核心理念

出版是科学事业不可或缺的一环，它深刻影响着科学家的职业发展。然而，出版工作（如格式化、引用管理、合并修改等）常常耗费大量精力。

本章旨在介绍一系列高效的工具与工作流程，帮助科学家**自动化处理通用的出版问题**，从而能更专注于**数据分析、内容质量**等更重要的部分。

本章主要内容

- **文档处理范围**：概述文档处理中两个相互竞争的范围。
- **文本编辑器**：如何在文档编辑中高效使用文本编辑器。
- **标记语言**：了解用于基于文本文档处理的标记语言。
- **文献管理**：如何有效管理引用并自动创建参考文献。

> "在科学中，科学家试图以人们能够理解的方式告诉人们之前所不知道的东西。但在诗歌中正好相反。"
> —— Paul Dirac

传统工具：所见即所得 (WYSIWYG)

- **定义**: "What You See Is What You Get"，格式与内容不可分割。
- **早期工具**: 石碑、纸莎草、图形文件处理软件等。
- **常见现代工具**:
  - Microsoft Word
  - Google Docs
  - Open Office
  - Libre Office

WYSIWYG在科学领域的挑战

1. **分散注意力**: 格式问题会分散本应集中在内容上的注意力。
2. **修改困难**: 修改文本格式成为一项艰难的任务。
3. **协作不便**: 文档的二进制格式增加了版本控制和合并更改的难度。

核心缺陷

- **内容与格式无法分离**: 这是其最大的缺点，无法将**内容**（文本和图片）与**格式**（字体和边距等）有效分开处理，不利于高效、可重复的科学论文编写流程。

“所见即所得” (WYSIWYG) 的问题

1. **分散精力**：在传统文字处理器中，作者需同时处理内容和格式（如字体、间距），无法完全专注于用词造句和逻辑梳理。
2. **投稿困难**：不同期刊有不同的格式要求。内容与格式的捆绑使得文章在被拒后，向其他期刊转投时，修改格式变得非常麻烦。

解决方案：“所见即所想” (WYSIWYM)

- **核心思想**: "What You See Is What You Mean"。作者只需专注于文章的**内容和逻辑结构**，而将**格式**交由独立的样式文件来处理。
- **优势**: 当需要切换期刊格式时，只需更换样式文件，论文内容无需任何改动，极大提高了效率。

推荐工具与技术

- **强烈推荐：LaTeX**
  - **优点**: 拥有强大的接口、美观的数学公式排版，在物理学等领域具有压倒性的普及率。是高效科研人员的基本技能。
- **其他工具**: DocBook, AsciiDoc, PanDoc.

额外好处

- **高效引用**：能够轻松实现参考文献的高效管理。
- **可再现性**：更适合跟踪和管理文档中的改动，提升科研工作的可再现性。

核心挑战：二进制格式与版本控制

- **传统存储**：许多“所见即所得” (WYSIWYG) 编辑器（如 Word）将信息存储为复杂的**二进制格式**。
- **版本控制难题**：
  - 二进制文件难以进行版本控制，因为不解码就无法理解文件间的逻辑差异。
  - 这导致很难有效地追踪和管理文档的修改历史。

现有工具的局限性

- **Word 的“修订”功能**：
  - 虽然 Word 提供了内置的“修订”工具，是一个进步。
  - **但其模型有缺陷**：通常需要一个人作为维护主内容的角色，手动合并其他人的内容，难以实现真正的多人**同步编辑**。

纯文本的优势

- **透明与强大**: 通过版本控制系统维护的**纯文本文件**，能够获得更透明、更强大的修改跟踪功能。
- **科学溯源**: 使用版本控制纯文本格式，能为整个科学研究过程提供清晰的**源谱 (provenance)**，记录下每一个改动。
- 强烈建议在文档处理中使用**纯文本标记语言**，并选择合适的**文本编辑器**来编辑它们，以实现高效、透明的版本控制。

20.2 文本编辑器：实现分离的工具

前面我们确立了**内容与格式分离**的核心理念，并提倡使用**纯文本**进行科学写作。**文本编辑器**在其中扮演了重要的角色。

编辑器选择的核心原则

- **自由与效率**: 在纯文本工作流中，协作者可以**自由选择各自顺手的编辑器**。这与 WYSIWYG 工具强制所有人使用相同软件形成了鲜明对比。
- **技巧大于工具**: 如何高效地**使用**一个编辑器，远比选择了**哪一个**编辑器更为重要。建议充分了解几款主流编辑器（如 Vi, Emacs, Nano）的特点后，独立做出选择。

