

# Linux 操作系统及应用

## 附录 G—用 L<sup>A</sup>T<sub>E</sub>X 撰写文档

王旭、唐晓晟、李亦农

`gnawux@gmail.com` `txs@bupt.edu.cn` `hoplee@bupt.edu.cn`  
SCHOOL OF INFORMATION COMMUNICATION ENGINEERING, BUPT



# 大纲

## ① L<sup>A</sup>T<sub>E</sub>X 导引



# 大纲

- ① L<sup>A</sup>T<sub>E</sub>X 导引
- ② L<sup>A</sup>T<sub>E</sub>X 的基本使用

# 大纲

- ①  $\text{\LaTeX}$  导引
- ②  $\text{\LaTeX}$  的基本使用
- ③ 书写中文



# 大纲

- ① L<sup>A</sup>T<sub>E</sub>X 导引
- ② L<sup>A</sup>T<sub>E</sub>X 的基本使用
- ③ 书写中文
- ④ 插入公式、图形、表格以及程序代码



# 大纲

- ① L<sup>A</sup>T<sub>E</sub>X 导引
- ② L<sup>A</sup>T<sub>E</sub>X 的基本使用
- ③ 书写中文
- ④ 插入公式、图形、表格以及程序代码
- ⑤ 参考文献



# 大纲

- ① L<sup>A</sup>T<sub>E</sub>X 导引
- ② L<sup>A</sup>T<sub>E</sub>X 的基本使用
- ③ 书写中文
- ④ 插入公式、图形、表格以及程序代码
- ⑤ 参考文献
- ⑥ 更多技巧与常用宏包



# 大纲

- ① L<sup>A</sup>T<sub>E</sub>X 导引
- ② L<sup>A</sup>T<sub>E</sub>X 的基本使用
- ③ 书写中文
- ④ 插入公式、图形、表格以及程序代码
- ⑤ 参考文献
- ⑥ 更多技巧与常用宏包
- ⑦ 参考文献与网上资料





# 大纲

## ① LaTeX 导引

LaTeX 是什么

LaTeX 的特点

安装 LaTeX

LaTeX 是如何工作的

## ② LaTeX 的基本使用

## ③ 书写中文

## ④ 插入公式、图形、表格以及程序代码

## ⑤ 参考文献

## ⑥ 更多技巧与常用宏包

## ⑦ 参考文献与网上资料



# T<sub>E</sub>X 是什么

- 字处理软件或说是一个排版软件，就像 MS Word
- 可编程、可扩展的工具，富于精确控制能力
- 强大的排版语言，多达 900 多条指令，用户还可以扩充命令
- 擅长逻辑结构严密的文章、特别是科学论文的排版工具
- 真正的跨平台工具，不论你在什么系统上都能使用，
- 实现了输出文档在任何平台上都同样的显示效果
- 不是所见即所得的工具，需编译才能生成最终文档
- 右边这张狮子的图案，是 T<sub>E</sub>X 的代表性 logo



Figure: T<sub>E</sub>X



# T<sub>E</sub>X 如何发音

## T<sub>E</sub>X

T<sub>E</sub>X 由大写的希腊字母“tec”( $\tau, \epsilon, \chi$ ) 组成, 在希腊语中这个词的意思是“科技”和“艺术”, 这也解释了它的发音。“T”和“E”就像在“technology”中的发音一样, 而“x”的发音类似于苏格兰语单词“loch”或德语单词“ach”中的“ch”, 也类似于西班牙语中的“j”或俄语中的“kh”。

T<sub>E</sub>X 系统由 Pascal (跨平台需要) 语言编写, 第一版于 1978 年面世, 1982 年的版本十分稳定, 其后没有大的变化, 当前版本号为 3.141592 /tek<sub>h</sub>/, 关键点: X 是希腊字母 ( $\chi$ ), 绝对不能读成 /ks/, 如果觉得不好发音, 可以读成 /k/。

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

/'lei'tek<sub>h</sub>/或/'la:'tek<sub>h</sub>/。



# T<sub>E</sub>X 之父



Figure: T<sub>E</sub>X 的缔造者: Donald E. Knuth(高德纳)

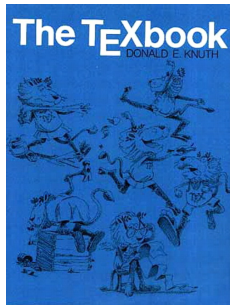


Figure: 著名的 T<sub>E</sub>XBook

# 各种 T<sub>E</sub>X 系统 I

为了便于使用，人们对 T<sub>E</sub>X 进行了二次开发，其中比较有名的有美国数学学会推荐的非常适合数学家使用的 AMS-T<sub>E</sub>X 以及适合一般文章、报告、书籍的 L<sup>A</sup>T<sub>E</sub>X 系统，此外，还有：

- **PDF<sub>T</sub>E<sub>X</sub>**：完全兼容标准的 T<sub>E</sub>X
- **e-T<sub>E</sub>X**：不仅完全兼容标准的 T<sub>E</sub>X，还支持一种“扩展模式”。在“扩展模式”中引入了许多新的特性和增强功能。这些新的特性和增强功能提供了更好的编程工具和对输出的控制。
- **Omega**：这是一个几乎是完全重新写过的，支持 Unicode 的 T<sub>E</sub>X 程序。Omega 有很多特征都超出了 Knuth 对 T<sub>E</sub>X 的原始设想。它采用了与 T<sub>E</sub>X 不同的方法来处理字符，使得可以处理非拉丁语系的文本。也支持从右到左，从上到下的排版方式。



# 各种 T<sub>E</sub>X 系统 II

- **NTS**: 表示 “New Typesetting System”。Knuth 已经停止发展 T<sub>E</sub>X，并决定任何超越 T<sub>E</sub>X 的扩展都不能被称为 T<sub>E</sub>X。NTS 现在还只是一些对于 T<sub>E</sub>X 的后续继承者的设想和概念的集合，还不是一个实际上的系统。自然，NTS 会继承 T<sub>E</sub>X 的绝大部分让我们热爱的特征，并且在很多方面对其加以发展。NTS 可能最终会取代 T<sub>E</sub>X 或 e-T<sub>E</sub>X。
- **MLT<sub>E</sub>X**: 这个 T<sub>E</sub>X 的扩展引入了命令 `\charsubdef`，使得可以更为简单方便的使用 8 bit 字符。不过，与 e-T<sub>E</sub>X 相比，这只是一个微小的改进。
- **ConT<sub>E</sub>Xt**: 是比 L<sup>A</sup>T<sub>E</sub>X 更高级的 T<sub>E</sub>X 宏包。相对 L<sup>A</sup>T<sub>E</sub>X 要灵活的多，几乎所有的环境都可以在一个统一的界面下配置。而且 ConT<sub>E</sub>Xt 没有文档类，所以无论是幻灯片还是打印文档都可以从相同的源文件产生。



# 各种 TeX 系统 III

## 小结

TeX 现在已经被 Knuth“冰封”(frozen)，新版本的 TeX 只有少许改进和错误修正。这些变动必须保证不会使得即使是写于 1982 年的 TeX 文档无法编译，TeX 自带了测试文件来确保这一点。如果测试文件文件的输出结果不同于预定的结果，那么这个排版系统就不能够被称为“TeX”。所以像一些对 TeX 有很大改进的软件，如“Omega”等将不能被称为“TeX”，尽管它也是基于“TeX”的。

接下来我们的话题只围绕 LaTeX 进行，对于我们来说，最常用的 TeX 发行版有 TeXlive 和 CTEx



# 为什么使用 (不用)L<sup>A</sup>T<sub>E</sub>X

- 使用简便、样式和内容分开，“所想即所得”
- 印刷级的优质排版，特别是英文
- 源文件的编辑更加高效
- 漂亮而方便的数学公式编辑
- 便捷的交叉索引和参考文献编排
- 众多的用户和大量成熟好用的模板
- 完美的 UNIX 哲学





# T<sub>E</sub>X 套装

一套完整的 T<sub>E</sub>X 系统包括:

- T<sub>E</sub>X 执行程序
- 多种样式, 对于我们来说, 需要的是 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>, 实际上还有 plain T<sub>E</sub>X, ConT<sub>E</sub>Xt...
- 大量宏包, 比如, 要包含图形就要用 graphicx 宏包。
- 需要的字体 — 要排出漂亮的文档, 当然要有漂亮的字体
- 辅助工具:
  - METAFONT & MetaPost 是很多精美绘图的缔造者
  - 有些套装中提供 **tex** 文件编辑器, 比方说 C<sub>T</sub><sub>E</sub>X, 不过, **vim** 才是最好的编辑器



