排版、印刷和网络出版

- 高德纳教授其人其事
- TEX 技术内幕一瞥
- 基于 TEX的宏软件包：LATEX、ConTEXt、Texinfo、Ω
- TEX 的许可证及其解读
- PostScript
- SGML/XML 与排版技术的结合
- 商业模式
- 出版行业在中国

题解

本章中提到的排版服务，不是指利用 Microsoft 的 Word，金山的 WPS，Adobe 的 Photoshop、PageMaker 或者 FrameMaker 之类的专有软件提供的排版服务，因为这类软件的灵活性和稳定性无论如何都无从与基于自由软件的排版排版软件相提并论。自由软件社团广泛使用的是 TEX 排版系统和 PostScript 页面描述语言。

TEX 排版系统由高德纳教授(Prof. Donald E. Knuth)发明，他发明该软件包后将版权置于公众领域（public domain），后立即为科技文章和出版物（特别是数学论文和书籍）的作者和出版社广泛接受。美国的 TEX 用户俱乐部的人数最多，早在 1990 年前就超过了 3,000人。德国的 TEX 用户俱乐部通过“雷曼专业书店”（Lehmanns Fachbuchhandlung）和“但丁公司”（Dante e. V.）的长期努力，现在其正式注册人数超过三千名，加上非注册俱乐部人员，群体人数按照保守的估计也在五千人以上，成为欧洲自由软件社团中的一大群体。除了欧美两大地区，TEX 的用户群体遍布全球各地，仅在印度一国，根据印度 TEX 用户协会的不完全统计，2002 年度就有超过八千名程序员以专业排版服务为生，可以说其规模大得相当惊人。

PostScript (以及派生的 PDF 技术)由美国 Adobe System 公司的两位创始人 John Warnock 和 Chuck Geschke 于二十世纪八十年代中期提出并实现。与绝大多数专有软件公司不同，由于该公司一直采取了一种独特的开放的商业策略（下面即将介绍），因此该项技术在自由软件社团里也赢得了相当广泛的认可。

由于这两项技术都已经发展得非常成熟健全，而且这两项技术被自由软件社团的黑客们巧妙地融合起来，创造出了全球学术界和印刷工业界最为广泛认可的事实上的标准，成为功能强大无比的排版业解决方案，自由软件社团在此基础上创造了一个庞大的排版服务行业。

在万维网联盟（W3C）于 2001 年 3月 15日公布了 XML 规范之后，排版、印刷和出版的概念都开始发生本质的变化，一个明显的发展趋势是 TEX 和 PostScript 技术开始被用作后台格式化引擎，而 XML 技术则广泛地运用于源文件的计算机电子化和网络化管理中，网络数据库管理技术开始进入出版领域，新的一些技术甚至开始使用 XML 技术制定样式表和格式化对象来直接格式化文档。可以预见，未来的十年将是传统出版行业脱胎换骨、完成新一轮技术革新的时期。

高德纳教授其人其事

在谈论 TEX 系统之前，我们先介绍高德纳教授。高教授是美国科学院院士和美国工程院院士，1938 年 1 月 10 日出生于美国威斯康星州的密尔沃基（Milwaukee），1956 年进入凯思工学院学习物理，1960 年毕业后进入加州理工学院研究生院，1963 年获得博士学位，并留校工作到 1968 年，1968 年转到斯坦福大学任教，直到 1992年退休。

高教授既是数学家，又是计算机科学家。纵观高德纳教授的学术生涯，他一生致力的众多研究领域中，两个主要项目占有显著地位，一个是编写经典著作《计算机编程的艺术》（*The Art of Computer Programming*），另一个就是发明了排版系统 TEX。《计算机编程的艺术》是计算机科学中的经典著作之一，其地位可以与逻辑学中由罗素和怀特海合著的《数学原理》相比较。这一著作的写作创意萌发于他二十世纪六十年代初期读博士期间，原计划共写作七卷：

- 第一卷《基础算法》(Fundamental Algorithms)，第一版于 1968年出版。
- 第二卷《半数字化算法》(Seminumerical Algorithms)，第一版于 1969年出版。
- 第三卷 《排序与搜索》(Sorting and Searching)，第一版于 1973年出版。
- 第四卷《组合算法》仍在创作之中。

  从目前的进展情况来看，其内容很丰富，第四卷将可能至少分成三个分册出版：

  - Volume 4A，枚举与回溯（Enumeration and Backtracking）。
  - Volume 4B，图与网络算法 (Graph and Network Algorithms)。
  - Volume 4C 或者 4D，优化与递归。
- 第五卷《句型算法》(Syntactic Algorithms)。

  这一卷仍在准备过程中，计划于 2009 年完成。

- 第六卷《上下文无关语言的理论》，尚未动笔。
- 第七卷《编译技巧》(Compiler techniques)，尚未动笔。

尽管这部著作取得的成就是世界公认的，但是《计算机编程的艺术》却是一部尚未完成的著作，作者目前仍然在继续其创作。为什么这部著作前面写作进度很快，但是后面的写作却长期停滞不前呢？

原来，高教授写作的这部著作主要讨论计算机软件设计中的数据结构和算法的问题。数据结构使用数学理论中基础的集合论构造，而且这一概念的提出深受布尔巴基学派和结构主义关系的影响，从数据结构的一般定义就可以见一斑：

一个数据结构 B 是指一个序偶 $B =< A, R >$，其中的 A 是由结点组成的一个非空有集，R 是定义在 A 上的关系的非空有限集合。

算法理论实际上就是数学的一个分支。所谓算法，简单地讲，就是求解问题的方法和步骤。有些问题是没有解的，有些问题有解，还有些问题本身是否有解是不可判定的。研究问题求解的可判定性是数理逻辑中的一个重大课题。而对于可以判定求解的问题，又存在着使用不同的算法求解，但相应的计算工作量或者计算效率却可能因算法不同而大相径庭，研究这些问题的学问称为算法复杂性分析。这些问题都是数学问题，所以实际上《计算机编程的艺术》可以视为一部标准的数学著作。[1]

数学论文和著作中往往充满了大量的数学符号、上标与下标、数学公式和图表等等这些当时计算机处理起来非常麻烦的排版对象。当时计算机在出版界应用还不像今天这么广泛，软件的水平不高，书稿交付出版商之后，高教授在审校清样时痛感当时的排版技术实在太落后了，计算机科学理论当时已经有相当大的进展，而排版技术却仍旧停留在中国古代印刷术的水平。为了让计算机能够漂亮地处理数学文章，他停下了这部著作的写作，转而专心致志来设计一套排版系统，以彻底解决这一问题。

在接下来整整九个年头里，高教授潜心研究，终于发明了 TEX 排版系统和 METAFONT 字库设计软件，这两个软件一发布，立即对整个西文印刷行业产生了革命性的影响，这两个软件包使高教授赢得了 1986 年度的 ACM 软件系统奖。《计算机编程的艺术》和 TEX 排版软件系统为高教授赢得了世界级的声誉。

[1] 重视算法研究是中国古代数学的优秀传统，中国古代的数学没有走上古希腊那种公理化道路，但是这不能说中国古代数学中没有系统科学的基础，实际上，中国古代哲学的系统性非常强，《易经》就可以视为系统科学的原胚。

TEX 技术内幕一瞥

TEX 的源代码是利用当时流行的系统设计语言 Pascal 编写的，但是高德纳教授没有直接使用 Pascal 语言来建立排版系统的数学模型，而是另外创作了一套称为 WEB 的系统。注意，这个 WEB 与今天互联网上 WWW 中的 Web 没有任何关系，仅仅是名字碰巧相同而已（注意，高教授的 WEB 中三个字母通常全部用大写字母）。由于 WEB 系统只是一个描述排版系统的数学模型，因此该模型表达的算法可以使用其他语言来实现[2]，而且在 TEX 的发展历史上的确出现了使用其他语言实现 WEB 的 TEX 系统，目前通用的 WEB2C 就是这么一个例子：先是 Tomas Rokicki 于 1987 年在 Howard Trickey 和 Pavel Curtis 工作的基础上实现了第一个 TEX–to–C 的系统，工作平台是 UNIX，Tim Morgan 遂即担任了系统维护员，在他担任系统维护员期间，这一系统的名称变更为 WEB-to-C。1990 年，Karl Berry 接手这一工作，在他维护期间内，又有一打以上的贡献者参与完善了这一系统，直到 1997 年，系统维护的接力棒交给了 Olaf Weber，在 Olaf 而维护期间，版本号码上升到了 WEB2C 7.3。之后英国人 Sebastuian Rahtz 开始维护这一系统，直到今天的 WEB2C 7.3.7 (打包在 2002 年 6 月发布的 TEX Live CD 7 中)。