高效文本编辑器的必备功能

这些编辑器之所以强大，是因为它们普遍具备以下功能，以辅助纯文本写作：

- **语法高亮**：清晰地区分标记和内容。
- **文本折叠**：专注于文章的特定部分。
- **多文件/并列窗口**：方便对照和引用。
- **内置 Shell**：无需离开编辑器即可执行命令。

通过使用这些强大的文本编辑器来编写**标记语言**，作者可以将注意力从易分散精力的格式调整中解放出来，真正实现对**写作内容本身**的专注。

我们已经知道了**为什么**要将内容与格式分离，也了解了使用**什么工具**（文本编辑器）来实践。接下来来看看标记语言。

核心定义

标记语言是一种语法，它通过在纯文本中插入**结构化指令（标记）**，来清晰地定义内容的**意义和结构**，而非其视觉样式。

- **工作流程**:
  [内容与结构文件 (如 `.tex`， `.html`)] + [样式与格式文件 (如 `.sty`， `.css`)] → **构建** → [最终文档 (如 `.pdf`， `.html`)]
- **经典示例：HTML 与 CSS**
  - 网站所有者使用 **HTML** 专注于**内容**（标题、段落）。
  - 网页设计师使用 **CSS** 专注于**样式**（字体、颜色、布局）。

重要的标记语言

- **LaTeX**: 本章重点，物理科学领域高质量文档的黄金标准。
- **Markdown & reStructuredText**:
  - **特点**: 语法简单、干净、可读性强。
  - **应用**: GitHub 的默认语言 (Markdown)，Python 文档的标准 (reStructuredText)。
- **MathML & OpenMath**: LaTeX 数学功能的替代品，但使用率不高。

实践优势

- **结构化**: 可通过引用、链接等方式，将多个子文件组合成一个大型文档。
- **轻量化**: 图片等复杂内容通过**路径引用**而非直接嵌入，源文件保持纯文本，仅在最终“构建”时才组合进来。

我们已经了解了多种标记语言，现在将深入探讨在科学领域，应用最广泛、功能最强大的工具——LaTeX

LaTeX 文档的核心构成

一个 LaTeX 项目由多个纯文本文件组成，实现了内容与格式的彻底分离：

- **内容文件 (.tex)**: **唯一必需的文件**。作者在此撰写文章的实际内容。
- **类文件 (.cls)**: 定义文档的**整体结构和类型** (如 article , book)。
- **样式文件 (.sty)**: 控制具体的**视觉格式** (如字体、边距)。
- **参考文献 (.bib)**: 独立管理引用的文献。

最基础的命令： \documentclass

所有 LaTeX 文档都始于这个命令，它用来指定文档所使用的**类文件**。

- **通用语法**: `\commandname[options]{argument}`
- **具体应用**: `\documentclass[12pt, a4paper]{article}`
  - **{article}**：**参数 (Argument)**，指定文档类型为“文章”。
  - **[12pt, a4paper]**：**选项 (Options)**，指定字体大小和纸张类型。

内容与格式分离

这套机制的强大之处在于切换格式的便捷性：

- **场景**: 我写完了一篇 `article` 格式的论文，现在需要投稿给 **Elsevier** 出版社。
- **传统方法 (Word)**: 手动调整整个文档的字体、标题、间距、引用格式...
- **LaTeX 方法**:
  1. 下载 Elsevier 提供的类文件 `elsarticle.cls` 。
  2. 只需修改**一行代码**：将 `\documentclass{article}` 改为 `\documentclass{elsarticle}` 。
- **结论**: 文档的**全部内容保持不变**，而整体格式瞬间符合了出版社的要求。这完美展示了 WYSIWYM (“所见即所想”) 思想的效率与威力。

LaTeX 环境

上面已经了解了如何用 `\documentclass` 来定义文档的**类型和全局样式**。现在，我们将学习使用**“环境” (Environment)** 这个概念来划定和组织文档的**实际内容**。

核心概念：环境

- **定义**: 环境是 LaTeX 文档的基本元素，可以像俄罗斯套娃一样相互嵌套，用于包裹和定义不同部分的内容。
- **语法**: 所有环境都由一对命令来界定：

```
\begin{environment}
... 这里是环境内的内容 ...
\end{end{environment}
```

