的是，要将 inner sep 设置为 0pt，这样才能用 minimum width 将反射镜的宽度设置为任意宽度，否则会附加 inner sep 的宽度。这里必须是宽度为大的值、高度为小的值，才能通过 node 的上下左右的锚点和旋转角度得到正确的结果，反之则不会按 node 的锚点进行旋转。而且一定要设定 draw 和 fill 都为 black，并且 draw 的线宽等于光路的线宽，更方便的是设置为极小，这样放大看图片后光路才是真正的反射效果。不能只 fill，不 draw，不设置线宽，否则放大效果就不好了，当然不放大实际上不大看得出来。反射镜的大小可以根据整体图片的尺寸进行调整，一般情况下同一张图尺寸都是一致的。特殊情况下可以特别设定尺寸，用 shift 语句移动反射点在镜子上的位置。可以将镜子 node 的设置自定义后重复使用。比较好计算的坐标可以分开写，不好计算的写到划线语句里面很方便。

```
\begin{tikzpicture}[>=Latex]
\tikzset{mymirror/.style={draw=black,line width=0.1pt,fill=black,inner
sep=0pt,minimum width=3pt,minimum height=6mm}}
\draw [red,line width=1pt] (0,0)--++(1,0)--++(0,1)--++(2,0) node
[right,mymirror,rotate=60] {} --++(-60:1) node [left,mymirror,rotate=60] {}
--++(2,0);
\node at (1,0) [right,mymirror,rotate=-45] {};
\node at (1,1) [left,mymirror,rotate=-45] {};
\end{tikzpicture}
```



偏振分光棱镜

```
\begin{tikzpicture}
\pgfdeclarelayer{laser}
\pgfdeclarelayer{element}
\pgfsetlayers{laser,element}

\tikzmath{
\ a=30;
\ b=180-\ a;
\ c=\ a-90;
}

\begin{pgfonlayer}{element}
\node (p) at (2,0) [rotate=\ a][line width=0pt,opacity=0.2, fill=cyan,minimum
size=1cm,inner sep=0pt] {};
\draw [gray] (p.north west)--(p.south east);
\end{pgfonlayer}

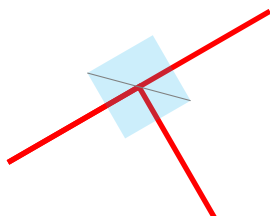
\coordinate (s) at ($(p.center)+(-\ b:2)$);
```

```

\coordinate (e) at ($(p.center)+(\c:2)$);
\coordinate (n) at ($(s)!2!(p.center)$);

\begin{pgfonlayer}{laser}
\draw [red,line width=2pt] (s)--(p.center)--(e);
\draw [red,line width=2pt] (s)--(n);
\end{pgfonlayer}
\end{tikzpicture}

```



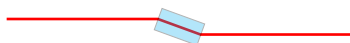
介质

同理，画光经过任意厚度的介质发生折射，也用 node 画，设置 opacity 的数值显示介质透明。这种情况下计算量其实很小，光线偏折用双加号极坐标，设置 node 宽度为光线偏折走的长度，高度为介质宽度，旋转点为入射点即可。放大很多倍看是有点问题的，考虑到光线宽度就不对了，但是基本上视觉感观是可以接受的。

```

\begin{tikzpicture}
\draw [red,line width=1pt] (0,0)---+(2,0)---+(-20:6mm)---+(2,0);
\node at (2,0) [right,draw=black, very thin,fill=cyan,opacity=0.3,inner
sep=0pt,minimum width=6mm,minimum height=3mm,rotate=-20] {};
\end{tikzpicture}

```



光阑

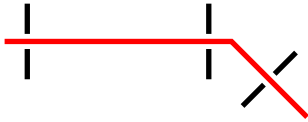
光阑过去讨论的画法这个版本中舍去了，新的画法更简单。利用 `\tikz{画图多行语句}` 是文字 (L^AT_EX 中图片其实是文字) 的特点，使用纯画图语句来实现。第一种方式采用 `\def` 命令，

```

\begin{tikzpicture}[>=Latex]
\def\ap{
\tikz{
\draw [line width=2pt] (0,0.1)---+(0,0.4);
\draw [line width=2pt] (0,-0.1)---+(0,-0.4);
}}

\draw [line width=2pt,red] (0,0)-- node [pos=0.1,black, align=center] {\ap} node
[pos=0.9,black, align=center] {\ap} (3,0) -- node [black,align=center,sloped] {\ap}
(4,-1);
\end{tikzpicture}

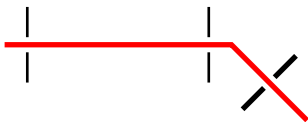
```

第二种方式采用`\newcommand`命令，并设定三个参数，光阑的线宽、1/2 间距（光阑半径）和光阑单边长度。

```
\begin{tikzpicture}[>=Latex]
\newcommand\ap[3]{
\tikz{
\draw [line width=#1] (0,#2)--++(0,#3);
\draw [line width=#1] (0,-#2)--++(0,-#3);
}}

\draw [line width=2pt,red] (0,0)-- node [pos=0.1,black, align=center]
{\ap{1pt}{0.1}{0.4}} node [pos=0.9,black, align=center] {\ap{1pt}{0.1}{0.4}} (3,0)
-- node [black,align=center,sloped] {\ap{2pt}{0.1}{0.4}} (4,-1);
\end{tikzpicture}
```



使用 pgf-spectra 的可见光光谱

凡是图片都可以添加到 node 里。

```
\begin{tikzpicture}
\node (a) at (0,0) [inner sep=0pt] {\pgfspectra[width=4cm,height=0.5cm]};

\draw (a.north west) --(0,1) --(a.north east);
\end{tikzpicture}
```



光栅色散

下面的代码还将使用 `patterns` 库，用来填充 node 显示为光栅。

```
\usetikzlibrary{patterns,patterns.meta}
```

画光栅色散可以使用`\foreach`语句，颜色必须设置一个中间变量先计算出来再使用。

```
\begin{tikzpicture}[>={Latex[scale width=0.7,scale length=0.7]}]

\draw [|->|] (-4,1)-- node [above] {$f$} ++(2,0);
\draw [|->|] (-2,1)-- node [above] {$f$} ++(2,0);
\draw [|->|] (0,1)-- node [above] {$f$} ++(2,0);
```

```

\draw [<->|] (2,1)-- node [above] {$f$} ++(2,0);

\draw [red,line width=1pt] (-3,-1)--(-4,0);
\draw [red,line width=1pt] (3,-1)--(4,0);

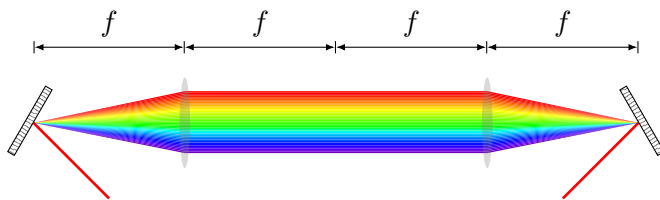
\foreach \x in {-0.4,-0.38,...,0.4}
{
\tikzmath{\c=\x*310+530;} %先计算出颜色的波长值
\definecolor{wavecolor}{rgb:wave}{\c} %加rgb:方便调用
\fill [wavecolor] (-4,0)-- ++(2,\x)-- ++(4,0)-- ++(2,-\x)-- ++(-2,{\x+0.02})--
++(-4,0)--cycle;
}

\fill [gray,opacity=0.3] (-2,0) ellipse (0.07 and 0.6);
\fill [gray,opacity=0.3] (2,0) ellipse (0.07 and 0.6);

\node at (-4,0) [left,draw,pattern={Lines[angle=-30,distance=0.5mm,line
width=0.3pt]},pattern color=gray,inner sep=0pt,minimum width=1mm,minimum
height=1cm,rotate=-30] {};
\node at (4,0) [right,draw,pattern={Lines[angle=30,distance=0.5mm,line
width=0.3pt]},pattern color=gray,inner sep=0pt,minimum width=1mm,minimum
height=1cm,rotate=30] {};

\end{tikzpicture}

```



4.21.3 tikz 自带的镜像参数

不需要添加镜像包 environ, tikz 自己有镜像功能, 还能镜像且缩放比例。测试了一下, 可以作为 node 参数和 scope 参数使用, 很方便。

```

\begin{tikzpicture}
\draw (0,0)-|(2,1)--cycle;

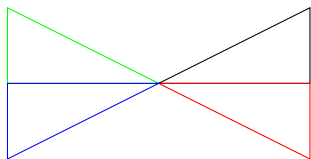
\begin{scope}[xscale=-1,green] %水平镜像
\draw (0,0)-|(2,1)--cycle;
\end{scope}

\begin{scope}[yscale=-1,red] %竖直镜像
\draw (0,0)-|(2,1)--cycle;
\end{scope}

\begin{scope}[scale=-1,blue] %反演

```

```
\draw (0,0)--(2,1)--cycle;
\end{scope}
\end{tikzpicture}
```



比例还可以缩放，但是要水平镜像还完全按比例缩放，就需要同时设置 `xscale` 和 `yscale`。

```
\begin{tikzpicture}
\draw (0,0)--(2,1)--cycle;

\begin{scope}[xscale=-0.5,yscale=0.5,green] %水平镜像，成比例缩放
\draw (0,0)--(2,1)--cycle;
\end{scope}
\end{tikzpicture}
```



4.21.4 一些实用技巧

`node` 坐标是无法穿透平移的 `scope` 子域使用的，给坐标起名的 `\coordinate` 命令也是如此。但是坐标可以被重新定义，因此只要复用定义代码，就可以在平移后的子域里重复画图。`\graph` 可以重复使用 `node` 坐标，但是功能单一，语法僵化，并不适合日常使用。最简单的办法是采用 `\def` 替换代码的形式复用命令，

```
\begin{tikzpicture}
\def\mycode{
\coordinate (m1) at (0,0);
\coordinate (m2) at (1,0);
}
\mycode
\draw (m1)--(m2);

\begin{scope}[yshift=-1cm]
\mycode
\draw [red] (m1)--(m2);
\end{scope}
\end{tikzpicture}
```



这里要注意的是，`\def` 仅仅是定义了 `\mycode` 包含的代码，第一次使用仍然需要调用 `\mycode`。

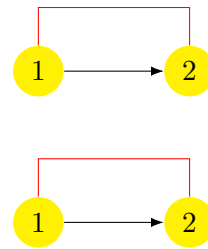
包括重复画相同部分的画图语句也是一样的，画过程图时可以提取最大公约的画图部分定义为新命令复用，展示行为过程。

```

\begin{tikzpicture}[>=Latex]
\def\mycode{
\node (m1) at (0,0) [fill=yellow,circle] {1};
\node (m2) at (2,0) [fill=yellow,circle] {2};
\draw (m1.90) [red] --++(0,0.5)-|(m2.90);
}
\mycode
\draw [->] (m1)--(m2);

\begin{scope}[yshift=-2cm]
\mycode
\draw [->] (m1)--(m2);
\end{scope}
\end{tikzpicture}

```



如果需要更换参数，就使用`\newcommand`语句，如同上面光阑的例子。

第五章 一些有用的包

本章介绍一个经常用到的类 standalone 和几个非常有用的包。

5.1 standalone 类

standalone是一个很好的类，可以将指定的图片环境编译为 pdf 文件中独立的一页，且页面大小随图片大小自动调整。平时使用时推荐每张图片一个 tex 文件，生成一页 pdf，方便将图片插入到文件中。在 standalone 类参数中设定单页环境，

```
\documentclass[tikz]{standalone} %tikzpicture环境
```

支持 tikz、minipage、animate、equation、chemfig、pstricks，还支持自定义环境。animate 情况只能是一个 animateinline 环境，生成一页动图，不能像 tikz 和 pstricks 那样为多页。其中 chemfig 环境在 chemfig 那章讲述。下面是基于 tikz 的文件框架，

```
\documentclass[tikz]{standalone}
\standaloneconfig{border=5pt} %设置图片四周留白

\usepackage{CJKutf8}
\usepackage{tikz}

\begin{document}
\begin{CJK}{UTF8}{gkai}

\begin{tikzpicture}
\fill [draw=black,fill=yellow] (0,0) circle (1);
\end{tikzpicture}

\end{CJK}
\end{document}
```

standalone 的另一个用处是用 minipage 环境来构建宽度固定长度自动的笔记，非常适合做计算机语言程序的笔记。下面的例子宽度其实也是可以随意调整的。类的参数必须写成multi=minipage，否则无法分页。

```
\documentclass[multi=minipage]{standalone}
\standaloneconfig{border={5mm 5mm 5mm 5mm}} %设置四周留白，左下右上

\usepackage{CJKutf8}

\begin{document}
\begin{CJK}{UTF8}{gkai}
```

```
\begin{minipage}{12cm} %设置宽度
内容
\end{minipage}

\end{CJK}
\end{document}
```

standalone 并不是支持所有环境，它支持那些可以明确尺寸的环境。比如说 tabular 是可以的（也需要加 multi=），一张表格的尺寸是明确的，但是可分页的 framed 环境就不行了。

standalone 还支持每个公式一页，但是必须通过 varwidth=宽度明确每页的宽度。

```
\documentclass[multi=equation,varwidth=6cm]{standalone}
```

standalone 支持多种环境

```
\documentclass[multi=minipage,tikz]{standalone}
```

使用 convert 选项，可以自动转换为 png（linux 下必须安装 imagemagick），使用这个选项，需要下面的编译手段

```
pdflatex -shell-escape 文件名
```

conver 写到包的参数里

```
\documentclass[multi=minipage,tikz,convert]{standalone} %默认是png格式
```

可以指定格式和分辨率等

```
\documentclass[multi=minipage,tikz,convert={ghostscript,gsdevice=tiffg4,outtext=.tiff,
density=300}]{standalone}
```

这是一次性的转换多个图片，不过直接用 convert 语句也一样。如果是单张图片，获得 pdf 后再使用 convert 命令很多时候更方便。

如果希望添加自己的环境，需要利用定义新环境的包 environ

```
\usepackage{environ}
```

美丽的 tcolorbox 环境不行，可以使用下面的方式，因为 tcolorbox 本身就加载了 environ，所以不用再加载。

```
\documentclass[multi=mybox]{standalone}
\standaloneconfig{border=3pt}

\usepackage{tcolorbox}

\NewEnviron{mybox}[1][]{
\begin{tcolorbox}[#1]
\BODY
\end{tcolorbox}
}

\begin{document}
```

```
\begin{mybox}[width=6cm,colback=green!5, colframe=green!35!black]
text
\end{mybox}

\end{document}
```

通过自定义环境方式，可以很方便自由使用 standalone 包。

5.2 计算机程序语法高亮: listings

本书的程序语法高亮就是用 listings 完成的。

```
\usepackage{listings} %加载listings包
```

5.2.1 样式设置

下面给出的是本书的设置。

```
%基本设置
\lstset{
language=[LaTeX]TeX, %设置语言，支持90多种常用语言
tabsize=2, %设置tab键宽度
frame=tlrb, %设置边框，tlrb表示四周都有边框
backgroundcolor=\color{gray!10}, %设置背景色
basicstyle=\small\ttfamily, %设置字号，代码长的话用小比较好看
commentstyle=\rmfamily\color{purple}, %设置注释颜色
keywordstyle=\bf\color{blue}, %设置关键词颜色，bf是直体加粗
morekeywords={\maketitle,chapter,subsection,subsubsection} %自定义关键词
columns=fullflexible, %强烈推荐这个设置，否则pdf有时产生多余空格
texcl=false, %支持某些tex命令，比如无编号公式
extendedchars=false, %建议为假
breaklines=true, %自动换行，xelatex对汉语注释换行支持最好
breakindent=0pt, %换行缩进，此处设置为无缩进
%showspaces=true, 显示不显示空格，显示空格的效果也不大好
aboveskip=6pt, %距离文本的上间距
belowskip=6pt, %距离文本的下间距
}
```

写几个\lstset{}都可以，可以分功能写成几个设置模块，还有一些常用设置，有许多是给别人讲代码时更有用的设置参数。

```
mathescape=true, %美元符号之间自动按tex行间公式，适用于非tex代码
escapechar=符号, %设置的符号对之间自动按tex，可以写带编号的公式，背景色支持不好
escapeinside=符号符号, %设置两个符号间自动按tex代码

numbers=left, %最左边显示行号，给人讲解代码时比较方便，一般不使用
numberstyle=\footnotesize, %行号字体
numbersep=5pt, %行号与语言间的距离
stepnumber=2, %每个几行显示行号
```

```

firstnumber=100, %第一行的行号
firstnumber=last, %接着上面的lstlisting环境编号
firstnumber=auto, %自动
%开始环境\{lstlisting\}[name=myC++code]时给每个环境起名, 名字相同的共享行号计数器
linrange={1-4,10-16}, %写一堆代码, 只显示指定行的代码

emph={square,root}, %对某些文字画重点
emphstyle=\underbar, %设置画重点的方式, 此处是下划线
emph={[2]base}, %多类重点, 可以分很多类, keyword也可以分类
emphstyle={[2]\color{red}}, %设置不同的重点标记方式

index={square}, %加索引, 后面打印索引才能看出效果
index={[2]root}, %分层

```

注释等也可以多重定义样式, 具体例子参看手册。

lstlisting环境可以像图、表、公式环境那样编号。

```

caption=[简短说明]代码内容说明文字, %如同图标一样对包含该设置的lstlisting环境自动编号
caption=\lstname, %说明自动添加为name=设定的内容
label=标签名, %引用为ref\{标签名\}
title=代码内容说明文字, %无自动编号, caption和title放在前面居中的位置
nolol=true, %caption有编号, 但是自动生产列表时不显示该条目

```

修改 caption 样式类似图表

```

\renewcommand\lstlistingnamestyle{}
\renewcommand\lstlistingname{代码} %默认是Listing

