



本科毕业论文（设计）
（二〇二一届）



题 目 怎样用 LaTeX 排版一个双面打印，标题长得没
必要的毕业论文

学 院 大数据与智能工程学院 专 业 数据科学与大数据技术

学生姓名 王晓林 学 号 20161152888

指导教师 GGG (凸凸), 凸凸凸 (凸凸), 凸凸凸 (凸凸)

评 阅 人 评阅人姓名（职称）

2021 年 5 月 20 日

原创性声明

本人郑重声明，所呈交的学位论文是本人在指导教师指导下进行的研究工作及取得的研究成果，论文成果归西南林业大学所有。尽我所知，除了论文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得西南林业大学或其他教育机构的学位或证书而使用过的材料。与我共同工作的同志对本研究所作的贡献均已在论文中作了明确的说明。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

作者签名：  日期： 2021 年 5 月 20 日

怎样用 LaTeX 排版一个双面打印，标题长得没必要的毕业论文

王晓林

(西南林业大学 大数据与智能工程学院 昆明 650224)

摘 要： \LaTeX^1 是一种基于 \TeX^2 文件格式的排版系统，由美国电脑学家莱斯利·兰伯特在 20 世纪 80 年代初期开发。利用这种格式，即使用户没有排版和程序设计的知识，也可以充分利用 \TeX 所提供的强大功能，在几天，甚至几小时内生成很多具有书籍质量的印刷品。对于生成复杂表格和数学公式，这一点表现得尤为突出。因此它非常适用于生成高印刷质量的科技和数学类文档。这个系统同样适用于生成从简单如信件到完整如书籍的所有种类的文档^[1]。

本文对如何利用 \LaTeX 来撰写西南林业大学本科毕业论文做一个简要的介绍。读者也可以将本文作为毕业论文模板来使用³。

关键词： \LaTeX ；西南林业大学；本科毕业论文模板；教程

¹<https://en.wikipedia.org/wiki/LaTeX>

²<https://en.wikipedia.org/wiki/TeX>

³本文所使用的毕业论文模板（`cls` 文件），`tex` 源文件，及其它相关文件都可以在下面的网址找到：<https://github.com/wx672/texmf/tree/master/doc/latex/swfu/swfcthesis/tutorial>

How to write your thesis in L^AT_EX

WANG Xiaolin

College of Big Data and Intelligence Engineering
Southwest Forestry University
Kunming 650224, Yunnan, China

Abstract: L^AT_EX is a document preparation system. When writing, the writer uses plain text as opposed to formatted text as users of word processors like Microsoft Word. The writer uses markup tagging conventions to define the general structure of a document (such as article, book, and letter), to stylise text throughout a document (such as bold and italic), and to add citations and cross-references. A T_EX distribution such as T_EXLive or MikT_EX is used to produce an output file (such as PDF or DVI) suitable for printing or digital distribution. Within the typesetting system, its name is stylised as L^AT_EX^[2].

This short tutorial shows you how to write an undergraduate thesis in L^AT_EX. It can also serve as a perfect template to ease your thesis writing⁴.

Key Words: L^AT_EX; T_EX; thesis; template; SWFU

⁴The class file used for typesetting this thesis, and the `tex` source of this file can be found at <https://github.com/wx672/texmf/tree/master/doc/latex/swfu/swfcthesis>.

目录

1	工欲善其事，必先利其器	1
1.1	工作环境	1
2	快速上手	5
2.1	用 LaTeX 写文章就是在编程	5
2.1.1	hello.c	5
2.1.2	hello.tex	6
2.1.3	什么是 <code>Ctrl</code> + <code>x</code> <code>Ctrl</code> + <code>f</code> ?	7
2.2	生活可以更轻松	8
2.2.1	Top matter	9
2.2.2	怎样写摘要	11
2.2.3	怎样分章节	13
2.2.4	什么是环境	16
2.2.5	制作表格	19
2.2.6	引用参考文献	23
2.3	小结	26
3	入门以后	29
3.1	插入图片	29
3.1.1	Figure 环境	30
3.1.2	<code>\label{}</code> 和 <code>\ref{}</code> 的用法	32
3.2	数学公式	34
3.3	插入程序代码	35
3.4	处理中文	36
4	完美的毕业论文	37

4.1	Class 文件	37
4.2	模板的使用	38
4.3	生成参考文献	39
4.4	小结	40
参考文献		43
指导教师简介		44
致 谢		46
附录 A 西南林业大学本科毕业论文（设计）撰写规范及要求（2019）		49
A.1	文档材料的写作规范	49
A.1.1	封面	49
A.1.2	原创性声明	49
A.1.3	中文摘要（含关键词）	49
A.1.4	外文摘要（含关键词）	49
A.1.5	目录	49
A.1.6	论文正文	51
A.1.7	设计图纸	52
A.1.8	注释和参考文献（注释采用尾注）	52
A.1.9	指导教师简介	52
A.1.10	致谢	52
A.1.11	附录	53
A.1.12	其它要求	53
A.2	毕业论文（设计）文本打印格式要求	54
A.3	毕业论文（设计）文本装订规范	55
A.4	其他要求	56
附录 B 与排版论文相关的软件清单		57

附录 C 中文 article 模板	59
附录 D 毕业论文模板	61
附录 E swfuthesis.cls 文件	63
附录 F tutorial.bib 文件	65

插图目录

2.1	Top matter	12
2.2	输出效果	17
2.3	itemize 的输出效果	18
2.4	itemize 的输出效果	19
3.1	Linux logo	32
3.2	插入代码示例	35
A.1	封面标准	50
A.2	单幅插图示例	53
A.3	插图示例	54

表格目录

2.1	常用快捷键	18
2.2	章节层次	22
A.1	文献类型和标志代码	54
A.2	方弯管内流动最大速度比较	54

1 工欲善其事，必先利其器

用 L^AT_EX 撰写毕业论文是一件赏心乐事，前提是你得会。网上关于 L^AT_EX 的入门教程很多，下面这几个就很不错，花一两天时间去熟悉一下，

1. L^AT_EX from Wikibooks¹
2. The very short guide to typesetting with L^AT_EX²
3. The not so Short Introduction to L^AT_EX³

有了对 L^AT_EX 的基础认识之后，再接着往下看。

1.1 工作环境

工作环境当然要清静、优雅。一杯热咖啡，配上轻音乐……除此之外，你还需要一台电脑，最好是像我用的电脑，

硬件： 不用太高级，但内存最好大一些，不少于 4G 为宜，因为在我看，CPU 已经足够快了，电脑运行是否顺畅，更多的是取决于内存是否充足；

操作系统： Debian GNU/Linux⁴ 的测试 (testing) 版。为什么不用 MS Windows⁵? 因为 ...当然不是它不好，而是我缺乏足够的耐心。我的 Debian 系统从来不考验我的耐心；

桌面环境： 不需要。有个 window manager 就足够了。我喜欢 Sawfish window manager⁶，它界面足够的简单，功能足够的丰富，配置足够的容易。而且和 Emacs 一

¹<https://en.wikibooks.org/wiki/LaTeX>

²<http://tug.ctan.org/info/latex-veryshortguide/veryshortguide.pdf>

³<https://www.ctan.org/tex-archive/info/lshort/english/?lang=en>

⁴<https://en.wikipedia.org/wiki/Debian>

⁵https://en.wikipedia.org/wiki/Microsoft_Windows

⁶[https://en.wikipedia.org/wiki/Sawfish_\(window_manager\)](https://en.wikipedia.org/wiki/Sawfish_(window_manager))

样，它也是用 Lisp⁷ 写成的，算是 Emacs⁸ 的近亲吧。这是我偏爱 Sawfish 的一个重要原因。

如果你有个大屏幕，那么可以试试 i3⁹，一个很不错的平铺式窗口管理器（tiling window manager¹⁰）。它可以自动让所有的窗口互不重叠地铺满整个屏幕，非常适合大屏幕操作。对于我 14” 的小屏幕，还是 Sawfish 更友好些，因为它能方便地让我全屏操作。Sawfish 和 i3 有个共同的优点，就是支持全键盘操作，有了它们，你可以完全忘掉鼠标的存在。

除了窗口管理器，显然你还需要几个“窗口”。下面这三个恐怕是必须有的，

Web 浏览器 原来我用 chromium¹¹，今年（2019 年）初开始，改用 qutebrowser¹²，因为它小、快、灵、省内存、支持全键盘操作；

PDF 阅读工具 我用 Emacs 的 pdf-tools 插件。在 Emacs 里，可以在一个 buffer 里写 tex，在另一个 buffer 里看 PDF。还可以很方便地在两个 buffer 中对应的位置跳来跳去；

终端 我用 st¹³ + tmux¹⁴。

「如果你爱他，逼着他用我的桌面环境；如果你恨他，逼着他用我的桌面环境。因为这里没有鼠标！」

L^AT_EX 套件： TeXLive¹⁵，它完备而庞大，但我们只需要其中很少的一些软件包就够了。

附录 B 中列出了利用本教程写论文所需的所有软件包，可供参考。

中文字体： 完全不必操心，因为 TeXLive 有很好的中文支持。如果抛开 T_EX 不谈，就日常使用而言，我比较喜欢 Noto 字体¹⁶。它是 Google 推出的开源字体。Debian

⁷[https://en.wikipedia.org/wiki/Lisp_\(programming_language\)](https://en.wikipedia.org/wiki/Lisp_(programming_language))

⁸<https://en.wikipedia.org/wiki/Emacs>

⁹[https://en.wikipedia.org/wiki/I3_\(window_manager\)](https://en.wikipedia.org/wiki/I3_(window_manager))

¹⁰https://en.wikipedia.org/wiki/Tiling_window_manager

¹¹[https://en.wikipedia.org/wiki/Chromium_\(web_browser\)](https://en.wikipedia.org/wiki/Chromium_(web_browser))

¹²<https://en.wikipedia.org/wiki/Qutebrowser>

¹³<https://en.wikipedia.org/wiki/Suckless.org>



¹⁴<https://en.wikipedia.org/wiki/Tmux>

¹⁵https://en.wikipedia.org/wiki/TeX_Live

¹⁶https://en.wikipedia.org/wiki/Noto_fonts

库里自带，装上就好。所谓“Noto”就是 No tofu 的意思，也就是说 Noto 字体的终极目标是消灭所有的豆腐块（缺字）。Noto 有专门的 CJK 字体包，只是尚缺楷体。楷体我用 Debian 库里自带的 arphic-ukai，它是台湾文鼎科技¹⁷推出的开源字体。你当然也可以选用 Windows 字体，只要从你的 Windows 系统里拷贝过来就行了。这也许是你人生中唯一需要 Windows 的时候。

编辑器： Emacs + AUCTeX¹⁸ + Yasnippet¹⁹ + pdf-tools²⁰；

- Emacs 是世界上最强大的编辑器^[3]。作为计科专业的学生，如果你不熟悉它的使用，怎么好意思写毕业论文呢？
- AUCTeX 是一个历史悠久、功能强大的 Emacs 插件，它为我们编辑 L^AT_EX 文件提供了丰富的快捷键操作^[4]；
- Yasnippet 也是 Emacs 的插件，它的功用就是为  键施加魔法^[5]。有了它，期待奇迹出现的时候，你只要左手小指在  键上轻轻一按……当然，你得先学会写魔咒 (snippets) 才行 ☺。放松，snippets 都是很简单的小东西，去看看 `.emacs.d/snippets/latex-mode/` 目录里的东西，我担保你能无师自通。
- pdf-tools，前面已经提过了。即使不写 T_EX，单纯地为了阅读 PDF 文件，pdf-tools 也是目前我认为最好用的阅读器，因为除了阅读，还可以标注 PDF。

简而言之，有了上面几个插件，Emacs 就成了一个强大的 L^AT_EX 排版 IDE²¹。有了它，写起论文来，就像领导说套话一样顺滑流畅。

附录 B 中列出了我的 Debian 系统上安装的与写论文相关的所有软件包，当然，清单里的东西并非都是必需。其实，除了中文字体之外，其它的一切，你都可以根据自己的偏好来选择。如果你打算用英文写论文的话，那么连中文字体都可以省略了。不管怎么说，上面提到的都是我个人的偏好，本教程也将以此为基础，逐步展开。