最重要的环境： `document`

- 这是顶层环境，所有希望在最终文档中**显示出来的内容**，都必须放置在 `\begin{document}` 和 `\end{document}` 之间。

- **文档结构**:
  - **前导区 (Preamble)**: `\documentclass` 和其他全局设置（如宏包、新命令定义）位于 `\begin{document}` 之前。
  - **正文区 (Body)**: 实际内容位于 `document` 环境之内。
  - **结束区**: `\end{document}` 之后的所有内容都会被忽略。

最小有效示例

下面是一个完整且有效的 LaTeX 文件：

```
\documentclass{article}
\begin{document}
Hello World!
\end{document}
```

这是一个完全有效的LaTeX文件。注意，其中不需要有关于字体、文档布局、页边距、页码或其他格式信息。现在还只是纯文本。为了将此文本以PDF来呈现，必须要构建文档。

构建文档：从纯文本到 PDF

我们已经了解了**纯文本内容**的 `.tex` 文件。现在，我们将执行“构建”步骤，将这份源文件编译成最终的、格式精美的PDF文档。

编译流程：执行命令

将 `.tex` 源文件转换为 `.pdf`，最常用和直接的方法是使用 `pdflatex` 命令。

- **命令 (在终端中运行):**

```
$ pdflatex hello.tex
```

- **编译过程**:
  1. `pdflatex` 程序读取并解析 `hello.tex` 文件。
  2. 它根据你在前导区定义的 `\documentclass{article}` **类文件**，自动应用所有预设的格式规则。
  3. 最终，它将格式化后的内容输出为一个 `hello.pdf` 文件。

(注：传统上也可以通过 `latex` 生成 `.dvi` 文件，再用 `dvipdf` 转换为 `.pdf`，但 `pdflatex` 一步到位，更为便捷。)

结果：格式的自动应用

如图 20-1 所示，生成了文档并只包含文本“Hello World!”。

Hello World!

图 20-1　Hello World!

图 20-1 Hello World!

尽管我们的源文件只有纯文本 `Hello World!`，但生成的PDF已经拥有了由 `article` 类所定义的标准页边距、字体和布局。

下一步：丰富文档内容

现在我们已经掌握了最基本的文档创建与构建流程，接下来将学习如何向文档中添加作者、标题等核心**元数据元素**。

LaTeX 元数据：定义文档信息

我们已经学会了如何创建并构建一个最基础的 LaTeX 文档。现在，我们将学习如何添加必要的元数据（如标题、作者、日期），让文档信息更加完整。

核心概念：元数据命令

- **定义**: 元数据是描述文档本身的信息。为了让这些信息（如标题）能在文档的多个位置（如页眉、标题页）被复用，它们需要在文档的**前导区 (Preamble)**，即 `\begin{document}` 之前进行定义。
- **常用命令**:
  - `\title{...}`：定义文档的标题。
  - `\author{...}`：定义作者信息。
  - `\date{...}`：定义日期，可留空或省略。

生成标题：`\maketitle` 命令

- **功能**: 这个命令会读取你在前导区定义的元数据，并根据当前文档类（`article`）的样式，自动生成一个格式化的标题块。
- **位置**: `\maketitle` 命令必须放置在 `document` 环境**内部**，通常紧跟在 `\begin{document}` 之后。

完整示例

以下代码展示了如何定义元数据并生成标题页。

```
% notes.tex (1) 注释: % 开头为注释。
\documentclass[11pt]{article} (2) 文档类: 创建一个 11pt 字体大小的 article 类型文档。

% --- 元数据定义（前导区）---
\author{Ada Augusta, Countess of Lovelace} (3)作者: 提供作者全名。
\title{Notes By the Translator Upon the Memoir: Sketch  标题: 提供完整的标题。
of the Analytical Engine Invented by Charles Babbage} (4)
\date{October, 1842} (5)日期: 提供可选的日期。

% --- 文档正文区 ---
\begin{document} (6)环境开始: 正文环境开始。
    \maketitle (7) 生成标题: 执行命令，将元数据渲染成标题。
\end{document} (8)环境结束: 正文环境结束。
```