```

手册还有一些设置, 如果需要可以自行查看测试。

有 5 个样式范例文件, 但是不如自己读手册自行配置更方便。

如果需要高亮的语言不在手册语言列表中, 可以自己写配置文件, 或者加载接近的语言, 然后自定义 keywords。如果经常做某个不在列表上的语言的笔记, 可以将自己的语言高亮定义放在安装路径/texlive/texmf-local/tex/listings之下, 文件名必须是下面的这三个

```

lstmisc0.sty %for local add-ons (see the developer's guide), 开发者
lstlang0.sty %for local language definitions, 添加语言只写这个就行
lstlocal.cfg %as local configuration file, 常用自定义样式

```

一定是这三个文件名, 不要自己起名。再次安装新版本时, install-path/texlive/texmf-local/路径下的文件不会被覆盖, 新版本会安装在install-path/texlive/2025/下。

参数设置的优先级别为, \begin{lstlisting}[这里的参数第一优先级], \lstset{这里的参数第二优先级}, lstlocal.cfg这个文件里的第三优先级。

一般情况新的语言如果是基于某个列表中的语言的语法时, 可以只添加新关键词, 下面是基于 C++ 的例子

```

\lst@definelanguage[新语言版本]{新语言调用名}[ISO]{C++}
{
  morekeywords={添加关键词, 逗号分隔}
}

```


5.2.2 lstlisting 环境和行间程序代码

使用时，可以随时更改指定的语言，例如

```
\begin{lstlisting}[language=C++]
#include <stdio.h>
int main()
{
    printf("hello, world\n");
}
\end{lstlisting}
```

用方括号 [language=] 方式定义语言，只对该环境有效，因此可以在一个文档内使用多种语言的语法高亮。其他设置，如背景色等，也都可以当前环境修改。

```
\begin{lstlisting}[language=HTML, backgroundcolor=\color{blue!20}]
<!DOCTYPE HTML>
<html>
<body>

</body>
</html>
\end{lstlisting}
```

当然也可以使用 \lstset 重新设置参数，对以下所有生效。

行间语言高亮为

```
\lstinline!语言!
```

遇到感叹号情况，因为配对无法作为计算机语言，可以采用下面的美元符号替代方式

```
\lstinline$!=$
```

整体引入代码文件内容时，不能简单的使用 \input，因为在 lstlisting 环境中，这个命令会被视为代码。所以使用下面的方法

```
\lstinputlisting[参数，如语言、设置等]{文件名}
```

这句单独使用，不放在 lstlisting 环境中。整体引入代码文件，结合指定显示的代码行，可以不用赋值就能很方便地向人展现代码。

listings 使用方式的代码展示不能简单地使用 listings 环境，因为不能嵌套环境，否则会产生嵌套错误（配对错误）。上面的例子是 tcolorbox 包里的 tcblisting 环境做出来的，这个环境下面还会讲。不想增加额外的包，也可以使用 verbatim 环境，这个环境将里面的内容当作文本如实展现。

在 beamer 中使用 listings 时，必须加载 frame 的 fragile 选项，否则会剧烈报错！

```
\begin{frame}[fragile]{
使用lstlisting环境
\end{frame}
```

还可以使用 \lstnewenvironment 语句设置新环境，方便更换语言。如果每次都是同一种语言就没必要了，在 vim 里定义宏生成 lstlisting 环境很方便。

```
\lstnewenvironment{环境名}[变量数目][默认值]
{开始代码}
{结束代码}
```

例如

```
\lstnewenvironment{mycode}[1] []
{\lstset{language=python,#1}} %可以随意添加设置
{}

\begin{mycode}[frame=trbl]
#coding=utf-8.cp936
import numpy
import scipy.optimize
\end{mycode}
```

5.2.3 显示 tex 代码及其结果

加载 showexpl 包，可以展现代码和结果

```
\usepackage{showexpl}
\lstset{explpreset={numbers=none}} %取消行号
\setboolean{@SX@varwidth}{true} %图片自动宽度，需要ifthen包
```

设置结果（图片）box 是 colorbox，取消框，并设置图片边缘留白 5pt。

```
\renewcommand\ResultBox{\colorbox{lightgray}}
\setlength\ResultBoxSep{5pt}
```

然后可以直接使用 LTXexample 环境

```
\begin{LTXexample}
\begin{tikzpicture}
\draw (0,0)--(1,0);
\end{tikzpicture}
\end{LTXexample}
```

得到左边是图片，右边是代码

```
\begin{tikzpicture}
\draw (0,0)--(1,0);
\end{tikzpicture}
```



需要修改什么，就阅读手册后面的源码，然后试着修改设置项。tikz 的手册是 lualatex 编译的，自定义了代码展示环境。tcolorbox 也自定义了 tcolorbox 样式的代码展示环境，下面会讲。一般的需求 showexpl 包就可以满足了，但是这个包对上下文竖直间距的处理很不好。这本书采用的是 tcolorbox 的代码展示环境。

5.3 tcolorbox

tcolorbox 的手册写得很好，自学也可以的。讲课的时候也只讲了几个简单例子。这个包非常漂亮，几乎年年更新。这里主要讲一下几个预先定义好的环境。

5.3.1 单个、上下和并排

单个很简单

```
\begin{tcolorbox}[colback=green!5,
colframe=green!35!black, fonttitle=\bfseries,
width=6cm, title=单个]
文字内容
\end{tcolorbox}
```

单个

文字内容

上下放置多个内容，每个小块都带标题

```
\begin{tcolorbox}[width=6cm, colback=red!10,
colframe=red!70!black, colbacktitle=yellow!50!red,
coltitle=black, fonttitle=\bf, title=Solid Laser]
介绍固体激光器
\tcbsubtile[before skip=\baselineskip]{光纤激光器}
解释光纤激光器
\tcbsubtile[before skip=\baselineskip]{半导体激光器}
解释半导体激光器
\end{tcolorbox}
```

Solid Laser

介绍固体激光器

光纤激光器

解释光纤激光器

半导体激光器

解释半导体激光器

上下放置两个内容，用\tcblower语句分开，

```
\begin{center}
\begin{tcolorbox}[colback=green!5,
colframe=green!35!black, width=6cm, title=天大校徽]
\includegraphics[width=3cm]{fig/tju.pdf}
\tcblower
天大校徽的设计理念...
\end{tcolorbox}
\end{center}
```

天大校徽



天大校徽的设计理念...

并排放置两个内容，默认是平均分配，但是可以设置左边内容长度和总长，总长是包含两个内容的全部左右边距的。

```
\begin{center}
\begin{tcolorbox}[colback=green!5, colframe=green!35!black, sidebyside, width=12cm,
lefthand width=3cm, title=天大校徽]
\includegraphics[width=3cm]{fig/tju.pdf}
\tcblower
天大校徽的设计理念...
\end{tcolorbox}
\end{center}
```



5.3.2 数学公式

`\tcboxmath{}`可以放在数学公式环境内。左边是默认效果，右边是修改的结果，大部分设置都需要写到这里面。

```
\tcboxset{}
\begin{equation}
\tcbset{fonttitle=\scriptsize}
\tcbboxmath{f(x)}=
\tcbboxmath[boxsep=2pt,top=0pt,bottom=0pt,left=0pt,right=0pt,sharp
corners,colback=cyan!20]{\sin x}
\end{equation}
```

$$f(x) = \sin x \quad (5-1)$$

`\tcbhighmath{}`用法接近，默认配色不同，可以通过 `highlight math style` 统一设置形式。可以与 `empheq` 包混用。

```
\tcboxset{highlight math style={tcolorbox参数}}
\begin{empheq}[box=\tcbhighmath]{equation}
f=\sin x
\end{empheq}
\begin{equation}
\tcbboxmath{f}=\tcbhighmath{\sin x}
\end{equation}
```

$$f = \sin x \quad (5-2)$$

$$f = \sin x \quad (5-3)$$

5.3.3 theorems

加载定理库。

```
\tcbuselibrary{theorems}
```

使用`\newtcbtheorem`命令定义 `tcolorbox` 形式的新的定理环境。

```
\newtcbtheorem[定理参数]{新定理环境名}{显示名}{tcolorbox的参数}{label separator}
```

下面是一个例子

```
\newtcbtheorem[number within=chapter]{solution}{习题答案}{colback=green!5,
colframe=green!35!black, fonttitle=\bfseries}{th}
\begin{solution}{此题主要练习...}{s1} %s1是该定理的标签
一个例子\ref{th:s1} %引用是添加前缀
\end{solution}
```

习题答案 5.1: 此题主要练习...

一个例子5.1

或者使用下面的方式也是等价的。

```
\newtcbtheorem[number within=chapter]{mysolution}{习题答案}{colback=green!5,
colframe=green!35!black, fonttitle=\bfseries}{th}
\begin{mysolution}[label=s2]{此题主要练习...}{s3}
一个例子\ref{s2}两种方式是等价的\ref{th:s3}
\end{mysolution}
```

习题答案 5.1: 此题主要练习...

一个例子5.1两种方式是等价的5.1

5.3.4 tcblisting

加载 listings 库。

```
\usepackage[most]{tcolorbox}
\tcbuselibrary{listings,skins}
```

定义了 tcblisting 环境，listing only 只显示代码；listing side text 一边显示代码，一边显示结果；listing side text，上边显示代码，下边显示结果。tcblisting 环境 xelatex 编译的时候会出现 tab 键显示不正确，pdfatex 编译的时候没有问题。所以使用这个环境且 xelatex 编译时，代码里不要包含 tab 键，尽量使用空格。

```
\begin{tcblisting}{listing only}
listing only只显示代码
\end{tcblisting}
```

\newtcblisting可以定义新的 tcblisting 环境。参数中，side by side，并排方式，不能分页。上下方式加 breakable 参数，可以自动分页。这本书定义了几个环境展示代码和代码结果。基本保持了与 lstlisting 环境显示差不多的效果。top 间距比起 2021 的手册，修改为 8pt。

```
\newtcblisting{pcode}{bicolor,colframe=white,boxsep=2pt,top=8pt,bottom=0pt,left=0pt,
right=0pt,sharp corners,sidebyside gap=5mm,colback=white,colbacklower=white,righthand
width=0.3\textwidth,listing side text,sidebyside align=center seam}

\newtcblisting{wcode}[1][]{bicolor,colframe=white,boxsep=2pt,top=0pt,bottom=0pt,left=0
pt,right=0pt,sharp corners,sidebyside gap=5mm,colback=white,colbacklower=white,listing
side text,sidebyside align=center seam,righthand width=#1}
```

```

\newtcblisting{vcode}{skin=enhanced,breakable,bicolor,colframe=white,boxsep=2pt,top=8
pt,bottom=0pt,left=0pt,right=0pt,sharp corners,sidebyside gap=5mm,colback=white,
colbacklower=white,middle=2pt,listing and text}

\newtcblisting{cpcode}{bicolor,colframe=white,boxsep=2pt,top=0pt,bottom=0pt,left=0pt,
right=0pt,sharp corners,sidebyside gap=5mm,colback=white,colbacklower=yellow!70!gray
!50,righthand width=0.4\textwidth,listing side text,sidebyside align=center seam}

\newtcblisting{cwcode}[1][{}]{bicolor,colframe=white,boxsep=2pt,top=0pt,bottom=0pt,left
=0pt,right=0pt,sharp corners,sidebyside gap=5mm,colback=white,colbacklower=yellow!70!
gray!50,listing side text,sidebyside align=center seam,righthand width=#1}

```

5.4 动图: animate

重点需要讲的是动图，也最容易出错。个人认为，animate 包最好结合 standalone 使用，然后用超链接方式将动图的 pdf 加载到文件或 ppt 里。因为 animate 的编译非常耗时。还有一个办法就是做好动图后注释掉，最终版本再加载。这样可以节省修改文件其余部分的编译时间。当然，下面的例子都是可以放在文章、书或 beamer 类里的。

5.4.1 利用多张图片实现动图

这种方式跟 gif 动态类似，实质上都是多张图片依次播放。此时必须加载图片包 graphicx。

```

\documentclass[animate]{standalone} %animate可以不加
\standaloneconfig{border=5pt}

\usepackage{graphicx} %必须有图片包
\usepackage{animate}

\begin{document}

\animategraphics[width=12cm,autoplay,loop]{12}{../fig/fiber_}{0}{23}

\animategraphics[width=12cm,controls,loop]{2}{../fig2/f}{1}{5}

\end{document}

```

\animategraphics的说明如下:

- 第一个是方括号: 设定动图参数。
自动播放动图使用参数 autoplay, 带播放按钮不自动播放动图使用参数 controls。参数 loop 是循环播放的意思, 否则只播放一次。一般情况总是 autoplay 和 loop 同时使用。controls 随意。如果图片过大, 需要设定宽度缩小一下。不推荐同时设定宽度和高度, 设定一个, 另一个会自动按比例缩放, 这样不会扭曲图片。但是如果图片的尺寸不一致, 宽度设置可能会带来问题, 不设置效果也不好。也可以设置缩放比例 scale, 对相同尺寸的图片效果一样好。当然, 一般多张图片产生的动图都是相同的尺寸。
- 第二个是花括号, 给出每秒多少帧。

- 第三个是花括号，给出图片名称。

图片名称必须包括路径，例子里图片并没有放在当前目录下，而是放在父目录下两个不同子目录里。真正的图片名称的规则是：相同的文件名和编号。此处给出的图片名是相同的文件名那部分，不带编号。第一个例子是推荐的下划线命名规则，比较清晰。第二个例子图片的名字就是简单的 f1、f2、f3。**动图所需的图片名必须连续编号**，但是可以不从 0 开始。起始编号可以任意。pdf_{late}x 编译方式允许的图片格式都可以使用，pdf、jpg 和 png 都行。不用给出具体的图片格式。**图片格式可以混用。**

- 第四个是花括号，给出起始编号。
- 第五个是花括号，给出终止编号。

需要起码两次编译。支持 pdf_{late}x 和 xel_{ate}x 编译方式，直接得到 pdf 文件。

必须使用带 javascript 功能的 pdf 阅读器才会出现动图效果，比如 adobe 的 acrobat reader，这个是免费的软件，最新版一般默认支持 javascript，不用再去设置。大部分小巧快速的 pdf 阅读器都无法展现动图效果，包括 texlive 自带的 pdf 阅读器也不行。必须切换到 adobe reader 查看动图效果。

adobe reader 是不支持自动更新的，因此修改并再次编译 tex 文件时，必须先关闭 adobe reader，编译两次后再用 adobe reader 打开查看。

除了 adobe reader 外，手册里还给出了 KDE Okular, PDF-XChange, Foxit Reader 三种 pdf 阅读器，笔者没有测试过。

其他参数

- 切边

有时候因为图片的问题，上面的语句加载会出现不希望的边框，可以用下面的参数剪裁

```
\animategraphics[width=12cm,trim=0.1cm 0.1cm 0.1cm 0.1cm,controls,loop]{2}{fig2/f}{1}{5}
```

个人觉得 trim 比较好用，切边，顺序是左下右上。width 决定最后的宽度。

还可以使用的参数是 bb 和 viewport，bb 是 bounding box 的意思。

```
\animategraphics[width=12cm,bb=0.5cm 0cm 22cm 10cm,controls,loop]{2}{fig2/f}{1}{5}
```

bb 和 viewport 是设置出图片的 bounding box，只是基准略有差别。

- palindrome

这个参数可以使得动画先顺序播放一遍，然后再逆序播放一遍。

- step

点击一次播放一页。此时加 controls 按钮没什么用处，autoplay 即使设置也无用。loop 参数还是有效的。

- autopause 和 autoresume

关闭动画的 pdf 文件后重新打开文件，前者是从第一页开始，后者是从退出时开始。

还可以设置 button 的颜色、尺寸、是否全部加载控制选项等。这些个人感觉保持默认也挺好。

5.4.2 tikz 直接画动态图

这小节讲的是一张图片中有动态模块这样的情况。比如说下面的例子。

```

\documentclass[animate]{standalone}
\standaloneconfig{border=0pt}

\usepackage{xcolor}
\usepackage{graphicx}
\usepackage{animate}

\usepackage{tikz,pgf}
\usetikzlibrary{shapes.arrows,calc}

\begin{document}

%必须先画个框，能把所有图框起来，这样图形才能稳定住，才能动起来
\begin{animateinline}[autoplay,loop]{12}
\multiframe{24}{iAngle=0+15}{\begin{tikzpicture}
\draw[line width=0.1pt, dashed] (-1.4,-1.2) rectangle (4.6,1.2);
\draw (0,0) circle (1);
\draw [color=cyan] (3,0) circle (1);
\coordinate (a1) at (\iAngle:1);
\fill[red] (a1) circle (4pt);
\coordinate (a2) at ({3+cos(-\iAngle)},{sin(-\iAngle)});
\shade [ball color=blue] (a2) circle (4pt);
\end{tikzpicture}}
\end{animateinline}

\end{document}

```

请使用 adobe reader 观看下面的效果:

另一个例子是加载图片并使图片旋转:

```

\documentclass[animate]{standalone}
\standaloneconfig{border=0pt}

\usepackage{xcolor}

\usepackage{graphicx}
\usepackage{animate}

\usepackage{tikz,pgf}
\usetikzlibrary{shapes.arrows,calc}

\begin{document}

%框的颜色可以设为white，框就不会显示出来
%图片是圆的，框的尺寸不能直接按直径，需要略大于2cm方块旋转的外轮廓尺寸。

```



```
%可能是graphics还有一个外框的缘故
\begin{animateinline}[autoplay,loop]{6}
\multiframe{24}{iAngle=0+15}{\begin{tikzpicture}
\draw[color=white,dashed] (-2,-2) rectangle (2,2);

\node [rotate=\iAngle] at (0,0) {\fbox{\includegraphics[width=2cm]{fig/tju.pdf}}};
\end{tikzpicture}}
\end{animateinline}

\end{document}
```