WEB2C 中所谓的 TEX 系统实际上包括一组程序：TEX 本身、METAFONT、METAPOST、BIBTEX等 25 个模块程序，尽管这一系统的规模非常庞大，但是设计的思想却非常直观，这里我们岔开来谈一谈。

高德纳教授非常喜爱中国文化，尽管我们不知道他在发明 TEX 系统时是否受到中国古代印刷术的启发，但是我们可以肯定地说 TEX 系统的原理与中国的活字印刷术是极为相似的。

印刷术是中国古代四大发明中的一项，由宋朝的毕升首创。根据史书记载，在毕升之前，印刷书籍采用雕版技术，每一页书需要一块木板，如果一本书有一百页，就需要工人制作一百块雕版，有了雕版，再将雕版上刷上油墨，并把纸张覆盖在雕版上拓印（类似我们今天在文件上盖章），印好再凉干、切割、装订。一旦书籍印完不用再版时，所有这一百块雕版便统统作废。由于木材受季节变化影响较大，时间一长，容易出现变形或者裂纹，因此雕版的保存期不是很长。大的印刷作坊往往还为储存这些堆积如山的雕版而头疼。

利用木材制作雕版是件非常复杂的事情，因为仅从木材的选材、切割、抛光等一套流程就颇费工夫，而且在工人刻字时必须雕上反字，这样才能被拓印为正字，一个字的一个笔画搞错就需要重来，整块木板便浪费了。

毕升的创举在于使用活字，由于汉语是由几千个常用汉字组成，因此只要将汉字当作一个又一个的标准零件做出来，那么就可以利用这些标准零件来组

[2] 除了利用 C 编写的 WEB2C, 还有利用 C++ 编写的 Ω 工程，以及笔者启动利用 Scheme 编写的 NeoTEX 系统，其中采用了 WEB2SCM，本章后面还将谈到。

装出一个雕版来。不同的雕版可以使用相同的一批活字来拼装，这样一来就不用消耗大量的木材和人力了。用现代数学语言来说，**毕升将变量带进了印刷业**。

下面是笔者于 2000 年访问韩国汉城时，在当地的博物馆拍摄的一张照片，这幅图片重现了古代印刷厂工人工作的情景。朝鲜古代的印刷术是从中国传入的，因此我们可以利用这幅图片联想中国古代印刷作坊的情形。值得一提的是，毕升使用的是用火烧制的泥活字，后人再改用木活字，而朝鲜是首次利用金属制造活字的国家。金属活字虽然造价比木活字昂贵，但是活字的使用寿命大为提高，而且金属活字嵌入到版面夹具之后不易松动，可以提高印刷质量。金属活字在印刷工业界一直使用到上个世纪。因此朝鲜人民也为人类印刷术的进步作出了很大贡献。

图 1 古朝鲜的印刷作坊

毕升发明活字印刷术的几百年后，欧洲德国的古腾堡（Gutenberg）才开始采用这项技术，古腾堡的创新在于使用印刷机替代了原来单一而固定的版式，同一台印刷机上可以制成不同的版式，因此大大提高了印刷的效率，印刷业告别了手工业进入了现代工业领域。他开创的印刷事业对于欧洲文艺复兴贡献很大，当今有一项很有名的工程称为“古腾堡计划”（The Gutenberg Project）就是以他的名字而命名，以纪念他对出版业作出的巨大贡献。该工程的目的就是将人类的经典著作全部制做为电子文档供大众使用。

汉字是方块形状的，每个汉字都放在一个格子中，这样一连串的汉字便组成词语和句子，古汉语没有标点符号，因此不需要断句，每一垂直竖行由多个

格子拼装起来，每一段由多个行从右到左拼装起来，然后多个段拼装起来组成一页，也就是一块雕版。

TEX 的原理也是这样的，只是它是面向西文处理，每一个字母放在一个称为“盒子”（box）的区域中，然后由多个盒子串接起来组成一个水平横行，一个水平的行又被视为一个更长的“盒子”，多个这样的“盒子”垂直拼装成为一段，一个段又可以被视为一个“盒子”，多个段组成一个完整的页面。“盒子”之间的距离可以填充“glue”（称为“胶水”），将盒字粘接起来，通过调整盒子之间胶水的值，就可以调整字符在页面上输出的位置。

排版系统中不可避免地需要使用各种字体。METAFONT 就是设计各种字体的工具，METAFONT 与 TEX 中的思想是一脉相承的，字体一旦用 METAFONT 生成，就可以当作“盒子”嵌进 TEX 文件中。有趣的是，METAFONT 不仅仅可以用来设计字体，还可以用来生成几何图形。

与人们的直觉相反，尽管西文的字符集比较小，英文字母表中只有 26 对大写和小写字母，加上其他字符，总共只有几百个，但是从计算模型的角度看，西文处理的难度反而要比汉字处理要大得多，西文字母的高度和宽度并不像汉字这样整齐划一，例如两个 f 在一起时，他们占有的宽度就少于两个 f 字母宽度之和，这在西文里面称为 kerning。另外，西文对于一个单词换行时如何折断字符串有严格的规定，例如，英文中 `software` 这个单词在一行的末尾写不下时，必须拆开成为 `soft-ware` 两个部分，其中，`soft-` 放在第一行的末尾，`ware` 放在第二行的开始。

人的姓氏与名字都不能随便拆开，否则便被认为是拙劣的排版，而且西方人的姓与名之间不能拆开成为两行来排版，必须放在一行中。很多人还有头衔，当头衔与姓名一起出现在文章中时，头衔和姓名必须作为一个整体来看待，否则就不符合西文的排版习惯。

至于处理数学符号、公式、表格等时，排版规则更加复杂。高教授的创作之一就是使用 WEB 系统先解决这一整套规则的数学模型，WEB 使用介乎数学语言和计算机编程语言之间的方法描述这一整套的排版规则，然后开发了针对 WEB 系统的用 Pascal 语言实现的软件系统，可以将含有排版格式指令的源文件编译转换生成为与硬件平台无关的文件格式，即 DVI 格式的文件。这种 DVI 文件则可以输出给任何支持它的设备：打印机或者屏幕。所以，WEB 系统相当于一个模板，它描述排版的标准动作(即排版的规则)，但是它不执行这些动作，而是交给 TEX 编译器的代码去执行这些动作。用泛系方法论的语言讲，就是高教授在多变的源文件和不变的 TEX 编译器之间构建了一个相对少变的中间层，从而建立了一种泛对称关系。

TEX系统的代码在 1982 年就已经稳定下来了，这一版本称为 TEX82。高教授本人无意再修改这一版本中含有的代码，但是注意，对于 WEB 系统，高教授则希望能不断地对它进行改进。实际上，对 TEX82 系统采用的 WEB 算法的改

进工作一直在进行着，到了 1990 年，所积累的算法改进工作已经相当多，高教授觉得是发布新一代 WEB 系统的时候了，这一次基于新 WEB 系统的代码却不再冠以 TEX 的称呼，而是称为 NTS。目前围绕 NTS进行的开发工作仍在进行，其中以捷克裔黑客司寇林 (Karel Skoupý)用 Java 开发的系统最有名。

在设计技巧上，WEB 系统的接口（参数的数量、类型、顺序等）一开始就尽量照顾到 Pascal 语言的特点，因此，WEB 程序模块与使用 Pascal 语言实现的 TEX 编译器之间的接口是相当清楚。尽管如此，由于系统中众多泛对称性的存在，软件作者与用户之间的信息不对称性的鸿沟便出现了。直接阅读高德纳教授编写的 TEX 代码对于初学者是非常晦涩难懂的（尽管他本人以此而自豪）。为了追求完美，便于其他人掌握这一系统，他编写了五卷宏篇巨著来解释说明 TEX 系统应该怎样使用，以及这套系统是怎样创作出来的。