# T<sub>E</sub>X 套装

## 常见的 T<sub>E</sub>X 套装

- C<sub>T</sub>E<sub>X</sub>, 中文 Windows 用户最常用的套装。  
C<sub>T</sub>E<sub>X</sub> 网站提供的一套中文 T<sub>E</sub>X 套装, 对 GBK 中文支持的更好, 可以利用 Windows 系统中的标准中文字体, 随软件包附带了 WinEdt 编辑器构成一个集成开发环境。
- T<sub>E</sub>XLive, 最完整的多平台套装。  
CTAN提供的全平台的 T<sub>E</sub>X 套装, 实际包含了几个服务于不同平台的核心执行程序以及大量的宏包, 每年发布更新, 已经成为当前 Linux & Windows 下安装 T<sub>E</sub>X 的主流。
- t<sub>e</sub>T<sub>E</sub>X, Linux 用户最常用的套装。  
Thomas Esser 维护的 UNIX 类操作系统使用的 T<sub>E</sub>X 套装, 几乎每个 Linux 发布版都会包含它, 内容非常丰富, 但目前已经停止维护。
- f<sub>p</sub>T<sub>E</sub>X, MikT<sub>E</sub>X...

# L<sup>A</sup>T<sub>E</sub>X 工作机制

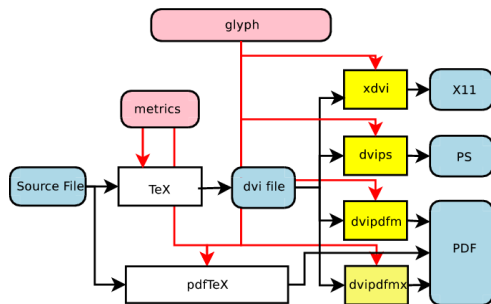


Figure: T<sub>E</sub>X 的工作流程

- 1 载入源文件
- 2 读入需要的字体维度信息 (盒子)
- 3 根据宏指令进行排版, 得到排版信息
- 4 输出为 DVI
- 5 载入字形信息 (具体的字)
- 6 最终输出

# L<sup>A</sup>T<sub>E</sub>X 的源文件

- 选择你最喜欢的编辑器，写入一段源文件，这是我们的第一个 L<sup>A</sup>T<sub>E</sub>X 文件：

```
1 \documentclass{article}
2 \title{First \LaTeX Document}
3 \author{Wang Xu}
4 \begin{document}
5
6 \maketitle
7
8 Hello, \LaTeX!
9
10 \end{document}
```



# 创建 pdf 文件

- 第一种方式:

```
latex first.tex= first.dvi  
dvips first.dvi= first.ps  
pspdf first.ps = first.pdf
```

- 第二种方式:

```
latex first.tex = first.dvi  
dvipdfm first.dvi= first.pdf
```

- 第三种方式:

```
pdflatex first.tex= first.pdf
```



# 用 L<sup>A</sup>T<sub>E</sub>X 写文章的方式

- 选择好要用的模板（不过，就算是没选好的话，将来换也可以）；
- 按照章节组织好文章；
- 插入公式、表格、图形...
- 在需要被引用的地方加上标记 (label), 在引用的地方引用 (ref) 标记；
- 添加参考文献和参考文献的引用 (cite)；
- 根据文章的需要，对需要强调的内容加以强调；
- 熟悉的话，可以做更多格式方面的调整和设置；
- 生成输出文档，查看、修改文档，直到达到目标为止。



# 大纲


- ① L<sup>A</sup>T<sub>E</sub>X 导引
- ② L<sup>A</sup>T<sub>E</sub>X 的基本使用
  - 源文件的基本要素：文本、命令和注释
  - 特殊字符的输入与逐字显示
  - 列表与枚举
- ③ 书写中文
- ④ 插入公式、图形、表格以及程序代码
- ⑤ 参考文献
- ⑥ 更多技巧与常用宏包
- ⑦ 参考文献与网上资料



# 源文件中的要素

LaTeX 的源文件, 尤其是开头, 更接近于程序而不是文章或书, 文章中有一些基本要素

- 注释: 以% 开头的是注释。这一行后面的内容都不会影响最终输出结果。
- # \$ & ~ \_ % { } 为特殊字符, 如果要在生成文档中出现, 必须使用\ 进行转义
- 以\ 开头的词是命令, 更确切地说是需要 LaTeX 特殊处理的内容。
  - 命令如

<code>\LaTeX</code>	<code>\bfseries{bold}</code>	<code>\rule[.5em]{5em}{1pt}</code>
LaTeX	<b>bold</b>	

- 此外, 还有许多可以加入大量内容的环境设置。如:

```
1 \begin{quote}
2 We use ``quote'' environment for words from others.
3 \end{quote}
```

- 除此之外的内容就是要被排版的文字了。



# 文字与空白

多于一个的空格以及换行都会被当作一个空格，而连续的两个换行 (也就是一个空行) 则会被当作是分段。



# 文字与空白

多于一个的空格以及换行都会被当作一个空格, 而连续的两个换行 (也就是一个空行) 则会被当作是分段。

## 示例

```
This is an          example for illustrate the organ%  
ization of  
a \LaTeXe file.
```

```
You can learn many commands and environments of it in  
this example.
```



# 文字与空白

多于一个的空格以及换行都会被当作一个空格, 而连续的两个换行 (也就是一个空行) 则会被当作是分段。

## 示例

```
This is an          example for illustrate the organ%  
ization of  
a \LaTeXe file.
```

```
You can learn many commands and environments of it in  
this example.
```

## 相应的结果

This is an example for illustrate the organization of a L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> file.  
You can learn many commands and environments of it in this example.

# 源文件的结构



# 源文件的结构

## 引言

```
\documentclass[a4paper,12pt]{article}
\usepackage{fancyhdr,listings,graphicx}
\usepackage{CJK,CJKnumb}
\usepackage[a4paper,%
  includehead,%
  includefoot,%
  top=2cm,%
  bottom=2cm,%
```

前导部分的功能就是引入宏包、定义命令...



# 源文件的结构

## 正文

```
\begin{document}
```

这里就是正文部分了...

```
\end{document}
```



# 源文件的结构

## 正文

```
\begin{document}
```

这里就是正文部分了...

```
\end{document}
```

## 另外

一篇 L<sup>A</sup>T<sub>E</sub>X 文章可以写在不同的源文件里, 通过 `\input{filename}` 或 `\include{filename}` 命令载入, 对于保持源文件的清晰和美观有很大好处。



# 文章的标题

文章的标题部分包括文章的标题、作者和日期:





# 文章的标题

文章的标题部分包括文章的标题、作者和日期:

首先是文章的题目:

```
\title{Hello, \LaTeX!\\  
My First \LaTeX Work}
```

这是整篇文章的题目, 如果需要让题目分行, 可以使用双反斜线。



# 文章的标题

文章的标题部分包括文章的标题、作者和日期:

作者列表:

```
\author{Wang Xu\\  
Beijing Univ. of Posts and Telecomm.\\  
gnawux@gmail.com  
\and  
Mr. Sleepy\\  
Who knows where he come from\\  
nobody@anywhere.net%  
}
```



# 章节与段落

L<sup>A</sup>T<sub>E</sub>X 按照章节组织文章的内容

对于article文档类

# 章节与段落

L<sup>A</sup>T<sub>E</sub>X 按照章节组织文章的内容

对于`article`文档类

- 可以有一个特殊的`abstract`环境，作为摘要；

# 章节与段落

L<sup>A</sup>T<sub>E</sub>X 按照章节组织文章的内容

对于`article`文档类

- 可以有一个特殊的`abstract`环境，作为摘要；
- 文章的不同大纲级别包括  
section, subsection, subsection

# 章节与段落

L<sup>A</sup>T<sub>E</sub>X 按照章节组织文章的内容

对于`article`文档类

- 可以有一个特殊的`abstract`环境，作为摘要；
- 文章的不同大纲级别包括  
section, subsection, subsubsection
- 对于超长的章节名称，可以有个短名字出现在页眉和目录中：

```
1 \section[short name]%  
2 {section has a long name}
```

# 章节与段落

L<sup>A</sup>T<sub>E</sub>X 按照章节组织文章的内容

对于article文档类

- 可以有一个特殊的abstract环境，作为摘要；
- 文章的不同大纲级别包括  
section, subsection, subsubsection
- 对于超长的章节名称，可以有个短名字出现在页眉和目录中：

```
1 \section[short name]%  
2 {section has a long name}
```