这里加了一个图片外框方便估计旋转时可能扫过了面积大小，外框需要比这个面积更大一些。可以加`\clip`语句保证旋转的面积只是圆本身。

5.4.3 tikz 多张画图方式

这小节的例子是用 tikz 画多种很不相同的图片，每个图片是一个 newframe。

```
\documentclass[animate]{standalone}

\usepackage{xcolor}
\usepackage{graphicx}

\usepackage{tikz,pgf}
\usetikzlibrary{calc}
\usepackage{animate}

\begin{document}

\begin{animateinline}[autoplay,loop,begin={\begin{tikzpicture}\useasboundingbox
(-0.2,-0.5) rectangle(3.3,0.8);},end={\end{tikzpicture}}]{1}

%后接多行tikz语句构成一幅图，每张图都不一样
\foreach \x in {0,1,2}
\fill (\x,0.5) circle (2pt); %三个点

\newframe \node at (0,0) [fill=red!20] {A}; %红色方块，写到一行也可以

\newframe
\node at (1,0) [fill=yellow!20] {B}; %黄色方块

\newframe
```

```
\node at (2,0) [fill=blue!20] {C}; %多行语句不需要加花括号，两个方块
\node at (3,0) [fill=green!20] {C};
\end{animateinline}

\end{document}
```

第一个 newframe 前的内容最先展示，第二个展示内容才是第一个 newframe 的内容。相当于省去了一个 frame。

5.4.4 时序控制

这一小节讲述通过时序文件控制显示顺序，组合加载 newframe 里的内容。首先是主 tex 文件如下：

```
\documentclass[animate]{standalone}

\usepackage{xcolor}
\usepackage{graphicx}

\usepackage{tikz,pgf}
\usepackage{pgfplots}
\usetikzlibrary{calc}
\usepackage{animate}

\begin{document}

%timeline=时序文件名
\begin{animateinline}[autoplay,loop,begin={\begin{tikzpicture}\useasboundingbox
(-0.4,-0.3) rectangle(2.3,0.3);},end={\end{tikzpicture}},timeline=a.timeline.txt]{2}
\newframe \node at (0,0) [fill=red] {A};
\newframe \node at (0.5,0) [fill=yellow] {B};
\newframe \node at (1,0) [fill=green] {C};
\newframe \node at (1.5,0) [fill=blue] {D};
\newframe \node at (2,0) [fill=cyan] {T};
\end{animateinline}
\end{document}
```

windows 系统下，时序文件名建议扩展名设置为 txt，linux 下可以不加任何扩展名。时序文件名本身随意。这里因为 tex 是 a.tex，所以时序文件名设置为了 a.timeline.txt。文件如下：

```
::0
*:1,2,3,4
:1:c,1
::2
::c,3
```

```

::c,4
::c,2x4,3,4
::1,4
::c,5
::1

```

上面的时序文件只有代码，timeline 的格式为

```
[*]:速率:frame页ID x 动画页数:代码
```

最前面的星号表示停顿，如果加星，该行显示后会停顿等待鼠标响应后再开始动画。如果是动画的最后一页加星停顿，且设置了循环 loop，循环会覆盖停顿，自动循环。

如果设置速率，会覆盖默认速率，对后面所有的动画页有效，直到再次开始循环，开始循环时回到默认速率。

c 表示清除。

2x4 表示对第 2 个 frame，也即 B，显示 4 页。2x0 则表示一直存在直到清除。仅仅只有 2，使用 2x1，仅当前页存在。

上面的例子没有代码，所以连后面的冒号一起省略。

最后的结果是：

1. 第一行，显示一个白色背景，可以没有。
2. 第二行，显示 1234，即 ABCD，停顿等待鼠标响应。
3. 第三行，c 表示清除掉前面所有的内容，显示 1，即 A。从这里开始速率为 1 秒 1 页。
4. 第四行，显示 2，即 B。默认只显示一页，所以不用加清除也只会显示 B。
5. 第五行，清除掉所有的内容，显示 3，即 C。
6. 第六行，清除掉所有的内容，显示 4，即 D。
7. 第七行，清除掉所有的内容，显示 234，即 BCD，设置 B 显示 4 页。
8. 第八行，显示 14，加上前面的 B 设置为在 4 页显示，所以即使没有写显示 B，得到的是 ABD。
9. 第九行，只显示 T，前面的 c 覆盖了 2x4 的 4 页显示设置，所以 B 不再显示。
10. 第十行，只显示 A，第九行的 c 对后面都有效，虽然 B 设置显示 4 页，实际只会显示 2 页
11. 循环

timeline 文件用分号分层，如果改写 timeline 文件为如下形式

```

::0
*::1,2,3,4
:1:c,1
::2
::c,3
::c,4
::c,2x4,3,4
::1,4x0;3x0
::c,5
::1

```

C 就会在后面一直存在，因为::c,5 只清除第一层，第二层的 3x0 会一直存在。如果写为

```
::c,5;c
```

则 C 就不存在了, 因为清除了两层, 只会留下 T。

5.4.5 转换成 svg 动图

dvisvgm 选项必须加在类参数上

```
\documentclass[dvisvgm]{standalone}
```

编译的时候使用 xelatex 编译

```
xelatex --no-pdf a.tex
```

得到 a.xdv 文件, 然后用 dvisvgm 命令转换

```
dvisvgm --zoom=1 --exact --page=1,- --font-format=woff2 a.xdv
```

得到的 svg 图片用浏览器打开, 可以看到矢量高清的动图。手册上是 `--zoom=-1`, 会充满整个 web 页面, 好像没有切边似的, 所以修改了这个值。如果觉得 `zoom=1` 太小, 可以改成其他值。新版本不用加上面的任何参数就可以转换成功。

也可以用 latex 编译, 得到的是 a.dvi, 转换命令一样, 此时转换的文件改为 a.dvi 即可。

5.5 试卷: exam 类

试卷 exam 类很方便教师排版试卷, 排版时带着答案排版, 编译时通过包的 `answers` 选项控制答案是否显示。

在网址<http://ull.tju.edu.cn/education/course/latex.html>放了一个天大试卷 8 开的样本 tex 文件。

天大本科生试卷基本样式

```
\documentclass[oneside,twocolumn,12pt]{exam} %空白试卷
\documentclass[answers,oneside,twocolumn,12pt]{exam} %显示答案
```

天津大学的本科生试卷一般是 8 开双栏页面

```
\usepackage[paperwidth=37cm,paperheight=26cm,top=3.6cm,bottom=1.8cm,left=1.7cm,right=1.7cm,bindingoffset=0.5cm]{geometry}
```

这里的边距设置是笔者目测出来的。天大的 word 模板, 前面居中的信息不是页眉做的, 就是直接居中的两行文字, 下面加了一个表格黑框。从实用的角度, 左右边距大一点更好, 最好的方式是分数批在边框外。实际装订效果看, 这里装订线边距设的有点小了, 但是也勉强够用, 终究是要尽量扩大卷面内容的面积才好, 这样放图表的题目时能够节省纸张。

有一个黑框, 用 `background` 包和 `tikz` 包做比较简单

```
\backgroundsetup[scale=1,contents={\tikz\draw [line width=1.7pt,black] (0,0)rectangle
++(33.5,21.1);},opacity=1,angle=0,hshift=5pt,vshift=-26pt}
```

但是这个做法有一个问题, 天大升级模板后页眉一部分字体改为黑体, 导致必须 xelatex 编译, 生成的 pdf 文件是高版本 pdf 格式, 打印卷子的地方 pdf 阅读器是 9.0 版本的, 太老旧

了, 无法正确显示第一页。同样的代码, 如果没有黑体, 仅仅是宋体, 用 pdf_latex 编译就完全没有问题。解决这个问题方法有很多种, 一个是第一章增加了 pdf 版本降级的方法, 一个是前面增加一个空白页并重置页码, 笔者现在用的是比较麻烦的 eso-pic 包和计算包 calc。这里的 calc 是计算包, 必须存在。这种方法背景黑框没有分层, 打印卷子那里可以正常显示所有内容, 代码就是略微复杂而已, 其实比 background 还要容易理解黑框的放置位置。因为 eso-pic 的初始坐标严格在页面左下角为 (0,0), 所以更容易计算出来黑框左下角的坐标。注意这里的\@tempdima和\@tempdimb不能更换名称。

```
\usepackage{eso-pic,calc}
```

下面的语句写在 document 之前

```
\makeatletter
\AddToShipoutPicture{%
  \begingroup
    \setlength{\@tempdima}{\textwidth+4mm}%计算黑框的长度
    \setlength{\@tempdimb}{\textheight+5mm}%计算黑框的高度
    \linethickness{2pt}
    \put(\LenToUnit{20mm},\LenToUnit{15mm}){%黑框左下角开始坐标
      \framebox(\LenToUnit{\@tempdima},\LenToUnit{\@tempdimb}){}}%
    \endgroup
}
\makeatother
```

先定义一下字体大小, 用到的是小四、小三和四号。

```
\newcommand{\xiaosi}{\fontsize{12pt}{14pt}\selectfont}
\newcommand{\sihao}{\fontsize{14pt}{16pt}\selectfont}
\newcommand{\xiaosan}{\fontsize{15pt}{17pt}\selectfont}
```

document 之前加竖直方向文字居于顶部

```
\raggedbottom
\begin{document}
```

exam 类包含了 fancyhdr, 所以直接用下面的语句就可以设置页眉, 页眉是四号字体。

```
\chead{\sihao 天津大学试卷专用纸\[\[6mm]{\heiti 学院\underline{\hspace{3.1cm}}专业\
underline{\hspace{5cm}}\hspace{1cm}\underline{\hspace{1cm}}班\hspace{1cm}年级\
underline{\hspace{2cm}}学号\underline{\hspace{5.5cm}}姓名\underline{\hspace{3cm}}}\hspace{1cm}共~~\protect\pageref{lastpages}~~页\hspace{1em}第~~\thepage~~页}

\pagestyle{head}
```

为了自动更新总页码, 这里在最后一页添加了一个 label, 可以自动显示共几页。

```
\label{lastpages} %最后一页添加一个label
\clearpage %这个命令必须存在
```

其他的设置基本是目测调出来的, 有些间距的细节做了一点调整, 主要是个人的美感不同。考试科目部分设置为 1.7 倍行距。也可以使用单倍行距, 然后采用\[\[距离]方式。

```
\begin{spacing}{1.7}
```

```
\centering\bf\heiti\xiaosan
\large 2024\ $\sim$ 2025 学年第~~1~~学期期末考试试卷\
《XXXX》(共~~\pageref{lastpages}~~页)\
\sihao(考试时间: 2024 年 11 月 22 日)
\end{spacing}
```

从这里开始字体都是小四, 回到正常的 12pt 字体即可。分数表部分和试卷的填空、选择部分设置为 1.5 倍行距以上, 排版时按需要调整。填空题用`\fillin[答案][横线长度]`命令。

```
\begin{spacing}{1.5}
\xiaosi
\vspace*{-1mm}
\setlength{\tabcolsep}{12pt}
\setlength{\extrarowheight}{2pt}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}\hline
题号 &一 &二 &三 &四 &五 &六 &七 &八 &成绩 &核分人签字\ \hline
得分 &&&&&&&&&&&\hline
\end{tabular}

\vspace{5mm} %这个间距目测调一下
一、填空(每空1分, 共20分) %1.5倍到填空题结束
\end{spacing}

\begin{spacing}{1.25}
二、其他类型题目等
\end{spacing}
```

笔者推荐在证明计算题前面放一个物理常数表, 省得每道题目中列出。这种情况后面的大题部分, 第一题建议是一个简单的小分值的题目, 扣除了数据表格外, 仍然半页可以放下的那种。证明计算题如果分小题, 个人喜欢题目部分直接强制回车手动标注序号, 这样空间上比较节省, 而答案部分使用 `itemize` 环境, 比较清晰。

常见题目类型

exam 类题目的基本框架是

```
\begin{questions}
\question %每小题目自动编号, 必须有一个question才不报错。
\end{questions}
```

常见题型也就下面这几种

■ 填空题

过程需要满足`\fillin[能量]`守恒和`\fillin[动量]`守恒。

默认有填空线长度(个人喜欢下划线), 可以临时手动调整

`\fillin[答案][12cm]`

■ 选择题

推荐采用下面的填空方式写答案, 更容易判卷。

```
\question
下面哪种描述是正确的? \fillin[ABD]
\begin{choices}
\choice 陈述1
\choice 陈述2
\choice 陈述3
\choice 陈述4
\end{choices}
```

选择题建议使用 1.25 倍行距压缩内容所占空间，但是期中小考的时候我经常填空和简答混出，所以小考的试卷我设置为 2 倍行距。填空题因为要留给学生写答案，1.5–2 倍行距比较合适。

■ 单行选项

如果选项字数比较少的话，推荐使用 `oneparchoices` 环境把选项放在同一行，节省空间和纸张。

```
\question
下面哪种食品有利于健康\fillin[AB] %下面需要空行或强制换行

\begin{oneparchoices}
\choice 苹果
\choice 牛肉
\choice 炸薯条
\choice 炸鸡
\end{oneparchoices}
```

■ 勾选方式的选择题

```
\question
下面哪种食品有利于健康?
\begin{checkboxes}
\CorrectChoice 苹果 %这条是正确选项，对上面的选择题例子也适用
\CorrectChoice 牛肉 %这条是正确选项
\choice 薯条
\choice 炸鸡
\end{checkboxes}
```

默认是对勾为正确答案，如果不显示答案，即空白试卷时，正确答案前面仍然是空心圆期待选标识。

■ 选项在同一行的情况

```
\question
下面哪种食品有利于健康? %下面需要空行或强制换行

\begin{oneparchcheckboxes}
\CorrectChoice 苹果
\CorrectChoice 牛肉
\choice 薯条
\choice 炸鸡
```

```
\end{oneparcheckboxes}
```

■ 简答题

简答题和证明计算题需要事先计算好宽度和高度, 采用 `parbox` 方式, 这样答案是否显示都不影响版面。

```
\parbox[] [4cm] [t] {0.47\textwidth}{
\question
为什么?

\begin{solution}
答案
\end{solution}
}
```

如果有图, 可以采用两种方式, 图在上面, 或者图和答案并排。我自己的试卷一般采用上下方式, 不用修改上面的代码。并排可以采用 `minipage` 环境, 也可以采用 `parbox` 嵌套方式, 下面是 `minipage` 方式的代码。此时仍然需要将整个题目 (包括答案) 用 `parbox` 封装起来。

```
\parbox[] [8cm] [t] {0.47\textwidth}{
\question
\begin{minipage}{0.27\textwidth}
\includegraphics[width=5cm]{../exam-fig/}
\end{minipage}
\begin{minipage}{0.2\textwidth}
\begin{solution}
答案
\end{solution}
\end{minipage}
}
```

证明题半页 (双栏时是半页) 写不下答案时, 可以将答案拆分到两个答案环境内, 手动分页。

```
\begin{lstlisting}
\parbox[] [20cm] [t] {0.47\textwidth}{
\begin{solution}
写不下的答案
\end{solution}
}
```

`solution` 环境支持分页, 这里需要手动调整的原因是 `\parbox` 不能分页 (换另外半页)。这是为了有答案和无答案时, 试卷版面各题位置严格不动引入的解决方案。布置作业时不需要 `\parbox`, 不存在拆分答案到两个 `solution` 环境的问题。

常用的设置语句

排出期末卷子, 常用设置可以分为两类。题目本身的设置和题目列表版面的设置。

- 设置默认填空横线长度,


```
\setlength\fillinlinelength{5cm}
```

- 横线向下沉一点距离

```
\setlength\answerclearance{0.5ex}
```

- 答案不加粗（个人觉得加粗不好看）

```
\CorrectChoiceEmphasis{\rmfamily}
```

- 题目间额外间距

这里需要说明的是, exam 的每道题自动有一点竖直间距 (`\itemsep`), 这个是天大模板没有的。这比较方便老师们隔开每道题目, 所以这里没有修改为天大模板的样式。修改这个间距, 手册里的语句没有成功, 就直接在 `questions` 环境内设置 `\itemsep` 就可以。

```
\begin{questions}
\setlength{\itemsep}{间距} %latex默认有间距, 数值为负才能取消间距
\question
\end{questions}
```

- 答案不加框（平时讲作业可以加框, 但是期末试卷个人觉得不加框好看。）

```
\unframedsolutions
```

- 不添加答案 title。默认是英文 Solution

```
\renewcommand{\solutiontitle}{\noindent}
```

题目的版面设置, 指调整题目序号对齐位置、题目本身起始文字对齐位置、选择题列表序号或勾选的对齐位置。

- 调整每个题型下小题目的缩进量

```
\renewcommand{\questionshook}{%
\setlength{\leftmargin}{2em}%
\setlength{\labelsep}{2ex} %可能这样更好看一点
}
```

因为题型的要求是大写汉字数字和中文顿号, 占 2em, `\leftmargin` 设置为 2em, 题目汉字起始与题型汉字起始对齐, 个人觉得这样整齐一些。`\labelsep` 调整的是每道题序号与题目内容的间距, 2ex 使得 1 对齐大写的一, 个人觉得挺好看。黑框和文字有一点间距, 如果题目序号两位数, 也不会触及黑框。

- 调整选择题选择项的缩进量

默认的好处是清晰, 缺点是占地方。有时候是描述型的选项, 就希望尽量省一些空间, 下面是按这个想法做的设置。

```
\renewcommand{\choiceshook}{%
\setlength{\leftmargin}{2em}% 内容缩进距离
\setlength{\labelsep}{2.0ex} %这里ex单位
}
```

这里的 2em 使得选项比题目缩进 2em, 选择题默认 ABCD 编号, 后一个值是调整编号到

选项文字的间距。下面勾选题目类似。

需要说明的是，单行情况与上面的设置无关，单行本质是换行。

■ 调整勾选选项的缩进量

```
\renewcommand{\checkboxeshook}{  
\setlength{\leftmargin}{3em}  
}
```

exam 类还包含了计分表，因为不使用就没有研究。设置语句还有很多，有需求的时候可以查阅手册。

5.6 其他一些不务正业的包

还有一些非常优秀的非理工科专业的包。

musixtex 五线谱包

mtx 五线谱和歌词，更简单一些

xpiano 钢琴键盘

horoscop 占星术

skak 国际象棋包

psgo 围棋包，基于 pstricks

sudoku 数独

sudokubundle 数独

qrcode 生成二维码

第六章 化学分子式 chemfig

chemfig 包是基于 tikz 的，单纯使用 chemfig 时，加载 chemfig 包并不需要额外加载 tikz 包。如果导言中还加载了 tikz 包，加载 chemfig 包时也不会产生错误。但是一些 tikz 的子库，比如 calc，是没有预先加载的，还需要自行添加。一些符号需要 amssymb 包。chemfig 功能强大，可以在其中使用 tikz 画图语句，对于熟悉 tikz 的人来说非常友好。