这五卷著作全部由 Addison–Wesley 公司于 1986 年出版，各卷书名分别是：

- 第一卷《A. The TEXbook》，483 页。

  详细介绍这一系统的使用方法，讲解的方式是启发式的，可读性很好，一切从基本的东西开始，一直深入到所有的细节。初学者可以在学完之后达到入门水平。即使是 TEX 方面的专家，每一次阅读这本书都可以温故而知新。

- 第二卷《B. TEX: The Program》，600 页。

  详细介绍 TEX 系统是怎样设计的，深入地说明了排版系统的数学模型。

- 第三卷《C. The METAFONTbook》，361 页。

  详细介绍了 METAFONT的使用方法。行文风格与第一卷如出一辙。

- 第四卷《D. METAFONT: The Program》，566 页。

  详细介绍了 METAFONT 系统是怎样设计的，深入地说明了 METAFONT 系统的数学模型。行文风格如第二卷。

- 第五卷《E. Computer Modern Typeface》，588 页。

  专门讨论 TEX 系统所附带字库的字形设计技术。

整部著作的篇幅总共有 2,598 页！（如果翻译出来成中文，长度很可能也会超过两千页。）

TEX 本质上是一个编译器，它的所作所为就是将人可以阅读的 ASCII 文本文件读入，然后按照一套规则输出生成一棵树，并且沿着树的分叉在各个叶子

中放进利用 METAFONT 制作的字符或者图形，并将最终的文件以 DVI 文件格式输出。

尽管高德纳教授声称 TEX 系统已经没有 bug，而且给任何一位报告发现了 bug 的人给予奖励，虽然他一直在提高奖金的金额（目前作者开出的奖金金额是每个错误 327.68 美元）。但是 TEX 系统的代码质量的确已经非常成熟稳定，加上能够理解 TEX 和 WEB 源代码的人极少，现在很难有人从中挑剔出暇疵了(不过，如果读者有兴趣，仍然可以试一试，或许你真的可以发现一条虫子，得到高教授的犒赏！)。

根据笔者的体会，要成为一名使用 TEX 排版的专家，至少需要一年的时间，因为仅仅阅读第一卷和第三卷著作，上机动手操作并能融会贯通，就会耗时很长。TEX 系统的最基本的命令（其专门术语成为“控制序列”，英文为 control sequence）约有 300 个左右，加上 600 个左右的扩展命令构成的 Plain TEX 系统，总共大约有 900 条控制序列。在 TEX 的作者与用户之间，形成了一道巨高的门槛(也就是本书一再说明的信息或者知识的不对称性。)

TEX 是无比灵活的系统，尽管系统是这么复杂，但是学习起来却很有趣。我们在“一、百、万”工程的“黑客道”培训计划中设置了一门课程，专门讲授 TEX 的排版，这一课程按照“黑客道与教育传统的复兴”一章中阐述的“目标牵引、内化成瘾”的教育学原理和方法，以学员自己动手实践为中心，让学员随时看到自己的排版结果，以激发和调动学员自身的学习兴趣，循序渐进地讲解 TEX 排版技术，学员可以在较短的时间内成为一名合格的“排版工程师”，自开设以来深得学员好评。

对于开发人员，要想跨越这一门槛，无疑在毅力和智力两方面都是一个巨大的挑战。仅理解 TEX 的程序，高教授所著第二卷的书中就含有 1,380 个小节！从头阅读这一著作难免一头雾水。根据笔者的体会，理解这一系统的关键在于彻底摸清“盒子”这一概念的数据结构和操作这一数据结构的算法，因此从 TFM 文件的规范入手，逐步理解这一系统的各个算法不失为一条捷径，或者说是掌握这一系统原理的要诀。

如果你仔细观察所有天才程序设计大师所创作的优秀软件包，它们都有一个共同点：就是底层结构非常稳定可靠，但是系统本身具有几乎无穷的可扩展性。斯托曼博士的文字编辑器 GNU Emacs 通过揉和 C 和 Lisp 达到了这一水平，底层的一些模块使用 C 语言编写，而文字编辑模块基本上使用 Lisp（Emacs Lisp 是 Lisp 语言的一个实现版本），正如本书已经介绍过的，可以使用 Lisp 语言可以编写出能生成程序的程序。因此，GNU Emacs 系统通过编程语言得到了几乎无穷的可扩展性。Miguel de Icaza 开发的 GNOME 在用户界面设计中采用了 CORBA 结构，CORBA 采用了对象请求代理（ORB），这一 ORB 相当于软件总线的概念，X Window 系统提供的函数统统被定制为对象库，而应用程序则可以使用不同的编程语言编写，从而在可扩展性上也

达到了一个相当高的水平。同样的，正如上面介绍的，通过 WEB 系统，高教授的 TEX 系统也是可以充分定制的，其扩展性也到达了一个很高的水平。

基于 TEX的宏软件包：LATEX、ConTEXt、Texinfo、Ω

LATEX

原始的 TEX 系统庞大而复杂，使得一般的用户望而生畏，这一点高教授自己也承认，他曾说：

"I never expected TEX to be the universal thing that people would turn to for the quick–and–dirty stuff. I always thought of it as something that you turned to if you cared enough to send the very best."

(我从来没有指望过 TEX 成为一种万能的工具，去完成各种方式的任务。我一直将它视为一种你可以充分优化以达最佳状态的东西。)

普通用户却可以利用一些宏软件包来模块化地使用这一系统。为了便于使用，目前已经有各种宏软件包出现。Leslie Lamport 和其他黑客在 TEX 基础上开发了 LATEX，这是目前最有名气的一组宏的集合。

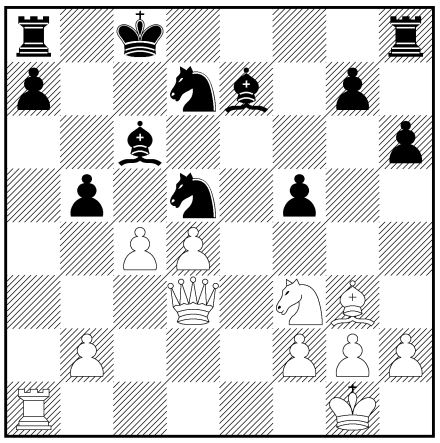
所谓"宏"(macro)，就是对底层的 TEX 系统再进行一次抽象处理，并提供一套更加简单的用户命令。如果说 TEX 提供的控制序列如同建筑房屋使用的砖、瓦、灰、沙、石等建筑材料，那么 LATEX 提供的宏就是用这些建筑材料制成的预制板、窗户、墙体和天花板。LATEX 是高度模块化的，今天的大多数用户嫌阅读高教授的原著麻烦，转向去直接使用 LATEX 来创作文件，直接阅读和使用 TEX 的人不像以往那么多了。(不过，依笔者所见，想发挥这一系统的潜力，阅读高教授的原著还是必须的，投入的时间也是可以得到大的回报的。)

LATEX2$_\varepsilon$ 是 LATEX 的最新版本，笔者根据 2002 年 6 月由 Sebastian Rahtz 编辑发行的 TEX Live CD 7 版本的进行统计，围绕这一版本开发的软件包(package)有 700 多个，打包进入第七版的 CTAN 档案整整充满了两张 CD–ROM，达到 1.2GB 的数据量！如果你每个月定期地观察一次 CTAN 网站，一定会发现有新的包提交出来。有了这么多的附加包，LATEX 的功能当然是很强大的了。

除了排版数学论文和书籍之外，它几乎可以排版任何出版物。下面的国际象棋插图就是作者利用 LATEX2$_\varepsilon$ 中的 chess 宏软件包排版而成的，图上记载的是 1997 年 5 月 11日的"人－机"大战最后局棋（第六局）的终局棋盘盘面[3]。

[3] 这一天，IBM公司的"深蓝"计算机在正常时限的比赛中以 3.5:2.5 战胜了世界象棋冠军加里·卡斯帕罗夫，这一胜利标志着人类在人工智能研究领域迈入了一个新的时代。

这个宏软件包的作者是 Piet Tutelaers，他于 1992 年在其他几位黑客工作的基础上创作。顺便你可以看出，文件中的代码全部都是 ASCII 字符，只要将这一代码块插入到 LATEX 文件中，就可以编译出下面所附的插图。



```
\usepackage{chess}
\board{r*k* * r}
      {p *nb p }
      { *b* * p}
      {*p*n*p* }
      { *PP * *}
      {* *Q*NB }
      { P * PPP}
      {R * * K }
\[ \showboard \]
```