- 带有 \* 的章节没有编号。

```
1 \section*{asterisk means \ldots}
```

# 章节与段落

L<sup>A</sup>T<sub>E</sub>X 按照章节组织文章的内容

对于book文档类





# 章节与段落

L<sup>A</sup>T<sub>E</sub>X 按照章节组织文章的内容

对于book文档类

- `\maketitle`命令会产生一个扉页，而不是article那样的标题栏。



# 章节与段落

L<sup>A</sup>T<sub>E</sub>X 按照章节组织文章的内容

对于book文档类

- `\maketitle`命令会产生一个扉页，而不是`article`那样的标题栏。
- 多出了两个更高的大纲级别:  
part, chapter



# 章节与段落

L<sup>A</sup>T<sub>E</sub>X 按照章节组织文章的内容

对于book文档类

- `\maketitle`命令会产生一个扉页，而不是`article`那样的标题栏。
- 多出了两个更高的大纲级别:  
part, chapter
- 缺省情况下，每一章都从右边的页开始，可以通过 `openany` 参数改变。



# 章节与段落

L<sup>A</sup>T<sub>E</sub>X 按照章节组织文章的内容

对于book文档类

- `\maketitle`命令会产生一个扉页，而不是`article`那样的标题栏。
- 多出了两个更高的大纲级别:  
part, chapter
- 缺省情况下，每一章都从右边的页开始，可以通过 `openany` 参数改变。
- 目录 (`\tableofcontents`) 等内容会成为一章，但没有编号



# 章节与段落

L<sup>A</sup>T<sub>E</sub>X 按照章节组织文章的内容

对于book文档类

- `\maketitle`命令会产生一个扉页，而不是`article`那样的标题栏。
- 多出了两个更高的大纲级别:  
part, chapter
- 缺省情况下，每一章都从右边的页开始，可以通过 `openany` 参数改变。
- 目录 (`\tableofcontents`) 等内容会成为一章，但没有编号
- 书的内容可以分为三个部分:  
frontmatter, mainmatter, backmatter



# 特殊文本元素

有些特殊的文本元素是由程序生成的，用于辅助文章的理解



# 特殊文本元素

有些特殊的文本元素是由程序生成的，用于辅助文章的理解

- 标题栏或扉页，章节，已经介绍过了。



# 特殊文本元素

有些特殊的文本元素是由程序生成的, 用于辅助文章的理解

- 标题栏或扉页, 章节, 已经介绍过了。
- 脚注

```
1 This sentence have  
2 footnote\footnote{blah blah\ldots}.
```





# 特殊文本元素

有些特殊的文本元素是由程序生成的, 用于辅助文章的理解

- 标题栏或扉页, 章节, 已经介绍过了。
- 脚注

```
1 This sentence have  
2 footnote\footnote{blah blah\ldots}.
```

- 标签

```
1 \section{Introduction}\label{sec:intro}
```



# 特殊文本元素

有些特殊的文本元素是由程序生成的, 用于辅助文章的理解

- 标题栏或扉页, 章节, 已经介绍过了。
- 脚注

```
1 This sentence have  
2 footnote\footnote{blah blah\ldots}.
```

- 标签

```
1 \section{Introduction}\label{sec:intro}
```

- 引用标签

```
1 We have mentioned in \S~\ref{sec:intro} on  
2 pp.~\pageref{sec:intro}\ldots
```



# 特殊字符

我们在上面已经看到很多特殊字符了，这里要介绍几个最常用的<sup>1</sup>:

---

<sup>1</sup>参考 lshort.

# 特殊字符

我们在上面已经看到很多特殊字符了，这里要介绍几个最常用的<sup>1</sup>:

- 引号

```
1 we use ``quote sign'' like this. So pretty.
```

---

<sup>1</sup>参考 lshort.



# 特殊字符

我们在上面已经看到很多特殊字符了，这里要介绍几个最常用的<sup>1</sup>:

- 引号

```
1 we use ``quote sign'' like this. So pretty.
```

- 横线

```
1 one means hyphe-nation\\  
2 two is a -- short bar\\  
3 three --- oh yeah\\
```

---

<sup>1</sup>参考 lshort.

# 特殊字符

我们在上面已经看到很多特殊字符了，这里要介绍几个最常用的<sup>1</sup>:

- 引号

```
1 we use ``quote sign'' like this. So pretty.
```

- 横线

```
1 one means hyphenation\\  
2 two is a -- short bar\\  
3 three --- oh yeah\\
```

- 波浪符

```
1 Tilde(\~{ }) can generate \~{ }.
```

---

<sup>1</sup>参考 lshort.



# 特殊字符

我们在上面已经看到很多特殊字符了，这里要介绍几个最常用的<sup>1</sup>:

显示效果

---

<sup>1</sup>参考 lshort.

# 特殊字符

我们在上面已经看到很多特殊字符了，这里要介绍几个最常用的<sup>1</sup>:

## 显示效果

- 引号

we use “quote sign” like this. So pretty.

---

<sup>1</sup>参考 lshort.





# 特殊字符

我们在上面已经看到很多特殊字符了，这里要介绍几个最常用的<sup>1</sup>:

## 显示效果

- 引号

we use “quote sign” like this. So pretty.

- 横线

one means hyphe-nation

two is a – short bar

three — oh yeah

---

<sup>1</sup>参考 lshort.



# 特殊字符

我们在上面已经看到很多特殊字符了，这里要介绍几个最常用的<sup>1</sup>:

## 显示效果

- 引号

we use “quote sign” like this. So pretty.

- 横线

one means hyphe-nation

two is a – short bar

three — oh yeah

- 波浪符

Tilde(~) can generate ~

---

<sup>1</sup>参考 lshort.

# 逐字显示

有的时候，我们需要刻意避免 L<sup>A</sup>T<sub>E</sub>X 将我们输入的字符当作是特殊指令。



# 逐字显示

有的时候，我们需要刻意避免 L<sup>A</sup>T<sub>E</sub>X 将我们输入的字符当作是特殊指令。

- 单行的逐字显示

```
1 \verb|%$#@~ oh yeah/ --- all are verb
```

- 整段的显示

```
1 \begin{verbatim}  
2 here ,  
3 #0%$  
4 verbatims\ldots  
5 \end{verbatim}
```

- 更复杂的逐字显示: listings, fancyverb...



# 调整显示效果

为了表示不同的内容，我们时常会指定不同的字体族



# 调整显示效果

为了表示不同的内容，我们时常会指定不同的字体族

- 强调命令与下划线

```
1 \emph{emphasis} and \underline{underline}
```



# 调整显示效果

为了表示不同的内容，我们时常会指定不同的字体族

- 强调命令与下划线

```
1 \emph{emphasis} and \underline{underline}
```

- 粗体

```
1 \textbf{bold face}
```



# 调整显示效果

为了表示不同的内容，我们时常会指定不同的字体族

- 强调命令与下划线

```
1 \emph{emphasis} and \underline{underline}
```

- 粗体

```
1 \textbf{bold face}
```

- 斜体

```
1 \textit{italic}
```





# 调整显示效果

为了表示不同的内容，我们时常会指定不同的字体族

- 强调命令与下划线

```
1 \emph{emphasis} and \underline{underline}
```

- 粗体

```
1 \textbf{bold face}
```

- 斜体

```
1 \textit{italic}
```

- 等线字体

```
1 \texttt{typewriter}
```

# 列表与枚举

将要说明的内容分点列出，是最常用的说明手法之一。

## 示例

```
\begin{itemize}
  \item ooollllwww
  \item ttttiirrr
  \item nnnmmmeee
\end{itemize}
```



# 列表与枚举

将要说明的内容分点列出，是最常用的说明手法之一。

## 示例

```
\begin{itemize}
  \item ooollllwww
  \item ttiiiirrr
  \item nnnmmmeeee
\end{itemize}
```

## 效果

- ooollllwww
- ttiiiirrr
- nnnmmmeeee

# 列表与枚举

将要说明的内容分点列出，是最常用的说明手法之一。

## 示例

```
\begin{enumerate}  
  \item first one  
  \item another one  
  \item yet another  
\end{enumerate}
```



# 列表与枚举

将要说明的内容分点列出，是最常用的说明手法之一。

## 示例

```
\begin{enumerate}  
  \item first one  
  \item another one  
  \item yet another  
\end{enumerate}
```

## 效果

- ① first one
- ② another one
- ③ yet another

# 列表与枚举

将要说明的内容分点列出，是最常用的说明手法之一。

## 示例

```
\begin{description}
  \item[itemize] bullet
  \item[enumerate] number
  \item[description] the term
\end{description}
```