¹⁷https://en.wikipedia.org/wiki/Arphic_Technology

¹⁸<https://en.wikipedia.org/wiki/AUCTeX>

¹⁹<https://www.emacswiki.org/emacs/Yasnippet>

²⁰<https://github.com/politza/pdf-tools>

²¹集成开发环境 (Integrated Development Environment)。

至于如何安装、配置好这样一个工作环境,如果你是本校的学生,那么当然可以直截找我帮忙。如果找不到我,那么你可以参考一下我曾经写过的一个比较潦草的《Debian 安装指导》²²,也许有点过时,但应该还是能对你有点帮助的。

²²<https://github.com/wx672/lecture-notes/blob/master/linux/tutorials/install/install.html>















2 快速上手

L^AT_EX 很强大，但对于初学者来说，你不必关心它有多强大，因为最为常用的命令和环境不过寥寥数个而已。而且 Emacs+AUCT_EX 提供了丰富的快捷键，稍加练习之后，你就再也不用去手工敲命令了。对于较复杂的格式需求，通常只要套用模版就可以解决问题了。所以，大家只要把 Emacs 用熟，一切迎刃而解。





2.1 用 LaTeX 写文章就是在编程

2.1.1 hello.c

我们先回忆一下用 Emacs 写一个 hello.c 的键盘操作过程：







1. +，如果你的系统配置和我的一样，那么只要按下 + 键，Emacs 窗口就出现在你面前了，而且（感谢 Sawfish）是全屏的；
2. ++，开始编辑一个新文件。这时，在 Emacs 窗口的最下面（也就是 mini buffer 里）有提示，输入你要编辑的文件的名字，也就是 hello.c，然后按 （回车键），或者 +，其实，后面你会发现，+ 带自动缩进，比按  更方便；
3. 现在可以在打开的空文件里写东西了：

```
1 | #include <stdio.h>
2 | int main()
3 | {
4 |     printf ("Hello, world!\n");
5 |     return 0;
6 | }
```





4. 存盘：++
5. 编译：`gcc hello.c`
6. 运行：`./a.out`

2.1.2 hello.tex

再看看用 L^AT_EX 写一个 hello.tex 文件的过程：




1. +, 打开 Emacs;
2. + +, hello.tex, 开始编辑;
3. 写入文件内容:

```
1 | \documentclass{article}
2 | \begin{document}
3 | Hello, world!
4 | \end{document}
```

4. 存盘: + +
5. 编译: xelatex hello.tex
6. 看结果: xpdf hello.pdf

怎么样? hello.c 和 hello.tex 的编辑过程没什么分别吧。只要把 Emacs 用熟, 触类旁通, 不管写什么程序, 都是这么个过程。你

- 不必学习 VC 去写 C/C++;
- 不必学习 eclipse 去写 Java;
- 不必学习 MS-Word 去写报告、幻灯片;
- 不必学习……

一句话, “Everything Emacs”, 你可以省下大量不必要的学习时间。人生苦短, 何必让你的生活被 VC/eclipse/MS-Word 搞得头昏脑胀呢? 简单而强大, 就是计科专业和非专业学生所应有的不同。如果你对 Emacs 操作还很陌生, 那么现在就打开 Emacs, + , 重温一下那些基本操作吧。

2.1.3 什么是 $\text{Ctrl} + \text{x} \text{Ctrl} + \text{f}$?

使用 Emacs 是可以（而且应该）完全抛开鼠标的。对于初次上手的人，抛开鼠标就像病人抛开拐杖一样痛苦。但只有不依赖拐杖的人才是健康的，不是吗？现在，你终于有了走向健康生活的冲动，那么我们开始吧。简短截说，

1. 先把你的双手在标准键盘上放好。然后，
2. 左手小指稍向左移，按在 CapsLock 键上¹，按住别放开，
3. 左手无名指稍向下移，在 x 键上轻按一下就放开，这就是 $\text{Ctrl} + \text{x}$ ；
4. 小指按在 CapsLock 上别放开，左手食指在 f 键上轻按一下就放开，这就是 $\text{Ctrl} + \text{f}$ ；
5. 现在左手小指可以放开了。

这就是 $\text{Ctrl} + \text{x} \text{Ctrl} + \text{f}$ ，最常用的 Emacs 快捷键之一，作用是要求打开一个文件²， f 代表 file。那么，告诉我

- 什么是 $\text{Ctrl} + \text{x} \text{Ctrl} + \text{s}$ ？
- 什么是 $\text{Ctrl} + \text{x} \text{2}$ ？什么是 $\text{Ctrl} + \text{x} \text{3}$ ？什么是 $\text{Ctrl} + \text{x} \text{o}$ ？什么是 $\text{Ctrl} + \text{x} \text{0}$ ？
什么是 $\text{Ctrl} + \text{x} \text{1}$ ？
- 什么是 $\text{Ctrl} + \text{x} \text{h}$ ？什么是 $\text{Ctrl} + \text{w}$ ？
- 什么是 $\text{Ctrl} + \text{g}$ ？
- 什么是 $\text{Ctrl} + \text{j}$ ？什么是 $\text{Ctrl} + \text{i}$ ？
- 什么是 $\text{Ctrl} + \text{k}$ ？什么是 $\text{Ctrl} + \text{y}$ ？
- 什么是 $\text{Ctrl} + \text{d}$ ？什么是 $\text{Ctrl} + \text{d}$ ？
- 什么是 $\text{Ctrl} + \text{a}$ ？什么是 $\text{Ctrl} + \text{e}$ ？什么是 $\text{Ctrl} + \text{f}$ ？什么是 $\text{Ctrl} + \text{b}$ ？
什么是 $\text{Ctrl} + \text{n}$ ？什么是 $\text{Ctrl} + \text{p}$ ？







¹如果你的系统配置和我一样，那么 CapsLock 就是 Ctrl 键。



²如果你要打开的文件不存在，那么 Emacs 会认为你要写一个新文件。

如果你还不熟悉上面这些快捷键，那么用起 Emacs 来，就会像西洋人用筷子一样不酷。（「嘿！把你的手从鼠标上拿开！」）刚甩开拐杖，走向健康生活的人，总会不自觉地扶点什么，这很正常。但你一定要坚持锻炼，不要让这种「正常」持续得太久。今后的生活能否轻松愉快，主要就取决于你的「健康」程度。如果有朝一日你真的抛开了鼠标，那么即使面对一个纯字符界面的终端，你也能写出漂亮的 PDF 格式的论文。作为计科专业的学生，好歹该比网吧青年们酷一些嘛。











2.2 生活可以更轻松





AU \TeX 是 Emacs 的一个功能模块，为 L \TeX 编程提供了巨大的便利。有了它，你的 L \TeX 生活可以像 Hello, world! 一样简单。现在就跟着我，手把手地领略一下简单的乐趣吧。

一切当然是从 +，打开 Emacs 开始。然后，+ +，让我们开始编辑一个新文件，就叫 simple.tex 吧。



在 Emacs 窗口的最下方，也就是 mini buffer 里，这时应该会有提示，让你输入文件名。输入 simple.tex，然后按 +。如果这时 mini buffer 里有如下提示：

```
Master file: (default this file) ...
```

直接按 + 就可以了。知道了吧，+ 就是我们的 （回车）键。如果你的手正放在「标准键盘」上，那么，左手小指向左一偏，按到的正是  键（ (CapsLock) 被我们改造成  了）。右手食指下不正是  键吗？怎么样，比  更方便吧。

现在，可以向 simple.tex 文件里写东西了，+ +，e 代表 environment。“环境”到底是什么呢？意会吧，用用就明白了。在 mini buffer 里会有提示，

```
Environment type: (default document)
```

这是在问你是不是要写一篇 document（文章）啊？你当然该用 + 来告诉它「是」。这时，mini buffer 又会提示，

```
Document class: (default article)
```

这是在问你是不是要写一篇 article 类型的文章啊？除了 article，通常还有 book, report, letter 可供选择。我们现在碰巧就是要写个短小的 article，所以，按 **Ctrl** + **j** 确认就好。这时，mini buffer 继续提示，

Options:

这是在问你是否有有什么特殊选项啊？用 **Ctrl** + **j** 来告诉它说「不需要」。现在，你的 simple.tex 文件里应该有如下几行东西了：

```

1 | \documentclass{article} % documentclass 可以是
2 |                        % article, book, report, letter...
3 | \begin{document} % 文章的开始
4 | |
5 | \end{document} % 文章的结束

```

这时，光标停在 `\begin{document}` 与 `\end{document}` 之间，等待你的输入。百分号 (%) 后面显然都是注释。

在第 2.1 节里，你已经会写 Hello, world! 了。现在，我们要写点像模像样的东西。偷懒起见，我直接套用 Andrew Roberts 写的 simple.tex³。我们把注意力集中在用 Emacs 写文章的过程上。

2.2.1 Top matter

先确保你的光标在 `\begin{document}` 和 `\end{document}` 之间，也就是文章的第 4 行。然后按 **Ctrl** + **c** **Ctrl** + **m**，这时 mini buffer 里会有如下提示：

Macro (default ref): \

这是系统在等待你输入一个 Macro，说白了就是“命令”。输入：title，**Ctrl** + **j**，这时你的文章会变成下面这样：

```

1 | \documentclass{article} % documentclass 可以是
2 |                        % article, book, report, letter...
3 | \begin{document} % 文章的开始
4 | \title{ }
5 | \end{document} % 文章的结束

```

³<http://en.wikibooks.org/wiki/LaTeX/simple.tex>

这时，光标停在 `\title{}` 的花括号里。不用说你也知道，该输入文章的标题了。那么就给它一个标题：

```

1  \documentclass{article} % documentclass 可以是
2                             % article, book, report, letter...
3  \begin{document} % 文章的开始
4  \title{How to Structure a \LaTeX{} Document}
5  \end{document} % 文章的结束

```

发现了吗？凡是以反斜杠开头的都是命令 (Macro)，比如 `\LaTeX{}`，它的唯一作用就是把 LaTeX 这五个字母输出成一副怪样子，`ℒℒℒℒℒ`。

好了，在 title 下新起一行。然后 `Ctrl`+`m`。你肯定知道 `Ctrl`+`m` 是干什么用的了吧，就是要输入一个 Macro。也许你会好奇，想知道总共有多少 Macro？那么现在可以按一下 `⌘` 键。看到了吗？在弹出的新窗口中，列出了近百个 Macro。还好，我们并不需要记住这么多。最常用的也就三、五个而已。

mini buffer 里又会有提示：

```
Macro (default title): \
```

Emacs 会把我们上次输入的 Macro，也就是 title，做为默认值提示出来。不用管它，输入：`author` `Ctrl`+`j`。然后在 `\author{}` 的花括号里输入作者的名字。当然，也可以把自己的通信地址、email 写在里面。就像下面这样：

```

1  \documentclass{article} % documentclass 可以是
2                             % article, book, report, letter...
3  \begin{document} % 文章的开始
4  \title{How to Structure a \LaTeX{} Document}
5  \author{Andrew Roberts\\
6          School of Computing,\\
7          University of Leeds,\\
8          Leeds,\\
9          United Kingdom,\\
10         LS2 1HE\\
11         \emph{andyr@comp.leeds.ac.uk}}
12 \end{document} % 文章的结束

```

注意，`\\`代表“强制换行”。现在，新起一行，加上日期：

1. `Ctrl`+`c` `Ctrl`+`m` date `Ctrl`+`j`
2. `Ctrl`+`c` `Ctrl`+`m` today `Ctrl`+`j`

其实，如果没有 `\date{\today}` 这一句，系统会自动把今天的日期添加上的。而且 `\date{}` 里面的日期你可以随意写，不一定非要是当天的日期。title, author, date 一般被叫做文章的 top matter（开头那点事）。

再新起一行，写 `\maketitle` `Ctrl` + `j`。`\maketitle` 自然是要排版 top matter 了。换句话说，不要标题的话可以省略掉这个命令。现在文章变成了这样：

```

1  \documentclass{article} % documentclass 可以是
2                               % article, book, report, letter...
3  \begin{document}         % 文章的开始
4  \title{How to Structure a \LaTeX{} Document}
5  \author{Andrew Roberts\\
6          School of Computing,\\
7          University of Leeds,\\
8          Leeds,\\
9          United Kingdom,\\
10         LS2 1HE\\
11         \emph{andyr@comp.leeds.ac.uk}}
12 \date{\today}
13 \maketitle
14 \end{document}          % 文章的结束