6.1 基本设置

加载包后可以修改一下基本设置，这是纯属个人感观。chemfig 的设置方式随着版本有较大变化，可能有些老的设置语句会逐渐退出。基本设置写在导言区比较好，但是可以临时修改基本设置。写在花括号内的设置，只对花括号内的语句临时有效，否则对后面的语句都有效。

```
\usepackage{chemfig}

%这版的设置单位都改成em了，这样改变字号时结果不会产生太大的变化。
\setchemfig{
atom sep=3em, %设置键连接的两个原子的间隔
bond offset=2pt, %设置原子和键之间的间隔，必须是pt
double bond sep=0.25em, %设置键之间的宽度，指双键、三键间宽度
bond style={line width=0.08em}, %设置键的线宽，还可以设置颜色，默认黑色
cram width=0.5em, %设置楔形角度
cram dash width=0.1em, %线宽
cram dash sep=0.15em, %间距
}
```

常用的设置有两个点必须说明，一个是 `bond offset=2pt`，这里不能是 `em` 这样的相对单位，必须是 `pt` 这样的绝对单位。上面其他的参数都可以是相对单位。另一个是下面这个参数，默认是 `false`，如果是 `true`，有些时候会造成键的错位，需要谨慎。

```
fixed length=false, %不设置就是false，修改为true会造成一些代码时键错位
```

参数设置可以写到紧跟着 `\chemfig` 命令的方括号里，仅对这条分子代码有效。

```
\chemfig[参数=值]{分子代码} %方括号里设置局域有效
```

`\chemfig{}` 视为文字，所以设置原子字符大小，直接使用设置字号的语句，这个是外置命令，不是 chemfig 的设置参数。

```
\small %写到document之前setchemfig里不会起作用
```

article 类或 book 类使用时，加局域花括号以免影响正常文字。

```
{\small \chemfig{ } }
```

chemfig 可以直接在 article、book 和 beamer 中使用，如果需要在 standalone 类中使用，可选择类参数为 tikz 或 chemfig，这两个参数得到的结果是不同的。

```
\documentclass[tikz]{standalone} %每一个chemfig一页
```

上课做练习用的是 tikz 选项，每个分子式一页，即每个 `\chemfig{}` 一页。或者使用 `\schemestart` 多个语句 `\schemestop` 方式。直接的 tikz 选项对高分子不适用，必须使用 chemfig 选项。tikz 选项对有 `\chemmove{}` 语句的情况也不适用。

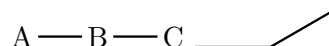
```
\documentclass[chemfig]{standalone} %所有的chemfig都在一页
```

如果希望一张图中画多个分子式，但是图片单独一页，可以采用这种方式。此时并不类似 tikz 选项对 tikzpicture 环境的作用，选择 chemfig 参数只会生成一张 pdf。

高分子式，情况比较复杂。可以采用 tikz、chemfig 和 minipage，根据不同的情况，某种做法可能更简单。另外，希望产生单独的一张分子式且图文混排有文字说明的结果，也可以设置 standalone 类的选项为 minipage，这种方式对高分子也适用。也可以采用 tikzpicture 环境加载 node 的方式。

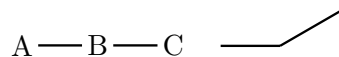
在文章或书中使用，或者 standalone 类选项选择 chemfig 时，并排放置分子式，竖直方向的对齐方式是按起笔原子位置对齐的，手册称之为首原子规则。更准确的描述应该是起笔原则，因为如果是键起笔，就不会跟原子起笔的分子的键在竖直方向上对齐。

```
\chemfig{A-B-C} \chemfig{--[:30]}
```



当画比较复杂的分子反应式时，需要充分考虑到这一对齐规则，规划好起笔原子。有时候需要引入不显示的原子帮助对齐。

```
\chemfig{A-B-C} \chemfig{\phantom{A}--[:30]}
```

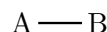


6.2 键的基本语法

化学键类型

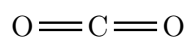
- 单键是连字符，

```
\chemfig{A-B}
```



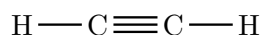
- 双键是等号

```
\chemfig{O=C=O}
```



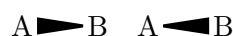
- 三键是波浪号

```
\chemfig{H-C~C-H}
```



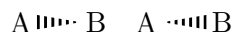
- 楔形键是大于号和小于号

```
\chemfig{A>B}\quad\chemfig{A<B}
```



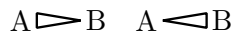
- 虚线样式的楔形键是大于号和小于号组合冒号

```
\chemfig{A>:B}\quad\chemfig{A<:B}
```



- 中空样式的楔形键是大于号和小于号组竖线合

```
\chemfig{A>|B}\quad\chemfig{A<|B}
```

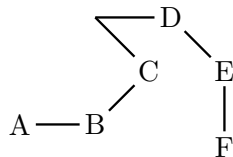


键的第一个参数：键角

键参数写到键的后面，对该化学键有效。键的第一个参数是角度，

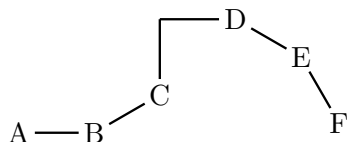
- 预先设定了 8 个方位，0-7，0 表示水平向左，即 0 度，是键角的默认值，可以不写。

```
\chemfig{A-B-[1]C-[3]-D-[7]E-[6]F}
```



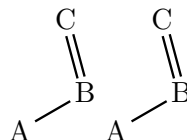
- 预设的方位相当于增量为 45 度，可以修改增量，一般不推荐用此种方式。如果设置为 30 度，就是 $360/30=12$ ，即 0-11 个方位，

```
\chemfig[angle increment=30]{A-B-[1]C-[3]-D-[11]E-[10]F}
```



- 如果不使用默认方位，直接写角度就可以，角度范围是 $[-360, 360]$ 。度数前单冒号是绝对角度，双冒号是相对角度。下面的例子结果是一样的。

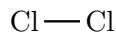
```
\chemfig{A-[:30]B=[:105]C}
\chemfig{A-[:30]B=[:75]C}
```



键的第二个参数：键长

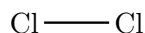
如果是两个字母的原子，默认设置使得原子间距固定，导致键长太短，如下

```
\chemfig{Cl-Cl}
```



如果设置为 true，键长是固定的，有些结构会引发扰乱，不推荐使用。

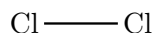
```
\chemfig[fixed length=true]{Cl-Cl} %方括号里设置局域有效
```



设置键长为固定值，旧版本的`\chemfig*{}`不再适用。

键长可以任意设定，设定方式是默认长度的倍数。这样可以在不修改`fixed length`情况下，拉长键长。键长是键的第二个参数，键的第一个参数是角度，所以设置键长时如果不需要设定角度，前面的逗号不能省略。

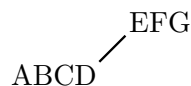
```
\chemfig{Cl-[1.5]Cl}
```



键的第三和第四个参数：连接的前后分子序数

键的第 3 个和第 4 个参数是连接的分子序数，此时不设定键长时，逗号不能省略。下面的例子，ABCD 的第二个分子是 B，EFG 第三个分子是 G。不设定连接的分子，默认连接分子会随键的方向变化。45 度时为 D-E，90 度时为 A-E。

```
\chemfig{ABCD-[1]EFG}
```



```
\chemfig{ABCD-[2]EFG}
```



所以需要设定哪两个原子连接，例如设定 B 连接到 G 时为

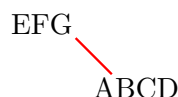
```
\chemfig{ABCD-[1,,2,3]EFG}
\chemfig{ABCD-[2,,2,3]EFG}
```



键的第五个参数：其他 tikz 参数

可以是键的颜色，线宽，线型。

```
\chemfig{ABCD-[3,1.2,2,3,red]EFG}
```



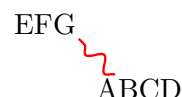
复杂分子讨论动力学时这个功能很有用，可以标记重点讨论的键为不同颜色。

还可以利用 tikz 的修饰库做蛇形键，此处要加入 tikz 的子库

```
\usetikzlibrary{decorations.pathmorphing}
```

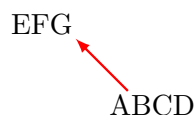
然后将键定义为修饰库里的蛇形

```
\chemfig{ABCD-[3,1.2,2,3,red, decorate, decoration=snake]EFG}
```



还可以画箭头

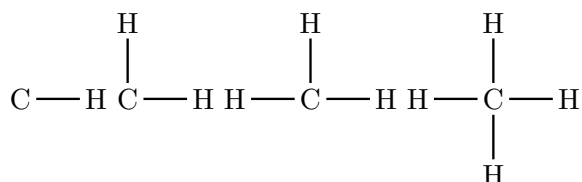
```
\chemfig{ABCD-[3,1.5,2,3,red,-latex]EFG}
```



键的分支

上面的例子都是单链结构，分支用圆括号，表示并在一个原子上的，

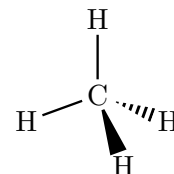
```
\chemfig{C-H}
\chemfig{C(-[2]H)-H}
\chemfig{C(-[4]H)(-[2]H)-H}
\chemfig{C(-[2]H)(-[4]H)(-[6]H)-H}
```



上面的例子很容易看出起笔对齐原则。都是从 C 起笔，所以四个分子的 C-H 在一条水平线上。

立体分子甲烷，C 上并了三个单键，加上原来的一个，共四个键。上面的楔形参数定义的是楔形的形状，< 纸面前，<: 纸面后

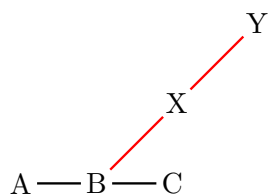
```
\chemfig{C(-[:90]H)(-[:200]H)(<[:290]H)<[:340]H}
```



键的进阶

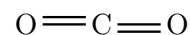
键的参数如果写到键的前面，表示对整个分支都有效

```
\chemfig{A-B([1,1.5,,,red]-X-Y)-C}
```



可以用上下标将键上下平移

```
\chemfig{O=^C=_O}
```

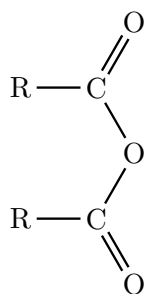


6.3 分子结构

旋转

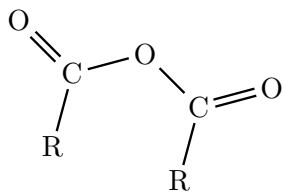
下面是酸酐的例子

```
\chemfig{R-C(=[::-60]O)-[::-60]O-[::-60]C(=[::-60]O)-[::-60]R}
```



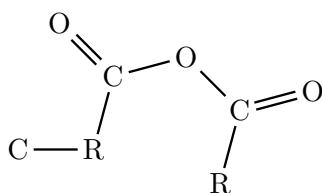
分子最前面加角度，整体旋转后面的分子

```
\chemfig{[:75]R-C(=[::-60]O)-[:--60]O-[:--60]C(=[::-60]O)-[:--60]R}
```



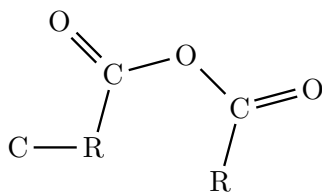
将整体旋转的分子连接到其他分子上，需要一个不存在的分子，第一个例子是将分子 C 设置为白色，不显示，整个酸酐并在这个不显示的 C 原子上，所以酸酐是在圆括号里面的。

```
\chemfig{C-\color{white}C}(:75]R-C(=[::-60]O)-[:--60]O-[:--60]C(=[::-60]O)-[:--60]R)}
```



也可以使用幻影原子``

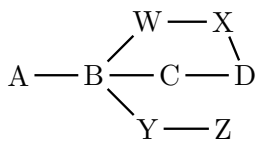
```
\chemfig{C-\phantom{C}(:75]R-C(=[::-60]O)-[:--60]O-[:--60]C(=[::-60]O)-[:--60]R)}
```



问号

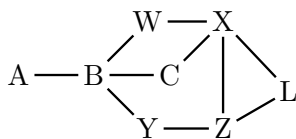
问号表示自动连接两个分子，可以自动成环，X 和 D 自动生成键的连接，键长自动计算出来。

```
\chemfig{A-B(-[1]W-X?)-([7]Y-Z)-C-D?}
```



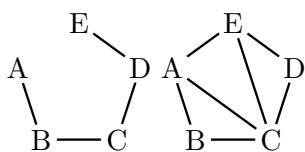
可以是多个分子自动连接

```
\chemfig{A-B(-[1]W-X?)-([7]Y-Z-[:30]L?)-C?}
```



问号可以设置编号连接分子，即多个问号。下面的例子，A 设置了问号 a，C 连接到 A，C 和 E 同时设置了问号 a,b，E 连接到 A 和 C。可以比较一下没有问号时候的分子。


```
\chemfig{A-[: -72]B-C-[:72]D-[:144]E}
\chemfig{A?[a]-[: -72]B-C?[a]?[b]-[:72]D-[:144]E?[a]?[b]}
```



问号还可以设定自动连接的键的种类，语法如下

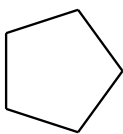
```
?[名字,{键}]
```

是后面的问号位置自动连接到前面的问号位置，所以前面问号位置处键是确定的，后面的问号位置自动连接的键的种类才可能是指定的（默认是单键）。对于复杂分子，使用问号时，画结构的顺序常常是重要的。

环

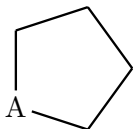
乘号表示环

```
\chemfig{*5(-----)}
```



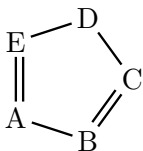
从 A 开始的环，环是等键长的

```
\chemfig{A*5(-----)}
```



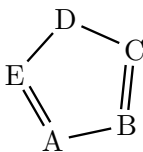
环可以是不同的键构成的

```
\chemfig{A*5(-B=C-D-E=)}
```



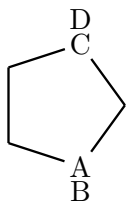
A 前加角度可旋转该环

```
\chemfig{[:30]A*5(-B=C-D-E=)}
```



有时候需要上下加分子，加`\vskip2mm`是因为这个时候会产生按 A 和 C 构成 box 的问题，导致上下间距不足。使用 standalone 类，需要修改上下的留白间距。

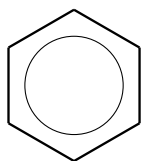
```
\vskip2mm\chemfig{*5(-\chembelow{A}{B}--\chemabove{C}{D}--)}
```



苯环

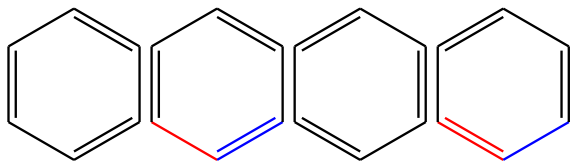
苯环有两种方式表示，一种是中间为圆环

```
\chemfig{**6(-----)}
```



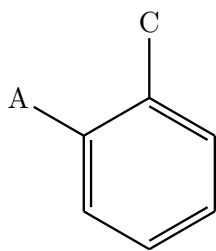
另一种是单双键形式，键按逆时针顺序，第一个键是左下负 30 度方向

```
\chemfig{*6(=====)}
\chemfig{*6(-[,,,,red]=[,,,blue]====)}
\chemfig{*6(=====)}
\chemfig{*6(=[,,,red]-[,,,,blue]====)}
```



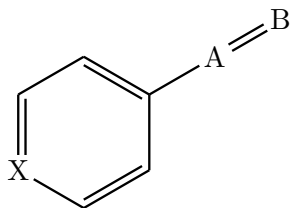
苯环连接到 A 分子，A 的负 30 度方向单键连接到苯环，苯环的第一个键为双键

```
\chemfig{A-[:-30]*6(=====(-C)-)}
```



如果分子在苯环内，分子直接写在乘号前

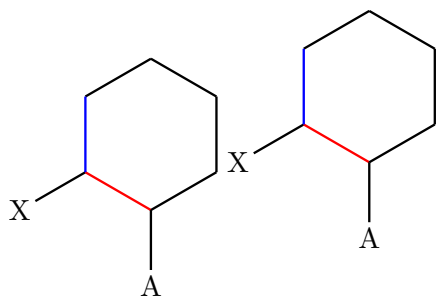
```
\chemfig{X*6(==(-A=B)===)}
```



在环上连原子时，红色起始键后端连 A，用圆括号并在红色键后。如果想在左下红色键前

端连 X，等价于蓝色键后端连 X，似乎应在蓝色键后端并圆括号，但是实际上要在红色键前并圆括号。

```
\chemfig{*6((-X)-[,,,,red](-[6]A)-----[,,,,blue])}
\chemfig{X-[:30]*6(-[,,,,red](-[6]A)-----[,,,,blue])}
```



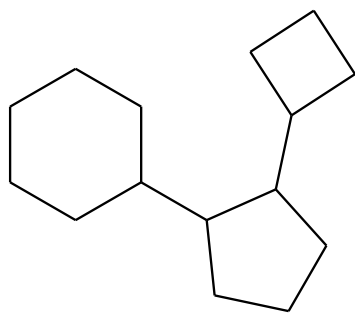
上面的两种写法效果一样，但是图片位置其实是不同的。第一个起笔在红蓝色键交点，第二个起笔在 X。安装起笔对齐，对齐的是 X 的 base 位置。