# 列表与枚举

将要说明的内容分点列出，是最常用的说明手法之一。

## 示例

```
\begin{description}  
  \item[itemize] bullet  
  \item[enumerate] number  
  \item[description] the term  
\end{description}
```

## 效果

itemize bullet  
enumerate number  
description the term

# 大纲

- ① L<sup>A</sup>T<sub>E</sub>X 导引
- ② L<sup>A</sup>T<sub>E</sub>X 的基本使用
- ③ 书写中文
- ④ 插入公式、图形、表格以及程序代码
- ⑤ 参考文献
- ⑥ 更多技巧与常用宏包
- ⑦ 参考文献与网上资料





# T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X 的中文支持

- T<sub>E</sub>X 本身并不支持中文编码，无法处理中文。
- T<sub>E</sub>X 的一个全新实现，OMEGA更适于处理中文这样的多字节编码。
- 经过一些包装和处理，T<sub>E</sub>X 也可以处理中文，如ConT<sub>E</sub>Xt可以处理中文。
- 但这些都不及 L<sup>A</sup>T<sub>E</sub>X 应用广泛。
- 早期将 L<sup>A</sup>T<sub>E</sub>X 引入中国的张林波老师开发了CCT用以支持中文。
- 但当时的 CCT 对 L<sup>A</sup>T<sub>E</sub>X 处理方法进行了改动，和大量的 L<sup>A</sup>T<sub>E</sub>X 宏包并不兼容。
- 还好德国人Werner Lemberg开发了CJK宏包，支持在 L<sup>A</sup>T<sub>E</sub>X 中使用中文。
- 还有CT<sub>E</sub>X为 L<sup>A</sup>T<sub>E</sub>X 的中文支持作出了巨大贡献。
- 现在，可以考虑使用 X<sub>Y</sub>T<sub>E</sub>X 更方便地处理多国语言文字的排版。



# CJK 宏包



# CJK 宏包

- CJK—Chinese, Japanese, Korean



# CJK 宏包

- CJK—**C**hinese, **J**apanese, **K**orean
- CJK 宏包是德国人 Werner Lemberg 的杰出贡献。



# CJK 宏包

- CJK—**C**hinese, **J**apanese, **K**orean
- CJK 宏包是德国人 Werner Lemberg 的杰出贡献。
- CJK 的基本处理方法



# CJK 宏包

- CJK—**C**hinese, **J**apanese, **K**orean
- CJK 宏包是德国人 Werner Lemberg 的杰出贡献。
- CJK 的基本处理方法
  - ① 处理**输入**源文件，一个一个读入亚洲字符集输入字符；



# CJK 宏包

- CJK—Chinese, Japanese, Korean
- CJK 宏包是德国人 Werner Lemberg 的杰出贡献。
- CJK 的基本处理方法
  - ① 处理输入源文件，一个一个读入亚洲字符集输入字符；
  - ② 将中日韩大字符集的逻辑字体拆分成多个字体，建立逻辑字体到多个具体字体的映射；



# CJK 宏包

- CJK—Chinese, Japanese, Korean
- CJK 宏包是德国人 Werner Lemberg 的杰出贡献。
- CJK 的基本处理方法
  - ① 处理输入源文件，一个一个读入亚洲字符集输入字符；
  - ② 将中日韩大字符集的逻辑字体拆分成多个字体，建立逻辑字体到多个具体字体的映射；
  - ③ 在输出文件中插入对应的字体。





# CJK 宏包

- CJK—Chinese, Japanese, Korean
- CJK 宏包是德国人 Werner Lemberg 的杰出贡献。
- CJK 的基本处理方法
  - ① 处理输入源文件，一个一个读入亚洲字符集输入字符；
  - ② 将中日韩大字符集的逻辑字体拆分成多个字体，建立逻辑字体到多个具体字体的映射；
  - ③ 在输出文件中插入对应的字体。
- 大部分  $\text{\LaTeX}$  宏包不加修改或稍加修改就可以和 CJK 和平共处。



## CJK 宏包

- CJK—Chinese, Japanese, Korean
- CJK 宏包是德国人 Werner Lemberg 的杰出贡献。
- CJK 的基本处理方法
  - ① 处理输入源文件，一个一个读入亚洲字符集输入字符；
  - ② 将中日韩大字符集的逻辑字体拆分成多个字体，建立逻辑字体到多个具体字体的映射；
  - ③ 在输出文件中插入对应的字体。
- 大部分 L<sup>A</sup>T<sub>E</sub>X 宏包不加修改或稍加修改就可以和 CJK 和平共处。
- 配合 C<sub>T</sub>E<sub>X</sub> 修改后的 GB.cpx，标准的 article 和 book 文档类都可以支持中文。



# 书写中文



# 书写中文

- ① 包含 CJK 宏包，以及辅助的 CJKnumb 包。



# 书写中文

- ① 包含 CJK 宏包，以及辅助的 CJKnumb 包。
- ② 使用 CJK 环境

```
1 \begin{CJK*}{GB}{song}
```

在其中就可以输入中文了。环境的参数:



# 书写中文

- ① 包含 CJK 宏包，以及辅助的 CJKnumb 包。
- ② 使用 CJK 环境

```
1 \begin{CJK*}{GB}{song}
```

在其中就可以输入中文了。环境的参数：

**GB** 字符编码，对于中文，还可能是 GBK, UTF-8。



# 书写中文

- ① 包含 CJK 宏包，以及辅助的 CJKnumb 包。
- ② 使用 CJK 环境

```
1 \begin{CJK*}{GB}{song}
```

在其中就可以输入中文了。环境的参数：

**GB** 字符编码，对于中文，还可能是 GBK, UTF-8。  
**song** 字体族，有很多种字体可选



# 书写中文

- ① 包含 CJK 宏包，以及辅助的 CJKnumb 包。
- ② 使用 CJK 环境

```
1 \begin{CJK*}{GB}{song}
```

在其中就可以输入中文了。环境的参数：

**GB** 字符编码，对于中文，还可能是 GBK, UTF-8。  
**song** 字体族，有很多种字体可选

- ③ 使用的字体还是可以改变的

```
1 {\CJKfamily{kai}}
```





# 细节调整



# 细节调整

- 章节标题中的汉字

1 `\CJKcaption{GB}`

这会载入 GB.cpx 文件，它原本是为了一组叫 Komascript 的文档类设计的，CT<sub>E</sub>X 的 **al**oft 修改了它，于是，标准的 article, book 都可以使用它了，标题会使用“第一章”这样的字样，以及“参考文献”，“目录”...



# 细节调整

- 章节标题中的汉字

1 `\CJKcaption{GB}`

这会载入 GB.cpx 文件，它原本是为了一组叫 Komascript 的文档类设计的，CT<sub>E</sub>X 的 **al**oft 修改了它，于是，标准的 article, book 都可以使用它了，标题会使用“第一章”这样的字样，以及“参考文献”，“目录”...

- CJKtilde。

CJK 的设计并不能很好地处理中英文之间的衔接，源文件中，中文和英文之间如果空一格会导致中文中的英文单词的前面紧贴着中文，后面空一格，这样并不美观，我们使用 CJKtilde 命令后，'~' 的行为被改变，成为一个很小的空间，在中英文之间放上它，可以让中英文混合的文章更漂亮。



# 细节调整

- 章节标题中的汉字

1 `\CJKcaption{GB}`

这会载入 GB.cpx 文件，它原本是为了一组叫 Komascript 的文档类设计的，CT<sub>E</sub>X 的 **aloft** 修改了它，于是，标准的 article, book 都可以使用它了，标题会使用“第一章”这样的字样，以及“参考文献”，“目录”...