```

好奇的话，现在可以编译一下，看看 PDF 文件的效果：

1. 编译： `Ctrl` + `c` `Ctrl` + `c` `Ctrl` + `j`
2. 查看： `Ctrl` + `c` `Ctrl` + `v`

这时，一个 PDF 文件应该显示在屏幕上了（图 2.1(a)）。效果还满意吧？保持你的好奇心。在下面的操作中，你随时可以编译一下看看效果。

2.2.2 怎样写摘要

好了，回到 Emacs。现在你的光标应该停在 `\maketitle` 的下面一行。我们开始写「摘要」部分。 `Ctrl` + `c` `Ctrl` + `e`，开始一个新的“环境”⁴。mini buffer 里提示：

```
Environment type: (default itemize)
```

这是在问你要开始哪个环境啊？默认是开始 itemize 环境，因为它是最常用的环境。但我们现在要写的是“摘要”，告诉它：abstract `Ctrl` + `j`。abstract 就是“摘要”的意思。科技论文都是要有摘要的嘛。于是，你的文章变成了这样：

⁴如果你好奇心强，想知道总共有哪些“环境”的话，现在可以按 `Ctrl` + `h` 键。

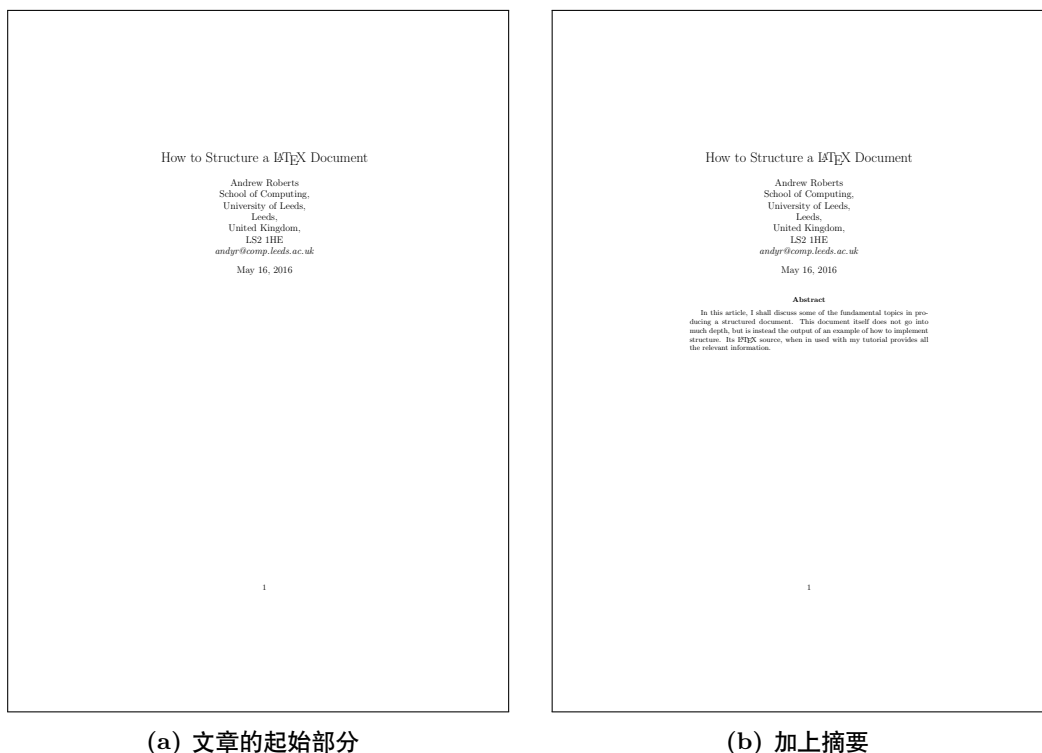


图 2.1 Top matter

```

1  % 此处略去十数行
2
3  \maketitle
4
5  \begin{abstract}
6  |
7  \end{abstract}
8  \end{document}           % 文章的结束

```

光标停在 `\begin{abstract}` 和 `\end{abstract}` 之间 (第 6 行)。好, 现在往摘要部分里填点东西:

```

1  % 此处略去十数行
2
3  \maketitle
4
5  \begin{abstract}
6  In this article, I shall discuss some of the fundamental
7  topics in producing a structured document. This
8  document itself does not go into much depth, but is
9  instead the output of an example of how to implement
10 structure. Its \LaTeX{} source, when in used with
11 my tutorial provides all the relevant information.
12 \end{abstract}
13 \end{document}           % 文章的结束

```

看看效果 (图 2.1(b))。

2.2.3 怎样分章节

接着上面的例子,我们来写点更多更长的东西。偷懒起见,文章末尾的 `\end{document}` 我也不再写出来了。

好,按 `Ctrl` + `n` 把光标移到 `\end{abstract}` 的下一行。然后, `Ctrl` + `c` `Ctrl` + `s`, 让我们开始文章的第一节。s 代表 section, “节”的意思。mini buffer 提示:

```
Level: (default section)
```

显然是在问你,要不要起一个新 section 啊? 没错,我就是要起一个新的章节,于是直接 `Ctrl` + `j`。mini buffer 又提示:

```
Title:
```

也就是问你,章节标题是……? 那就给它个标题吧,就叫“Introduction”。`Ctrl` + `j` 之后, mini buffer 继续提示:

```
Label: sec:introduction
```

这是在问你,要不要给这个新章节打个标签,比如 `sec:introduction`,以后也许要索引到它呢? 这个暂时无关紧要, `Ctrl` + `j` 就行了。于是,文中又有了下面的第 5、6 两行。

```
1 | % 此处略去十数行
2 |
3 | \end{abstract}
4 |
5 | \section{Introduction}
6 | \label{sec:introduction}
```

给这一节添加内容:

```

1 % 此处略去十数行
2
3 \end{abstract}
4
5 \section{Introduction}
6 \label{sec:introduction}
7
8 This small document is designed to illustrate how easy
9 it is to create a well structured document within
10 \LaTeX\cite{lamport94}. You should quickly be able to
11 see how the article looks very professional, despite the
12 content being far from academic. Titles, section
13 headings, justified text, text formatting etc., is all
14 there, and you would be surprised when you see just how
15 little markup was required to get this output.

```

注意到了吗？在这一节里有一个新命令 `\cite{}`，这是在引用一个参考文献。先不管它，后面再说。

如法炮制，再添加几个章节：

```

1 % 此处略去十数行
2
3 \end{abstract}
4
5 \section{Introduction}
6 \label{sec:introduction}
7
8 This small document is designed to illustrate how easy
9 it is to create a well structured document within
10 \LaTeX\cite{lamport94}. You should quickly be able to
11 see how the article looks very professional, despite the
12 content being far from academic. Titles, section
13 headings, justified text, text formatting etc., is all
14 there, and you would be surprised when you see just how
15 little markup was required to get this output.
16
17 \section{Structure}
18 \label{sec:structure}
19
20 One of the great advantages of \LaTeX{} is that all it
21 needs to know is the structure of a document, and then it
22 will take care of the layout and presentation itself. So,
23 here we shall begin looking at how exactly you tell
24 \LaTeX{} what it needs to know about your document.
25
26 \subsection{Top Matter}
27 \label{sec:top-matter}
28
29 The first thing you normally have is a title of the
30 document, as well as information about the author and
31 date of publication. In \LaTeX{} terms, this is all

```

```

32 | generally referred to as \emph{top matter}.
33 |
34 |

```

注意到 `\emph{}` 了吗？它代表 *emphasize*，“强调”。英文习惯用斜体字来表示强调的东西，那么 `\emph{hello, world}` 自然就是把 *hello, world* 排版成 *hello, world* 了。

注意到 `\subsection{}` 了吗？一会儿，我们还会看到 `\subsubsection`。不用解释吧，文章的章节次序是这样：

- | | | |
|------------|-----------------------|----------------------|
| 1. chapter | 3. subsection | 5. paragraph |
| 2. section | 4. subsub-
section | 6. subpara-
graph |

其中的 `chapter`，只有在 `book` 和 `report` 中才能使用，而 `article` 只能用 `section` 以下的东西。

现在我们就来增加一个 `subsubsection`。不出所料的话，光标现在应该在第 34 行。那么就 `Ctrl` + `c` `Ctrl` + `s`，mini buffer 提示：

```
Level: (default subsection)
```

当然输入： `subsubsection` `Ctrl` + `j`。mini buffer 提示：

```
Title:
```

输入： `Article Information` `Ctrl` + `j`。mini buffer 提示：

```
Label: sec:article-information
```

似曾相识吧？敲 `Ctrl` + `j`，于是，文章中又有了如下两行：

```

1 | \subsubsection{Article Information}
2 | \label{sec:article-information}

```

也就是说，我们有了一个 `subsubsection`。

2.2.4 什么是环境

现在，我们来添加一个 environment。[Ctrl]+[c] [Ctrl]+[e]，mini buffer 提示：

```
Environment type: (default abstract)
```

我们当然不再需要 abstract 了，现在我们要的是 itemize，也就是“不带序号的列表”。那么当然输入：itemize [Ctrl]+[j]。于是看到：

```
1 | \begin{itemize}
2 | \item |
3 | \end{itemize}
```

光标停在 \item 的后面。非常好，这正是我想要的。于是直接输入如下文字：

```
\verb|\title{}| --- The title of the article.
```

输入之后，[Alt]+[↵]，也就是，左手拇指按住 [Alt] 键，同时右手小指去敲 [↵]。你会看到这样的效果：

```
1 | \begin{itemize}
2 | \item \verb|\title{}| --- The title of the article.
3 | \item
4 | \end{itemize}
```

也就是说，不仅换了行，而且自动有了 \item 等待你输入新的东西。

你一定注意到了 \verb|| 这个新命令。它的作用和 bash 命令行的单引号 (') 是一样的。还记得吧，在命令行，单引号里的东西是原样输出的。 \verb|| 里的东西也一样。verb 是 verbatim 一词的缩写，就是“原样引用”的意思。好奇的话，可以编译一下，看看效果（图 2.2）。

好，继续输入：

```
\verb|\date| --- The date. Use:
```

得到：

```
1 | \begin{itemize}
2 | \item \verb|\title{}| --- The title of the article.
3 | \item \verb|\date| --- The date. Use:
4 | \end{itemize}
```

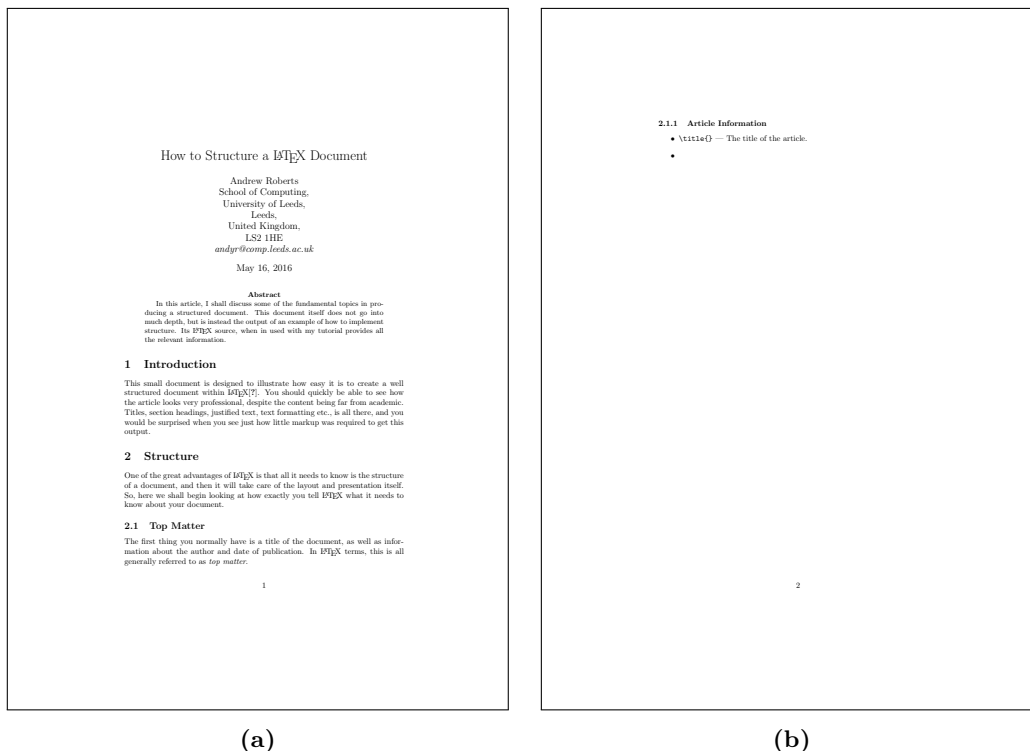


图 2.2 输出效果

没什么好说的。现在我们要在 `itemize` 环境里面再套一个 `itemize`。光标现在应该在第 3 行的最后。敲：`Ctrl` + `c` `Ctrl` + `e` `Ctrl` + `j`，于是得到：

```

1 \begin{itemize}
2 \item \verb|\title{}| --- The title of the article.
3 \item \verb|\date| --- The date. Use:
4   \begin{itemize}
5     \item
6   \end{itemize}
7
8 \end{itemize}
```

简单吧？不用说了，你肯定知道下面这些是怎么来的了吧。

```

1 \begin{itemize}
2 \item \verb|\title{}| --- The title of the article.
3 \item \verb|\date| --- The date. Use:
4   \begin{itemize}
5     \item \verb|\date{\today}| --- to get the date that
6       the document is typeset.
7     \item \verb|\date{}| --- for no date.
8   \end{itemize}
9 \end{itemize}
```

编译之后的效果应该和图 2.3 差不多。好了，请你现在照猫画虎，再来一个 `subsub-section`，标题叫 `Author Information`。模仿上面的东西，来得到下面的东西：

2.1.1 Article Information

- `\title{}` — The title of the article.
- `\date` — The date. Use:
 - `\date{\today}` — to get the date that the document is typeset.
 - `\date{}` — for no date.