很多时候，特别是复杂分子时，需要先画出环，定下环结构的方位，第二个例子需要计算 X 连向环的键的键角才能把环放得跟第一个例子一样。虽然第二个例子看上去结构上更简单明了，和手册例子接近，但是其实需要额外的计算量。有些时候是不方便计算键角的，所以第一个例子才是常用的情况。

并环

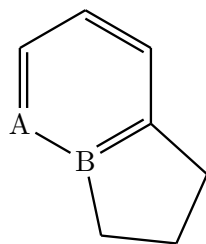
连接是写分支，并环是共享环中的某个键。环通过键连接在一起，环内的键个数是一致的。

```
\chemfig{*6(--(*5(----(*4(----))--))----)}
```



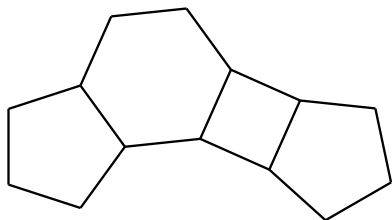
并环，5 环结构不用再写共享那个双键，即 B 后面那个双键，所以并环时五环只写 4 个键，比独立五环少一个共享键。

```
\chemfig{A*6(-B*5(----)=--=)}
```



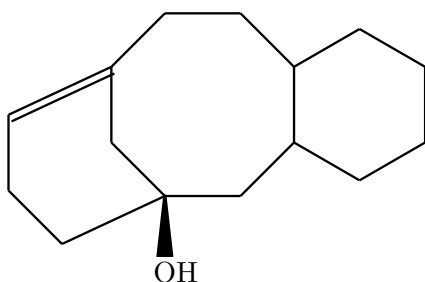
多次并环的一个例子

```
\chemfig{*5(--*6(-*4(-*5(-----)--)-----)----)}
```



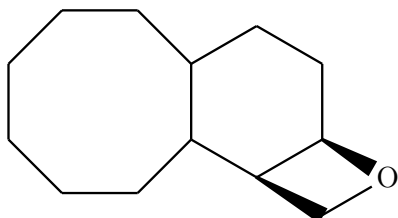
上面的并环都是正多边形的，如果不是正多边形结构，并环就需要设置问号、短键和幻影原子来协助完成。

```
\chemfig{*8(-?[b](<[: -90]OH)--*6(-----) ----(=[:205,1.5]-[: -90]-[: -45]?[b])-)}
```



与并环一样，设置问号自动连成环时少一个键，所以从双键那里分支出来，只有三个键，然后接着是问号。笔者在这个复杂的分子式中也犯过错，问号是自动连接，不需要设置键角和键长，仅仅是有些时候需要设置键的类型，比如

```
\chemfig{*8(---*6(-(<[: -30]-[:30]O?[f])-[f,{>}]---)-----)}
```



因为起笔是正的 8 环，所以左边的四环结构，画结构时必须从 8 环的第 7 个键末端分支，这样左边的四环结构才能正确地自动连接到第 1 个键末端先设定好的问号位置。起笔影响键的顺序，会影响第一个问号的位置，从而影响不规则环结构的起笔和画法。右边的四环结构还有其他画法，这里设置到 O 原子位置了，也可以设置到正 6 环上，右边需要设置键以楔形键方式连接。紫杉醇这样的复杂分子，笔者的实践，只有从这个并环开始画才容易，笔者先画的是 8 环 + 6 环 + 右边的不规则并环，后画的是左边不规则并环，上面的例子将代码拆分了一下，容易看清楚怎么设置问号。

6.4 结构复用

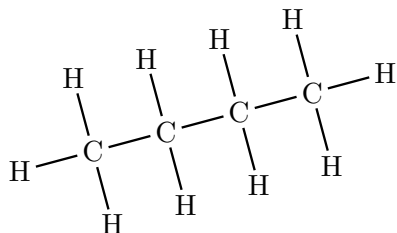
定义 submol 可以复用结构

```
\definesubmol{xy}{CH_2} %定义复用单元
\chemfig{H_3C-!{xy}-!{xy}-!{xy}-CH_3}
```



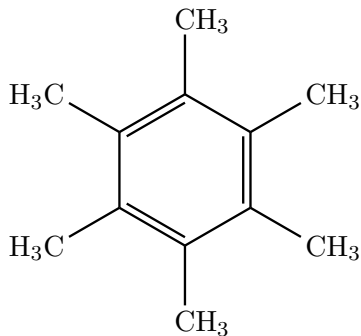
定义替代结构，跟上面本质相同，写法不同，增加可读性

```
\definesubmol\xx{C(-[:90]H)(-[:-90]H)}
\chemfig{[:15]H-!\xx-!\xx-!\xx-!\xx-H}
```



可以方括号定义键从右到分子时的代码，花括号定义键从左到分子时的代码，使用时自动选择

```
\definesubmol\zz[H_3C]{CH_3}
\chemfig{*6((-!\zz)=(-!\zz)-(-!\zz)=(-!\zz)-(-!\zz)-)}
```



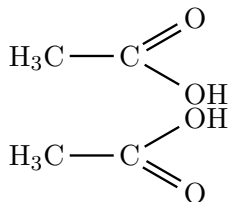
`\definesubmol`语句还可以用于将大分子拆成较小的分子来画，方便构建结构、查找错误。

6.5 对称

`\vflipnext`得到上下对称，中间空行是为了更方便地看两个分子是上下对称的。

```
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}

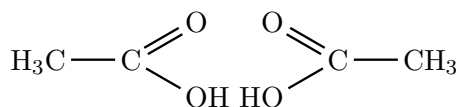
\vflipnext
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}
```



`\hflipnext`得到左右对称，中间不空行，方便看左右对称。

```
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}

\hflipnext
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}
```



6.6 Lewis 形式

新版本废弃了 `lewis`，改用宏 `charge`，`lewis` 形式在 1.6a 及以后的版本中已经不可用。语法规则为：

```
\charge{[参数] 电荷}{原子} %原子省略电荷标在键交点处
\charge{电荷[参数]}{原子} %Charge也可以
```

注意写入参数的方括号在第一个花括号里面，参数是用来设置电荷的。

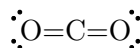
```
\Charge{45=\:,135=\.,225=\",-45=\l}{X}
```



- \. 表示不成键的单个电子，单点
- \: 表示不成键的孤对电子，双点
- \" 是空心细长矩形
- \l 表示成键的共享电子对，单线

比如二氧化碳分子

```
\Charge{135=\:, -135=\:}{O}=C=\Charge{45=\:, -45=\:}{O}
```



参数写到前面对所有的电荷都适用，如下单电子和双电子的点都变大且为蓝色

```
\Charge[.radius=1.5pt, .style={draw=none, fill=blue}]45=\:, 135=\.,
225=\"}{X}
```



参数写到后面则是对当前电荷有效。如下，点可以设置为不同颜色。

```
\Charge{45=\:[.radius=1pt, .style={draw=none, fill=red}]},
135=\., 225=\"[{"width=3pt, "style={fill=green}}]{X}
```

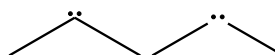


参数是四种电荷形式相关的，

- 单点的参数有 `.radius=` 点半径和 `.style={tikz参数}`。
- 双点复用单点的参数，还有额外的两点间距 `:sep=` 距离。
- 单线就是直接的 `tikz` 参数 `|style={tikz参数}`。
- 空矩形可以定义宽度 `"width=` 距离和 `"style={tikz参数}`。空矩形的长度不超过原子高度，是字号决定的。

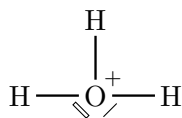
`\Charge{}` 和 `\charge{}` 有区别，大写会覆盖使得键长变短。推荐小写。有原子的时候大小写其实差别不大，关键是没有原子时，应该是小写。

```
\chemfig{-[:30]\charge{90=\:}{-[:30]}}
\chemfig{-[:30]\Charge{90=\:}{-[:30]}}
```



下面是水合氢离子的例子，网上标的是一对孤立电子，跟手册上的不一样。

```
\chemfig{H-\charge{45:1.5pt=$\scriptstyle+$,-45=\|,-135=\"}{O} (-[2]H)-H}
```



从这个例子可以看出，利用`\charge`很容易将符号围着原子放置，也可以将原子围着原子放置

```
\chemfig{\charge{90:3pt=B}{A}}
```



6.7 一些细节

opus 的位置

```
\chemfig{A^\oplus-[2,,1]B}
```



这个写法类似于两个原子，实际上希望键连在 A 上，因此代码调整为

```
\chemfig{A\rlap{$^\oplus$}-[2]B}
```



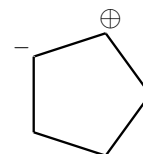
也可以用`\charge`，此时键是连在原子上的，电荷是原子的修饰，使用`\charge`可以对电荷位置精准调控。

```
\chemfig{\charge{40:0.5pt=$\scriptstyle\oplus$}{A}-[2]B}
```



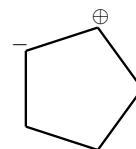
环上时需要设置一个不显示的短键

```
\chemfig{*5(---(-[,0.2,,,white]\oplus)-(-[,0.2,,,white]
\scriptstyle{-})-)}
```



也可以使用`\charge`

```
\chemfig{*5(---\charge{60:1pt=$^\oplus$}{-}
-\charge{135:1pt=$\scriptstyle -$}{-})}
```



从这个几个例子可以看出`\charge`与`\chemfig`键上连接的原子的细节不同。`\charge`里是文字，`\chemfig`里是行间公式，所以`\chemfig`里可以直接使用`\oplus`或`CH_3`，`\charge`里不行。

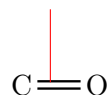
6.8 宏

@ 表示定义宏（这个翻译也不知道对不对了），宏可以定义在键上，此时的语法是

[@{宏的名称,在键的位置}键参数]

宏的定义与后面的键参数之间没有逗号。宏的名称是任意的，沿键方向的位置的值就是 tikz 里的 pos 的值的意义，默认是 0.5。也如 pos 的值一样，系数可以超过 [0,1] 的范围。沿垂直于键方向的位置，则是单键时的位置。此时的宏类似于 tikz 里的 \coordinate，是一个坐标。只要使用宏，就必须至少编译两次才会有正确结果。

```
\chemfig{C=[@{a,0.3}]O}
\chemmove{
\draw [-,red,shorten <=1pt] (a)--++(0,1);
}
```

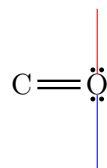


chemfig 的作者提供了缩短长度的两个命令

```
shorten <=起点缩短的距离
shorten >=终点缩短的距离
```

宏可以定义在原子上，一定要写在原子前，有电荷标识时宏定义直接写在 \charge 命令前，此时位置参数失效，默认位置在原子字母的顶部。这时的宏名可以当作 node 名使用。

```
\chemfig{C=@{b}\charge{90=\:, -90=\:}{O}}
\chemmove{
\draw [-,red] (b)--++(0,1);
\draw [-,blue] (b.-90)--++(0,-1);
}
```



如果写在原子后，宏名是原子后基准线位置处的坐标，不能当作 node 名使用，所以一定要写在原子前。

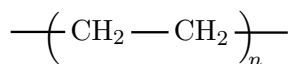
6.9 高分子与宏

高分子是通过定义宏方式得到方括号或者圆括号。定义宏后可以在 chemfig 之外调用宏，这个功能很灵活很方便。必须编译两次才能正确显示高分子。在 standalone 类中输出高分子时，必须选择类参数为 chemfig，需要手动扩大图片留白才能不切边。

```
\documentclass[chemfig]{standalone}
```

宏定义的花括号里第二个参数是位置，按键的长度比例定义括号的位置。第二个语句是定义括号的高度、深度和脚标。高度是调整距离键（水平位置）上面的括号高度，深度是调整下面的括号高度。如果括号在竖直方向是基本对称的，只设置高度就行，不对称时需要两个长度都设置。

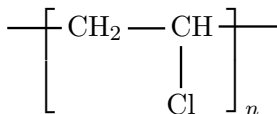
```
\chemfig{\vphantom{CH_2}-[@{left,0.75}]CH_2-CH_2-[@{right,0.25}]}
\polymerdelim[height=5pt, indice=\!|\!n]{left}{right}
\vspace{3mm} %article类可能要添加竖直间距，standalone类要增加上下留白
```



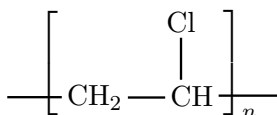
前面加 \vphantom{CH_2} 是为了并排画两个高分子时，两个高分子的主链位置在竖直方向一

致。里面的分子必须是第一个高分子主链上的第一个原子组，否则竖直方向会有细微差别。

```
\chemfig{\vphantom{CH_2}-[@{left,0.7}]CH_2-CH(-[6,,1]Cl)-[@{right,0.3}]}
\polymerdelim[delimiters={},height=5pt,depth=30pt,indice=n]{left}{right}
```



```
\chemfig{\vphantom{CH_2}-[@{aa,0.75}]CH_2-CH(-[2,,1]Cl)-[@{bb,0.25}]}
\polymerdelim[delimiters={},height=30pt,depth=5pt,indice=n]{aa}{bb}
```



手册上还有括号位置不在同一个水平链上的情况，需要的时候可以自学。

6.10 在 tikz 的 node 里加载高分子

这里给出一个 tikzpicture 里加载高分子的例子，高分子是比较麻烦且容易出错的。需要编译两次。

```
\documentclass[tikz]{standalone} %这里是tikz，因为是tikzpicture环境
\standaloneconfig{border=30pt} %有时需要手动设置留白

\usepackage{graphicx}
\usepackage{chemfig}

\begin{document}
\begin{tikzpicture}
%node里一定要同时加载高分子的那两句，不能只加载一句，顺序不能颠倒
\node at (0,0) {
\chemfig{\vphantom{CH_2}-[@{aa,0.75}]CH_2-CH(-[2,,1]Cl)-[@{bb,0.25}]}
\polymerdelim[delimiters={},height=5pt,depth=30pt,indice=n]{aa}{bb}
};
\end{tikzpicture}
\end{document}
```

6.11 chemmove、debug、内置 node 和 schemestart/schemestop

6.11.1 chemmove 和 standalone 类使用 [chemfig] 选项

\chemmove的用法是

```
\chemmove{多条tikz画图语句}
```

当使用\chemmove语句时，standalone 类里必须使用 [chemfig] 选项。需要看一个简单的例子才能说明使用 [chemfig] 选项会发生什么。

```
\documentclass[chemfig]{standalone}
\standaloneconfig{border={0.7cm 5pt 5pt 1.6cm}} %左下右上
```

```

\usepackage{chemfig}

\begin{document}

a \chemfig{*6(-----)} b

\chemmove{
\draw [-] (0,0)--(-3,0);
\draw [cyan] (0,0)--(0,3);
\fill [red] (0,0) circle (2pt);
\fill [blue] (-3,0) circle (2pt);
}

\end{document}

```

单条`\chemfig`是图片，等同文字，当 standalone 类使用 `[chemfig]` 选项时，上面例子中的文字 a 和 b、`\chemfig`画出的分子式和`\chemmove`里面的画图语句结果，都会同时显示在一个 pdf 文件里。特别需要注意的是，花括号里的 tikz 画图语句不能有空行。`\chemmove`也是文字，相当于`\tikz{}`。此时本质上涉及了多个环境，standalone 的自动切边功能失效，需要手动调整边缘才能够完整显示图片。

`\chemmove`中 tikz 画线语句自动在终点添加箭头，如果不想要箭头，需要额外设置参数取消箭头。`\chemmove`中 tikz 画图语句与 tikz 的原始画图语句有一些细微差别，下面还会讲 node 语句的差别。

6.11.2 debug 和内置 node 名

`\chemfig`内置了许多 node 名，画图时可以临时设置下面的语句查看

```
\setchemfig{debug=true}
```

画完图后再关闭这个选项（默认是 false，注释掉这条语句就可以了），就不会显示显示内置 node 名了。

显示完这些内置 node 名就可以使用它们画图了，注意调整 standalone 的边距，使得 node 编号可以完全显示出来。

```

\documentclass[chemfig]{standalone}
\standaloneconfig{border={3pt 0.5cm 3pt 13pt}}

\usepackage{chemfig}

\begin{document}
\setchemfig{debug=true}
\chemfig{ABC-EDF-H}

\chemmove{
\draw (n1-1)---+(0,-0.5)-|(n2-3);
\draw (n3-1.center)---+(0,-0.5);
\draw (n3-1.0)---+(0,-0.5);
}

```

```
\end{document}
```

红色编号是 node 编号，灰色编号是 node 里面的原子编号，调用方式是 n 后面紧跟着 node 编号，连字符，连字符后面紧跟着原子编号。单原子也要写编号，原子编号不能省略。所有 node 名都支持 anchor 坐标和角度坐标。

chemfig 对环还内置了中心位置，但是环中心在 debug 下不显示，有环就有环中心 node，就可以调用。环中心的 node 名为 cyclecenter 后面紧接着编号。原子编号和环编号是独立的，都是按先画原则编号，所以下面的例子中，苯环上的原子编号是不连续的。如果不想从画分子的语句中分析出环的编号，就填充个点测试一下也可以。环指的是正多边形情况，不包括问号构成的不规则的环。

```
\documentclass[chemfig]{standalone}
\standaloneconfig{border={3pt 0.5cm 3pt 13pt}}

\usepackage{chemfig}

\begin{document}
\setchemfig{debug=true}
\chemfig{*6(==(-[:30]H)=)}

\chemmove{
\draw (cyclecenter1)--++(1.2,0);
}

\end{document}
```