- CJKtilde。

CJK 的设计并不能很好地处理中英文之间的衔接，源文件中，中文和英文之间如果空一格会导致中文中的英文单词的前面紧贴着中文，后面空一格，这样并不美观，我们使用 CJKtilde 命令后，'~' 的行为被改变，成为一个很小的空间，在中英文之间放上它，可以让中英文混合的文章更漂亮。

- CJKindent: 标准的英文文章第一段的段首不缩进，使用这条命令后，会让第一段的段首有空格，而且将所有缩进都设置成中文习惯的缩进长度。



# 大纲

- ① L<sup>A</sup>T<sub>E</sub>X 导引
- ② L<sup>A</sup>T<sub>E</sub>X 的基本使用
- ③ 书写中文
- ④ 插入公式、图形、表格以及程序代码
  - 插入公式
  - 插入表格
  - 插入图形
  - 插入程序代码
- ⑤ 参考文献
- ⑥ 更多技巧与常用宏包
- ⑦ 参考文献与网上资料



# T<sub>E</sub>X 的数学模式

## 数学模式

- 数学模式几乎是 T<sub>E</sub>X 诞生的原因，而很多人使用 L<sup>A</sup>T<sub>E</sub>X 也是因为漂亮的数学排版。
- 数学模式是 T<sub>E</sub>X 的内建支持，包含了各种数学用的符号和排版的规则，所有你看到的数学书中的数学公式排版，都可以由 T<sub>E</sub>X 实现。
- 数学模式中的所有的公式都可以用纯文本录入，这意味着你在输入公式的时候无须将手移开键盘。



# 公式环境

- 我们可以在文章的字里行间插入公式，如

```
1 $\min_R D_{\text{rec}}(R) + \lambda \sum_{m=1}^M \sum_{l=1}^L R_{ml}$
2 \sum_{l=1}^L R_{ml}$
```

- 结果就像  $\min_R D_{\text{rec}}(R) + \lambda \sum_{m=1}^M \sum_{l=1}^L R_{ml}$
- 也可以断开行，突出显示

```
1 \begin{equation}
2 \min_R D_{\text{rec}}(R) + \lambda \sum_{m=1}^M \sum_{l=1}^L R_{ml} \\
3 \sum_{l=1}^L R_{ml},
4 \end{equation}
```

- 显示结果如

$$\min_R D_{\text{rec}}(R) + \lambda \sum_{m=1}^M \sum_{l=1}^L R_{ml},$$



# 多行公式

- 使用`eqnarray`支持多行公式
- 如下所示：

```
1 \begin{eqnarray}
2 D_{\text{dec}}=&D_0+\theta/(R-R_0)+ \nonumber \\
3 &\quad &\kappa(P_r+(1-P_r)e^{-(C-R)T/L}).
4 \label{equ:distort}
5 \end{eqnarray}
```





# 多行公式

- 使用`eqnarray`支持多行公式
- 如下所示：

```

1 \begin{eqnarray}
2 D_{dec} &=& D_0 + \theta / (R - R_0) + \nonumber \\
3 && \kappa (P_r + (1 - P_r) e^{-(C-R)T/L}) .
4 \label{equ:distort}
5 \end{eqnarray}

```

显示效果如

$$D_{dec} = D_0 + \theta / (R - R_0) + \kappa (P_r + (1 - P_r) e^{-(C-R)T/L}). \quad (2)$$

# 绘制表格

## 使用 tabular 环境绘制表格

```
\begin{tabular}{|l|lc|p{5em}|}  
\hline  
number & type & example & comments \\  
\hline  
\hline  
1 & bf & \textbf{bf} & for the  
defination etc.\\  
2 & italic & \textit{italic} & emph.\\  
\hline  
\end{tabular}
```



# 绘制表格

显示效果如

number	type	example	comments
1	bf	<b>bf</b>	for the definition etc.
2	italic	<i>italic</i>	emph.

每一列的对齐方式对应环境的声明

```
\begin{tabular}{|l|l|c|p{5em}|}
```



# 浮动表格环境

- 我们常常希望表格的位置不是固定的，而是在页面的某些位置浮动，于是，使用`table`环境构成浮动图形。



# 浮动表格环境

- 我们常常希望表格的位置不是固定的，而是在页面的某些位置浮动，于是，使用`table`环境构成浮动图形。
- 同时，我们还用“`caption`”为表格加标题，用“`label`”来打标记。



# 浮动表格环境

- 我们常常希望表格的位置不是固定的，而是在页面的某些位置浮动，于是，使用`table`环境构成浮动图形。
- 同时，我们还用“`caption`”为表格加标题，用“`label`”来打标记。
- 如示例

```
1 \begin{table}[htb]
2 \centering
3 \caption{Sample Table}
4 \label{tab:sample}
5 \begin{tabular}{|l|l|c|p{5em}|}
```



# 支持的图片格式

- 我们在  $\text{\LaTeX}$  文档中，通常使用 **graphicx** 宏包来插入图形。
- 以下格式的图片是被支持的:

	图形格式	说明
$\text{\LaTeX}$	eps	嵌入式 PS 文件
pdf $\text{\LaTeX}$	pdf	对就是那个
	png	一种位图
	jpg	另一种位图



# 支持的图片格式

- 我们在  $\text{\LaTeX}$  文档中，通常使用 **graphicx** 宏包来插入图形。
- 以下格式的图片是被支持的：

	图形格式	说明
$\text{\LaTeX}$	eps	嵌入式 PS 文件
pdf $\text{\LaTeX}$	pdf	对就是那个
	png	一种位图
	jpg	另一种位图

- eps 图形格式
  - 是一种矢量图
  - 可以直接绘制
  - 可以从 PS 文件中剪裁或通过 PS 虚拟打印机得到
  - 可以通过其他工具，如 (Imagimagick) 转换格式得到。





# 插入图片



## 插入图片的命令行

以上图片使用如下命令行插入:

```
\includegraphics[height=2cm]{figures/wti-logo}
```



# 插入图片



## 常用命令行参数

参数	用途
width/height	图片宽度或高度, 如本例
scale	图片缩放的系数, 如 <code>scale=0.5</code>
angle	图片旋转的角度, 如 <code>angle=90</code>
clip	eps 图片切边

# 浮动图形

- 与表格类似，图片也可以以浮动图形的方式出现在文档之中。
- 格式如下

```
1 \begin{figure}[htb]
2   \begin{center}
3     % \includegraphics{}
4   \end{center}
5   \caption{Sample figure}
6   \label{fig:sample}
7 \end{figure}
```



# listings 宏包

- `verb`命令和`verbatim`环境可以用来原封不动的保持输入的内容，理论上讲，我们可以使用它们来展示程序代码。
- `listings` 宏包的特色
  - 两个命令 (`lstinline`和 `lstinput`) 与一个环境 (`lstlisting`), 其中 `lstinput` 这个命令最为贴心, 可以读入其它文件。
  - 支持多种程序语言的语法, 对关键字、注释等进行语法加亮
  - 支持边框、背景色、字体风格、行号等特征
- 本幻灯片中大部分代码是通过 `listings` 宏包加入的。



# 设置与调用 listings 宏包

## 本幻灯片中的 listings 设置

```
\lstset{extendedchars=false}  
\lstset{linewidth=.8\textwidth}  
\lstset{basicstyle=\small\tt}  
\lstset{keywordstyle=\tt\color{red}}  
\lstset{numbers=left}  
\lstset{numberstyle=\tiny\color{blue}}  
\lstset{frame=trbl}
```

## 本文中使用 listings

```
\lstinputlisting[firstline=295,lastline=295]{tex.tex}
```

# 设置与调用 listings 宏包

## 常用参数

参数名称	说明
caption	一般的 caption 的内容，文章的题目
label	一般的 label 的含义，引用此列表的标记
frame	边框设置
language	设置的语言
firstline	引用文件的起始行号
lastline	引用文件的终止行号



# 大纲

- ① L<sup>A</sup>T<sub>E</sub>X 导引
- ② L<sup>A</sup>T<sub>E</sub>X 的基本使用
- ③ 书写中文
- ④ 插入公式、图形、表格以及程序代码
- ⑤ 参考文献
- ⑥ 更多技巧与常用宏包
- ⑦ 参考文献与网上资料



# 参考文献列表

参考文献列表很象 itemize 环境

```
\begin{thebibliography}{10}  
\bibitem{bib:ubi91weiser}  
  M. Weiser,  
  The computers for the twenty-first century,  
  Scentific American,  
  vol. 265, pp. 94-100, Sept. 1991.  
\bibitem{bib:wwrf}  
  The Wireless World Research Forum,  
  http://www.wireless-world-research.org .  
\end{thebibliography}
```