图 2.3 itemize 的输出效果

```

1  \subsubsection{Author Information}
2  \label{sec:author-information}
3
4  The basic article class only provides the one command:
5  \begin{itemize}
6  \item \verb|\author{ }| --- The author of the document.
7  \end{itemize}
8
9  It is common to not only include the author name, but
10 to insert new lines (\verb|\\|) after and add things
11 such as address and email details. For a slightly more
12 logical approach, use the AMS article class (\emph{amsart})
13 and you have the following extra commands:
14
15 \begin{itemize}
16 \item \texttt{address} --- The author's address. Use the
17   new line command (\verb|\\|) for line breaks.
18 \item \texttt{thanks} --- Where you put any acknowledgments.
19 \item \texttt{email} --- The author's email address.
20 \item \texttt{urladdr} --- The URL for the author's web page.
21 \end{itemize}

```

显示效果如图 2.4 所示。怎么样，不太困难吧？目前为止，我们用到的无非是表 2.1 中列出的这几个快捷键操作而已：

表 2.1 常用快捷键

快捷键	功用
Ctrl + j	换行带缩进
Ctrl + c Ctrl + m	输入 Macro
Ctrl + c Ctrl + s	新起一个章节
Ctrl + c Ctrl + e	新起一个环境
Alt + ↵	换行带 <code>\item</code>

好，趁热打铁，再起一个小节，

1. **Ctrl** + **c** **Ctrl** + **s** subsection **Ctrl** + **j**

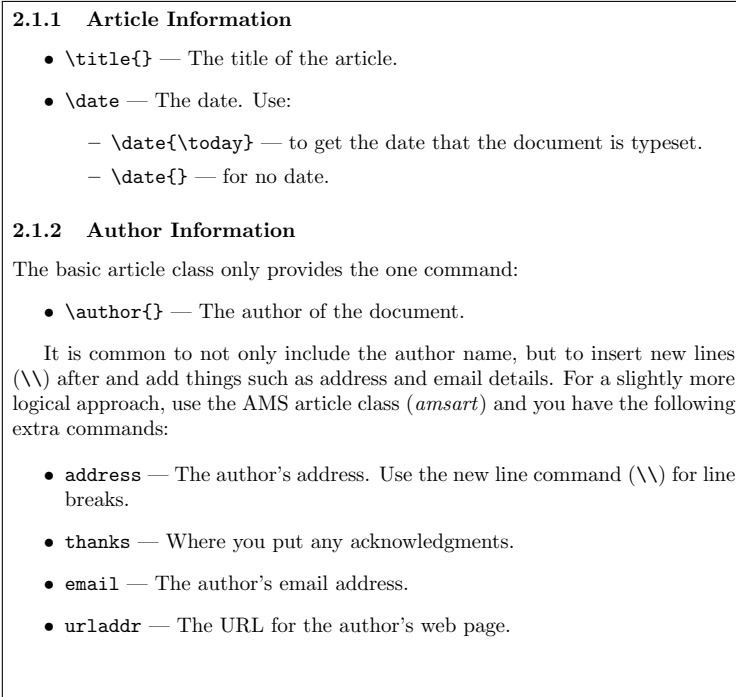


图 2.4 itemize 的输出效果

2. Sectioning Commands `Ctrl` + `j` `Ctrl` + `j`

再添加一些文字，得到：

```

1  % 此处略去数十行
2
3  \subsection{Sectioning Commands}
4  \label{sec:sectioning-commands}
5
6  The commands for inserting sections are fairly intuitive.
7  Of course, certain commands are appropriate to different
8  document classes. For example, a book has chapters but a
9  article doesn't.
10
11  % A simple table. The center environment is first set up,
12  % otherwise the table is left aligned. The tabular
13  % environment is what tells Latex that the data within
14  % is data for the table.
```

没什么新鲜东西，就不看效果了。下面抓紧说说怎么画表格。

2.2.5 制作表格

在这一小节，我们来尝试一下表格的输入。先起一个新“环境”，center，自然是“居中”的意思：

Ctrl + **c** **Ctrl** + **e** center **Ctrl** + **j**

得到：

```

1      % 此处略去数十行
2
3      \subsection{Sectioning Commands}
4      \label{sec:sectioning-commands}
5
6      The commands for inserting sections are fairly intuitive.
7      Of course, certain commands are appropriate to different
8      document classes. For example, a book has chapters but a
9      article doesn't.
10
11     % A simple table. The center environment is first set up,
12     % otherwise the table is left aligned. The tabular
13     % environment is what tells Latex that the data within is
14     % data for the table.
15
16     \begin{center}
17
18     \end{center}
```

在 center 环境里面，我们添加一个 tabular（表格）环境：

Ctrl + **c** **Ctrl** + **e** tabular **Ctrl** + **j**

这时你会看到这样的提示：

(Optional) Position:

Optional 是可有可无的意思，也就是说，你如果在意表格的位置 (Position)，那么就提供位置信息；如果不在意，那么就不用管它。现在我们连「位置」意味着什么都不清楚，自然就不必管它了。直接 **Ctrl** + **j**，又看到提示了：

Format:

这是问你，表格的格式，比如该有几列？每列之间要不要有竖线分割？等等。我的答案是这样：

```
|1|1|
```

也就是：竖线 (|)，小写 L (l)，竖线 (|)，小写 L (l)，竖线 (|)。小写 L 代表 left，也就是“左对齐”的意思。那么，你应该恍然大悟了，不就是……竖线-左对齐-竖线-左对齐-竖线嘛。那么，举一反三，除了小写 L，我们还会见到 r（右对齐）和 c（居中）。现在 `Ctrl` + `j`，得到如下结果：

```

1      % 此处略去数十行
2
3      \subsection{Sectioning Commands}
4      \label{sec:sectioning-commands}
5
6      The commands for inserting sections are fairly intuitive.
7      Of course, certain commands are appropriate to different
8      document classes. For example, a book has chapters but a
9      article doesn't.
10
11     % A simple table. The center environment is first set up,
12     % otherwise the table is left aligned. The tabular
13     % environment is what tells Latex that the data within is
14     % data for the table.
15
16     \begin{center}
17         \begin{tabular}{|l|l|}
18             &
19         \end{tabular}
20     \end{center}

```

现在我们开始画表格，先画一条横线：

`\hline` `Ctrl` + `j`

所谓 `\hline`，顾名思义，就是 horizontal line。画完横线，开始第一行，

Command & Level `\\` `\hline` `Ctrl` + `j`

那个 `&` 就是两列之间的分隔符，“`\\`”我们见过，表示强制换行。照猫画虎，把所有的行都加上，得到如下结果：

表 2.2 章节层次

Command	Level
<code>\part{}</code>	-1
<code>\chapter{}</code>	0
<code>\section{}</code>	1
<code>\subsection{}</code>	2
<code>\subsubsection{}</code>	3
<code>\paragraph{}</code>	4
<code>\subparagraph{}</code>	5

```

1  % 此处略去数十行
2
3  \begin{center}
4    \begin{tabular}{ll}
5      \hline
6      Command & Level \\ \hline
7      \verb|\part{}| & -1 \\
8      \verb|\chapter{}| & 0 \\
9      \verb|\section{}| & 1 \\
10     \verb|\subsection{}| & 2 \\
11     \verb|\subsubsection{}| & 3 \\
12     \verb|\paragraph{}| & 4 \\
13     \verb|\subparagraph{}| & 5 \\
14     \hline
15   \end{tabular}
16 \end{center}

```

这张表格的效果如表 2.2 所示。好了，表格画完了。再添加点文字：

```

1 % 此处略去数十行
2 \begin{center}
3   \begin{tabular}{ll}
4     \hline
5     Command & Level \\ \hline
6     \verb|\part{}| & -1 \\
7     \verb|\chapter{}| & 0 \\
8     \verb|\section{}| & 1 \\
9     \verb|\subsection{}| & 2 \\
10    \verb|\subsubsection{}| & 3 \\
11    \verb|\paragraph{}| & 4 \\
12    \verb|\subparagraph{}| & 5 \\
13    \hline
14  \end{tabular}
15 \end{center}
16
17 Numbering of the sections is performed automatically by
18 \LaTeX{}, so don't bother adding them explicitly, just
19 insert the heading you want between the curly braces. If
20 you don't want sections number, then add an asterisk (*)
21 after the section command, but before the first curly
22 brace, e.g., \verb|section*{A Title Without Numbers}|.

```

现在编译一下，看看效果：

2.1.1 Article Information

- `\title{}` — The title of the article.
- `\date{}` — The date. Use:
 - `\date{\today}` — to get the date that the document is typeset.
 - `\date{}` — for no date.

2.1.2 Author Information

The basic article class only provides the one command:

- `\author{}` — The author of the document.

It is common to not only include the author name, but to insert new lines (`\`) after and add things such as address and email details. For a slightly more logical approach, use the AMS article class (*amsart*) and you have the following extra commands:

- `address` — The author's address. Use the new line command (`\`) for line breaks.
- `thanks` — Where you put any acknowledgments.
- `email` — The author's email address.
- `urladdr` — The URL for the author's web page.

2.2 Sectioning Commands

The commands for inserting sections are fairly intuitive. Of course, certain commands are appropriate to different document classes. For example, a book has chapters but a article doesn't.

Command	Level
<code>\part{}</code>	-1
<code>\chapter{}</code>	0
<code>\section{}</code>	1
<code>\subsection{}</code>	2
<code>\subsubsection{}</code>	3
<code>\paragraph{}</code>	4
<code>\subparagraph{}</code>	5

Numbering of the sections is performed automatically by \LaTeX , so don't bother adding them explicitly, just insert the heading you want between the curly braces. If you don't want sections number, then add an asterisk (*) after the section command, but before the first curly brace, e.g., `section*{A Title Without Numbers}`.