最终效果

执行 \maketitle 命令后，会生成如下图所示的标准化标题：

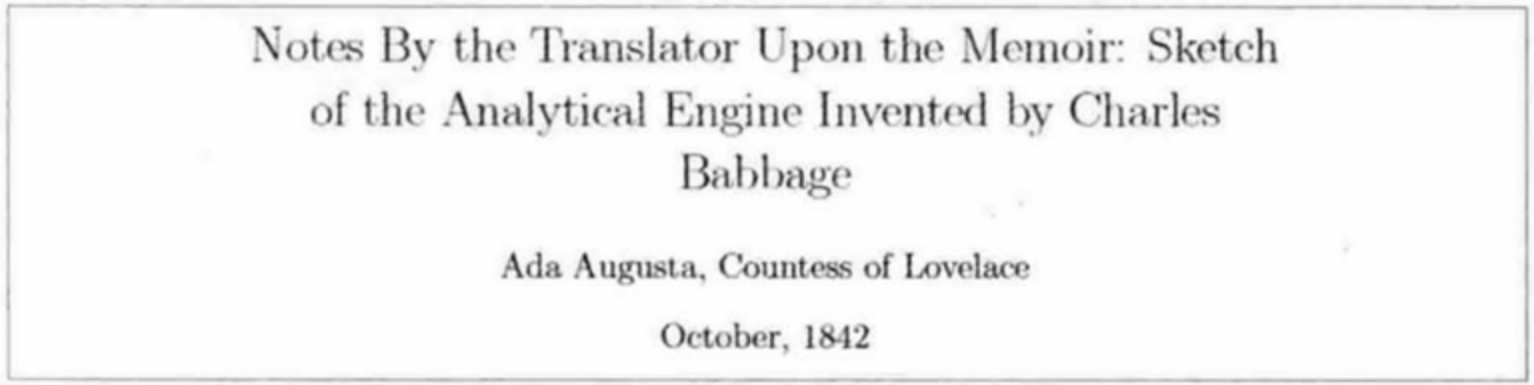

图 20-2　LaTeX 中的标题

我们已经定义了文档的元数据并生成了标题。现在，我们将学习如何使用结构化命令来组织文章的正文，并通过模块化的方式让长文档的管理变得更高效。

1. 创建文档结构：`\section` 与 `\subsection`

- **功能**: 在文档主体（document 环境）中，使用带标签的命令来声明结构化元素。
- **常用命令（article 文档类）**:
  - `\section{标题}`
  - `\subsection{子标题}`
  - `\subsubsection{次级子标题}`
  - `\paragraph{段落}`

- **示例**: Ada Lovelace 的笔记包含从 A 到 G 的七个部分，可以分别定义为 section 。

```
\begin{document}
\maketitle

\section{Note A}
\section{Note B}
\section{Note C}
...
\section{Note G}

\end{document}
```

2. 模块化写作：`\input` 命令

将所有内容都放在一个巨大的 `.tex` 文件中会变得难以编辑和管理。一个更聪明的做法是将每个 `section` 的内容保存在各自独立的 `.tex` 文件中。

- **核心命令**：`\input{filename}`
- **功能**：该命令会读取指定文件（如 `intro.tex`）中的所有文本和 LaTeX 语法，并将其完整地插入到当前位置。

- **模块化后的主文件示例**:

```
% --- 主文件: notes.tex ---
\documentclass{article}
... (元数据) ...

\begin{document}
\maketitle

\section{Note A}
\input{intro} % 插入 intro.tex 的所有内容

\section{Note B}
\input{storehouse} % 插入 storehouse.tex 的所有内容

...

\section{Note G}
\input{conclusion} % 插入 conclusion.tex 的所有内容

\end{document}
```

3. 模块化带来的优势

- **简化编辑**: 将长文档分解为多个独立的子文件（如 `intro.tex`， `conclusion.tex`），可以专注于单一部分的编辑，思路更清晰。
- **轻松重排**: 只需在主文件中调整 `\input{...}` 命令的顺序，即可改变整个文档的章节顺序，无需剪切和粘贴大量文本。
- **内容重用**: 可以创建标准化的文件（例如，一段通用的致谢 `acknowledgements.tex`），并在多篇不同的文章中方便地插入，确保一致性。

承接上文：从结构到内容细节

我们已经学会了如何组织文档的宏观结构（ section ）和模块化（ input ）。现在，我们将深入探讨 LaTeX 最具标志性的功能——高质量的数学公式排版，这也是科学文档写作的核心需求。