# 参考文献的引用

## 使用`cite`命令引用参考文献

- 可以使用最简单的引用方式

```
1 in \cite{bib:ubi91weiser}, \ldots
```

- 也可以一次引用多个参考文献

```
1 in \cite{bib:ubi91weiser,bib:wwrf}, \ldots
```

- 还可以加上更多的说明

```
1 in \cite[pp. 99-105]{bib:wwrf}, \ldots
```



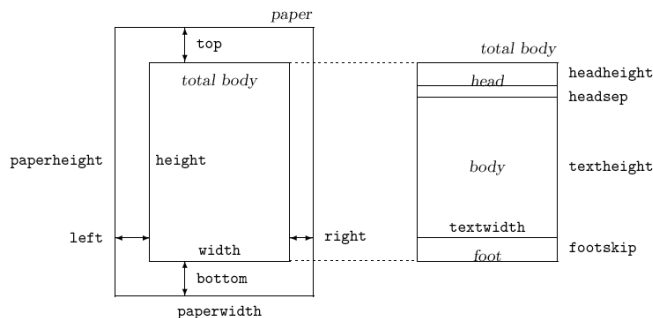
# 大纲

- ① L<sup>A</sup>T<sub>E</sub>X 导引
- ② L<sup>A</sup>T<sub>E</sub>X 的基本使用
- ③ 书写中文
- ④ 插入公式、图形、表格以及程序代码
- ⑤ 参考文献
- ⑥ 更多技巧与常用宏包
  - 页面维度设置
  - 页脚页眉设置
  - 自定义命令与环境
- ⑦ 参考文献与网上资料



# 页面维度参数

LaTeX 把页面用以下几个长度描述



# geometry 宏包

## geometry 的典型使用

```
\usepackage[a4paper,%  
  includehead,%  
  includefoot,%  
  top=2cm,%  
  bottom=2cm,%  
  left=2cm,%  
  right=2cm,%  
  head=15pt,%  
  foot=15pt%  
{geometry}
```



# 页面风格

## LaTeX 的内建风格

- **empty**是最简单的，也就是什么都没有。
- **plain**是很简单的，缺省的风格，**article** 的首页和 **book** 的每章开始的地方通常也会是这个风格，只有下面有页码。
- **heading**页眉放置章节的标题。



# 页面风格

## L<sup>A</sup>T<sub>E</sub>X 的内建风格

- **empty**是最简单的，也就是什么都没有。
- **plain**是很简单的，缺省的风格，**article** 的首页和 **book** 的每章开始的地方通常也会是这个风格，只有下面有页码。
- **heading**页眉放置章节的标题。

## fancyhdr宏包

- 页脚上方和页眉下方划线
- 左中右三段式的页脚和页眉的设置



# fancyhdr 宏包

## article中常用的 fancyhdr 用法

```
\fancyhf{}  
\pagestyle{fancy}  
\lhead{\rightmark}  
\chead{{\color{blue} $\spadesuit$ gnawux $\spadesuit$}}  
\rhead{\bfseries\thepage}  
\lfoot{Manipulated by \LaTeXe.}  
\rfoot{K.I.S.S.---Keep It Simple Stupid}  
\renewcommand{\headrulewidth}{0.5pt}  
\renewcommand{\footrulewidth}{0.5pt}  
\fancypagestyle{plain}{%  
  \fancyhf{} % get rid of headers on plain pages  
  \cfoot{\bfseries\thepage}  
  \renewcommand{\headrulewidth}{0pt} % and the line  
  \renewcommand{\footrulewidth}{0.5pt}  
}
```

# 自定义命令与环境

我们还可以修改一些命令或环境的行为

- `newcommand` 和 `newenvironment` 命令可以定义新命令和环境
- `renewcommand` 和 `renewenvironment` 命令可以修改已有的命令和环境





# 自定义命令与环境

我们还可以修改一些命令或环境的行为

- `newcommand` 和 `newenvironment` 命令可以定义新命令和环境
- `renewcommand` 和 `renewenvironment` 命令可以修改已有的命令和环境
- 这里也有一个例子

```
1 \renewcommand{\sectionmark}[1]  
2 {\markright{\thesection\ #1}}
```



# 大纲

- ① L<sup>A</sup>T<sub>E</sub>X 导引
- ② L<sup>A</sup>T<sub>E</sub>X 的基本使用
- ③ 书写中文
- ④ 插入公式、图形、表格以及程序代码
- ⑤ 参考文献
- ⑥ 更多技巧与常用宏包
- ⑦ 参考文献与网上资料



# 文章推荐

## 入门必读

- ① lshort-cn
- ②  $\text{\LaTeX}$  2 $\epsilon$  使用手册 ( $\text{\LaTeX}$  Manual,  $\text{\LaTeX}$  科技文献排版指南电子版)
- ③  $\text{\LaTeX}$  插图指南 (epslatex) 中文版

## 常用快速参考手册

- $\text{\LaTeX}$  Command.
- $\text{\LaTeX}$  Cookbook.



其它推荐文章:

- CT<sub>E</sub>X-FAQ
- lshort-en
- L<sub>A</sub>T<sub>E</sub>X 插图指南 (Using Imported Graphics in L<sub>A</sub>T<sub>E</sub>X 2<sub>ε</sub>) 英文版
- T<sub>E</sub>XBook
- L<sub>A</sub>T<sub>E</sub>X Companion Ch8 (经典书籍 L<sub>A</sub>T<sub>E</sub>X Companion 的第八章, 详细讲解数学输入)
- latex-help-book
- short math guide
- math mode



# 网站推荐

- CT<sub>E</sub>X: 中文 T<sub>E</sub>X 中坚站, 文档、下载、论坛
- 新水木T<sub>E</sub>X 版
- CTAN: Comprehensive T<sub>E</sub>X Archive Network, T<sub>E</sub>X 大全



# 谢谢!

2008 年 5 月 3 日



# 大纲

- ⑧ 八卦 Knuth
- ⑨ 使用 X<sub>Y</sub>TeX 编写文档
- ⑩ 使用 vim 编辑 L<sup>A</sup>T<sub>E</sub>X 文档



# 早期经历 I

- 1938 年 1 月 10 日生于美国威斯康星州密尔沃基市。他在模式方面辨别和熟练操作的能力在八年级的时候开始显现出来。当时，当地的一家糖果制造商举办了一项比赛，比赛要求选手用其品牌“Ziegler's Giant Bar”中的字母组成新的单词，规定时间内组成单词数量最多者获胜。Knuth 参加了比赛，并以单词总数 4500 余个远远超过了裁判的 2500 个的标准，轻松赢得头奖。赛后，Knuth 说道，如果自己当初想到回答时用些省略符号的话，还能写出更多。这次比赛 Knuth 为学校赢得了一台电视机，还为每个同学赢得了一根糖果棒。





## 早期经历 II

- Knuth 多产的出版事业开始于他的高中时代，当时他的科技设计被 *Westinghouse Science Talent Search* 光荣提及。他的 “Potzbie System of Weights and Measures” 的基础章节被登在 *Mad* 杂志第 26 号，“Power” 的基础章节被叫作 “whatmeworry”。*Mad* 的编辑认识到了年轻的 Donald 著作的重要性，以 25 美元买下了他的文章，并刊登在了其 1957 年 6 月的期刊上。
- 虽然 Knuth 的等级平均分是学校历史上最高的，但是他和他的指导老师还是对他能否成功学习大学数学持怀疑态度。Knuth 说在他高中阶段和大学早期一直有一种自卑感，这个问题一度是他的一个障碍。作为一个大学新生，Knuth 没有对于失败的恐惧，他花了许多时间攻克额外的数学难题，几个月后，他在这方面的能力已经远远超过了其他同学。



## 早期经历 III

- 当 Knuth 在 Case 科学院（现在的 Case Western Reserve）获得物理奖学金时，梦想成为一个音乐家的计划改变了。Knuth 回去继续研究数学是在大二，当时一个爱出难题的教授提出了一个特殊的问题，并说哪个学生能解决这个问题就立刻记成绩“A”。Knuth 跟大多数同学一样，也认为那是道解不出来的题目，直到有一天，他错过了公共汽车，只能步行去看一个演出，Knuth 利用路上这点空闲时间决定尝试一下。那阵子他运气真的是非常好，不仅问题很快就解开了，得到了“A”，还成功地经常逃课。虽然 Knuth 也承认，逃课让他有负罪感，但是很明显，他完全有能力补上落下的功课，接下来的一学年，他的离散数学就又得了个“A”，而且还获得了给自己不能参加的课程评定论文等级的工作机会。
- 1956 年，作为 Case 的新生，Knuth 第一次接触到了计算机，那是一台 IBM 650。Knuth 说直到一年后，女孩才进入了他的生活。这又是计算机科学界一直以来亏欠科学家们的一个事例之一。