2

2.2.6 引用参考文献

处理参考文献，在 \LaTeX 中有两个选择：

- 传统的 \BibTeX ；
- 时尚的 `biblatex`。

至于说两者的区别，粗略地说，`biblatex` 就是 \BibTeX 的改进版。所以，下面我们只简要介绍一下基于 `biblatex` 的参考文献排版。

第一步：准备好一个 .bib 文件

Bib 就是 Bibliography（参考文献）一词的前三个字母。顾名思义，在这个 .bib 文件里放的都是你要用到的参考文献。我们这个小教程所用到的 tutorial.bib 文件就是个典型的例子。节约篇幅起见，我们只列出该文件中的前三条记录如下：

```

1  @Misc{biblatex,
2    author = {Philipp Lehman and Philip Kime and Audrey Boruvka and Joseph
3             ↪ Wright},
4    title = {The biblatex Package},
5    month = 11,
6    year = 2016}
7
8  @misc{ simple,
9    author = "Andrew Roberts",
10   title = "A simple article to illustrate document structure",
11   year = 2003,
12   url = "https://en.wikibooks.org/wiki/LaTeX/simple.tex",
13 }
14 @Book{Goossens94a,
15   Title = {The LaTeX Companion},
16   Author = {Michel Goossens and Frank Mittelbach and Alexander Samarin},
17   Publisher = {Addison-Wesley},
18   Year = 1994,
19   Edition = {2nd revised},
20 }
```

怎么样，不难理解吧？照猫画虎地写出你自己的 .bib 文件应该不是件太困难的事情。

第二步：引用参考文献

准备好了你自己的 .bib 之后，剩下的事情就很简单了。简而言之，只要在你的 .tex 文件里做三件事情……

首先，在 .tex 文件的 preamble 部分（也就是 `\begin{document}` 之前）加上如下一行（假设你的 .bib 文件名字是 myref.bib）：

```
1 | \addbibresource{myref.bib}
```

然后，在 .tex 文件中的适当地方，你要引用到你的参考文献（也就是 .bib 文件中的相应条目）。举个例子：

L^AT_EX:

在 `\biblatex` 手册中，作者说到，`\biblatex` 是对 `\latex` 的参考文献处理模块的彻底改进 `\cite{biblatex}`。

PDF:

在 `biblatex` 手册中，作者说到，`biblatex` 是对 L^AT_EX 的参考文献处理模块的彻底改进^[6]。

上例中的 `\cite{biblatex}` 就是要引用 `myref.bib` 中的第一条记录。在编译后输出的 PDF 文件中我们会看到 “[6]”。至于为什么是 “6” 而不是 “1” 或者其它什么数字，这取决于生成参考文献时的排序选择，我们暂时不用关心它。

在 `.tex` 文件中要做的第三件事情是，在参考文献应该出现的地方加上如下一行：

```
1 | \printbibliography{}
```

通常，“参考文献” 应该作为一个章节出现在你的文章（或论文）的末尾。

第三步：编译

在上面的两步中，我们分别准备好了 `.bib` 和 `.tex` 文件，现在调用 `latexmk` 来编译一下就行了。

前面我们都是用 `xelatex` 编译 `.tex` 文件，从来没提过 `latexmk`。其实，不用 `latexmk` 当然也可以完成文件的编译，具体步骤就是：

1. `xelatex texfilename`
2. `biber texfilename`
3. `xelatex texfilename`
4. 也许还要再执行一次上一行命令

上面的第二步（调用 `biber`）就是用来处理参考文献的。如果觉得敲这三四个命令麻烦，那我们就改用 `latexmk` 算了，唯一的前提条件是准备好一个小配置文件 `.latexmkrc`，内容如下：

```
1 | $pdf_mode = 1; $postscript_mode = $dvi_mode = 0;
2 | $pdflatex = 'xelatex -interaction=nonstopmode -8bit --shell-escape %0
   | ↪ %S';
3 |
4 | $bibtex_use = 1;
5 | $biber = 'biber --debug %0 %S';
```

把这个小文件放到你自己的\$HOME 目录下就行了。然后，只要：

```
1 | latexmk texfilename
```

不出错的话，一个带参考文献的 PDF 文件就诞生了。参考文献的样式就和本文的第 43 页差不多。

2.3 小结

本章我们利用 Andrew Roberts 的 `simple.tex` 了解了一下 L^AT_EX 中最常用的命令和环境，同时也熟悉了一下 Emacs 的基本键盘操作，在此做个小结。

L^AT_EX 最常用到的命令和环境

<code>\title{}</code>	<code>\author{}</code>	<code>\date{}</code>
<code>\section{}</code>	<code>\subsection{}</code>	<code>\subsubsection{}</code>
<code>itemize</code>	<code>center</code>	<code>tabular</code>

最基本的 Emacs 快捷键（大多以 `Ctrl` + `x` 开头）

文件操作

打开文件： `Ctrl` + `x` `Ctrl` + `f`

存盘： `Ctrl` + `x` `Ctrl` + `s`

关掉文件： `Ctrl` + `x` `k`

编辑操作

终止命令： <code>Ctrl</code> + <code>g</code>	设置标记： <code>Ctrl</code> + <code>~</code>
换行缩进： <code>Ctrl</code> + <code>j</code>	缩进： <code>Ctrl</code> + <code>i</code>
取消操作： <code>Ctrl</code> + <code>/</code>	删除字符： <code>Ctrl</code> + <code>d</code>
剪切： <code>Ctrl</code> + <code>k</code>	粘贴： <code>Ctrl</code> + <code>y</code>

移动光标

向前: <code>Ctrl</code> + <code>f</code>	向后: <code>Ctrl</code> + <code>b</code>
行首: <code>Ctrl</code> + <code>a</code>	行尾: <code>Ctrl</code> + <code>e</code>
下一行: <code>Ctrl</code> + <code>n</code>	上一行: <code>Ctrl</code> + <code>p</code>
下一页: <code>Ctrl</code> + <code>v</code>	上一页: <code>Alt</code> + <code>v</code>

窗口操作

横分: <code>Ctrl</code> + <code>x</code> <code>2</code>	纵分: <code>Ctrl</code> + <code>x</code> <code>3</code>
保留我: <code>Ctrl</code> + <code>x</code> <code>1</code>	关掉我: <code>Ctrl</code> + <code>x</code> <code>0</code>
切换: <code>Ctrl</code> + <code>x</code> <code>o</code>	

寻求帮助

教程: <code>Ctrl</code> + <code>h</code> <code>t</code>	info: <code>Ctrl</code> + <code>h</code> <code>i</code>
快捷键: <code>Ctrl</code> + <code>h</code> <code>k</code>	函数: <code>Ctrl</code> + <code>h</code> <code>f</code>
变量: <code>Ctrl</code> + <code>h</code> <code>v</code>	

最基本的 AUCTeX 快捷键 (大多以 `Ctrl` + `c` 开头)

环境: <code>Ctrl</code> + <code>c</code> <code>Ctrl</code> + <code>e</code>	章节: <code>Ctrl</code> + <code>c</code> <code>Ctrl</code> + <code>s</code>
命令: <code>Ctrl</code> + <code>c</code> <code>Ctrl</code> + <code>m</code>	换行: <code>Alt</code> + <code>↵</code>
编译: <code>Ctrl</code> + <code>c</code> <code>Ctrl</code> + <code>c</code>	查看: <code>Ctrl</code> + <code>c</code> <code>Ctrl</code> + <code>v</code>

有了这些入门基础，我们已经可以应付要求不甚严格的文章排版了。但如果想排版出高质量的毕业论文，hmm...，同志仍需努力。

在后续章节里，我们将简要介绍如下一些内容：

1. 如何插入图片

2. 如何写数学公式
3. 如何插入程序代码
4. 如何写中文
5. 如何使用毕业论文模版

3 入门以后

站在上一章的入门基础之上，本章我将介绍一些更为有趣的东西。这些貌似“高级”的技术其实也不复杂，无非是再多认识几个命令和环境罢了。

3.1 插入图片

```
1 \documentclass{article}
2
3 \usepackage{graphicx}
4 \graphicspath{{./figs/}{./}}
5
6 \begin{document}
7 \includegraphics[width=5cm]{tux}
8 \end{document}
```

怎么样，能看明白吗？插入图片用到了三个新命令：

1. `\usepackage{graphicx}`，这是在说「我要用到一个名字叫 `graphicx` 的 package(宏包)」。这很类似于我们 C 编程时常用的 `#include<stdio.h>`。`\includegraphics{}` 就是这个宏包提供的命令之一。想详细了解 `graphicx` 的话，你可以打开一个命令终端，敲命令：`texdoc graphicx`。`texdoc` 是 TeXLive 提供的专门用来看各种宏包手册的小工具。你可以通过 `texdoc -h` 命令来粗略了解它的用法。
2. `\graphicspath{{./figs/}{./}}`，显然这是在指明 `graphics`(图片) 所在的 `path`(路径，位置)，也就是说，当你编译的时候， \LaTeX 会到你指定的地方去找要插入的图片。在这里，我指定了两个地方：
 - (a) `./figs/`，当前目录下的 `figs` 目录。如果在这里没有找到，那么就去下面的目录里接着找；
 - (b) `./`，也就是当前目录。如果在这个目录里还是没找到想插入的图片，那么编译器就要报错了。
3. `\includegraphics[width=5cm]{tux}`，这显然就是在插入图片了。
 - (a) 图片的名字叫 `tux.pdf`，后缀 (`.pdf`) 可以被省略掉。显然 `tux.pdf` 应该被存放在 `./` 或者 `./figs/` 中，才能被找到。我喜欢 PDF 图片，因为它可以自由缩放。你当然可以插入 `jpeg`、`png` 图片。

(b) 宽度是 5cm，也可以是相对宽度，比如 `[width=.5\textwidth]` 就表示宽度等于 0.5 倍的行宽。

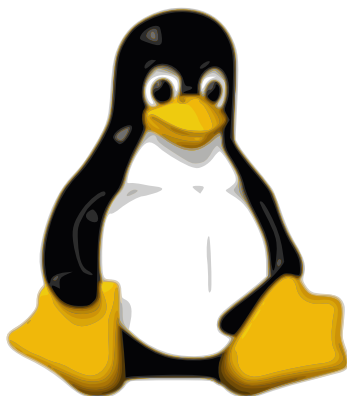
如果你希望图片“居中”摆放，那自然是要用到 `center` 了：

```

1 \documentclass{article}
2
3 \usepackage{graphicx}
4 \graphicspath{{./figs/}{./}}
5
6 \begin{document}
7 \begin{center}
8   \includegraphics[width=5cm]{tux}
9 \end{center}
10 \end{document}

```

编译后的效果大致就是这样，居中，5cm 宽。



3.1.1 Figure 环境

“哎，似乎应该加上图片说明吧？比如，【图 1: Linux 图标】？”这个容易，只要用到一个新的 environment，叫 `figure`。

`Ctrl` + `c` `Ctrl` + `e` `figure` `Ctrl` + `j`，mini buffer 提示：

(Optional) Float position:

这是在问你图片放在那里比较好啊？是靠上？还是靠下？还是懒得操心？如果没概念，那还是让 LaTeX 来决定吧，`Ctrl` + `j`，mini buffer 提示：

Caption:

这是在提示你输入图片的说明文字。那么输入：

Linux logo

mini buffer 提示:

Center? (y or n):

当然选:

y

mini buffer 提示:

Label: fig:

这是要你给图片打个标签，以后方便索引到它。那么就给个标签:

linux-logo

于是得到:

```

1  \documentclass{article}
2
3  \usepackage{graphicx}
4  \graphicspath{{./figs/}{./}}
5
6  \begin{document}
7  \begin{figure}
8    \centering
9
10   \caption{Linux logo}
11   \label{fig:linux-logo}
12 \end{figure}
13 \end{document}

```

现在你建立了一个完美的图片环境，别忘了把图片放进去。当然放在第 9 行:

```

1  \documentclass{article}
2
3  \usepackage{graphicx}
4  \graphicspath{{./figs/}{./}}
5
6  \begin{document}
7  \begin{figure}
8    \centering
9    \includegraphics[width=5cm]{tux}
10   \caption{Linux logo}
11   \label{fig:linux-logo}
12 \end{figure}
13 \end{document}

```

编译后的效果如图 3.1 所示。

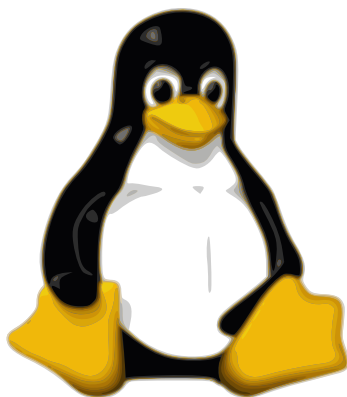


图 3.1 Linux logo

3.1.2 `\label{}`和`\ref{}`的用法

`\label{}`到底怎么用？来看下面的例子就明白了。

```
1  \documentclass{article}
2
3  \usepackage{graphicx}
4  \graphicspath{{./figs/}{./}}
5
6  \begin{document}
7  \begin{figure}
8    \centering
9    \includegraphics[width=5cm]{tux}
10   \caption{Linux logo}
11   \label{fig:linux-logo}
12 \end{figure}
13
14 Figure~\ref{fig:linux-logo} is the famous Linux Tux!
15 图\ref{fig:linux-logo}所示就是大名鼎鼎的 Linux 吉祥物!
16 \end{document}
```

编译一下，看看效果吧：

Figure 3.1 is the famous Linux Tux! 图 3.1 所示就是大名鼎鼎的 Linux 吉祥物!