（人－机大战终局盘面的 TEX 代码）

图 2 人－机大战终局盘面

除了排版象棋（国际象棋、中国象棋、围棋）棋谱，LATEX 系统还可以排版音乐的五线谱、化学分子式、十字形猜字游戏、电路图，等等。

ConTEXt

LATEX 是目前用户数量最多的 TEX 宏，现在仍在不断地发展中。不过，一些专家认为它还不是性能最好的扩展包。为了方便杂志和图书的出版商高效率地使用 TEX 系统，荷兰的黑客 Hans Hagen 等人开发了 ConTEXt。ConTEXt 是具有工业力度的排版系统，它在 TEX 和 Plain TEX 的基础上，利用 Perl 语言编写并加入了很多宏。欧洲有相当多的杂志和专业期刊利用它排版，原因之一在于它能够从 TEX 直接生成 PDF 格式的文档。（PDF 是 Adobe 公司在 PostScript基础上开发的技术，有很多新的技术特性，下面还将提到。）

关于 Perl 语言，我们在本书的“不散的宴席”一章和其他部分曾经提到过，Perl 的设计出发点就是扫描文本，从文本中提取所需要的信息，并将提取出来的信息按照给定的格式打印出来。这一特性使得它非常适合于开发 TEX 的扩展宏。ConTEXt 所做的事情就是扫描 TEX 源文件，从中理出各种宏及其作用域，然后将相应的文本按照指定的格式输出。

Hans 等人从 1990 年开始，前后花费了四年的设计时间才将 ConTEXt 系统稳定下来，后来又相继开发了支持德文、法文、波兰文、俄文等语种的接口，ConTEXt 的文档资料非常丰富、详细，它与数据库的接口非常成熟，而且输出的格式多种多样，系统定制的弹性很大。可以说它已经成为目前最好的基于

TEX 的排版系统。如果想在较短的时间里使用集成度高的系统，ConTEXt 是一个相当好的选择。

顺便提一句，ConTEXt 在 2002 下半年推出的版本可以支持支持 UTF–8 格式，ConTEXt 对汉语排版的支持已经非常好，本书就是在 GNU/Linux 系统上采用 GNU Emacs 作为编辑器写作编辑、利用 ConTEXt 系统完成排版和格式化的。ConTEXt 在 ε–TEX子系统中 TEX–XET 的基础上，加入了从右到左的排版支持，并且支持从上到下的垂直排版模式。这在 TEX 社团内是独特的。

Texinfo

Texinfo 是自由软件基金会的作品，用来创作 GNU 工程的文档。目前，全部 GNU 文档资料都是按照 Texinfo 的格式编写的，Texinfo 可以看作是 TEX 的一组宏，与 LATEX 类似。Texinfo 的输出格式有三种：info 文件、HTML 文件或 TEX 文件。info 文件可以在 GNU 系统或者 UNIX 系统的文字终端上作为在线文档直接阅读，或者用 GNU Emacs 阅读，HTML 为网页格式，TEX 文件则用于打印纸媒体的文档。

Texinfo 与 LATEX 相比，功能要少很多，而且在 Texinfo 上没有那么多第三方开发人员创作的宏软件包。不过这也是 Texinfo 的优点，因为简单易学，其学习曲线就比 LATEX 短很多了。

Ω

当然，作为一个三十多年前构思而成的排版系统，TEX 在设计思想上仍然有一些改进的空间。以往对 TEX 的意见主要集中在两点上。

第一个问题是这个系统在开发时是以支持英文为出发点来考虑的，因此系统规定输入一律采用 ASCII 字符。但是目前地球上的被广泛使用的语言文字系统远不止英文一种，很多语种的字符集的基数远远大于 ASCII 字符集。

这一点已经在《The TEXbook》中已经提到了，高教授在 137 页引用了 Michael Spivak 在 1982 年所著的“The Joy of TEX”中的见解：

TEX has no regard for the glories of the Greek tongue — as far as it is concerned, Greek letters are just additional weird symbols, and they are allowed only in math mode. In a pinch you can get the output $\tau\epsilon\chi$ by typing `$\tau\epsilon\chi$`*, but if you're actually setting Greek text, you will be using a different version of TEX, designed for a keyboard with Greek letters on it, and you shouldn't even be reading this manual, which is undoubtedly all English to you.*

为了解决这一问题，目前已经出台了几种方案。第一种最为简单直接，就是将 ASCII 字符集替换成为 Unicode 字符集合。当然，这么一改动，整个系统就得进行非常大的改造。幸运的是，高教授的 TEX89 已经可以支持 8–bit 编码的输入，而目前的 Unicode 编码格式中有一种 UTF–8 的格式，就是采用了八位编码。

目前进行的 Ω 项目就是朝着支持 Unicode 这一方向努力的，Ω 系统的核心开发人员，澳大利亚新南威尔士大学的约翰 • 普莱斯教授（Prof. John Plaice）就说，Ω 系统从头支持 Unicode，这一新的系统从第二版开始使用 C++ 语言和 STL (标准模板库) 编写，使用 31 位编码的 Unicode 编码。目前 Ω 系统开发小组很活跃，除了普莱斯教授，伊朗的黑客卜纳德，加上一批印度的黑客正在努力完善这一系统。已经发布的 Ω–1 是一个可以使用的系统，日本商界对这一系统很重视；Ω–2 是正在进行开发的一个版本，Ω–3 已经动手构思，看来很有希望获得成功。

第二种方案是对大字符集的字符先进行预处理，将字形绑定到预处理之后的(不可理解的)双字节的 ASCII 编码上。这一方案已经成功，目前处理中文的几个软件包都是按照这一思路进行处理的。例如荷兰黑客 Werner Lemberg 开发的 CJK 软件包，天元软件包和台湾的学者开发的几种软件包。七位的 ASCII 代码扩展到八位之后，如果最高位置 0，就是 ASCII 代码；如果最高位置 1，就表示将由两个连续的字节表示一个字符，例如汉字字符。这一方案的优点是可以将汉字和英文字符同一处理，而不用修改 TEX 底层系统的代码。当然首先必须有相应的软件包支持汉字的编辑和修改，幸运的是，GNU Emacs 软件包可以没有障碍地支持汉字的编辑和显示了。

在这种方案下，一个汉字的双字节编码对应于一个汉字的字形“盒子”，因此计算机中必须在内存中保存所有汉字的字形以供选择，而且双字节的编码需要进行预处理，增加了软件包的在编译时的时间，另外编译文件时提示的编译信息以及输出到日志文件中的汉字内码毫无可读性可言，让人去记忆至少六千多个乃至几万个双字节或者多字节编码是不合理的，纠错的难度也随之略有上升。

TEX 系统设计中的一个不尽完美之处是 TFM 文件的格式规范只容许每一个 TFM 文件只允许含有 255 个字符。因此按照这一方式进行编码而得到每一个汉字字库 TFM 文件将浪费不少空间，而在编译文件时，每一次查找字库中的每个汉字都会多多少少浪费一些处理时间，如果排版文件的长度相当长(例如一本近千页的著作的 TEX文件)，那么这些点点滴滴的时间积累起来对编译性能就是一个相当大的影响。

为了解决这一问题，笔者提出在非 ASCII 的大字符集中引入数据结构的概念，使用逻辑语言来描述字符的语义。通过语义块来匹配汉字字形的“盒子”名称，即采用哈希表的数据结构。尽管 TEX 不接受双字节的汉字编码直接输入，而只接受可读 ASCII 输入，但是如果一条汉语句子被编码成为可读

ASCII 编码序列，而新的编码序列对应于表达汉字字形的“盒子”，那不就解决了可读性的问题了吗？

笔者观察到了在西文中表达新符号的方式可以借用来处理中文信息的排版。例如，2002 年 1月 1日开始投入使用的欧元符号 € 以前在 TEX 发行版本中是不存在的，但是人们开发了 `\euro` 来代表这一新的符号。汉字本质上是一种表意符号，因此，这一处理方法可以借用到汉字的排版处理中来。