LaTeX 数学排版的强大之处

- **设定标准**: LaTeX 为技术文档的数学排版设立了行业标杆。
- **功能全面**: 它能优雅地处理：
  - 美观的希腊字母和拉丁字母。
  - 复杂的逻辑、数学符号数组。
  - 各种显示需求：与段落**内联**、**单独成行**、多行**对齐**、自动**编号**等。

核心模式一：内联数学模式 (Inline Math Mode)

- **功能**: 用于将数学符号或简短的方程式无缝嵌入到文本段落中。
- **语法**: 使用一对美元符号 `$` 将数学表达式包裹起来。

示例与语法解析

- **对应的 LaTeX 源码**:

```
The particular function whose integral the Difference Engine was constructed to
tabulate, is $\Delta^7u_x = 0$. The purpose which that engine has been specially
intended and adapted to fulfil, is the computation of nautical and astronomical
tables. The integral of $\Delta^7u_x = 0$ being $u_x = a+bx+cx^2+dx^3+ex^4+fx^5+gx^6$,
the constants a, b, c, &c. are represented on the seven columns of discs,
of which the engine consists.
```

- **语法分解**:
  - `$...$`：指示内联数学公式的**开始**和**结束**。
  - `\Delta`：通过**反斜杠** `\` + 名称来输入希腊字母 (Δ)。
  - `^`：用**插入符号** `^` 表示**上标** (例如 `^7`)。
  - `_`：用**下划线** `_` 表示**下标** (例如 `u_x`)。

目标文本:

The particular function whose integral the Difference Engine was constructed to tabulate, is $\Delta^7 u_x = 0$. The purpose which that engine has been specially intended and adapted to fulfil, is the computation of nautical and astronomical tables. The integral of $\Delta^7 u_x = 0$ being $u_z = a + bx + cx^2 + dx^3 + ex^4 + fx^5 + gx^6$, the constants a, b, c, etc. are represented on the seven columns of discs, of which the engine consists.

图 20-3 内联公式

核心模式二：陈列数学模式

我们已经掌握了在段落中嵌入数学公式的**内联模式**（`$...$`）。然而，对于更重要或更复杂的公式，需要将其从文本中分离出来，使其在单独的行上居中显示，并自动编号。这就是陈列数学模式的作用。

`equation` 环境：单个公式的展示

- **功能**: 用于在单独的行上展示一个**带自动编号**的居中公式。
- **语法**:

```
\begin{equation}
    ... 你的数学公式 ...
\end{equation}
```

- **特性**:
  1. **自动分离**: 将公式从文本中抽出，独立成行。
  2. **自动居中**: 完美地将公式水平居中。
  3. **自动编号**: 在公式右侧自动添加一个序号，如 (1)。

示例与效果

- **LaTeX 源码**:

```latex
\begin{equation}
    F(x, y, z, \log x, \sin y, x^p)
\end{equation}
```

- **最终渲染效果**:

这样就将公式从文本中抽出并自动给定了一个公式号，如图 20-4 所示。

The following is a more complicated example of the manner in which the engine would compute a trigonometrical function containing variables. To multiply

$$F(x, y, z, \log x, \sin y, x^p) \tag{1}$$

which is, it will be observed, a function of all other functions of any number of quantities.

图 20-4　公式环境

我们已经学会了使用 `\input` 命令将文本内容模块化。这一思想同样适用于非文本元素，如表格和图形。将这些元素保存在独立的外部文件中，不仅能简化主文档的操作，也便于在不同文档（如文章和演示文稿）中重复使用。

核心环境： `figure`

在 LaTeX 中，我们使用 `figure` 环境来插入、定位并描述图形。作者只需在主文档中关注与排版相关的元素，如**位置**、**说明**和**相对大小**。

完整示例与语法解析

以下代码展示了如何在文档中插入一个图形，并为其添加标题和可供引用的标签。

- **LaTeX 源码**:

```
\begin{figure}[htbp] (1)
    \begin{center} (2)
        \includegraphics[width=0.5\textwidth]{var_diagram} (3)
    \end{center}
    \caption{Any set of columns on which numbers are inscribed, represents
    merely a general function of the several quantities, until the special
    function have been impressed by means of the Operation and
    Variable-cards.} (4)
    \label{fig:var_diagram} (5)
\end{find{figure}
```