`\label{}` 和 `\ref{}` 总是配合使用的，一个用来打标签，另一个用来找到它。而且这两个命令可以被用在任何你需要的地方，非常方便。比如，

```

1 % 本文的第一章标题
2 \chapter{工欲善其事，必先利其器}
3 \label{cha:pre-requisite} % 章标签
4
5 % 本文中的某张表格
6 \begin{table}[!htbp]
7   \centering
8   \caption{常用快捷键}\label{tab:keys} % 表格标签
9   \begin{tabu*}{to .5\textwidth}{X[r]|X[l]}
10    \hline
11    \rowfont[c]{\bfseries}快捷键&功用\\\hline
12    \Cj{}&换行带缩进\\
13    \Cc\Cm{}&输入 Macro\\
14    \Cc\Cs{}&新起一个章节\\
15    \Cc\Ce{}&新起一个环境\\
16    \MEnter{}&换行带{\verb|\item|}\\\hline
17  \end{tabu*}
18 \end{table}
19
20 % 在文中某处有这样一行
21 最基本的 Emacs 快捷键（大多以\Cx{}开头）\label{p:keys} % 任意标签
22
23 %% 下面让我们来使用（索引）上面的标签
24
25 \begin{enumerate}
26 \item 本文的第~\ref{cha:pre-requisite}章介绍了一个简单、% 索引某章
27   高效的工作环境；
28 \item 在表~\ref{tab:keys}中列出了几个最常用的快捷键； % 索引某表
29 \item 在第~\pageref{p:keys}页列出了更多的快捷键。 % 索引某页
30 \end{enumerate}

```

上述代码就可以生成如下的效果，不仅数字是正确的，而且它们都是 hyperlink，用鼠标点一下试试。

1. 本文的第 1 章介绍了一个简单、高效的工作环境；
2. 在表 2.1 中列出了几个最常用的快捷键；
3. 在第 26 页列出了更多的快捷键。

快捷地插入标签和索引

插入标签和索引也是有快捷键的。**Ctrl**+**C****I**就是要插入标签，mini buffer 提示：

Label: cha:

如果你不是要插入章标签，那么可以把 cha 改成其它你认为合适的字符。通常 Emacs 会根据光标所在的环境给出不同的提示，如果光标在

- Figure 里，它就提示 `fig:`;
- Table 里，它就提示 `tab:`;
- 正文中其它什么地方，它就提示 `cha:` 或者 `sec:`。

现在，在提示下输入一个简短而好记的标签名称，以便后面可以轻松找到它。

要插入索引 (`\ref{}`) 的话，敲 `Ctrl+C`，mini buffer 提示：

```
SELECT A REFERENCE FORMAT
```

```
[^M] \ref
```

```
[p] \pageref
```

这是让你选择索引格式，如果想索引某页的话，就选 `[p]`。其它任何情况，都选择 `[^M]`，也就是直接回车，`↵`。根据你的选择，Emacs 会弹出新的 buffer，方便你找到要引用的标签。

3.2 数学公式

举个简单的例子吧：

```
1 \documentclass{article}
2 \begin{document}
3 This is a simple math example:  $c^2=a^2+b^2$ 
4 \end{document}
```

结果是这样：

This is a simple math example: $c^2 = a^2 + b^2$

美元符号 (\$) 在 $\text{L}^{\text{T}}\text{E}^{\text{X}}$ 里面是特殊字符。夹在两个美元符号之间的东西，会被当做数学公式来排版。如果想让数学公式独占一行的话，就用双美元符号 (\$\$)，比如，

$$(1+x)^n = \sum_{k=0}^n \binom{n}{k} x^k$$

就是 `$$ (1+x)^n = \sum_{k=0}^n \binom{n}{k} x^k $$` 的输出结果。还不难看懂吧？

- `\sum` $\Rightarrow \sum$

代码：

```

\documentclass{article}
\usepackage{minted}
\begin{document}
\begin{minted}{c}
#include <stdio.h>
int main()
{
    printf("Hello, world!\n");
    return 0;
}
\end{minted}
\end{document}

```

效果：

```

1      #include <stdio.h>
2      int main()
3      {
4          printf ("Hello, world!\n");
5          return 0;
6      }

```

图 3.2 插入代码示例

- $\backslash\text{binom}\{n\}\{k\} \Rightarrow \binom{n}{k}$
- 下划线 ($_$) 后面跟下标，如果下标不止一个字符，那么就要用花括号 ($\{ \}$) 括起来。比如， $A_1 + A_100 \Rightarrow A_1 + A_{100}$ ；
- 上箭头 ($\^$) 后面跟上标，用法和下划线一样。比如， $2\^2 \backslash\text{times} 2\^32 \Rightarrow 2^2 \times 2^{32}$ 。

那么，去哪里找这些数学符号呢？很简单，Google 一下 “latex math”，就什么都有了。“天啊，谁能记住那么多数学符号啊？！”。 \LaTeX 的数学排版功能博大精深，各式各样的数学符号、怪异字符无所不及，当然用不着都记住。你只要记住上面我们提到的几条，应该就足以应付毕业论文了。如果你要经常对付复杂数学公式的话，那么最好把《The \LaTeX Companion》^[7]这本书的第八章 (Higher Mathematics) 打印下来放在手边，随用随查就好了。Google 一下 “latex math chapter 8”。

3.3 插入程序代码

还是从一个小例子开始吧，如图 3.2 所示，插入代码首先要 $\backslash\text{usepackage}\{\text{minted}\}$ 。 minted 是用于美化代码排版的 \LaTeX 宏包，有了它，你可以把几乎所有编程语言的代码排版得像你们老师那样道貌岸然，而且是彩色的。不过，对于毕业论文来说，还是采用黑白排版比较好。彩色排版如果黑白打印，效果就很模糊了。当然你可以选择彩色打印，但那很费钱啊。

想详细了解 minted 的用法, 就: `texdoc minted`。minted 宏包提供了一个新环境, 就叫 minted, 把你的程序放在 `\begin{minted}` 和 `\end{minted}` 之间就行了。`\begin{minted}` 后面的 `{c}` 当然是说, 插入的程序是用 C 语言写的。

怎么? 程序太长, 拷贝进来太麻烦? 那么可以这样:

```

1  \documentclass{article}
2  \usepackage{minted}
3  \begin{document}
4
5  \inputminted{c}{hello.c}
6
7  \end{document}


```

简单吧? `hello.c` 当然要和你的 $\text{T}_{\text{E}}\text{X}$ 文件在同一个目录下, 否则你要指明详细路径。

minted 宏包提供了丰富的命令, 可以支持数十种编程语言, 后台调用强大的 pygments 来打扮你的程序代码, 可以把程序以各种你能想到的方式排版出来。当然, 使用 minted 的前提条件是, 系统里已经安装好了 python 和 pygments。这一点在 minted 的手册里已经说得很清楚了。关于强大的 pygments, 你该去它的网站看看: <http://pygments.org/>。

3.4 处理中文

如果你的 Emacs 配置和我的一样, 那么输入中文就是 “piece of cake”, 当然你得安装了中文输入法, 我用的是 fcitx¹, 挺好用的。

现在, 打开一个全新的 `tex` 文件, 比如 `hello.tex`, 在里面写 `ctexart`, 然后敲  (TAB) 键, 一个现成的 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 文件模板立时会展现在你的面前。这个模版里有你写一篇漂亮中文文章所需要的一切。光标就停在 `\title{}` 的花括号里, 那么就开始用中文填空吧。Happy $\text{T}_{\text{E}}\text{X}$ ing!

附录 C 中列出了这一 `article` 模板的全部内容。重点就是第一句:

```

1 | \documentclass[scheme=chinese]{ctexart}

```

也就是说, 只要直接采用 C $\text{T}_{\text{E}}\text{X}$ 提供的 `ctexart` class, 中文问题就不需要我们操心了。

¹<https://fcitx-im.org/wiki/Fcitx>

4 完美的毕业论文

写毕业论文，像结婚一样，是一生一次的大事情。潦草而失败的论文，就像失败的婚姻，总要一次次地返工，在痛苦中煎熬，直到你遇见 ta，带你摆脱迷茫、痛苦，登上幸福的彼岸¹。

写普通文章要用 `\documentclass{article}`;

写报告要用 `\documentclass{report}`;

写书要用 `\documentclass{book}`;

写信要用 `\documentclass{letter}`;

那么写毕业论文自然要用毕业论文模版了：`\documentclass{swfuthesis}`。

如此关系人生的大事情，当然要为它专门建立一个目录吧。目录建好了，把论文模板 `swfuthesis.cls` 文件拷贝进去。然后就可以用它来写你的论文了。

4.1 Class 文件

Class 文件²，它决定了你的文章样式，比如说，纸张尺寸、页边距、行距、字距、字体、标题样式等等在 class 文件中都做了设置。除此之外，我们在写 `tex` 文件的过程中用到的命令（Macro）也都是 class 文件提供的。

这里有一个值得注意的概念，排版这件事情，是由排版软件根据你（在文章中提供）的命令来进行的。只要你的命令正确，文章的格式就必然是正确的。这就是排版软件和字处理软件（比如 MS-Word）的区别所在。利用字处理软件来写文章，你不得不既操心文章的内容，也操心文章的格式。而利用排版软件，比如 \LaTeX 来写文章，你只需要关心文章的内容，而格式的事情，排版软件会根据你的命令来替你完成。所以，你输入的命令必须正确、合法、合情理才行。

¹我说的当然是 \LaTeX 。

²也就是后缀为 `.cls` 的文件，也就是我们常说的模板文件。


排版软件只能理解 class 文件中提供的命令³，所以，我们当然要对这些命令有个基本的了解。附录 E 中就是 swfuthesis.cls 文件的全部内容。我们在此做个简要介绍。简而言之，swfuthesis.cls 提供了如下一些基本命令：



- | | |
|--------------------------|------------------------|
| 1. \title{论文标题} | 7. \Month{这里填月份} |
| 2. \author{作者} | 8. \Year{这里填年份} |
| 3. \enTitle{论文标题} | 9. \Subject{这里填专业} |
| 4. \enAuthor{作者} | 10. \maketitle: 用于生成封面 |
| 5. \Advisor{指导教师姓名} | 11. \makebib: 用于生成参考文献 |
| 6. \AdvisorTitle{指导教师职称} | |

和如下一些环境：

- | | |
|--------------------|------------------------|
| 1. abstract: 中文摘要 | 4. EKeyword: 英文关键词 |
| 2. keyword: 中文关键词 | 5. advisorInfo: 指导教师简介 |
| 3. EAbstract: 英文摘要 | 6. acknowledgment: 致谢 |

4.2 模板的使用

具体如何使用这些命令呢？简而言之，**Ctrl** + **x** **Ctrl** + **f**，输入一个崭新的文件名，**Ctrl** + **j**。现在，面对空无一字的 Emacs 窗口，你有生以来最重要的一篇文章就要开篇了，你在思考什么？其实什么都不用想，直接敲 thesis  (Tab)，一个论文框架就呈现在你的眼前了，你要做的就是用你前面学到的那些 L^AT_EX 命令来填空。

在本文开始的时候（第 3 页），我们提到过“ 大法”是 yasnippet 的贡献。所以，你在期待  键能带来奇迹之前，必须先确保你的系统里已经装好了 yasnippet。如果还没装上，那么就执行下面的命令^[8]：

```
1 | sudo apt update && sudo apt upgrade
2 | sudo apt install yasnippet yasnippet-snippets
```

装好之后，重新加载 Emacs 的配置文件。很简单，在 Emacs 窗口里，敲

³我说谎了，实际上，在 tex 文件中，你可以利用 \newcommand{} 和 \renewcommand{} 来定义自己的命令。但对于初学者来说，你暂时还不必操心这个，class 文件所提供的命令应该足以应付你目前的需求了。