## 早期经历 IV

- Knuth 熬夜读 IBM 650 的说明手册，自学基本的程序设计。那时，在高等计算机语言发明之前，程序编写只能用第二代或是汇编语言。这个工作既耗时又困难，因为指令必须根据每台机器特定的构造编写，而实际上指令只须一步就可从二进制 0、1 系列转存到计算机硬盘上。Knuth 说，有了第一次使用 650 的经历，他便肯定自己能编写出比说明手册上介绍的更好的程序。
- Knuth 很快便开始编写可以执行数学函数的程序。他的第一个程序是把数字转化为素数，第三个是做井字游戏（或者说是让计算机在改正每次输的错误的过程来学会玩井字游戏）。作为学校篮球队的经理，Knuth 编写了一个根据不同成绩标准评定每个运动员对球队贡献等级的程序，他的努力赢来了那些认为这样做有助于球队赢得同盟冠军的教练的好评。Knuth 的成就成了新闻周刊的标志，他和教练、计算机的照片也被刊登在 IBM 650 后来的说明手册上。



## 早期经历 V

- 1960 年, Knuth 从 Case 毕业时享有着最高荣誉, 在由全体教员参加的选举上, 他因其公认的出众成就获得了硕士学位。1963 年, Knuth 回到加利福尼亚理工学院攻取了数学博士学位, 之后成为了该院的数学教授。在加利福尼亚理工学院任教期间, Knuth 作为 Burroughs 司的顾问继续从事软件开发工作。1968 年, 他加入了斯坦福大学, 九年后坐上了该校计算机科学学科的第一把交椅。1993 年, Knuth 成为斯坦福大学 “The Art of Computer Programming” (计算机程序设计艺术) 的荣誉退休教授。



## 早期经历 VI

- 1962 年, Knuth 还是个研究生的时候就开始了为不同的机器编写编译程序。在不知道众多软件公司正高额寻求成百上千的编程者的情况下, Knuth 编写了一个程序, 赚得 5000 美元, 他的名字立刻响誉了整个行业。世界上一流的出版社 Addison-Wesley 找到 Knuth, 请他写一本关于编译程序的书。到 1966 年, Knuth 已经发表了 3000 页的手写设计草图, 并且发明了一种综合方法, 用于分析或决定结构翻译所客观需要的文法规则。最近, 关于他的那第一部著作, Knuth 自己这样评述: “用三年半的时间写第一章可并不是件好事。”



## 早期经历 VII

- 当 Knuth 的出版商计算出他的那 3000 页的笔记打印成文章大约需要 2000 页时，大家才发现这实际上是一项多么大的工程。Knuth 决定将它详述，成为一部更大的关于程序设计科学的纵览，共分为七个部分。一部巨著就这样诞生了。《计算机程序设计艺术》，至今仍是各程序类图书书架上标志性的书籍。微软首席执行官比尔·盖茨在 1995 年接受一次采访时说，“如果你认为你是一名真正优秀的程序员，就去读第一卷，确定可以解决其中所有的问题”。盖茨还说：“如果你能读懂整套书的话，请给我发一份你的简历。”



## 早期经历 VIII

- 依 Knuth 本人所讲,《计算机程序设计艺术》是他毕生最重要的事业,其目的是“组织和总结所知道的计算机方法的相关知识,并打下坚实的数学、历史基础”。Knuth 撰写的前三卷被翻译成多种语言,到 1976 年为止,已卖出超过一百万册。他目前正全神贯注地编写第四卷,他期望第四卷的篇幅约为 2000 页,并分为三个独立的章节。为了完成丛书的其余部分,Knuth 现在进入了一种引退的状态,全身心地投入这项工作。Knuth 说,一般说来,他更喜欢在一段时间内集中精神完成一项工作,正像他自己在书中提出的:按“批处理”的模式。
- Knuth 从他主要的工作计划中拿出了十年,即从 1976 年起,致力于对数字排版的研究,设计了著名的  $\text{T}_{\text{E}}\text{X}$  系统,字体生成程序 METAFONT。这项工作带来的值得注意的副产品是用于结构文件和“文章性编程语言”的 WEB 和 CWEB 语言。



## 早期经历 IX

- 现在，Knuth 和他的妻子 Jill，两个孩子 John 和 Jennifer 一起，住在斯坦福大学校园里。他继续着《计算机程序设计艺术》第四卷的编写工作。虽然说 Knuth 是全身心的投入这一项工作，但他还是能挤出时间研究 MMIX（计算机编程艺术一书中的虚拟计算机汇编指令体系）的设计，那是一台 64 位 RISC（精简指令集计算机）。而他的业余爱好仍然是音乐，还一直邀请那些能够即兴演奏四手联弹钢琴曲的人们给他留下便条，以便安排一些活动。





# 成就 I

**编译程序** 编译程序能够实现高级语言和二进制机器语言之间的翻译。Knuth 教授在这方面最著名的成就是 LR(k) 分析的研究，那是一个能使确定一串字符文法规则的过程更加顺畅的值得注目的方法。

**属性文法** 在编译程序的工作之后，Knuth 教授走上了形式上定义程序语言意义、语义的研究道路。他建立起一个更加经济的方法去编译联合规则，他把这种方法称作“属性规则”。该方法创立的同时，计算机科学的子域被称作“属性文法”。

**算法** 也许 Knuth 教授在计算机科学领域最原创的贡献就是他对于算法的分析。算法是编写一个程序，使之能去完成一项任务的基础，例如搜索或分类等。他所著的《计算机程序设计艺术》是一个详尽的算法实践和科学的综述。



## 成就 II

**数字化排版** 由于对计算机排版的校样的低质量感到无法忍受，Knuth 教授拿出了十年时间，来开发一个高质量的计算机排版系统，包括用于文件排版和字体生成的软件系统，这两个系统现在已被世界大多数出版社运用。也即  $\text{T}_{\text{E}}\text{X}$  用于出版业的科学排版和“优美文章”的产品； $\text{METAFont}$  一个字体生成程序。

**结构化文件和文章性程式语言** Knuth 教授的排版研究，引领他发明了文件构造的两种语言和一个方法论，叫作“文章性程式语言”。语言分别是  $\text{WEB}$  和  $\text{CWEB}$ ，它们促进了程序编写向“文学作品，是用来阅读的”这个方向发展。使程序员能够同时创建两个不同的系统程序，一个面向人，另一个面向机器。



# 八卦 I

- 传说 Knuth 写书写文章的第一稿都是用铅笔写的，很多人不明白他为什么不用键盘。其实原因是这样，Knuth 曾经参加过一个训练小秘的学习班，练习打字每分钟 80 个词以上。到了后来，他发现他打字的速度大大高于他思考的速度，如果他用键盘，就会出现很多停顿。所以他决定用铅笔，这样可以与自己的思考速度保持一致。
- 很多人都对  $\text{\TeX}$  断行的算法感到满意，其实只有 Knuth 觉得担心。他设计  $\text{\TeX}$  的时候听说有一本书叫做 *Aesthetic Measures*，作者是美国 No.1 的数学家 George David Birkhoff。是说怎样用数学公式来衡量“美”。他查阅了 7 个 Harvard 图书馆，其中有一个图书馆有一个拷贝，但是却被人借走了。无奈，跑到 MIT 去借。还好，借到了。后来他就在  $\text{\TeX}$  里加入了一个变量叫做 *badness*，用来衡量一行文字的美感。*badness* 越小这行文字就越美。但是与 Birkhoff 不同，Knuth 对这个公式没有多少信心。也许是因为谦虚。



## 八卦 II

- Knuth 的书都是自己用  $\text{\TeX}$  排版的，但是却不都是自己设计的。传说 Knuth 和 Graham, Patashnik 合作写 *Concrete Mathematics* 的时候请了一位有名的图书版面设计家为他们设计好了书的尺寸，字体大小，标题样式，后来 Hermann Zapf 专门设计了一种数学字体叫做 Euler。自此，数学家 Euler 的灵魂浮游于 CM 当中……此外，这本书本来不会做的那么花哨的，结果后来他们专门为那本书设计了字体，页面样式，所有习题都给出了出处。几乎所有页面都至少有一个涂鸦，连《爱利丝漫游奇境记》都被列入参考文献……
- 这是怎么回事呢？原因就是 Knuth 在写书期间去看了一部电影：“白雪公主与七个小矮人”。看完之后 Knuth 感叹道：难以置信，这样完美的艺术品竟然可以在 1937 年完成。我一定要把我的书做成一个艺术品，而且要在三个月之内完成！结果他说到做到了。