作者受到汉语中存在的“字义基元化、词义组合化”现象的启发，创造性地提出使用逻辑语言 Lojban 的根词来描述字义基元，并使用 Lojban 的格语法规则来构造汉字结构，并在词义表的基础上结合 Lojban 的造词规则构造表意符号词典。[4] 熟悉语言学的学者可以从中看出许多自然语言的优点。由于计算机检索表意符号的哈希表(你可以将这种数据结构理解为一种“词典”)的速度非常快，因此软件可以动态地快速地找到意符号词典中词语对应的表意符号（也就是汉字字形的序列），并将找到的汉字字形序列当作一个“盒子”传递给 TEX 系统，投入编译计算。

在这一方案下，对于 TEX 系统提供的输入不再是两个字节排列的汉字，而是具有语义定义的语义块，用 Lojban 作为元语言表达，因此都是人可以阅读的 ASCII 字符串，与汉字字库建立哈希表的对应关系，从字形库中寻找相应的汉字“盒子”插入进入 TEX 编译树的叶子中。

这一方案打破了多年以来汉字编码以双字节为基本编码单位的陈规，而语义块的描述方法又符合逻辑语言的规则，便于利用计算机来处理。Lojban 已经含有 1,300 个左右的字义基元，这些基元可以用来描述任意的词组合。如果与汉字输入法再结合起来，就可以得到一种高效率的排版处理方法。

笔者还注意到新版本的 PostScript 规范中对 CJK 编码的支持，由于可以利用 PostScript 语言的字典堆栈存放 CMap 编码，因此对于在 PostScript 文件中结合这一编码方案支持具有 CID 的汉字编码也是可能的。

与现有的汉字编码方案相比较，逻辑语言的字义基元的描述方法既可以使用 ASCII 字符集来描述，也可以使用 Unicode 字符集或者 UCS–4 字符集来描述。ASCII 代码是几乎所有计算机上都支持的编码方案，因此使用 ASCII 代码(或者说扩展的八位的 UTF–8) 来处理汉字语义的表达是最具有移植性的。在这一方案下，TEX 系统本身不需要作修改就可以处理汉语排版了，而 TFM 文件原来空缺和浪费的空间也得到充分利用。[5]

[4] Lojban 语言的设计的出发点是“沙皮尔－沃尔夫假说”，这一假说认为语言的结构是妨碍思想表达了沟通的最大障碍，因此作为一门人工语言，Lojban 力求将语言的结构设计得尽可能简单，Lojban 作为一种元语言已经通过了计算机的无二义性检测。

[5] 一个副产品就是汉语信息在网络上又具有一种新的编码方案。现有的计算机硬件，软件和网络协议都不用进行修改。

当然，因为不更改 TEX 系统代码，这一方案应该先对准备处理的汉字文档作预处理，再使用 TEX 系统进行编译。在 TUG2002 大会上一提出，立即引起了在场的来自英国、荷兰、美国等语言学家和 TEX 专家的兴趣，他们认为这一方案为更好地在 TEX 系统下处理汉字信息的排版提出了一条崭新的思路。

TEX 体系设计的第二个问题是系统本身对于图形处理的支持较弱。

这曾是对 TEX 系统批评最多的一个方面。当然这不能完全说是高教授设计中的问题，因为当初高教授设计 TEX 时，市场上能支持图形的计算机系统还很昂贵，而且 TEX 系统在设计时主要针对数学文章和书籍的处理，所以 TEX 系统本身没有提供任何图形处理能力。只是为了创作数学符号和字库，高教授才发明了 METAFONT。METAFONT 是开发其他字库的设计工具，它提供了一些图形处理的功能，因此用户又可以使用 METAFONT 创作图形，然后将得到的图形作为一“盒子”插入到 TEX 文件中去。TEX 可以与 METAFONT 密切配合，在与 METAFONT 通信时，可以从 METAFONT 代码中接受“盒子”信息，来完成数学文章和书籍的排版。METAFONT 的这些图形处理功能除了用来设计字库字形外，还可以用来画图(字形本身也是一种图形)。

METAFONT 的设计原理如同 WEB 系统，也是采用“两步走”的机制，即先用元语言“描述”字符的形状，然后编译出最终的字形并放进“盒子”中。这种分两部走的处理手法对于字库是绝对行之有效的(因为字库一旦设计完毕，除非有特殊的理由，几乎就不会变动了)，而对于一般的图形这样处理起来却显得不方便。幸好，TEX 系统在设计时还留下了一个对外的接口：使用 `\special` 控制序列可以将外部的图形插入到 TEX 文档中，这样一来，就为使用外部图形系统提供了可能性。

下面即将谈到的 PostScript 语言恰好就是这样一种方案。由于有这一接口，现在我们可以使用 PostScript 技术生成图形，并将它作为一个外部对象插入到 TEX 文件中。

PostScript 技术可以描述任意的图形、文字和图片，而且有相当强大的颜色处理模式，彻底地弥补了原来 TEX 系统“色盲”的缺陷。因此黑客们将这两项技术巧妙地结合起来，形成了具有工业力度的排版系统。TEX 系统除了可以插入 PostScript 外部图形文件之外，还可以插入很多其他工具生产的图形文件，因此现在社团里不再有对 TEX 系统图形处理能力差的批评之声了。

TEX 的许可证及其解读

TEX 许可证比较复杂，因为当我们谈及这一软件包时，TEX 软件社团组织里面的软件包作者群体非常庞大，不同的作者在制定许可证时的想法不一定完全相同。用户手头使用的软件包往往是经过定制的，很难用同一许可证涵盖所有的情形。因此，需要具体情况具体分析。

先让我们从源头说起，考察由高教授创作的 TEX 的许可证。理解高教授的许可证条款本身也是一个比较困难的事情。TEX 的许可证显然不是 GPL，因为在 TEX 开发成功和发布时，自由软件基金会还没有成立，GPL 等法律文件还没有制定出来，后来等到 GPL 公布之后，TEX 社团组织内部也没有人明确提出将 GPL 追溯以施加到 TEX 系统上。

尽管这样，我们还是可以找到与 TEX 相关的版权声明，它们前后出现在三个地方。

1）下面这一声明出现在 `tex.web` 文件的顶部：

```
This program is copyright (C) 1982 by D. E. Knuth; all rights are reserved.
Copying of this file is authorized only if (1) you are D. E. Knuth, or
if (2) you make absolutely no changes to your copy. (The WEB system
provides for alterations via an auxiliary file; the master file should
stay intact.) See Appendix H of the WEB manual for hints on how to install
this program. And see Appendix A of the TRIP manual for details about how
to validate it.
```

参考译文：

本程序的版权属于 D. E. Knuth 所有，出版时间为 1982 年。所有的权利都由作者保留。

在以下情况下，你可以得到授权拷贝这一文件：

（1）如果你是 D.E.Knuth; 或者，
（2）如果你对文件的拷贝不加任何修改。WEB 系统可以通过一个辅助文件形成替代系统，但是主文件应该完整无缺地保留。

本程序的安装提示参见《WEB 手册》的“附录 H”。使系统生效的详情参见《TRIP手册》的“附录 A”。

解说：

TEX 系统的版权属于高德纳教授所有，这一点当然是肯定无疑的。集合论在研究等价关系的定义时，有自返性的定义，自返性就是自己与自己的关系。出于数学家的职业习惯，高教授在许可证中将自己的权利也明确列出了。今天从律师实践的角度看，这一笔纯属画蛇添足，因为第一句“版权所有”的声明就已经足以体现作者的这一权利了。

对于第二条，有必要略加解释。我们在前面介绍过 TEX 的技术内幕，这一系统在生成排版结果时采取了两步走的机制，先由 WEB 系统完成数学模型，然

后由数学模型出发实现生成排版结果的代码。WEB 的数学模型可以由其他语言实现，例如 C、C++ 或者 Lisp。

从上面的声明来看，WEB 系统的数学模型的代码隐含地是可以自由运行的。拷贝的自由度则受到了一定的限制，用户必须保持作品的完整性。WEB 系统本身没有提供自由修改的自由度。这一声明隐含地指出，除了作者之外任何其他人不能修改高教授编写的代码并且命名为 TEX。

2）下面的声明印刷在“计算机与排版丛书”第二卷《B. TEX：The Program》的版权页上：

```
The program for TeX is in the public domain, and readers may
freely incorporate the algorithms of this book into their own
programs. However, use of the name ‘TeX’ is restricted to software
systems that agree exactly with the program presented here.
```

参考译文：

TEX 的程序属于公众领域，读者可以自由地将本书中的算法合并到他们自己的程序中去。但是，对于“TEX”这一名称的使用，仅限于同意精确地使用了这里所列明的程序的软件系统。