Alt+**x** load-file **Ctrl**+**j** ~/.emacs **Ctrl**+**j**

跟着本教程写论文的前提条件就是，Emacs 必须工作正常，如果发现问题必须及时处理，不能将就。尤其是在入门阶段，不要指望用一个别扭的 Emacs 来顺畅地写出完美的论文。

附录 D 中列出了完整的毕业论文框架，也就是你在键入 `thesis` 后应该看到的东西。如果奇迹没有发生，那么就要操心一下你的 Emacs 配置了。

假设你的 Emacs 环境良好，按 键之后，本来空荡荡的 Emacs buffer 里现在有了一个论文框架，而且光标就停在你要填空的第一个位置，也就是 `\title{论文标题}` 的花括号里。这时，你只要键入任何文字（通常应该是你自己的论文标题），（感谢 yasnippet）花括号里的“论文标题”四字就会被自动替换掉。写好了论文标题，还是按 键，光标会自动跳到下一个你需要填空的位置。如此一直 下去，直到光标不再跳开了，那就是你该写论文第一章内容的时候了。

论文框架里有足够多的注释，再加上你在前几章学到的本领，我相信写出一个规范、漂亮的毕业论文应该是手到擒来的事情了。

4.3 生成参考文献

在第 2.2.6 节中我们已经介绍过如何处理参考文献。大致步骤：

1. 准备好一个 .bib 文件（假设名字叫 myref.bib）；
2. 在 .tex 文件中做三件事：
 - (a) 在 preamble 中加一句 `\addbibresource{myref.bib}`；
 - (b) 在正文中适当的地方利用 `\cite{}` 命令引用 myref.bib 文件中的参考文献条目；
 - (c) 在文章的结尾部分加一句 `\printbibliography{}`；
3. 用 latexmk 来编译 .tex 文件。

附录 F 中列出了本文所用到的 .bib 文件（tutorial.bib）。你可以对照着第 43 页的参考文献部分看看，找找感觉。

这个文件是怎么来的？ 当然可以手写。bib 文件的格式清晰易懂，照猫画虎地手写并不困难。但生活可以更轻松，如果你引用的书籍、资料不是太冷门，那么通常只要 google 一下「书名 bibtex」，就可以找到相应的 bib 条目了。有时你甚至可以从网上下载到现成的 bib 文件，比如全套 RFC⁴ 的 bib 文件。

怎样使用这个文件？ 很简单，三件事：

1. 第一件事不用做，因为在我们的论文模板文件 (swfuthesis.cls) 里已经做了。在模板文件里有如下一行：

```
\RequirePackage[backend=biber,style=gb7714-2015]{biblatex}
```

也就是说，帮我们排版参考文献的是 biblatex 宏包；

2. 在 tex 文件（你的论文）的 preamble 部分，也就是在 \begin{document} 之前，加上如下一行：

```
\addbibresource{tutorial.bib}
```

很显然，这是在告诉 biblatex 从当前目录下的 tutorial.bib 文件里去读取参考文献信息。

3. 这件事也不用做了，因为在论文的框架文件 (tex 文件) 里已经做好了。在 tex 文件的 appendix（附录）部分，你能看到如下一行：

```
\makebib
```

这就是要求排版输出参考文献。

4.4 小结

在此我们简单回顾一下，要写出漂亮的毕业论文所应具备的必要条件：

1. 一套高效的工作环境

(a) Debian sid;

(b) T_EXLive;

(c) Emacs+AUCT_EX+Yasnippet;

⁴Request for Comments. https://en.wikipedia.org/wiki/Request_for_Comments

2. 对 Emacs 基础快捷键的熟练使用；
3. L^AT_EX 的基础知识；

只要具备了以上条件，轻松地写出一份漂亮、规范的毕业论文应该不成问题了。先到这里吧。以后如果有时间，我会继续介绍一下

1. 如何用 L^AT_EX Beamer 做出漂亮的演示幻灯片；
2. 如何用 Emacs org-mode 快速生成 PDF 文件。

另外，配合本教程，我制作了一个简单的小视频，`tutorial.mkv`，它应该和本文放在同一个目录下⁵。

关于本文的任何疑问和建议，欢迎反馈到 `wx672ster@gmail.com`。

Happy T_EXing!

⁵<http://cs6.swfu.edu.cn/~wx672/swfcthesi/tutorial/>

参考文献

- [1] 维基百科. LaTeX — 维基百科, 自由的百科全书. 2016. <https://zh.wikipedia.org/w/index.php?title=LaTeX&oldid=39426205>.
- [2] Wikipedia. LaTeX — Wikipedia, The Free Encyclopedia. 2016. <https://en.wikipedia.org/w/index.php?title=LaTeX&oldid=720051950>.
- [3] Et AL. R S. GNU Emacs Manual. 17th ed. The Free Software Foundation, 2015.
- [4] THORUP K K, ABRAHAMSEN P, KASTRUP D, et al. AUCT_EX— A sophisticated T_EX environment for Emacs. 11.88. The Free Software Foundation, 2014.
- [5] Pluskid, TÁVORA J, POSTAVSKY N. Yet another snippet extension. 2016. <http://capitao.morte.github.io/yasnippet/>.
- [6] LEHMAN P, KIME P, BORUVKA A, et al. The biblatex Package. 2016.
- [7] GOOSSENS M, MITTELBACH F, SAMARIN A. The LaTeX Companion. 2nd revised. Addison-Wesley, 1994.
- [8] BURROWS D. Aptitude user's manual. 2016.

指导教师简介

\$ dict google

From The Free On-line Dictionary of Computing (05 May 2016) [foldoc]:

Google

<web> The {web} {search engine} that indexes the greatest number of web pages - over two billion by December 2001 and provides a free service that searches this index in less than a second.

The site's name is apparently derived from "{googol}", but note the difference in spelling.

The "Google" spelling is also used in "The Hitchhikers Guide to the Galaxy" by Douglas Adams, in which one of Deep Thought's designers asks, "And are you not," said Fook, leaning anxiously foward, "a greater analyst than the Googleplex Star Thinker in the Seventh Galaxy of Light and Ingenuity which can calculate the trajectory of every single dust particle throughout a five-week Dangrabad Beta sand blizzard?"

{{(http://google.com/)}}.

(2001-12-28)

致 谢

感谢美国计算机教授高德纳 (Donald Ervin Knuth) 编写的功能强大的排版软件 $\text{T}_{\text{E}}\text{X}$ 。感谢美国计算机科学家莱斯利·兰波特 (Leslie Lamport) 教授为 $\text{T}_{\text{E}}\text{X}$ 开发的简单易用的 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 宏包。感谢本文作者能有如此多的闲功夫和好心情来饶有兴致地做这样一件不能当饭吃的事情。

A 西南林业大学本科生毕业论文（设计）撰写规范及要求（2019）

毕业论文（设计）是学生本科学习阶段最后一个重要学习环节，是学习深化与升华的重要过程。它既是学生学习、研究与实践成果的全面总结，又是对学生素质与能力的一次全面检验，而且还是对学生的毕业资格及学位资格认证的重要依据。为全面规范本科生毕业论文（设计）写作、装订、归档等各环节工作，特制定本规范。

A.1 文档材料的写作规范

论文（设计说明书）应包括封面、原创性声明、目录、中英文摘要与关键词、正文、参考文献、指导教师简介、致谢、附录等部分。

A.1.1 封面

采用学校规定的统一格式（图 A.1），按要求填写论文中文题目、学生姓名、学号、专业、指导教师、评阅人、完成毕业论文（设计）的时间等内容¹。论文题目要简明、具体、确切，一般不超过 20 个汉字（英文题目不超过 15 个实词），必要时可加副标题。题目中应避免使用非公知公用的缩略语、字符、代号以及结构式和公式，中间不使用标点。

A.1.2 原创性声明

原创性声明须附于学位论文摘要之前，需学生本人签字。

A.1.3 中文摘要（含关键词）

中文摘要应简洁明了，以精炼的文字对毕业论文（设计）的内容、观点、方法、成果和结论进行高度概括，具有独立性和自含性，自成一篇短文，具有报导作用。摘要中对作者所做工作的评价应客观，不要使用带有渲染、夸张作用的词藻，字数为 300 字左右。一般不分段、不用图表。

内容包含本项毕业论文（设计）工作的目的、意义、研究方法、研究过程、研究成果及结论、关键词等。突出毕业论文（设计）工作中具有创造性成果和新见解的部分。

关键词是供检索用的主题词条，应采用能覆盖论文主要内容的通用技术词条，一般是 3~8 个，关键词之间以分号隔开。

A.1.4 外文摘要（含关键词）

外文摘要、关键词内容应与中文摘要对应，紧接其后。要求用词准确、语法规则、意思完整。

A.1.5 目录

目录独立成页，包括论文中全部章、节的标题，一般列到 3 级标题，文字表述与正文一致，并标明页码。

¹学校未对封面上的图片大小、字号、字距、行距做任何规范。也没有提供一个 PDF 格式的封面样本，只提供了一个 .doc 格式的模板，用不同的文字处理器打开，可以看到各种各样的效果。

A4 纸宽	
3cm	2cm
页眉（宋体小 5 号字，居中）	
2cm	
<div><p>西南林业大学 SOUTHWEST FORESTRY UNIVERSITY</p><p>本科毕业论文（设计） （二〇二〇 届）</p><p>小 2 号黑体字，居中</p><p>题 目 怎样用$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$排版毕业论文</p><p>学 院 大数据与智能工程学院 专 业 计算机科学与技术</p><p>学生姓名 王晓林 学 号 20161152888</p><p>指导教师 Google (G), Stackoverflow (🐼)</p><p>评 阅 人 评阅人姓名（职称）</p><p>对于校标、校徽，以及封面上除论文题目之外的所有其它文字，学校规范中未给出任何数据，亦未给出 PDF 格式的模板，只提供了一个.doc 文件，用不同的工具打开，会产生不同的效果。跟着感觉走吧。</p><p>文本宽度</p><p>2020 年 5 月 20 日</p><p>页脚（宋体小 5 号字，居中）</p></div>	
A4 纸高	文本高度
3cm	2cm
2.5cm	3cm

图 A.1 封面标准

A.1.6 论文正文

毕业论文正文总字数不少于 8000 字。设计创作类作品不做字数及格式限制，原则上应包括设计说明、设计图纸、展板或模型等。毕业论文的基本结构应由题目、作者、原创声明、中文摘要、外文摘要、关键词、目录、前言、研究方法、结果与分析、主要结论、参考文献、指导教师简介、致谢、附录等基本部分组成（有设计的须有设计图纸）。

正文一般由标题、主体部分、表格、图和公式五个部分构成。写作内容可因课题的性质不同而变化。一般可包括（可结合论文特点进行调整）：

1. 前言（或取名引言、序等）。说明本论文（设计）课题的来源、目的、意义、应解决的主要问题及应达到的技术要求；简述本课题在国内外发展概况及存在的问题，属设计的还应说明设计的指导思想。
2. 方案（方法）论证。说明设计原理（方法思路）并进行方案（方法）选择，阐明为什么要选择这个设计方案（方法）（包括对其进行分析、比较）以及所采用方案（方法）的特点。
3. 过程（设计或实验）论述。指作者对自己的研究工作的详细表述。要求论理正确、论据确凿、逻辑性强、层次分明、表达确切。
4. 结果分析。对研究过程中所获得的主要数据、现象进行定性或定量分析、得出结论或推论。
5. 结论或总结。对整个研究工作进行归纳和综合，阐述本课题研究存在的问题、技术障碍及进一步开展后续研究工作的思路、见解和建议。

毕业论文的正文章节可选用以下三种形式：

1. 第一种²（供理工类专业使用）：一级标题：1, 2, 3...；二级标题：1.1, 1.2, 1.3, ...；三级标题：1.1.1, 1.1.2, 1.1.3, ...。
2. 第二种（供文科类专业使用）：一, 二, 三,...；（一）（二）（三）...；1. 2. 3. ...；（1）（2）（3）...。
3. 第三种（供毕业设计使用）：毕业设计的正文内容较多，应按结构顺序编写，建议论文目录和内容书写按章、节、小节编排，具体格式可参照如下：

```

第一章 × × ×
  1.1 × × ×
    1.1.1 × × ×
    1.1.2 × × ×
      (1) × × ×
      (2) × × ×
  1.2 × × ×
    1.2.1 × × ×
    1.2.2 × × ×
    
```