# 八卦 III

- 大家都知道 1974 年图灵奖授予 Knuth 主要是因为他写了一部巨著叫做 *The Art of Computer Programming* 但是不幸的是，很多人不能理解，甚至不相信他为这部书起了这么一个不“科学”的名字。后来很多人的著作里出现这样的文献引用：“The Act of Computer Programming, Donald Knuth.”
- Knuth 是个喜欢自夸的人，这是毫无疑问的。在他出版 *The Art of Computer Programming* 之前就已经有这种苗头了。还没有出版的时候，在一次会议上，有个人知道他的这种性格，就说：“我猜你正在写的这本书的书名肯定是 ‘An Introduction to Don Knuth’。” Knuth 回答说：“正好相反。我要以你的名字来命名它”。原来这个人的名字叫 “Art Evans”。



# 八卦 IV

- Knuth 是 Caltech 数学系博士毕业的, 但是他常常说: “我戴着一顶计算机科学家的帽子, 而不是一顶数学家的帽子。” 这说明他似乎对数学家有某些看法。在他看来数学家只知道 “What is it”, 而他还知道 “How to do it”. 这就是他认为的数学与计算机科学的区别。
- Knuth 回到 Stanford 时, 学校让他自己给自己一个头衔他就选了一个 Professor Emeritus of The Art of Computer Programming 他其实觉得 “计算机科学” 不是科学。虽然大家很希望计算机编程变成科学, 但是 Knuth 觉得奇怪为什么大家这么喜欢用 “科学” 这个字眼, 以致于他们瞬间把程序设计变成了科学, 方法就是叫它 “计算机科学”。在他眼里, 计算机科学其实仍然是门艺术。
- 在 Knuth 的眼里, 科学与艺术有什么区别呢? 艺术是人创造的, 而科学不是。艺术永远是可以无止境的提高的, 而科学不是。艺术需要天赋才能掌握, 而科学不需要, 按部就班就行。所以...



## 八卦 V

- Knuth 的 *TAOCP* 开始写的也不那么好。传说有一天 Bob Floyd 给 Knuth 一封信，开门见山就说：“Don，请不要用那么多感叹号！”信的结尾至少打了五个感叹号。看了之后，Knuth 发现 *TAOCP* 里竟然平均每页有两个感叹号！！
- 有人说 Knuth 写完三卷 *TAOCP* 就去研究  $\text{T}_{\text{E}}\text{X}$ ，其实是因为害怕写第四卷。很多人早就希望他放下  $\text{T}_{\text{E}}\text{X}$ ，继续写书。Knuth 说：“一个人要把事情做的完美，只有当他跟上帝的意图保持和谐，现在上帝要我去写第四卷了。”
- Knuth 很推崇随机算法。他批改作业时，一般都是翻到随机一页，仔细看那一页，之后就对学生的作业有了一个概貌，其它的部分就看不那么仔细了。Knuth 看书的时候首先看第 316 页，如果书很短就看第 100 页。仔细看那一页。之后他就可以说那本书好不好。据说这样做出判断的正确率很高。不知道是否有很多人跟他学，看 316 和 100。以后写书要注意把第 316 页或者 100 页写好呀！



# 八卦 VI

- Knuth 发明了一种程序设计方法叫做 **Literate Programming** (文学编程), 把程序当成文学作品来写。这样可以创造永恒的作品, 甚至几十年后还有人用它作为茶余饭后的读物。他为什么要发明这个东西。原因有:
  - 他想让一个程序员 (也许是他自己) 在某一天拿到普利策奖。
  - 他想让提出 “**Structured Programming**”(结构化程序) 的那些家伙在写 “非文学程序” 的时候, 就像他当年写 “非结构化程序” 的时候一样觉得自己有罪。

他的 “文学编程” 思想最早是在英国 **Computer Journal** 发表的。当人问他为什么不在美国发表。他说, 美国人没文化, 他们不能理解这个东西





# 八卦 VII

- Knuth 喜欢在他的作品里用 “we” 作为主语，虽然很多时候文章是他一个人写的。有人认为使用被动语态好。但是 Knuth 认为不应该大量使用被动语态。“用 We 可以减少被动语态带来的麻烦。‘we’ 是指你和你的读者。” 那么怎么称呼作者？答案是: the authors, the first author, 或者直接用名字。但是他确实反对使用 “I”，除非你是身名显赫，人人尊敬的君王式人物，否则最好不要在论文里用 “I”。在你描述你的程序时，喜欢说 “we insert the element in the heap”，还是 “it inserts the element in the heap”？Knuth 总是喜欢用 “we”。显然他已经融于算法的动作之中了。
- 虽然他不喜欢论文里用 “I”，但是他喜欢让他的程序自称 “I”。看这里：



## 八卦 VIII

```

This is TeX, Version 3.14159 (Web2C 7.3.7x)
! I can't find file `kkkk.tex'.
<*> kkkk.tex
Please type another input file name:

```

有很多人跟着他学，把这种称呼顽皮的发挥到极致：

```

Welcome to Scheme 48 0.57
Copyright (c) 1993-2001 by R Kelsey and J Rees.
Type ,? (comma question-mark) for help.
> (define (sq x) (* x x))
; no values returned
>
Exit Scheme 48 (y/n)? <按 Ctrl-D>
I'll only ask another 100 times.

```



## 八卦 IX

```
Exit Scheme 48 (y/n)?  
I'll only ask another 99 times.  
Exit Scheme 48 (y/n)?  
I'll only ask another 98 times.  
Exit Scheme 48 (y/n)?  
I'll only ask another 97 times.  
Exit Scheme 48 (y/n)?  
.....
```



# 八卦 X

- Knuth 曾经在 American Mathematical Monthly 发表过一篇叫做 *The Toilet Paper Problem* 的论文，据说是一个研究怎样合理使用厕纸的算法。现在收录于 *Selected Papers on Analysis of Algorithms*，这篇论文投到 Monthly 时，在节标题使用了大量“粪便学”词汇。编辑警告 Knuth 说，笑话在我们这里是危险的，请你三思！不得已啊，Knuth 后来把小节标题里的那种词改掉了。可是他很不想改文章的标题。怎么办呢？他给编辑一封信说，我已经以这个标题做过两次演讲，这个主题已经被广泛的采用和讨论……云云。最后编辑回信说：“你的厕纸被接受了！” (Your toilet paper is accepted!)

P.S. 附录：增加这篇论文的信息：

Stanford 计算机科学系楼里的厕纸架上可以并排放两筒厕纸。人上厕所时有可能从两个中的一个取纸。两个卷筒不一样大的时候，喜欢从大的那筒拿纸的人叫做 **big-chooser**，喜欢从小的那筒拿纸的人叫做 **little-chooser**。如果两筒大小差不多，一般人都会从离自己最



# 八卦 XI

近的那筒取纸。厕纸用完的时候一般由 janitor(门口老大爷?) 提供新的纸。如果一边的用完了, 那就换掉那一边, 如果两边同时用完, 那么有人就有麻烦了……(我怀疑 Knuth 遇到了那么一次:)

Knuth 似乎在计算那种两筒同时用完的窘境出现的概率……

- 高老爷子发布了  $\text{\TeX}$  的源码, 同时宣布支付给每一个发现 bug 的人 2.56 美元 (256 美分为 16 进制的 1 美元)。此后奖金逐年翻倍, 直至目前冻结在 327.68 美元。但由于程序的设计及编写极其优秀, 被发现的 bug 极少, 加之高德纳教授的权威声望, 以至于有发现 bug 的获奖人将高德纳教授亲笔签名的奖金支票装裱进相框内留作纪念。



# 大纲

- 8 八卦 Knuth
- 9 使用 X<sub>Y</sub>TeX 编写文档
- 10 使用 vim 编辑 L<sup>A</sup>T<sub>E</sub>X 文档



当前 X<sub>Y</sub>TeX 的版本为 (0.99991)



# X<sub>Y</sub>TeX 说明

- 基于 e-TeX 设计
- 更好的支持 Unicode 字符集
- 可以使用现代化的字体技术
- 对字体更好的渲染
- 一个示例文件 xelatex.tex





# 大纲

- 8 八卦 Knuth
- 9 使用  $\text{\XeTeX}$  编写文档
- 10 使用 vim 编辑  $\text{\LaTeX}$  文档



# 安装配置 vim-latex-suite

- 安装 vim-latexsuite 软件包
  - 直接将该软件包解压到用户根目录下的.vim 目录
  - 修改.vim/ftplugin/tex.vim 文件，添加：
    - `set sw=2`
    - `set iskeyword+=:`
- 修改.vimrc 文件，添加：
  - `filetype plugin on`
  - `set shellslash`
  - `set grepprg=grep\ -nH\ $*`
  - `filetype indent on`
- Happy vim & texer!



# The End

# The End of Appendix G: $\LaTeX$ .