6.11.3 内置 node 名和宏 node 名混用

下面这个例子，还用 @ 自定义了 node 名，然后混用画图。这在画反应动力学的图时非常方便。

```
\documentclass[chemfig]{standalone}
\standaloneconfig{border=10pt}

\usepackage{chemfig}
\usetikzlibrary{calc}

\begin{document}

\setchemfig{debug=true}

\chemfig{N(-[@{b1,0.5}:-120]*6(==)))([,,,blue]-[@{b2,0.5}:120]*6(==))=N
-[:60]*6(==)}

\chemmove{
\draw [-latex] [blue] (b1) to [out=140,in=220] (b2);
\draw [red] (cyclecenter1) to [out=150,in=210] (cyclecenter2);
\draw [-] [dashed,cyan] (cyclecenter1) -| (cyclecenter3);
\coordinate (a) at ($(b1)!0.5!(b2)$);
```

```
\node [at=(a),xshift=-5mm,rotate=90] (new-name) {$120^\circ$};
\fill [red] (new-name) circle (2pt);
}
\end{document}
```

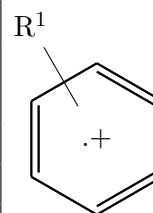
这里需要额外说明的是 node 的用法与 tikz 里的语法略有不同。

```
\node [at=(放置位置的坐标),node参数, 包括移动和旋转] (给这个node起名) {文字内容};
```

凡是 tikz 的坐标都可以用 calc 子库进行计算，上面这个例子自以为比手册里的键角例子容易、好懂，前提是熟悉 tikz 语法。

在\chemmove里定义新 node 和使用 node 是非常有用的功能，下面再放个书上的例子。

```
\chemfig{*6(-----)}
\chemmove{
\node [at=(cyclecenter1)] (.)+.
node [at=(cyclecenter1),shift=(120:1.75cm)] (end){\printatom{R^1}};
\draw[-,shorten <=.5cm](cyclecenter1)--(end);
}
```



6.11.4 schemestart/schemestop 与宏

如果是多个分子，可以使用 schemestart/schemestop 方式，chemfig 定义了 compound 这个概念。每个\chemfig里的分子是一个 compound，compound 自己有另外的编号。然后还可以定义宏 node 名。

```
\documentclass[chemfig]{standalone}
\standaloneconfig{border=10pt}

\usepackage{chemfig}

\begin{document}

\setchemfig{debug=true} %显示每个chemfig内分子的内置node名
\setchemfig{scheme debug=true} %显示每个compound的node名

\schemestart
\chemfig{ABC-@{aa}ED}
\arrow{0} %必须使用箭头
\chemfig{*6(-[@{bb}]=---=)}
\schemestop

\chemmove{
\draw [cyan] (c2.north west)-|(c1.north east);
\draw [blue,dashed] (aa.-90)|-(bb);
\draw [magenta] (n3-1)--++(0,-0.7)-|(n1-2); %不重名的仍然可以使用
}
\end{document}
```

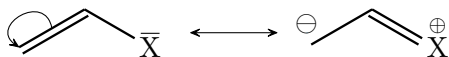
这个例子有几点需要说明。放在\schemestart和\schemestop之间的\chemfig可以同时显示。如果

使用`\arrow`语句设定后面分子的位置的话, 每个`\chemfig`有自己的 node 名。但是遗憾的是, 原子的 node 名在第二个 compound 里被重置。原子的 node 名如果重名的话, 就只能识别最后一个 compound 里的位置。如上所示, 如果不重名, 仍然可以用来画图。手册说 **debug 编号不起作用了, 严格讲是重名部分不起作用了**。对于重名的 E 原子 (node 名是 n2-1), 就只能使用 `@` 定义宏 node 名方式了, 略微麻烦一点。

注意: 在`\schemestart`和`\schemestop`之间和`\chemmove{}`花括号内不能存在空行。

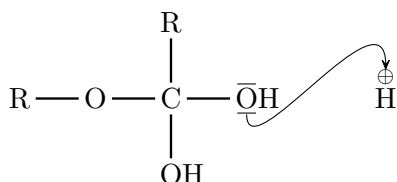
下面的两个例子是手册上的。

```
\schemestart
\chemfig{@{a1}=_[@{db}::-30]-[::-60]\charge{90=\!}{X}}
\arrow{<->}
\chemfig{\chemabove{\vphantom{X}}{\ominus}-[:30]=_[::-60]
\chemabove{X}{\scriptstyle\oplus}}
\schemestop
\chemmove{\draw(db) .. controls +(100:5mm) and +(145:5mm).. (a1);}
```



下面这个例子有一点要额外说明。宏定义在原子上, 如果是带 lewis 电荷形式的原子, 位置就不合适了。可以用 `shorten` 来缩短线的长度, 相当于移动了起点和终点的位置。但是这种方法需要移动的距离与字号相关, 如果修改字号, 缩短的距离也会有细微变化。

```
\chemfig{R-O-C(-[6]OH)(-[2]R)-@{aa}\charge{90=\!,-90=\!}{O}H}
\hspace{1cm}
\chemfig{@{bb}\chemabove{H}{\scriptstyle\oplus}}
\chemmove{\draw[shorten <=2pt,shorten >=7pt](aa) .. controls +(south:1cm) and
+(north:1.5cm).. (bb);}
```



上面语句中的 **shorten** 后面的空格不能省略。

6.12 arrow 与分子位置的精确放置

`chemfig` 里内置了箭头语句, 帮助在`\schemestart`和`\schemestop`之间精确放置多个`\chemfig`分子。下面是一个箭头使用的说明性例子

```
\schemestart
\chemfig{ABC}
\arrow %默认水平1cm, 带末端箭头
\chemfig{EFG}
\arrow[,2,red,dashed]
\chemfig{JJ}
\arrow{0}[-20,0.5]
```

```
\chemfig{HH}
\arrow{->[above][below]}[0,2,blue,-latex]
\chemfig{OH}
\arrow{0}[.,0]
\chemfig{OH}
\schemestop
```

ABC \longrightarrow EFG \dashrightarrow JJ

HH $\xrightarrow[\text{below}]{\text{above}}$ OHOH

箭头的主要用法有

- `\arrow`，画一个水平长度为 1cm 的向右箭头。
- 箭头起点和终点位置与两个分子（两个 `compound`）之间存在距离，所以两个分子之间的实际距离等于箭头默认的 1cm 长度加上 2 倍的预留间距。这个间距不能设为 0，否则箭头有可能连在分子的键上，效果不好。

```
\setchemfig{arrow offset=1em} %默认1em
```

但是这个值使得箭头的实际长度变得很复杂。

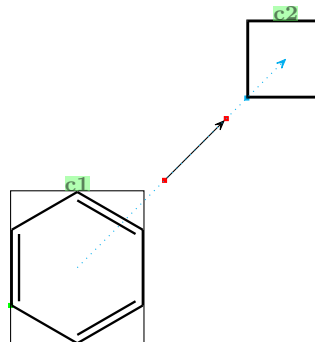
- 箭头参数写到后面的方括号里，参数分别是`\arrow[角度,长度,tikz划线参数]`，角度不需要冒号。

- 角度和箭头起始点的关系

箭头的角度是两个 `compound` 中心的连线的方向。起始点是连线与两个 `compound` 边框交点向外位移 `arrow offset` 间距后的位置。

```
\setchemfig{scheme debug=true}
\schemestart
\chemfig{*6(==--==)}
\arrow[45]
\chemfig{*4(----)}
\schemestop

\chemmove
{
\draw [cyan,dotted] (c1.center)--(c2.center);
}
```



`\setchemfig{scheme debug=true}`后可以很清楚地看出这些关系。

- 设定箭头的长度略微有些复杂。

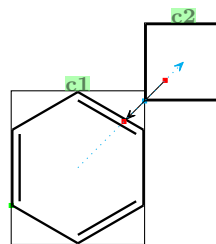
箭头实际的长度会受到 `compound sep` 的控制。其默认值为 5em。只有箭头长度超过 `compound sep` 时，箭头实际长度才是设定值。否则，箭头实际长度是 `compound sep` 的值。如果设定 `compound sep` 的值为 0，箭头无论多长都不起作用。关系虽然有些复杂，其实很好理解。**`compound sep`** 的值是两个分子（两个 `compound`）之间的最小绝对间距。箭头长度是两个 `compound` 之间的距离。如果需要两个 `compound` 紧挨着，不要修改 `compound sep` 的值，这会使得所有的箭头都失效。设置箭头长度为 0，两个分子 `node` 边框就会紧挨着。紧挨着的意思是两个 `compound` 边框不交叠，两个 `compound` 的 `node` 中心的连线方向仍然由角度确定。

```

\setchemfig{scheme debug=true}
\schemestart
\chemfig{*6(====)}
\arrow[45,0]
\chemfig{*4(----)}
\schemestop

\chemmove
{
\draw [cyan,dotted] (c1.center)--(c2.center);
}

```



此时箭头反向，因为箭头的起点和终点距离两个 compound 的边框的间距 arrow offset 不为 0。这种情况一般是不显示箭头的，所以按下面的设置方式取消箭头显示即可。

- 不显示箭头

当两个 compound 紧挨着时，我们其实不想画箭头，所以应该将箭头设置为不显示

```
\arrow{0}[45,0]
```

chemfig 的 v1.66 版下，花括号里不能是其他数值，要么加{0}不显示箭头，要么不显示箭头。即使没有长度，设置了角度后，角度仍是生效的，上面的结果，两个分子仍然按 45 度方向放置。花括号里为 0，不显示箭头，但是各种角度和长度逻辑关系仍然存在，因此可以用箭头语句在指定位置上放置 compound。

```
\arrow{0}[45,2]
```

- 箭头上下标文字

```
\arrow{->[箭头上方的文字][箭头下方的文字]}[角度,长度,tikz样式]
```

->是在箭头上下添加说明的语法。

\arrow 的长度关系虽然复杂，但是该语句可以将每个 compound（每个 \chemfig{} 分子）放置在指定位置。然后再利用 compound 的 node 名和自定义的宏名画分子动力学的各种指示箭头。

6.13 chemfig 的几个例子

目前键长和键角的语句还没有打包好，还不如直接定义宏和调用 tikz 语句方便，下面是一个氢键的例子。

```

\chemfig[atom
sep=4em]{\charge{90:3pt=$\scriptstyle-\delta$}{O}(-[a1]:128.5]H)(-[:231.5]H)
-[,1.5,,blue,loosely dotted,very thick]\charge{90:3pt=$\scriptstyle+\delta$}{H}
-[: -51.5]\charge{90:3pt=$\scriptstyle-\delta$}{O}(-[:231.5]H)-[,1.5,,blue,loosely
dotted,very thick]\charge{90:3pt=$\scriptstyle+\delta$}{H}
-[@{r,0.7}:-51.5]O(-[:231.5]H)}

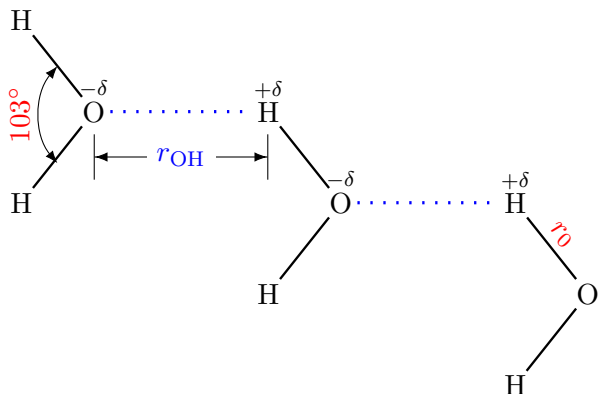
\chemmove{\draw [{Bar[width=6mm]Latex}-{Latex[]Bar[width=6mm]},thin]
($ (n1-1)+(0,-0.6)$)-- node [fill=white,text=blue] {$r_{\mathrm{OH}}$}}

```

```

($\n4-1)+(0,-0.6)$);
\node [at=(r),above,red] {\rotatebox{-45}{\textcolor{red}{r}_0}};
\draw [Latex-Latex] (a1) arc [radius=1, start angle=140, end angle=220] node
[pos=0.5,sloped,above, rotate=180,text=red] {\textcolor{red}{103}^\textcolor{blue}{circ}};
}

```



这里标识角度的做法与前面的例子不同。前面的例子，是取了两个键的中点，然后用 tikz 的 calc 子库计算两个键中点连线的中点，在连线中点 node，加旋转、平移，表示张角的弧用 to[out=角度, in=角度] 语句。这里是取了一个键的中点，然后画弧，在画弧语句的中间 [pos=0.5] 位置做 node，然后 [sloped, above, rotate=旋转角度]。所以氢键这个例子，只取了两个键的中点加了宏名。

第七章 演示稿 beamer

beamer 类使用起来不难，自定义模板比较麻烦一点。

7.1 beamer 模板

beamer 的基本框架

beamer 将一页 ppt 分为了五个部分，这五个部分不会同时存在，否则样式太难看。上面可以有 headline，下面可以有 footline，左边可以有左边栏 sidebar left，右边可以有右边栏 sidebar right，中间的展示内容为 frame 环境。手册第 63 页现在有一张图说明 beamer 每页的基本框架的上下顺序。

- 最底层是 background canvas，其上是 background。
- 然后是左边栏、左边栏内容，右边栏、右边栏内容。左右边栏一般不会同时存在。
- 默认右下有一行灰色的导航条（一组有翻页和跳转等功能的导航按钮）。这个实际是插在右边栏的。即使右边栏不存在，宽度为 0，仍然可以在右边栏插入东西，比如导航条和 logo。设置了 logo 图片后才会显示 logo 图片。
- 然后是 headline。
- 然后是 frame，frame 环境包括两个部分，frametitle 和 frame 的内容。因为灰色导航条的顺序在 frame 下面，所以 frame 里的内容（文字）可以覆盖灰色导航条。即使导航条被覆盖，仍然不影响使用导航条，点击仍然可以跳转。
- 最后是 footline。footline 一般很窄。

内置了一些模板，基本上可以分为使用 sidebar 的样式（一般是左边栏样式），和使用 headline 与 footline 样式（两个都用或者只用一个）。当使用边栏样式时，必须设置边栏宽度，边栏和 frame 里的内容是互斥的，本质上它们是并排的 box。

笔者个人喜欢默认，只有 frametitle 和 frame 内容，其余都没有，默认右边栏插入灰色导航条。

beamer 类的 option

使用 beamer 类，beamer 类的最常用选项为字体。

```
\documentclass[12pt]{beamer} %8,9,10,11,12,14,17,20
```

默认字体为 10pt。这里的 10pt、11pt 和 12pt 不是 article 的字体大小，已经是相对应扩大的结果。12pt 是一套很大的字体，很适合大教室讲课。10pt 和 11pt 可以放置更多内容，比较适合学术类讲座和自己做笔记。

beamer 类的另一个最常用选项是比例，默认是 43, 4:3，过去的投影仪一般是这个比例。新的投影仪是 16:9，使用 aspectratio 参数可以修改比例为 16:9，但是建议设置为 16:10。

```
\documentclass[12pt,aspectratio=1610]{beamer} %默认43
```

天津大学教室里的投影仪有时调整的不好，如果采用 default 模板，几乎没有上下和左右的留白余地，16:9 可能会导致内容在白屏外，16:10 保险一点。其余比例可以查看手册。还有就是有的学生的平板可能是 4:3 的，导致他们喜欢默认的 43 比例。笔者经常用 beamer 做笔记，如果笔记内容有很长的公式，16:10 比较好。ppt 一般是短句子，纵向比例太小，单页内容就会受限，43 比例和 1610 比例哪个好很难说。

默认定理等环境不编号，如果希望使用编号，可以写入下面的参数

```
\documentclass[envcountsect]{beamer}
```

与文章类一样，可以对公式编号样式进行设置

```
\documentclass[leqno]{beamer}
\documentclass[fleqn]{beamer}
```

字体样式、主题样式和配色样式

beamer 模板分开设置为字体样式、主题样式和配色样式。内置了一些样式可以使用。

字体样式推荐 serif，默认的个人觉得非常难看。使用\usefonttheme加载。

```
\usefonttheme{serif}
```

beamer 的默认主题是 default，不设置主题的情况就是 default 模板样式，没有 headline、footline 和左右的 sidebar，只有 frame 环境和灰色导航条。这个灰色导航条很具特色，一般大家都不取消，几乎就是 beamer 做的 ppt 的标志。

内置主题使用\usetheme加载。最常用的两个模板是 Warsaw 和 Berkeley。

```
\usetheme{Warsaw} %headline和footline
```

Warsaw 模板设置了 headline 和 footnote。headline 和 footnote 都是两个等分的 box，headline 里显示当前页所在的小节和小节，footline 里显示作者和演示稿的 title。

Berkeley 模板设置了左边栏，左边栏内显示题目、作者、当前页所在的小节和小节。

```
\usetheme{Berkeley} %左边栏
```

个人认为 Berkeley 模板比较适合 16:10，Berkeley 的 frametitle 占比太大了，从美学角度可能这样的比例更适合，但是不利于更多的内容展示。Warsaw 模板适合 4:3，headline 和 footnote 都是极窄的，占用空间不是很大。笔者最喜欢 default，感觉没有必要让导航或其他如 title 这样的挤占展示空间。很多导航的功能只是显示讲述人的姓名、单位、报告题目和讲到哪里了，听讲的人未必真的关注这点。只要 tex 文件有\section，使用超链接包，编译生成的 pdf 就有书签，用 pdf 的书签跳转可以很方便快速定位内容。hyperref 包可以添加 pdf 的信息，包括作者名之类的。

每个主题模板内置了一套设置，如果觉得可以接受，就直接使用模板，这样代码量很小。不喜欢地方可以随后设置覆盖掉默认参数。

上面这两个模板的默认配色是深蓝色系，推荐保留默认配色。

设置配色样式的语句为\usecolortheme:

```
\usecolortheme{seahorse} %浅蓝色系
```

seahorse 是浅蓝色系，笔者认为用在 Berkely 模板不好看。beamer 默认了一系列颜色主题，个人觉得好看的并不多。beamer 手册后面有一些主题和配色的展示。个人觉得 beamer 的功能还是比较强大的，样式和配色感觉很一般，很多时候可能添加一个设计合理的 background 更好看。

7.2 frame

每一个 frame 环境是一页 ppt，beamer 的基本框架为：

```
\documentclass{beamer}
\begin{document}
\begin{frame}{填写frametitle} %frametitle可以省略
内容
\end{frame}
\end{document}
```