²大数据与智能工程学院采用此格式。

A.1.7 设计图纸

图纸要求图面整洁,布局合理,线条粗细均匀,圆弧连接光滑,尺寸标注规范,文字注释必须使用工程字书写。原则上要求学生使用计算机绘图;对于曲线图表,要求所有曲线、图表、线路图、流程图、程序框图、示意图必须按国家规定标准或工程要求采用计算机或手工绘制等,不准徒手画;图号按章序编号,如:图 3-2 为第三章第二幅图。如果图中含有几个不同部分,应将分图号标注清楚,如:图 3-2-1、3-2-2 等,并在图题下列出各部分内容。

A.1.8 注释和参考文献(注释采用尾注)

毕业论文(设计)须在论文的最后列出参考文献,注释多用于文科类论文。

正文中按引用顺序在注释和参考文献出处的文字右上角用[]标明,[]中序号应与注释和参考文献中序号一致,正文之后则应列出注释和参考文献。注释和参考文献中应包含一定的外文文献,参考文献一般不少于 10 篇。

注释和参考文献的著录,按著录/题名/出版事项顺序排列,常用参考文献编写规定如下:

- 著作图书类文献——[序号] 著者. 书名[M]. 版次. 出版地: 出版者, 出版年. 起止页码
- 翻译图书类文献——[序号] 著者. 书名[M]. 译者. 版次. 出版地: 出版者, 出版年. 起止页码
- 学术期刊类文献——[序号] 作者. 题名[J]. 刊物名. 出版年, 卷号(期号), 起止页码
- 学术会议类文献——[序号] 作者. 题名. 编者名. 会议名称, 会议地址, 年份, 出版地: 出版者, 出版年: 起止页码
- 学位论文类文献——[序号] 作者. 学位论文题目. 学校及学位论文级别. 答辩年份: 起止页码
- 报纸文献——[序号] 作者. 文章名[N]. 报纸名, 出版日期(版次)
- 在线文献——[序号] 作者. 文章名. 电子文献的出处或可获得地址, 发表或更新日期/引用日期

A.1.9 指导教师简介

介绍指导教师的基本情况,包括基本信息,从事的研究方向,取得的研究成果等。

A.1.10 致谢

致谢应以简短的文字对在课题研究和论文撰写过程中曾直接给予帮助的人员(如指导教师)表示自己的谢意,这不仅是一种礼貌,也是对他人劳动的尊重,是治学者应有的思想作风。要求内容要实在,语言要诚恳。



图 A.2 单幅插图示例

A.1.11 附录

主要列入正文过分冗长的公式推导；研究方法和技术更深入的叙述；以备查读方便所需的辅助性工具或表格；重复性图表；使用的主要符号的意义、单位、缩写、程序全文及说明等。附录可采用“附录 1”、“附录 2”或“附录一”、“附录二”等序号格式。

A.1.12 其它要求

1. 使用普通语体文写作，要文句通顺，体例统一，无语法错误，简化字要符合规范，正确使用标点符号，符号的上下角标和数码要写清楚且位置准确。
2. 采用中华人民共和国国家标准（GB3100~3102-93）规定的计量单位和符号，单位用正体，符号用斜体。
3. 使用外文缩写代替某一术语时，首次出现的，应用括号注明其含义，如 CPU (Central Processing Unit, 中央处理器)。
4. 国内工厂、机关、单位的名称等应使用全名，如不得把“西南林业大学”简写成“西南林大”。
5. 图、表、公式等一律用阿拉伯数字分章连续编号，如图 1.3、表 2.1、(3.2) 等。图、表、公式等与正文之间间隔 0.5 行。

图标题采用相应的黑体小四号和小四号 Times News Roman 字体，均居中，位于图下方。

表标题采用相应的黑体小四号和小四号 Times News Roman 字体，均居中，位于图上方。

插图，如图 A.2 所示，大小一般为宽 6.67cm × 高 5.00cm。特殊情况下，也可宽 9.00cm × 高 6.75cm，或宽 13.5cm × 高 9.00cm。总之，一篇论文中，同类图片的大小应该一致，编排美观、整齐。

一幅图如有若干幅分图，均应编分图号，用 (a), (b), (c), ... 按顺序编排，且各分图的分题注直接列在各自分图的正下方，总题注列在所有分图的下方正中，如图 A.3 所示。

如表 A.1 所示，表格中各物理量及量纲均按国际标准（SI）及国家规定的法定符号和法定计量单位标注。一律使用三线表，与文字齐宽，顶线和底线线粗 1.5 磅，中线线粗 1 磅。表格必须通栏，即表格宽度与正文版面平齐。



图 A.3 插图示例

表 A.1 文献类型和标志代码

文献类型	标志代码	文献类型	标志代码
普通图书	M	会议录	C
汇编	G	报纸	N
期刊	J	学位论文	D
报告	R	标准	S
专利	P	数据库	DB
计算机程序	CP	电子公告	EB

在三线表中可以加辅助线，以适应较复杂表格的需要，如表 A.2 所示。

公式：公式中各物理量及量纲均按国际标准（SI）及国家规定的法定符号和法定计量单位标注，禁止使用已废弃的符号和计量单位。公式后应注明编号，公式号应置于小括号中，如（2-3）。写在右边行末，中间不加虚线。例：

$$y = ax^3 + bx + \frac{c}{x} + d \tag{A-1}$$

A.2 毕业论文（设计）文本打印格式要求

- 1. 毕业论文（设计说明书）成文文稿一律采用计算机打印，采用 A4 规格复印纸单面输出，左边留一定装订边距，用学校统一封面格式装订。打印时，页面设置为：
 - (a) 上下页边距均为 3cm，左页边距为 2.5cm，右页边距为 2cm，装订边 0.5cm；
 - (b) 页眉与页脚的边距均为 2cm，奇数页页眉内容为“论文中文题目”，偶数页页

表 A.2 方弯管内流动最大速度比较

项目	层流		紊流	
	0° 截面	90° 截面	0° 截面	90° 截面
理论值 $V_{max}/m \cdot s^{-1}$	0.04	0.03	1.30	1.25
计算值 $V_{max}/m \cdot s^{-1}$	0.04	0.03	1.26	1.21
误差/%	0.00	3.12	3.07	3.20

- 眉内容为“第 X 章 XXX”，如“1 前言”，采用宋体小 5 号字居中排写；
- (c) 正文文字间距为标准，行间距为 1.5 倍行距，段前段后不空行；
 - (d) 页码一律位于页面底端（页脚），居中标明。

2. 毕业论文（设计）文本字体和格式要求

- (a) 中文题目使用小 2 号黑体字、居中，署名使用 4 号宋体字³；
- (b) 中文摘要、关键词的内容为 5 号宋体字，“摘要、关键词”5 个字用小 4 号黑体字；
- (c) 英文题目使用 3 号 Times New Roman 体、居中；
- (d) 英文摘要、关键词内容使用小 4 号 Times New Roman 体字；
- (e) 正文统一使用小 4 号宋体和 Times New Roman 字体；
- (f) 图、表内容使用 5 号宋体和 Times New Roman 字体⁴；公式字体、字号与正文相同；
- (g) 文本一级标题使用小 3 号黑体字，二级标题使用 4 号黑体字，三级标题使用小 4 号黑体字；
- (h) 注释、参考文献使用 5 号宋体字，“注释、参考文献”六个字用 4 号黑体⁵。

A.3 毕业论文（设计）文本装订规范

1. 毕业论文（设计）文本按如下次序装订成册。

- (a) 封面；
- (b) 原创性声明；
- (c) 中文摘要及关键词；
- (d) 英文摘要及关键词；
- (e) 目录；
- (f) 正文；
- (g) 参考文献；
- (h) 指导教师简介；
- (i) 致谢；
- (j) 附录（必要时可加此部分，如程序清单等）；
- (k) 封底。

³显然这是在说摘要页题目和署名。封面上的题目该用几号字，没有规范。

⁴图片中的字号，本来就是有大有小，怎么可能统一用 5 号？图片字号比正文略小就行了。

⁵第 A.1.8 节中说，注释要用尾注方式。因为是“尾注”，也就是在每章末尾列出注释，所以“注释”二字排版与二级标题一致，采用 4 号黑体字，这是没问题的。但第 A.1.8 节中也说到，参考文献要列于论文的最后，也就是说，不是以尾注的形式出现，而是以独立一章的形式出现，故“参考文献”四字显然应以章标题形式出现，用小 3 号黑体字。

2. 附件另行装订。毕业论文（设计）材料较多，且不宜收入正文中的有关材料，如译文及原文、专题调研报告、过长的公式推演过程、非软件设计题目中篇幅较大的计算机程序等，可按如下次序装订成册：
 - (a) 封面；
 - (b) 目录；
 - (c) 调研报告、文献综述；
 - (d) 外文翻译及原文（译文在前，原文在后）；
 - (e) 公式推演过程、计算机程序等；
 - (f) 封底。
3. 某些特殊专业毕业论文（设计）文本、图纸等较多时，应按要求整理完毕后装入专用资料袋，其封面要用仿宋字认真填写，做到资料齐全、工整美观；
4. 毕业论文（设计）任务书、毕业论文（设计）实习计划表、毕业论文（设计）开题报告、毕业论文（设计）中期检查表、毕业论文（设计）答辩记录表、毕业论文（设计）指导教师意见、毕业论文（设计）评阅意见、查重检测报告、毕业论文（设计）、答辩评分表及其它归档材料一并装入“西南林业大学大学毕业论文（设计）档案袋”，由学生所在学院归档保管。

A.4 其他要求

专科学学生毕业论文（设计）撰写规范参照本科执行。相关表格可到教务处网页下载。

B 与排版论文相关的软件清单

1. emacs25 - GNU Emacs editor (with GTK+ GUI support)
2. emacs25-common-non-dfsg - GNU Emacs common non-DFSG items, including the core documentation
3. fcitx - Flexible Input Method Framework
4. fcitx-pinyin - Flexible Input Method Framework - classic Pinyin engine
5. fonts-noto - metapackage to pull in all Noto fonts
6. fonts-noto-cjk-extra - "No Tofu" font families with large Unicode coverage (CJK all weight)
7. fonts-arphic-ukai - "AR PL UKai" Chinese Unicode TrueType font collection Kaiti style
8. fonts-symbola - symbolic font providing emoji characters from Unicode 9.0
9. fonts-tlwg-purisa - Thai Purisa font (dependency package)
10. latexmk - Perl script for running LaTeX the correct number of times
11. sawfish - window manager for X11
12. sawfish-merlin-ugliness - More flexible functions for sawfish
13. texlive-bibtex-extra - TeX Live: BibTeX additional styles
14. texlive-extra-utils - TeX Live: TeX auxiliary programs
15. texlive-generic-extra - TeX Live: transitional dummy package
16. texlive-generic-recommended - TeX Live: transitional dummy package
17. texlive-lang-chinese - TeX Live: Chinese
18. texlive-lang-english - TeX Live: US and UK English
19. texlive-latex-recommended - TeX Live: LaTeX recommended packages
20. texlive-luatex - TeX Live: LuaTeX packages
21. texlive-xetex - TeX Live: XeTeX and packages
22. python-pygments - syntax highlighting package written in Python
23. xterm - X terminal emulator

⁰这份清单是用如下命令获得的：

```
aptitude search '~i "texlive|fonts|latexmk|pygments|emacs|fcitx|xterm|sawfish"' > pkg-list
```

注意，这是一个不完全的软件包列表。但是，在安装这些软件包时，其它一些被依赖的软件包会被自动安装上。另外，Emacs 的很多插件，比如 AUCT_EX, pdf-tools 等，可以通过 Emacs 的插件管理模块来安装、更新，在此没有列出。

C 中文 article 模板

```
1 | sudo apt update && sudo apt upgrade
2 | sudo apt install yasnippet yasnippet-snippets
```


D 毕业论文模板

```
1 | sudo apt update && sudo apt upgrade
2 | sudo apt install yasnippet yasnippet-snippets
```


E swfuthesis.cls 文件

```
1 | sudo apt update && sudo apt upgrade
2 | sudo apt install yasnippet yasnippet-snippets
```


F tutorial.bib 文件

```
1 | sudo apt update && sudo apt upgrade  
2 | sudo apt install yasnippet yasnippet-snippets
```