- **语法分解**:
  1. **\begin{figure}[htbp]**：开始 figure 环境。方括号内的选项 [htbp] 建议了图形的**浮动位置**： h (此处)， t (页顶)， b (页底)， p (独立页面)。
  2. **\begin{center}**：将环境内的元素（此处为图形）水平居中。
  3. **\includegraphics[...]**：**核心命令**，用于插入图像文件。
     - {var_diagram} 是图像的文件名。
     - [width=0.5\textwidth] 是一个选项，指定图形的宽度应为当前文本宽度的 50%。
  4. **\caption{...}**：添加图形的说明。LaTeX 会自动为其编号（例如 “Figure 1”）。
  5. **\label{...}**：为图形添加一个**唯一的标签**。这个标签是实现交叉引用的关键，后续可以在文本中通过这个标签引用该图。

最终效果

代码执行后，会生成一个包含编号、说明和图形的完整元素。

该语法的结果如图 20-6 所示。文档中插入了图像，根据代码的要求安排了图形的编号、大小和标题。

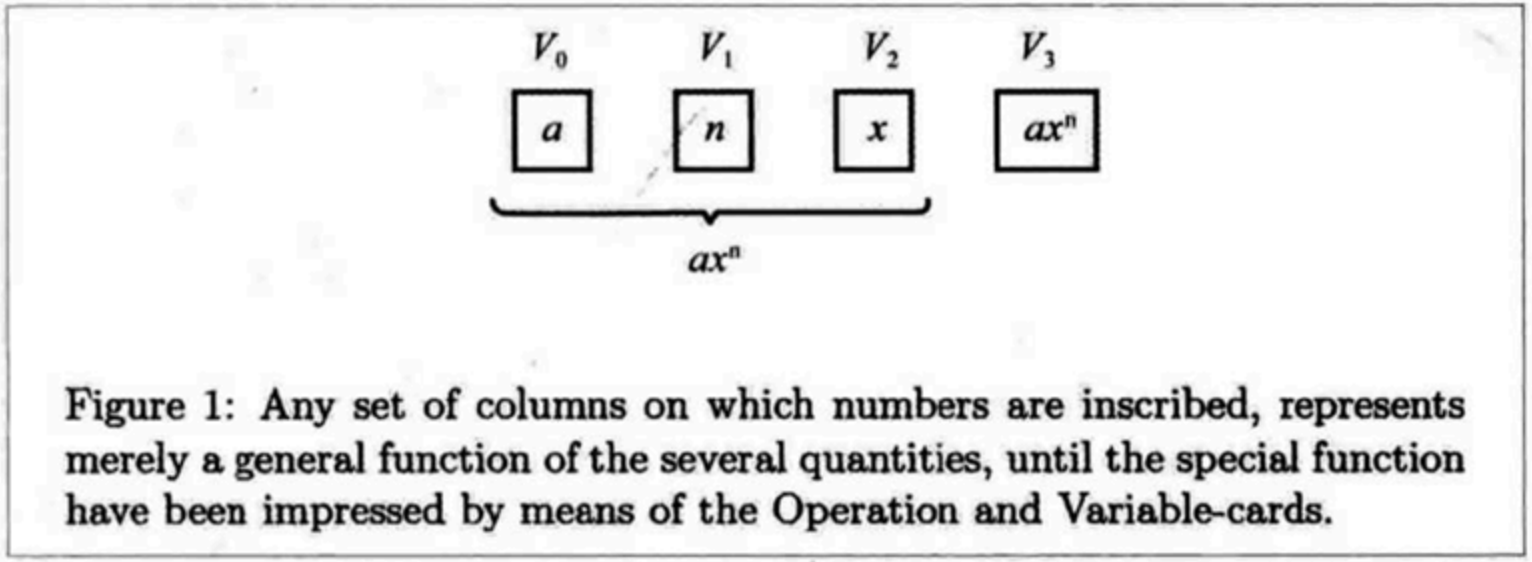

图 20-6　LaTeX 中的标签

这种干净、可自定义的内部**交叉引用**语法是 LaTeX 的一个标志性特性，它能极大地提升写作效率。接下来详细介绍其工作方式。

内部引用 (Cross-Referencing)

上一节我们在 figure 环境中使用了 \label{fig:var_diagram} 命令。现在，我们将揭示这个命令与 \ref{} 命令如何协同工作，构成 LaTeX 中最强大、最高效的功能之一：**内部引用**。