解说：

这一段声明将 TEX 程序置于了公众领域。这里有必要将“公众领域”(Public Domain) 这一法律术语作一番解释。法律界认为，属于“公众领域”的著作属于全人类的作品，即人人都可以自由地使用它，而没有向作品的作者交纳版税的义务。属于公众领域的作品一般不受到著作权法律的保护。因此在公众领域的软件的基础上，可以将原来的程序进行修改，转化成为专有软件发布。这一情形在 TEX 后来流行的过程中的确真的发生了。例如，英国就有一家公司投入了很多人力研究 TEX 系统的代码，并且在消化吸收了 TEX 的代码基础上开发了具有图形用户界面的排版系统，而新的系统在改头换面之后被作为专有软件发行。TEX 的许可证不能防止这一类情形发生，从某种程度上，它的许可证甚至隐含地鼓励出现这种努力（尽管这不是高德纳教授的原意）。这一情况有些类似于 X Window 系统的许可证引发的问题（参见本书第二部分第十章的讨论细节），只不过 X Window 系统不是属于公共领域的软件。

3）下面的声明出现在 `plain.tex` 文件的顶部：

```
% This is the plain TeX format that’s described in The TeXbook.
% N.B.: A version number is defined at the very end of this file;
%       please change that number whenever the file is modified!
%       And don’t modify the file under any circumstances.
```

参考译文如下：

这是《The TEXbook》中描述的 Plain TEX 格式。注意：本文件的最后定义了版本号码；每次修改本文件时请改变这一号码！在任何情况下不要修改本文件。

TEX 许可证综评：

综合上面的三个文件中的有关声明，TEX 许可证的条款就比较完整了。

TEX 是公用领域的软件包，因为 WEB 系统从算法到代码都是处于公用领域内，将 WEB 系统的代码进行修改并按照另一个名字发布是没有限制的。后期由 Karl Berry 等黑客开发的 WEB2C 系统的 C 代码已经按照 GPL 发布。后期衍生出来的 LATEX 软件包实际上是基于 WEB 模型的一组宏，它几乎没有设计底层的 TEX 系统的代码，因此 LATEX 也是基于自由开放的精神而发布的。

这种许可证的问题在于如果有的公司在 TEX 的基础上作出改进，并将改进的软件包作为专有软件发行，这一点是 WEB 的许可证所允许的，但是正如我们在本书第一部分所分析的，使用专有软件的用户的自由被剥夺了。现在有道德修养水平不高的一些公司就在这样干。所幸的是，在 TEX 社团内部，绝大多数人受到高教授的宽阔胸襟的感召，一般都将自己开发的代码也无偿地贡献给社团，真正走专有软件道路的个人或者公司数量并不多。

使用 TEX 这一名称是有真正限制的。正如我们在本书第二部分中提到的，自由软件基金会定义的自由软件的定义中有四个自由度，其中要求软件必须可以自由地修改，从这一定义看，TEX 不是现在自由软件基金会的严格意义上的自由软件。

可见，尽管高德纳教授是世界闻名的数学家和计算机科学家，在学术上取得了巨大的成就，包括这一套 TEX 排版系统在学术界取得了很大成功，拥有众多的用户群体，但是 TEX 许可证在自由度上却存在一定的缺陷，以今天的观点来看，其软件许可证制定得很业余，其严谨度和自由度远远没有达到 GPL 那种理性思维的境界。

不过，鉴于高德纳教授在计算机科学发展历史上的独特影响，TEX 社团组织的成员都理解所有由高德纳创造的软件可以没有限制地使用、再发行。软件的源代码可以自由地复制和修改，只要修改的代码使用另一个名字即可，这样可以遵从高德纳教授将所有在 TEX 系统下的代码是他真正创作的这一意愿。考虑到这一软件包出现的年代比自由软件基金会的自由软件要更早一些这一历史事实，因此我们自由社团又将 TEX 系统视为一种自由软件，因为它在实际中向用户提供了自由运行、复制、再发行和修改的自由度。TEX 的名称使用限制在现实中并不是一个太大的障碍。虽然 TEX 的许可证存在这些问

题，但是瑕不掩玉，它毕竟是自由软件，其许可证上的缺陷并不妨碍自由软件社团采用和维护它。

关于商标

出现在 `tex.web` 文件的顶部，紧接着版权声明之后，还有关于商标所有权的声明：

```
% TeX is a trademark of the American Mathematical Society.
% METAFONT is a trademark of Addison-Wesley Publishing Company.
```

TEX 的商标属于美国数学学会所有，而 METAFONT 商标属于 Addison–Wesley 公司所有。

读者可能感到奇怪，为什么 METAFONT 的商标属于 Addison–Wesley 公司所有呢？原因是高教授与这个公司签署有出版合同，目前高教授的《计算机编程的艺术》前三卷和《计算机与排版丛书》的五卷都是由 Addison–Wesley 出版的。[6] 这家出版公司当时十分看好 METAFONT 字库设计软件，因此让高教授将商标转让给了该公司。

高教授原来的设计意图是非常明显的，即使用 TEX 作为编译器，处理字形盒子之外的工作；而 METAFONT 则作为字形和图形设计工具，处理盒子内部的工作，与 TEX 密切配合。男主外，女主内 —两者原是高教授意中亲密的一对情侣。但是后来，随着 METAPOST 等技术的推出，情况发生了很大变化：TEX 和 METAFONT 两者仍然可以合作，但是 TEX 这头雄狮已经不再专情于 METAFONT 了，他的身边又多出了其他一些“女朋友”，连高教授本人在新版的《计算机编程的艺术》中，也特意让助手将原有的使用 METAFONT 创作的插图改用 METAPOST 重新绘制，看来 METAFONT 的前景十分堪忧。对于 TEX 社团而言，METAFONT 的商标权属于谁已经真的变得无关紧要了。

TEX 的商标属于美国数学学会($\mathcal{AMS}$)的原因则不难理解：前面提到过，TEX 发明的初衷首先就是为了解决数学论文和书籍创作和出版过程中的问题，而美国数学学会拥有大量的数学家会员。因此，这一软件包一发布，就有大量的数学家采用它，最终大家觉得这一软件包实在太好了，因此大家都采用它来写作数学论文和书籍，连美国数学学会权威性的数学期刊《数学评论》也指定作者投稿时必须以 TEX 文件格式投稿，而且专门制作了稿件的样式规范和相应的宏软件包，称为 $\mathcal{AMS}$–TEX。$\mathcal{AMS}$–TEX 的影响很大，乃至在 LATEX 出现后，TEX 内部强烈希望 LATEX 合并 $\mathcal{AMS}$–TEX 的特性。最终在社团经过努力后，达到了这一目的。现在的 LATEX2$_\varepsilon$ 中就含有 $\mathcal{AMS}$–TEX 的样式文件了。正

[6] 遗憾的是，这些著作都是专有出版物。

是由于美国数学学会中 TEX 的用户群体人数这么多，所以美国数学学会和高教授很早达成了协议，将 TEX 的商标转让给了美国数学学会。

在 TEX 系统中，TEX 这一商标已经有专门的控制序列，只要在文章中输入 `\TeX`，那么系统就会编译生成 TEX 这一模样来。注意，这一商标的形式是有特定的设计规格的，设计的代码本身就是用 TEX 语言编写。在“计算机与排版丛书”第一卷《The TEXbook》中，高教授在第 66 页第六章专门详细讲解了如何利用 TEX 的原始控制序列来生成这一商标，以下就是这一商标在 TEX 系统中的定义：

```
\setbox0=\hbox{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125em X}
```

TEX 系统将这一商标放置在内部的寄存器 `box0` 中。回头看前面的技术性解说，这一定义将 T、E、X 这三个字母分别放入到一个小“盒子”中，并将这三个盒子定义为一个水平的大盒子。在大盒子的内部，将字母 E 与 T 靠得近一些，`kern` 是靠近的意思，靠近多少呢？答案是 0.1667 个 em。并且将 E 的位置降低一些，`lower` 是降低的意思，降多少呢？0.5 个 ex。字母 X 与 E 也贴得比较近，近的距离比正常的两个字母少 0.125 个 em。