也可以采用 frametitle 命令来设置标题

```
\begin{frame}
\frametitle{当前页的标题} %等价于上一个例子
...
\end{frame}
```

可以设置子标题，但是必须 frametitle 存在时才会显示子标题。

```
\begin{frame}
\frametitle{当前页的标题}
\framesubtitle{当前页的子标题} %如果没有frametitle不生效
...
\end{frame}
```

frame 的参数

frame 有几个比较常用的参数

1. 内容在顶部，默认是上下居中的 c，底部 b 很少会用，s 是分散。这几个选项可以写在类参数里，对所有 frame 都有效。

```
\begin{frame}[t]{} %顶部
...
\end{frame}
```

2. allowframebreaks: 内容太长自动分页

```
\begin{frame}[allowframebreaks]{}
...
\end{frame}
```

这个功能并不完美，经常会遇到分页时希望一页塞得很满，一页相对比较空的情况。所以一般笔者都是用手动调整内容的方式分页。目录太长是需要这个选项的。

3. shrink 自动缩小内容匹配尺寸，手册称 shrink 参数 **evil**，不推荐，这个选项相当于将所有的东西都按比例缩小，强行将内容放在一页，所以比较难看。类似的还有 squeeze 参数，

竖直压缩间距，如果内容太长仍然会超出范围，也不是很好用。基本上还是需要手动调整内容的。

```
\begin{frame}[shrink]{}
内容
\end{frame}
```

4. fragile 参数，lstlisting 环境必须使用该参数，否则会产生严重编译错误

```
\begin{frame}[fragile]{}
\begin{lstlisting} %必须添加fragile参数
code
\end{lstlisting}
\end{frame}
```

5. 注释掉某个 frame，可以使用下面的参数

```
\begin{frame}}<presentation:0>[noframenumering]{frametitle}
frame内容
\end{frame}
```

可以省去前面的presentation:，很多时候后面的选项也可以不加。

```
\begin{frame}}<0>{frametitle}
```

这里存在一个小问题，<0>选项令该页 frame 不显示出来，这页 frame 仍然会正常编译，不能有编译错误。但是有些时候需要注释掉的是代码，在 frame 环境内，可以使用百分号注释掉单行语句，但是不能使用\iffalse...\fi注释掉多行语句。只能放在 frame 环境的外面，使用它注释掉一个或多个 frame 环境。

```
\iffalse
\begin{frame}{}
存在错误代码无法通过编译
\end{frame}
\fi
```

7.3 beamer 自定义的分栏环境

beamer 类有一个自定义的分栏环境非常好用，个人觉得，要是能独立出来构成一个包就好了。

```
\begin{frame}{}
\begin{columns}[t] %多栏上下对齐方式，默认居中
\begin{column}{8cm} %分栏，花括号里定义宽度
左边内容
\end{column}
\begin{column}{0.33\textwidth} %宽度的另外一种定义方式
右边内容
\end{column}
\end{columns}
\end{frame}
```

如果需要更多分栏，就多嵌套几个 column 环境即可。所有分栏宽度加起来不能超过文字宽度。上面例子中的系数 0.33 是按默认页面比例、default 模板情况下设定的，和第一分栏的 8cm 宽度加起来，在文字左右边距各 4mm 情况下，不超过文字宽度。

7.4 目录

目录

beamer 里可以加载目录，一般用单独一页 frame 加载目录。

```
\begin{frame}{目录}
\tableofcontents
\end{frame}
```

很多时候我们希望逐条展示目录，可以在后面加暂停选项 pausesections。

```
\tableofcontents[pausesections]
```

每一小节前，还有可能以突出当前目录的形式展示目录，可以添加选项 currentsection

```
\tableofcontents[currentsection]
```

小节前当前目录多次复用，可以使用下面的语句。

```
\AtBeginSection[]
{
\begin{frame}[label=current-section]{Outline}
\tableofcontents[currentsection]
\end{frame}
}
\section{第一个小节}
```

这段需要放在第一个小节之前。

默认目录不带小节号，可以用下面的语句设置

```
\setbeamertemplate{section in toc}[sections numbered]
\setbeamertemplate{subsection in toc}[subsections numbered]
```

可以用下面的语句在目录中取消掉 subsection，只包含 section，此时 pdf 的目录仍然包含 subsection。

```
\tableofcontents[hideallsubsections]
```

[hideallsubsections] 这个选项会导致当 section 增多时，目录条不能自动压缩间距。目录条是固定间距的。此时 shrink 选项仍然是有效的。如果希望修改这个固定间距，必须修改文件 beamerbaseloc.sty，建议 copy 到当前目录下修改。2024 版，第 137 行的 1.5em，可以修改为更小的值。

如果目录条比较多，但是自动压缩情况下还能放下，隐藏 subsection 的方法还是下面的基本设置语句最好：

```
\setcounter{tocdepth}{1} %目录只显示到section
```

设置 `tocdepth` 的数值后, `bookmarksdepth` 的深度会重置为 0, 必须再手动设置 `bookmarksdepth`, 不需要设置 `bookmarks=true`, 相当于分别设置了 ppt 显示的目录条深度和 pdf 的书签目录条深度。

```
\hypersetup{
bookmarksdepth=3 %包含subsection
}
```

目录间距

如果目录条目很多, 会自动压缩纵向间距, 使得目录尽量在一页中显示。frame 的 `shrink` 选项对目录是有效的, 当然这个选项是 evil 的, 并不美观。

目录条目很多时, 自动分页功能对目录内容是有效的。遵循先自动压缩纵向间距、如果还放不下然后再断页的规则。但是缺点是, 如果分页后目录条目比较少, 第二页的目录条间距很大。

笔者目前理解, beamer 的目录样式文件能传递的参数不包括间距, beamer 不支持单纯的行距调整包和 `spacing` 环境。frame 里内容的竖直间距, 有许多预设的自动调整功能。这是为什么竖直间距这个问题比较复杂的原因。

最近想出了一个很好的办法解决这个问题, 可以充分利用 beamer 设置好的间距自动调整功能,

```
\vspace{上面希望留的竖直间距} %需要自动分页时不加前面这句
\tableofcontents
\parbox[][下面希望留的竖直间距][1cm]{}
```

需要自动分页的情况, 根据最后的目录条数目, 只在下面加一个合适高度的 `\parbox`。frame 里, 当下面没有内容时, 不支持 `\vspace` 和 `\vfill`, 所以调整下面的竖直间距时, 必须填充一个指定高度的 box。另外, 对于目录, frame 的参数 `[t]` 也是无效的。

7.5 复用、暂停、显示顺序和跳转

复用 frame

```
\begin{frame}[<+>][label=labelname]{frametitle}
内容
\end{frame}
...
\againframe[<+>]{labelname} %显示相同的一页
```

如果该标记名 frame 定义了显示顺序, 则 `\againframe` 命令中也需要添加相同的顺序选项。这种 frame 逐渐显示多页的情况, 也可以指定直接复用到哪一页。

```
\againframe<指定复用时的显示规则>{labelname}
```

如果前面是 `[<+>]`, 指定规则仍然也需要添加这个选项

```
\againframe<3->[<+>]{labelname}
```

暂停

如果希望在某处暂停，最直接的方式是加`\pause`语句：

```
\begin{itemize}
\item 首先\pause
\item 其次\pause
\item 最后\pause
\end{itemize}
```

经常会遇到列表时希望逐条显示，每个 item 后加语句太麻烦，可以采用下面的方式：

```
\begin{itemize}[<+>->]
\item 首先
\item 其次
\item 最后
\end{itemize}
```

如果一页中有多个 itemize 环境，可以将选项加到 frame 那里，每个 itemize 环境都会有逐条显示功能。

```
\begin{frame}[<+>->]{直接加到frame选项}
\begin{itemize}
\item A
\item B
\end{itemize}\pause %实际多了一个停顿，但是必须有
文字\pause %有文字时要这么加才是按上下顺序显示
\begin{itemize}
\item A1
\item B1
\end{itemize}
\end{frame}
```

`\item`是有逐条顺序的，但是文字本身并没有。

显示规则

加减号只能按顺序逐条显示，可以在`\item`后面加显示规则，注意显示规则加在尖括号里。

```
\begin{frame}{显示规则}
\begin{itemize}
\item<1-> 首先 %n-表示从n往后都显示
\item<2-> 其次
\item<3-> 最后
\end{itemize} %这个例子的结果与上面的按顺序显示相同

\begin{itemize}
\item<1-> 第一
\item<2-> 第二
\item<-3> 第三 %-n表示从1到n后显示
\item<2,4> 第四 %2和4次显示，其余不显示
\end{itemize}
```



```
\end{frame}
```

重点和要点

beamer 默认定义了重点\alert为红色，要点\structure为蓝色。

```
\alert{重点}
\structure{要点}
```

在逐条显示中也可以加入 alert，突出当前条目

```
\begin{frame}{alert}
\begin{itemize}
\item<1-|alert@1> 首先
\item<2-|alert@2> 其次
\item<3-|alert@3> 最后
\end{itemize}
\end{frame}
```

这种情况突出的条目是 alert 的默认红色，希望改换不同的颜色，可以用下面的语句定义两个颜色，突出时的和不突出时的，也可以加粗。加粗的例子对 pdf_latex 有效，x_el_atex 时依赖于设置语句，有时需要改为设置中文黑体。还可以设置 colorbox 等方式来突出条目。

```
\begin{itemize}
\item<1->\alt<1>{\color{red}首先}{\color{gray!30}首先}
\item<2->\alt<2>{\bf\color{purple}其次}{\color{red!30}其次}
\item<3->\alt<3>{\colorbox{blue!30}{最后}}{\color{gray!20}最后}
\end{itemize}
```

下面是个在 tcolorbox 里做逐次显示的例子。

```
\documentclass{beamer}
\usefonttheme{serif}
\usepackage[many]{tcolorbox}
\begin{document}
\begin{frame}
\begin{tcolorbox}[title=My title,fonttitle=\bfseries,
enhanced,colframe=red!50!black,colback=orange!10,colbacktitle=red,
sidebyside,righthand width=3cm,
lowerbox=invisible,lower separated=false,
drop lifted shadow,
only=<1>{colbacktitle=yellow,coltitle=red!50!black,colframe=red},
only=<3>{colback=yellow!50,watermark text={Attention!}},
only=<3->{lowerbox=visible} ]
This is a test.
\begin{itemize}[<+>-]
\item One
\item Two
\item \alert<3>{Three}
\item Four
\end{itemize}
\end{tcolorbox}
\end{frame}
\end{document}
```



```

\tcblower
\begin{equation*}
\int\limits_{1}^x \frac{1}{t} dt = \ln(x).
\end{equation*}
\end{tcolorbox}
\end{frame}
\end{document}

```

跳转

beamer 里可以设置调整按钮，下面的例子是跳转到当前页第二次显示。

```

\begin{itemize}[<+>][label=here]{逐条显示}
\item 首先
\item 其次
\item 最后
\end{itemize}
\hyperlink{jumpsecond}{\beamergotobutton{Jump to second item}}
\hypertarget<2>{jumpsecond}{}

```

必须另外的方括号里写 label，不能使用同一个方括号。设定顺序在前，设置 label 在后，不能更换前后位置。这个 label 是下面的例子要用的，这个例子本身并不需要。

然后可以在后面的某个 frame 里设置按钮，可以跳转到“逐条显示”这页，如果是上面的逐条显示的某页，还可以直接设置到某页某次显示，下面的例子就是跳转到第二次显示。

```

\hyperlink{here<2>}{\beamerbutton{Jump to other frame}}

```

还可以往前跳转到某 frame 某条目那里，这个 frame 需要设置好[label=名字]，注意上面的例子中必须是两个方括号分别写入参数，不能合并。

```

\begin{frame}[<+>, label=here]{逐条显示} %错误的写法
\begin{frame}[<+>][label=here]{逐条显示} %正确的写法

```

可以使用\uncover命令逐行显示公式

```

\[
\begin{aligned}
f(x)&=a+b+c\\
\uncover<2>{\&=d\\}
\uncover<3>{\&=e}
\end{aligned}
\]

```

表格逐行显示可以采用\pause命令，

```

\begin{tabular}{l!{\vrule}cccc}
Class & A & B & C & D \\ \hline
X & 1 & 2 & 3 & 4 \pause\\
Y & 3 & 4 & 5 & 6 \pause\\
Z & 5 & 6 & 7 & 8
\end{tabular}

```

逐列显示则采用\onslide命令，加载列格式的后置语句中

```
\begin{tabular}{l!{\vrule}c<\onslide<2->c<\onslide<3->c<\onslide<4->c<\onslide>c}
Class & A & B & C & D \\
X & 1 & 2 & 3 & 4 \\
Y & 3 & 4 & 5 & 6 \\
Z & 5 & 6 & 7 & 8
\end{tabular}
```

7.6 图片

beamer 里加载图片直接用\includegraphics[]{}语句，这一小节主要讲图片的一些特殊情况。

局部放大

对图片进行局部放大

```
\begin{frame}
\framezoom<1><2>[border](0cm,0cm)(2cm,1.5cm)
\framezoom<1><3>[border](1cm,3cm)(2cm,1.5cm)
\framezoom<1><4>[border](3cm,2cm)(3cm,2cm)
\includegraphics[height=5cm]{fig/tju.pdf}
\end{frame}
```

这里定义了三个 framezoom，对图片天津大学校徽进行局部放大，border 参数表示显示放大区域的边框，后面的坐标表示局域放大区域的对角线坐标。最后的效果是，展示整个校规，然后每翻页一次，依次放大图片的不同区域。这个功能做学术报告时非常有用。

上面这个例子不能直接加 frametitle，否则展示局域放大图片时 title 的存在会影响展示效果，用下面的语句，使得第二次展示，也就是第一次局域放大时，title 被取消掉。

```
\begin{frame}<1>[label=tjuzoom]
\frametitle<1>{title}
\framezoom<1><2>[border](0cm,0cm)(1cm,1.5cm)
\framezoom<1><3>[border](1cm,3cm)(2cm,1.5cm)
\framezoom<1><4>[border](3cm,2cm)(3cm,2cm)
\pgfimage[height=5cm]{tju.pdf} %另一种加载图片的方式
\end{frame}
\againframe<2->[plain]{tjuzoom}
```

多图覆盖

在相同位置展示多张图片，后面的图片覆盖前面的图片，用\llap{\includegraphics<>[]{}}, 尖括号内设定显示规则，方括号内设置图片宽高和角度等，花括号内设置图片名字。

```
\centering
\includegraphics<1->[width=6cm]{tju.pdf} %第一句不加llap
\llap{\includegraphics<2>[width=9cm]{2.jpg}}
```

```
\llap{\includegraphics<3->[width=3cm]{d-2.jpg}}
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
\end{frame}
```

`\llap`语句有一个缺点，如果图片宽高一致，效果还行，如果不一致，后面的图片将以右下角对齐第一张图片的方式覆盖显示。类似的语句都是相同的缺点。

可以使用 `xmpmulti` 包加载多图，

```
\usepackage{xmpmulti}
```

假设图片名为 `d-0`、`d-1` 和 `d-2`，此时语句比较简单：

```
\begin{frame}{使用xmpmulti包}
\center
\multiinclude[graphics={width=7cm},format=jpg]{d}
\end{frame}
```

这样图片将以相同宽度展示，竖直方向的位置是底边对齐。`multiinclude` 还支持选择起始编号，对一组排序图片进行多图展示。

使用下面的语句，覆盖时前一张图片不显示：

```
\multiinclude[<+>][graphics={width=7cm},format=jpg]{d}
```

使用 `xmpmulti` 包对于宽高比不同的图片，尤其是比例差别极大的情况，仍然无法满足展示需求，最好使用 `tikz` 的 `node` 来完成多图展示，必须加载 `tikz` 包。

```
\begin{frame}{}
\center
\begin{tikzpicture}
\node at (0,0) {\includegraphics<1->[width=7cm]{tju.pdf}};
\node at (0,0) {\includegraphics<2>[width=6cm]{2.jpg}};
\node at (0,0) {\includegraphics<3->[width=6cm]{d-2.jpg}};
\end{tikzpicture}
\end{frame}
```

这样图片的中心在一个位置上，展示效果是最好的。同理，尖括号控制显示规则。这种方式也可以精确控制多图的显示位置，达到最好的显示效果。

`beamer` 内可以使用动画包，上面讲的并不是动画，而是逐次展示，这两者之间是不同的。

7.7 参考文献

在 `beamer` 里添加参考文献，推荐使用 `biblatex` 包，使用 `biber` 编译。这样可以做到每个 `frame` 里引用参考文献，或者每个 `frame` 里用脚注方式引用参考文献，最后再打印一份全部的参考文献列表。

```
\begin{frame}{单页引用}
脚注方式引用参考文献\footfullcite{wang}

内容处引用参考文献\
\fullcite{wang}
\end{frame}
```

```
\begin{frame}{参考文献}
\printbibliography
\end{frame}
```

7.8 简单的设置语句

文字对齐方式

两边对齐的设置推荐使用 ragged2e 包，中英文混排效果更好

```
\usepackage{ragged2e}
\justifying\let\raggedright\justifying
```

设置字体

字体模板的加载前面已经说过了，beamer 里设置字体的语句是 `\setbeamerfont{beamer-font name}{字体}`，beamer-font name 是 beamer 已经定义好的字体名。

default 模板最好将 frametitle 设置为加粗的，CJKut8 加载中文，`\bfseries` 对汉语是生效的。

```
\setbeamerfont{frametitle}{series=\bfseries}
```

还可以同时设置大小

```
\setbeamerfont{frametitle}{size=\large,series=\bfseries}
```

如果是 xelatex 编译，则需要定义汉语字体，比如设置汉语字体为黑体，或者设置汉语的 `\bfseries` 为黑体，依赖于具体的字体设置。

```
\setbeamerfont{frametitle}{series=\bfseries\heiti}
```

设置 itemize 和 enumerate 的字体，两个参数分别是字体大小和 baseline 的大小。

```
\setbeamerfont{itemize/enumerate body}{size*={14pt}{20pt}}
```