核心理念：告别手动编号

- **痛点**: 在传统写作中，如果你在文章开头新增了一个章节或图片，那么后面所有章节和图片的编号都需要手动修改，引用处也要逐一检查，极其繁琐且容易出错。
- **LaTeX 解决方案**: 使用**有意义的标签**来引用，而非写死编号。LaTeX 会在**构建时**自动处理编号和匹配。

工作机制： \label 与 \ref

1. **标记 (\label)**: 在任何可以编号的元素（ section , figure , table , equation ）之后，使用 \label{一个独一无二的标签名} 来为其命名。
2. **引用 (\ref)**: 在文档的任何地方，使用 \ref{之前定义的标签名} 来插入该元素对应的**编号**。

实践案例：Ada 在 Note D 中引用 Note B

假设 Ada 的笔记分为多个文件， `storehouse.tex` 包含了 Note B 的内容，而 `example.tex` 包含了 Note D 的内容。

- **第一步：在 `storehouse.tex` (Note B) 中设置标签**

```
% storehouse.tex
\section{Note B}
\label{sec:storehouse} % (2) 为 Note B 设置一个有意义的标签
That portion of the Analytical Engine here alluded to is called the
storehouse...
```

  - **命名约定**: 使用 `sec:` 作为章节标签的前缀是一个好习惯（同理，图形用 `fig:`，表格用 `tab:`）。这并非强制，但能极大提高代码的可读性。

- **第二步：在 `example.tex` (Note D) 中使用引用**

```
% example.tex
We have represented the solution of these two equations below, with
every detail, in a diagram similar to those used in Note
\ref{sec:storehouse}; ... % (2) 使用标签名引用 Note B
```

- **自动化**: 当 Ada 构建整个文档时，LaTeX 会自动将 `\ref{sec:storehouse}` 替换为 Note B 对应的**实际章节号**（例如 "B" 或 "2.2"）。即使未来 Note B 的顺序被移动到 Note E 之后，只需重新编译，这个引用就会自动更新，无需任何手动修改。

我们已经掌握了如何引用文档**内部**的元素。下一节将介绍如何处理另一种重要的引用类型：**参考文献（引证）**，其处理方式略有不同。

20.3.2 参考文献 (References & Citations)

我们已经掌握了如何引用文档**内部**的章节和图表。现在，我们将学习 LaTeX 更为强大的功能：自动化管理和格式化**外部参考文献**，彻底解决科学写作中的一大痛点。

LaTeX 通过 BibTeX (或 BibLaTeX) 系统来管理参考文献。其核心是一个数据库文件，后缀为 `.bib` 。

- **`.bib` 文件**: 这是一个纯文本文件，用于存储所有参考文献的元数据信息。
- **条目 (Entry) 结构**: 每个参考文献都是一个独立的条目，有唯一的“引用键”(Citation Key)。

```
% refs.bib
@article{menabrea_sketch_1842,
  series    = {Scientific Memoirs},
  title     = {Sketch of the Analytical Engine Invented by Charles Babbage},
  volume    = {3},
  journal   = {Taylor's Scientific Memoirs},
  author    = {Menabrea, L.F.},
  month     = oct,
  year      = {1842},
  pages     = {666--731}
##‘menabrea_sketch_1842’ 就是这个条目的引用键。
}
```

三步工作流：引用、声明、构建

要在正文中使用该引用并生成参考文献列表，Ada 必须完成三件事：

第一步：在正文中引用（.tex 文件）

使用 `\cite{引用键}` 命令在需要的位置插入引文标记。

```
% intro.tex
... explained in the Memoir itself \cite{menabrea_sketch_1842}) the law of development ...
```

第二步：在文档末尾声明样式和数据库

在 \end{document} 命令之前，添加两行代码来指定参考文献的样式和 .bib 文件的位置。

```
% notes.tex (主文件)
...
\input{conclusion}

\bibliographystyle{plain} % 指定参考文献的样式 (如 plain, MLA, Chicago)
\bibliography{refs}       % 指向 .bib 文件 (此处为 refs.bib)

\end{document}
```

第三步：特殊的构建（编译）过程

由于引用和文献列表是分开处理的，需要一个特殊的多步编译流程来将它们关联起来：

```
$ latex notes       # 第一次运行，LaTeX 记录下需要引用的条目
$ bibtex notes      # BibTeX 读取记录，并根据样式生成格式化的参考文献列表
$ latex notes       # 第二次运行，LaTeX 将引文标记 ([1]) 插入正文
$ latex notes       # 第三次运行，确保所有交叉引用都正确无误
```