过去一个 em 曾经是你正在使用的字号中的大写字母 M 的宽度，一个 ex 就是你正在使用的字号中小写字母 x 的高度。但是高德纳教授在设计 TEX 系统时打破了这个约定，对于每一种字体，字体设计人员现在都可以自由定义 em 和 ex 的值，所以 em 和 ex 的值究竟是多少取决于你正在使用的字体的字体设计人员所作的定义。em 用来定义水平方向的单位长度，ex 则用来定义垂直方向的单位长度。如果你正在使用“计算机现代罗马字体”(cmr)，在字号为 10pt 时，一个 em 等于 10pt，而一个 ex 约等于 4.3pt。

pt 是 point 的简写，它的定义又有一套来历。美国字库协会在 1886 年约定一个 pt 精确地等于 0.013837in，因此传统意义上的一个点等于 0.99999999pt，“精度”达到了 10^{-8}，但是英寸的定义在 1959 年发生了变化，它从原来的 1/0.3937cm 缩小到 2.54cm，故现在一英寸等于 72.27pt，既便于记忆、又便于计算。

这里要提到一个不太引人注意的细节：就是 PostScript 技术中的 point 与 TEX 中的 pt 是不同！在 PostScript 中，1in = 72 point，因此 PostScript 的一个点要比 TEX 系统中的一个点要大一些，人们常称之为“大点”。从某种意义上说，TEX 在设计计算单位时比 PostScript 的开发人员考虑得要更为精细一些。从另一方面看，TEX 的内部设计机制使文件的最大尺寸不得超过 5.7583 米(一般的技术文档绝对不会超过这一尺寸)，但是对于制作室外大型广告或者招贴画时，一个文件的版面就可能显得不够用了(但可考虑用多个文件拼接来变通解决)，这一方面 PostScript 技术的限制要宽松一些。

美国数学学会拥有 TEX 商标的第二个原因就是它是一个非营利性学术组织。这一点与高教授设计 TEX 这一系统的初衷是吻合的，高教授希望自己设计的系统能使所有写作数学论文和书籍的人都可以受益。因此他选择了美国数学学会这个公益性的学术团体作为 TEX 商标的所有者。

理查德·斯托曼说过："凡是不自由的软件终归趋于平庸"。从另一方面理解，只有自由的软件才有可能变得伟大。由于 TEX 公开了系统的所有源代码，并且提供了详尽的文档资料，因此其他人才在这些基础上参加系统的完善和改进。TEX 后来的发展过程正是这样，而且今天还保持着旺盛的活力。这正是自由软件内部发展机制具有强大生命力的有力佐证之一。

PostScript

在谈到为 TEX 提供图形支持时，必须提及 PostScript 语言。尽管为 TEX 提供图形支持的软件包很多，但是 PostScript 技术影响力最大。

正如上面提到的，在高德纳教授着手发明 TEX 系统时，由于图形计算需要大量的存储器，而且对 CPU 浮点运算速度要求较高，存储器和 CPU 这两样东西曾经都很昂贵，因此当时计算机系统对于图形终端的支持还非常少，因此 TEX 被设计成基于文本界面的格式化工具。

到了二十世纪八十年代中期，由于个人计算机的出现和普及，局面开始大有改观，CPU 的浮点计算速度大大提高了，而且存储器的单位成本随着工业化大规模集成电路技术的成熟和生产规模的上升而不断下降。支持图形的计算机开始出现，图形用户界面也开始流行起来，PostScript 技术就是在这种大环境下出现的。

PostScript 是基于"堆栈"(stack)的编程语言。堆栈是一种"后进先出"的数据结构，它既可以由软件实现，也可以由硬件实现。堆栈可以很方便地体现程序中递归的思想，因此发展成为计算机科学中一个基本的概念。历史上还一度流行过由硬件直接实现的堆栈计算机。到了二十世纪八十年代时，基于堆栈的技术已经不新鲜，在 PostScript 技术出现以前就有过 Forth 语言，Forth 语言是为了控制天文望远镜系统而发明的语言，PostScript 的设计人员显然受到了 Forth 语言的很多启发。

基于堆栈的语言一般采用逆波兰表示法编写程序指令，也就是操作码与操作数顺序倒置的表示法。如果你熟悉 Intel x86 的指令，一定知道一条指令是先写"操作码"（称为 opcode），也就是先用指令告诉计算机干什么，然后后面跟着"操作数"（称为 operand），也就是被指令操作的数据。例如，将数字 10 与 eax 寄存器中的数字相加：

```
add %eax 10      ; add 10 to the register eax
```

基于堆栈的编程语言，例如 Forth，写法则不同：

```
%eax 10 +
```

实现起来的方式也大不一样，先将数字 10 压入堆栈，然后将 `eax` 寄存器中的值压入堆栈，然后 + 将两个数字弹出堆栈，在逻辑和数字运算单元执行 + 的操作，并将结果压入堆栈。

在 Intel x86 体系上实现的 C 编译器在编译时会将 C 源代码中的临时变量、函数的返回地址、参数和参数列表、中间计算需要的表达式等统统地放置在同一个堆栈中，在程序的运行时通过帧框活动过程记录来反映堆栈的生长和下降情况。完全基于堆栈的编程语言中的代码和数据既可以各自单独存放在一条堆栈中，也可以混和在一起存放于同一条堆栈中，通常为了体现程序运算的算法，将完成算法的步骤的返回地址单独存放在一条堆栈中。由于返回地址按照堆栈存放，正如前面介绍过的，堆栈是一种后进先出的数据结构，因此程序的算法实现上可以变得非常清晰，而且进程的切换的速度很快——一旦堆栈中提供的返回地址发生变化，那么进程的上下文切换几乎可以立即完成。

老一代基于堆栈的语言主要的特点在于“支持多线程代码”和“无操作数指令”方面。它们被集中地应用在工业的实时控制方面，但是新出现的 Java 语言却在这一方面有革命性突破。尽管 Java 的语法很像 C 语言，但是 Java 的虚拟机也是基于堆栈的，阅读 Java 虚拟机的代码与阅读 PostScript 或者 Forth 代码差不了多少，而且 Java 从 Lisp 语言继承了很多特性（Java 语言也支持多线程，还有一大堆的其他特性）。

无论是 TEX，还是 PostScript，它们最终要将图形输出到点阵设备上，不过，PostScript 将这一技术发展到了极致。利用 PostScript 语言编写的代码具有紧凑、易读、灵活的特点。作为全功能的页面描述语言，PostScript 内置了大量关于图形描述的运算符，所有的这几百个运算符在 PostScript 规范中被划分成三个等级：LanguageLevel 1、LanguageLevel 2、LanguageLevel 3。第三级抽象程度最高，是在 PostScript 语言规范第三版中才出现的。显然，与 TEX 系统一样，由这么多运算符组成的 PostScript 系统也是非常复杂的，要想彻底熟练掌握 PostScript 语言，也得花相当长的时间才行。

奇妙的是，PostScript 语言是基于文本的，而且是可以打印出来的 ASCII 文本，PostScript 代码就是利用这些 ASCII 字符来描述图形，由于 ASCII 字符是几乎任何计算机上都支持的编码，因此，PostScript 代码具有良好的移植性。

尽管系统如此复杂，但是基本的编写程序的思路却不难掌握。现在举一个例子说明，假设我们想画一个圆圈，相应的 PostScript 程序可以这样来写（注意，这是一个普通的文本文件，您可以使用任何文本编辑器来创建这个文件。自由软件开发人员一般使用 GNU Emacs 来创建 PostScript 的源程序）：

```
newpath

300 300 100 0 360 arc
stroke
closepath

showpage
```

PostScript 程序举例之一

程序解说：

newpath 表示我们想新设置一条路径，在这条路径上我们画出圆圈。(路径是 PostScript 语言中一个非常基本的概念，PostScript 解释器在作解释计算时一般将路径存放在一个堆栈中，这样画出一条路径的计算可以转化为将堆栈中的数据倒出的操作。一副图上可以存在多条路径，当多条路径都画出来后，图的图形也就基本上呈现出来了。)

由于圆圈可以看作是一条完整的圆弧，因此，PostScript 采用 arc 运算符，给定的参数是圆心的坐标（离纸面左下角的横向和纵向距离，300，300），半径的长度（100），是起始角度（0），以及弧线终止角度（360）。

```
newpath

300 300 100 0 360 arc
0.8 setgray fill
stroke
closepath

showpage
```