尺寸

beamer 中定义尺寸的语句是 `\setbeamersize{}` 文字宽度，default 下可以设置为 4mm

```
\setbeamersize{text margin left=4mm, text margin right=4mm}
```

设置左右边栏宽度，如果自定义模板，必须设置左右边栏宽度，然后设置的模板才有效。如果只是利用左右边栏插入 logo、导航条或者页码，不用设置宽度，因为此时并不是真正使用边栏样式。真正使用边栏样式，边栏和 frame 是并列关系。而不使用边栏样式，在边栏里插入导航条（默认样式），导航条在 frame 内容的下面，frame 里的文字内容可以覆盖导航条。

```
\setbeamersize{sidebar width left=1.5cm} %right是右边栏
```

一个常数的尺寸设置是 description 环境的文字宽度

```
\setbeamersize{description width=8em}
```

另外一种设置方法很贴心，直接给出最长的那条文字即可，会自动计算出文字宽度。

```
\setbeamersize{description width of=最长描述条目文字}
```

设置 beamer 颜色

beamer 颜色是一对颜色，前景色和背景色。

比如定义演示稿 title 的颜色，得到的是 title 变成了 colorbox 形式。

```
\setbeamercolor{title}{bg=cyan!20,fg=black}
```

设置每页演示的题目，这里要用 titlelike，这样对 framesubtitle 也有效

```
\setbeamercolor{titlelike}{fg=white,bg=blue!70!black}
```

单独设置某项的颜色并不好看，最好是设计模板时使用这些命令。或者在采用内置的模板和配色时，局部修正某项为自己喜欢的颜色，这样更美观一些。

设置 beamer 模板

设置 beamer 模板的语句为 `\setbeamertemplate{模板名}{内容}`，这个语句是模板设计的基础，这一小节先学习插入页码和一些简单设计。

在 footline 插入页码，`\insert...` 是 beamer 内置的导航条，可以插入页码、作者、单位、题目、小节名、小小节名等等。

```
\setbeamertemplate{footline}{\tiny\insertpagenumber}
```

在 default 模板，设置在 footline 插入页码，会使得自动生成的灰色导航条上移。目前笔者更喜欢在左侧边栏插入页码。

在左边栏插入页码，`\vfill` 自动竖直填充，后面的强制换行符后加了 2pt，使得页码位置略靠上一点，否则太靠近底部了。因为只插入页码，所以不用设置左边栏宽度，仅仅靠默认的留白就足以得到需要的结果。

```
\setbeamertemplate{sidebar left}{\vfill\tiny\insertpagenumber\\[2pt]}
```

beamer 里，pagenumber 和 framenumbers 是两个计数器，前者是 pdf 页数，后者是 frame 个数。逐渐显示的内容，pagenumber 是多页，framenumbers 是一页。frame 环境可以设置当前 frame 不显示 framenumbers

```
\begin{frame}[noframenumbering]{title}
frame contents
\end{frame}
```

但是如果插入的是 pagenumber，总是存在并计数的。如果有逐渐显示的内容，推荐使用 framenumbers。

设置前引符

设置目录的前引符，默认是没有的。

```
\setbeamertemplate{section in toc}[ball] %有编号
\setbeamertemplate{subsection in toc}[square] %无编号
```

设置 `itemize` 和 `enumerate` 的前引符形式，一个三层，`item`、`subitem` 和 `subsubitem`，`items` 是三者全部都设置为一个样式。

```
\setbeamertemplate{itemize item}[square]
\setbeamertemplate{itemize subitem}[square]
\setbeamertemplate{enumerate items}[ball]
```

`caption` 默认没有编号，下面的语句设置 `caption` 带编号

```
\setbeamertemplate{caption}[numbered]
```

背景色

简单地改变背景色可以用设置颜色的语句

```
\setbeamercolor{background canvas}{bg=blue!20}
```

更复杂一点的，则需要使用设置模板的语句，比如设置渐进色背景。

```
\setbeamertemplate{background canvas}[vertical shading][top=white,bottom=blue!40]
```

在背景上打格子，这句与上面的渐进色一起使用效果还可以

```
\setbeamertemplate{background}[grid][step=1cm]
```

取消渐进或格子，改成 `default` 就可以了。

插入 logo

插入 `logo` 的语句很简单，只需要设定 `logo` 即可，因为 `logo` 的位置在模板里都设定好了

```
\logo{\includegraphics[height=0.5cm]{fig/tju.pdf}}
```

对于 `default` 模板，`logo` 的默认位置在右边栏的右下角位置，竖直方向在灰色导航按钮之上。这个位置不是很好看。如果修改这个位置，必须先清空右边栏的 `logo`，否则会出现两个 `logo`，清空右边栏的 `logo` 的语句为：

```
\setbeamertemplate{sidebar right} %右边栏无内容
{
\vfill
% 保留默认的灰色导航按钮
\llap{\usebeamertemplate***{navigation symbols}\hskip0.1cm}
\vskip2pt
}
```

`\llap{}` 必须有，表示内容右端对齐，否则不会显示。这句使得不设置右边栏宽度，仍然可以显示灰色导航按钮。

然后重新设置 `logo` 的位置，比如加在左边栏，放在左下角

```
\setbeamertemplate{sidebar left}
\vfill
\rlap{\insertlogo} %左端对齐
}
```

default 模板，无论 logo 在左边栏还是右边栏，如果 logo 尺寸比较大且左右文字边距比较小，都会占用和重叠文字空间。还有一种可能是放在边栏，logo 尺寸很小，占用的是 text margin 的空白，这样不会重叠文字空间。有 sidebar 的模板，logo 设置在 sidebar 是很不错的方案。headline 比较宽的模板，logo 放在 headline 也好看。

个人觉得 logo 最佳位置是在 frametitle 的左边（前面）或右边（后面），这个设置的代码修改量稍微有些大。将 logo 插入右边栏最上面，上和右各留一点边距，然后修改相应模板里的 outertheme 文件中的 frametitle 格式，比如修改 beamerouterthemedefault.sty。将 frametitle 对应的 box 宽度调整得小一点，默认是 `\textwidth`，根据 logo 的大小调整一下 box 的宽度。不是调整 `\textwidth` 的大小，是调整模板设置中的 `beamercolorbox` 宽度。不要直接用 `\setbeamertheme{frametitle}` 修改 frametitle 样式，虽然手册给了一个例子，但是这么做非常不好，frametitle 的设置还包含条件语句，有 framesubtitle 和没有时怎么办，手册上那个简单的例子会导致 framesubtitle 无法显示。比如说默认 43 比例时，box 宽度修改为 `0.9\textwidth`，logo 图片 0.7cm 或者 0.8cm，上和右留 0.1cm 边距。

如果放在 frametitle 左边，则最好将插入 logo 的命令写到左边栏最上面，修改 frametitle 对应的 box 宽度，比如修改为 `0.92\textwidth`。在该 box 前面计算好水平移动量，将 box 左移。只是，如果 logo 的尺寸比较大，这个方法不会影响 frametitle 的 box 内容，但是会影响 frame 的文字内容。说到底，笔者不想 logo 挤占 frame 内容的空间，所以才需要控制 logo 的大小。如果没有 frametitle，文字会在 logo 之上。上面的讨论和参数的修改都是基于 default 模板，而且保证 frametitle 自动折行和 framesubtitle 可用。

可以 copy 模板文件到当前 tex 目录下，这样可以测试修改默认模板，不会影响原来的安装文件。beamer 应该是默认当前目录下文件是首选，这样可以很方便学习修改模板文件和测试修改结果。如果觉得 frametitle 上面留白太多，也需要修改这个文件的设置。

7.9 beamer 内置的 box 环境

beamer 内置了两个 box 环境，其中 `beamercolorbox` 是设置模板的主要工具。简版的 `beamercolorbox` 称为 `beamerboxesrounded`，

```
\begin{frame}
\setbeamercolor{uppercol}{fg=black,bg=green!30}
\setbeamercolor{lowercol}{fg=black,bg=gray!30}
\begin{beamerboxesrounded}[upper=uppercol,lower=lowercol,shadow=true,width=\textwidth]
]{公式}
\[
f(x)
\]
\end{beamerboxesrounded}
\end{frame}
```

`beamerboxesrounded` 的效果很一般，推荐使用 `tcolorbox`。

`beamercolorbox` 环境的参数更多一些，下面是一个例子

```
\setbeamercolor{bgcyan}{fg=black,bg=cyan!20}
\begin{beamercolorbox}[wd=\textwidth,left,sep=3pt,shadow=true,rounded=true]{bgcyan}
\bf \large beamercolorbox is a colorful box with many parameters.
\end{beamercolorbox}
```


beamercolorbox 环境的参数中，特别要提到的是参数colsep*。在leavevmode命令下，这个参数设置为合适的值，两个水平放置的beamercolorbox才能够无缝拼接在一起。不同设置方式，水平位置略有差异。但是写在模板里，不用设置就可以拼接。

7.10 beamer 模板设计

beamer 模板设计的基本语法框架为

```
\setbeamertemplate{名字}
{
\begin{beamercolorbox} %一般都会有一个box
...
\end{beamercolorbox}

\begin{beamercolorbox} %可以重叠多个box，计算好尺寸即可
...
\end{beamercolorbox}
}
```

单从效果来说，上面的语句就可以把想做的效果做出来。设计的时候，需要考虑 beamer 参数的各种影响，不能只适用一些情况。

更复杂的代码结构，推荐使用下面的方式。

```
\defbeamertemplate*
\usebeamertemplate***
```

模板文件在texmf-dist/tex/latex/beamer目录下，主模板文件内容很少，主要设置分在innertheme 和 outertheme 文件中，以及字体和颜色文件中。innertheme 主要设置 frame 环境里的内容的格式，outertheme 主要设置 frametitle、headline、footline 和 sidebar 的格式。

虽然理论上，beamer 模板设计主要依赖于\setbeamertemplate语句和 beamercolorbox 环境，但是 beamer 模板设计还是相对繁琐的。L^AT_EX 总是做简单的事情很复杂。大多数情况还是推荐在内置模板基础上修改，如果想完全从头设计模板，建议从读 default 的四个主要模板文件开始。

如果只是在 headline 和 footline 做效果，beamer 内置了两个命令\addheadbox和\addfootbox，插入 2 个是两等分，插入 3 个是三分等，非常方便。

```
\addheadbox{beamercolor颜色}{显示内容}
```

7.11 两个重新定义 frame 的例子

第一个例子是 default 模板，用 tcolorbox 做的效果。整个 frame 就是一个 tcolorbox。直接写了 fragile 参数，所以 lstlisting 环境可以直接使用。设置 tcolorbox 大小时，采用\textwidth和\textheight为基准，修改 beamer 的尺寸时不受影响。

```
\documentclass[aspectratio=1610]{beamer}
\usefonttheme{serif}
\usepackage{CJKutf8}
\usepackage{listings}
\usepackage{tcolorbox}
```



```

\setbeamersize{text margin left=3mm,text margin right=3mm}

\newenvironment{myframe}{\begin{frame}[t,fragile,plain,environment=myframe]\startbox
}{\stopbox\end{frame}}
\newcommand\startbox[1][\vspace{-2mm}\begin{tcolorbox}[colback=cyan!5,colframe=blue
!60!black,width=\textwidth,height=\textheight-4mm,#1]}
\newcommand\stopbox{\end{tcolorbox}}

\begin{document}
\begin{myframe}[title=frame]
here
\[
f(x)
\]
\end{myframe}

\begin{myframe}[title=frame,colback=red!5,colframe=red!60!black]
here
\[
f(x)
\]
\end{myframe}
\end{document}

```

第二个例子 frametitle 在 tcolorbox 外部, tcolorbox 没有 title

```

\documentclass{beamer}

\setbeamersize{text margin left=4mm,text margin right=4mm}

\usepackage{CJKutf8}
\usepackage{listings}
\usepackage{tcolorbox}

\newenvironment{myframe}[1][\begin{frame}[fragile,environment=myframe]\frametitle{\
hskip5pt #1}\startbox}{\stopbox\end{frame}}
\newcommand\startbox[1][\begin{tcolorbox}[colback=green!5,colframe=black!20,width=\
textwidth,height=\textheight-1.5cm,#1]}
\newcommand\stopbox{\end{tcolorbox}}

\begin{document}
\begin{myframe}[media][colback=green!5,colframe=blue!50!black]
here
\[
f(x)
\]
\end{myframe}

```

```
\begin{myframe}[media][colback=red!5,colframe=red!50!black]
here
\[
f(x)
\]
\end{myframe}
\end{document}
```

这两个例子功能比较少，更复杂的结果需要带入更多的参数。tcolorbox 很漂亮，以重新定义环境的方式将 tcolorbox 融合到 beamer 里还是不错的。不过这两个例子都不支持 \framesubtitle，因为当添加子标题时，标题和子标题加起来占据的竖直距离会变大，所以需要添加判断语句，设置出不同情况时的样式。笔者一直觉得子标题时，应该保持标题竖直间距不变，改变标题和子标题的上下位置。

7.12 一些其他功能

可以将每页 ppt 打印到一页

```
\pgfpagesuselayout{2 on 1}[a4paper,border shrink=5mm]
```

给 ppt 做笔记，并且放在并列的位置。无论有没有添加笔记，并列的 note 页会在上面自动显示当前页的树状目录和一个缩小的当前页内容显示。添加了 note 后，在并列的这两部分下面，显示 note 内容。

```
\setbeameroption{show notes on second screen}
```

在frame环境内写当前页的 note

```
\begin{frame}{}
ppt的内容
\note{ppt当前页的的笔记}
\end{frame}
```

如果 frame 存在逐次显示的内容，note 也可以指定哪次显示的 note。下面这种情况可以不加[item]。

```
\begin{frame}{title}
\begin{itemize}[<+>]
\item 111
\item 222
\end{itemize}
\note[item]<2>{Tell joke}
\end{frame}
```

如果[<+>]是设置在 frame 环境参数内的，就不能添加[item]。

```
\begin{frame}[<+>]{title}
\begin{itemize}
\item 111
\item 222
\end{itemize}
```

```
\note<2>{Tell joke}  
\end{frame}
```


索引

abstract, 56
amscd, 114
amsmath, 94
amssymb, 115
anyfontsize, 20
array, 43

background, 88
balance, 56
beamer, 241
biber, 10, 62
biblatex, 10, 62
bibtex, 8, 59
bibunits, 62
bicaption, 40
bigints, 108
bm, 108
booktabs, 45

caption, 79
chapterbib, 60
chemfig, 219
CJKutf8, 2
colortbl, 45
ctex, 7

empheq, 115
epstopdf, 17
esint, 107
eso-pic, 88
exam, 212

fancyhdr, 86
ffuserguide, 92
float, 90
flowfram, 91
footmisc, 87
footnpag, 87
framed, 36

geometry, 31
glossaries, 14
graphicx, 38

hangindent, 23
hyperref, 52

ifthen, 14
imakeidx, 12
include, 5
indentfirst, 23
input, 5

letter, 88
longtable, 51
lscap, 32

makecell, 47
makeglossaries, 15
makeidx, 11
makeindex, 11, 13
marginnote, 26
mathabx, 117
mathrsfs, 117
midpage, 25
minipage, 26
multicol, 89
multirow, 49

natbib, 8
natlib, 59
nicematrix, 111
nomencl, 12
ntheorem, 119

oplotsymb, 116

paracol, [91](#)
 pdfpages, [32](#)
 ps2eps, [17](#)
 psnfss2e, [22](#)
 pstricks, [15](#)

 setspace, [21](#)
 showexpl, [202](#)
 standalone, [197](#)
 subcaption, [42](#)
 subfig, [42](#)
 subfigure, [41](#)

 texdoc, [1](#)
 threeparttable, [50](#)
 threeparttablex, [51](#)
 tikz, [123](#)

- 3d, [172](#)
- angles, [138](#)
- arrows.meta, [129](#)
- backgrounds, [144](#)
- calc, [173](#)
- circuitikz, [182](#)
- circuits.ee, [179](#)
- circuits.ee.IEC, [179](#)
- circuits.logic.IEC, [179](#)
- decorations.markings, [167](#)
- decorations.pathmorphing, [167](#)
- decorations.shapes, [167](#)
- hf-tikz, [186](#)
- intersections, [176](#)
- karnaugh, [188](#)
- math, [145](#)
- matrix, [178](#)
- mindmap, [179](#)
- perspective, [173](#)
- pgf, [123](#)
- pgf-spectra, [186](#)
- pgfornament, [189](#)
- pgfornament-han, [189](#)
- pgfplots, [183](#)
- pinoutikz, [183](#)
- positioning, [149](#)
- quantikz, [187](#)
- quotes, [138](#)
- sa-tikz, [183](#)
- shadings, [142](#)
- shapes.arrows, [164](#)
- shapes.geometric, [160](#)
- shapes.multipart, [164](#)
- shapes.symbols, [164](#)
- through, [176](#)
- tikz-3dplot, [171](#)
- tikz-among-us, [189](#)
- tikz-cd, [188](#)
- tikz-dependency, [188](#)
- tikz-dimline, [188](#)
- tikz-feynhand, [187](#)
- tikz-feynman, [187](#)
- tikz-network, [188](#)
- tikz-optics, [187](#)
- tikz-palattice, [187](#)
- tikz-planets, [187](#)
- tikz-relay, [183](#)
- tikz-timing, [183](#)
- tikz-trackschematic, [188](#)
- tikz-truchet, [189](#)
- tikzducks, [189](#)
- tikzlings, [189](#)
- tikzmark, [188](#)
- tikzmarmots, [189](#)
- tikzorbital, [187](#)
- tikzpeople, [189](#)
- tikzrput, [188](#)
- tikzsymbols, [189](#)

 titleps, [66](#)
 titlesec, [66](#)
 titletoc, [66](#)

 ulem, [28](#)
 umoline, [28](#)

 write18, [9](#)

 xcolor, [33](#)
 xdvipdfmx, [16](#)
 xeCJK, [6](#)
 xmpmulti, [251](#)