经过上述步骤

正文中: \cite{menabrea_sketch_1842} 会被自动替换为 [1]。

文档末尾: LaTeX 会自动生成一个标题为 “References” 的章节，并包含格式化好的参考文献条目。

科学家再也无需手动处理不同期刊（如 MLA 风格）参考文献中的标点符号、斜体和顺序。只需更换 \bibliographystyle 的样式，LaTeX 和 BibTeX 就能自动完成所有格式化工作。

引用管理 (Citation Management)

我们已经了解了 BibTeX 的强大功能，但手动创建和维护 `.bib` 文件本身是一项枯燥且耗时的工作。本节将介绍如何使用**引用管理器**来完全自动化这一过程。

核心问题

为每一篇参考文献手动输入作者、标题、期刊、卷号、日期等元数据，既繁琐又容易出错。

引用管理器是专门用于收集、组织和引用参考文献的软件工具。它们的核心功能是：

1. **自动收集**：帮助研究人员自动抓取期刊文章和其他文献的完整元数据。
2. **高效组织**：提供一个可视化的界面来管理所有的参考文献资源。
3. **一键导出**：可以轻松地将整个参考文献库或特定条目导出为格式规范的 **`.bib` 文件**，直接供 LaTeX 使用。

主流的开源引用管理器

- **BibDesk EndNote JabRef Mendeley RefWorks Zotero** 等

最终工作流

通过使用这些工具，研究人员可以将精力集中在阅读和写作上，而将参考文献的格式化与管理工作完全交给自动化流程，从而极大提升科研效率。

我们已经了解了如何高效管理和引用传统的**学术文献**。然而，为了实现真正的**可重现科学 (Reproducible Science)**，我们还必须能够引用构成研究基础的另外两个核心元素：**代码和数据**。

- 传统的 BibTeX 格式主要为书籍（`@book`）和文章（`@article`）设计，缺乏专门用于引用代码或数据集的元数据格式。

数字对象标识符 (DOI)

DOI (Digital Object Identifier) 提供了一个统一的标准来持久化地标识和引用任何数字对象，包括软件和数据。

- **类比理解**:
  - **图书** 使用 **ISBN** 来唯一区分。
  - **Git 的每次提交** 使用**哈希值 (Hash)** 来唯一区分。
  - **数据和软件** 则可以使用 **DOI** 来唯一区分和引用。

如何为你的代码/数据获取 DOI?

- 存在专门的**线上归档服务 (Archiving Services)**，它们可以为你的软件或数据集生成一个 DOI。
- 其中一些服务甚至是**免费和开源**的。

总结

对于任何旨在实现完全可重现的出版物，都应该为其分析代码和所用数据获取 DOI，并在参考文献中进行引用。虽然具体如何使用这些归档服务超出了本章的范围，但了解并使用它们是现代科研工作的重要一环。

出版的成本与效率

- **出版是科学事业的“货币”**: 它是分享和评判科研工作的传统方式。
- **传统方法的痛点**: 科学家花费大量时间在格式调整而非内容创作上。
- **本章目标**: 介绍以 LaTeX 为核心的一系列工具和理念，用以生成高质量的科学文档，从而将精力回归内容本身。

核心理念回顾

- **内容与格式分离**: 标记语言的根本优势在于将**写什么**（内容）与**长什么样**（格式）彻底分开。
- **版本控制**: 基于纯文本的标记语言文档，天然易于进行版本控制。
- **LaTeX 的威力**: 在众多标记语言中，LaTeX 是一个功能尤其强大的科学出版工具。

在 LaTeX 的环境中，我们现在应该知道如何执行以下工作：

- **基础**: 生成简单的文档并定义其**结构**（`\section`）。
- **数学**: 添加**内联**（`$..$`）和**陈列**（`equation`）数学公式。
- **图形**: **包含**图形（`figure`）并对其进行**内部引用**（`\label`，`\ref`）。
- **文献**: **引用**参考文献（`\cite`）并**自动创建**文献列表（`BibTeX`）。

下一步：深入学习

以下是两个广受推荐的在线资源：

- 《**The Not So Short Introduction to LaTeX2ε**》：作者 Tobias Oetiker et al.
- **Tex-LaTeX Stack Exchange 网站**: 活跃的问答社区。