PostScript 程序举例之二

程序解说：

这一段代码在上一个例子的基础上增加了两条指令，即将圆圈涂上灰色。颜色的填充命令是 fill。灰度设置命令为 0.8 setgray。

```
newpath

/Times-Roman findfont 40 scalefont setfont

10 600 moveto
(Free Software: New Game Rules) show
10 580 moveto
2 setlinewidth
550 0 rlineto
10 560 moveto
2 setlinewidth
550 0 rlineto
stroke
closepath

showpage
```

PostScript 程序举例之三

程序解说：

这一段代码先是选择了文本的字体，然后在程序指定的地方写出文本 “Free Software: New Game Rules”，并在文本下方画出两条横线。最终得到的图形就是本书英文版的扉页。

基于堆栈的 PostScript 程序在编写时有很多技巧，善用这些技巧可以直接提高程序的执行速度。对于一个编译型的 C 语言程序，想将程序的运行速度提高一倍往往是很难的事情，但是对于 PostScript 程序，经常稍加改进，优化了的程序代码的运行速度就可以提高三倍，甚至更多到十倍。

Philips Koopman Jr. 教授关于 Forth 程序员的评论同样也适用于 PostScript 语言程序员，现在摘录并翻译如下：

The extensibility of Forth does have mixed blessings. Forth tends to act as a programmer amplifier. Good programmers become exceptional when programming in Forth. Excellent programmers can become phenomenal. Mediocre programmers generate code that works, and bad programmers go back to programming in other languages. Forth also has moderately difficult learnning curve, since it is different enough from other programming languages that bad habits must be unlearned. New ways of conceptualizing solutions to problems must be acquired through practice. Once these new skills are acquired, though, it is a common experience to have Forth–based problem solving skills involving modularization and partitioning of programs actually improve a porgrammer's effectiveness in other languages as well.

“Forth 语言的可扩展性是件好坏参半的事情。Forth 似乎可以扮演程序员放大器的角色。良好的程序员在编写 Forth 程序时可以很出色，而优秀的程序员则可以表现得极其出彩。普通的程序员可以编写出可以工作的代码，而蹩脚的程序员会去使用其他的语言编写程序。Forth 学习起来不是那么轻松容易，因为它与其他语言的差异实在太大，学习 Forth 时必须避免学习一些坏的习惯。程序员在学习过程中必须通过实践来掌握解决问题的新思路和新概念。但是一旦掌握了这些新的技巧，那么使用基于 Forth 解决问题的这些技巧就可以使程序达到良好的模块化程度，并且合理地切分程序成为小的程序，这些技巧在使用其他语言编写程序时可以如法炮制,以提高程序的效率。”

这一段评论是很有见地的。原来有些程序员曾经打算使用 PostScript 编写图形应用之外的其他应用程序，但是这一做法已经不被新一代的程序员认同。PostScript 主要还是应用在印刷领域。

PDF 是 Adobe 公司在 PostScript 的技术基础上开发除了的另一套技术。与 PostScript 技术相比，PDF 在以下方面有明显不同：

- PDF 不含有内置的编程语言
- PDF 的文件格式保证页面的独立性
- 支持超文本连接和安全性
- 字体的字形文件可以不包含在文件中
- PDF 文件的代码更加紧凑，因此文件的尺寸更小一些。

pdfTEX

在 PDF 问世之后，一个新的思路出现了，既然 TEX 系统可以输出 DVI 格式，那么能否直接输出 PDF 格式的文件呢？答案是肯定的。果然不出所料，越南裔学者 Hàn Thế Thành 已经成功地发明出了 pdfTEX 软件包，其输出的文件格式就是 PDF。

METAPOST

高德纳教授设计 METAFONT 时，曾经得到过贝尔实验室一位研究员的鼎力协助，这位研究员就是 John Hobby。在 METAFONT 和 TEX 系统一起推出后，John Hobby 在原来研究的基础上，又推出了 METAPOST 系统。

与 METAFONT 相似，METAPOST 本质上也是一个编译器，但是和 METAFONT 编译生成与设备无关的点阵图形不同的是，METAPOST 的编译输出结果是与设备无关的矢量图形形，即符合 PostScript 规范的代码，因此目标文件可以使用像 Ghostscript 这样的 PostScript 解释器来解释它的编译结果。这

一系统出现后在 TEX 社团内部迅速扩散开来，成为目前字库设计和图形设计的首选工具之一。

METAFONT 和 METAPOST 编译的目标文件都需要相应的硬件驱动程序才能被显示或者打印，显然有多少不同的硬件设备就需要多少相应的硬件设备驱动程序，但是 METAPOST 输出的 PostScript 文件在工业界得到的支持更加广泛，使用面无疑要更大一些。[7]

自由软件基金会中国研究院的“一、百、万工程”的第一个项目就是开发基于 PostScript 汉字字库。由于 METAPOST 的最终输出文件格式是 PostScript，它比直接使用 PostScript 描述汉字字形更加方便，因此我们确定了设计的技术原则，就是以 METAPOST 来设计 PostScript 字库的描述文件，然后从这些 METAPOST 文件生成符合 PostScript 规范的汉字字库。在设计汉字的 METAPOST 代码的过程中，我们制订了一套规范，对于汉字的偏旁部首、部件合成原则，以及汉字的结构都作了详细定义，并且所有的汉字设计均按照这一规范进行设计。

下面是用 METAPOST 描述的汉字“一”的代码：

```
% This is the METAPOST code for ‘‘pa’’,
% or D2BB in GB18030, or U+4200 in Unicode.
% Copyright Hong Feng 2003. All rights reserved.
% RON’s Datacom Co., Ltd.

beginfig(1) ;

path p ;
path a ;
a := (8u,2u) ;
path a1 ;
a1 := (10u,3u) ;
path b ;
b := (92u,3u) ;
path c ;
c := (80u,15u) ;
path d ;
d := (74u,7u) ;
path e ;
e := (4u,8u) ;
path f ;
f := (8u,2u) ;
```

[7] METAPOST 与 METAFONT 技术上的详细比较可以参见《METAPOST 用户手册》。

```
p := a..a1--b--c--d--e..f--a..cycle ;
p := p shifted (-8u,-2u) ;

fill p ;
draw p ;
endfig ;
\bye
```

图 –3 是由这一代码生成的汉字字形。

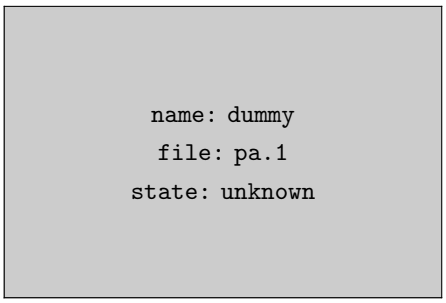


图 3 由 METAPOST 代码生成的汉字“一”的字形

我们的想法是将所有的汉字字形都当作图形符号，因此在 TEX 文件中插入的不是汉字编码, 而是汉字编码对应的图形符号, 图形符号的编码是用笔者开发的一个名为 `cn2tex` 的小程序进行汉字编码和汉字内码转换的。

“一、百、万工程”的 METAPOST 汉字代码已经为字库族取名为“首义”，英文为 mnm，因此宋体汉字字库的写法为 mnm–shouyi–songti。这套字库的 METAPOST 代码许可证是按照 GNU GPL 发布的，但是字库名称版权属于武汉荣世数据通信有限责任公司及其开发人员。

Adobe 公司的商业策略

在人们的心目中，Adobe 公司对于自由软件开发人员不甚友好，原因是该公司曾经起诉在美国打工的俄罗斯程序员迪米特里·斯克里亚罗夫。天才的迪马发明了一种算法，可以高效率地反向破解出加密的 PDF 文件的原始代码，Adobe 公司感到这一技术威胁到公司的利益，随即利用美国的数字千年版权法案（DMCA，Digital Millenium Copyright Act）向迪马提出起诉，只身一人在国外的迪马琅铛入狱，此事在美国软件界引起了轩然大波，后来 Adobe 公司迫于公众舆论的强大压力，由双方律师庭外调解而息事宁人。

Adobe 公司基于 PostScript 技术的软件产品都是专有软件，包括可以从其网站免费下载的 Acrobat PDF 阅读器软件也是这样，尽管任何人都可以免费使用这一阅读器，但是 Adobe 公司不提供该阅读器的源代码，因此它不是自由软件，而只能算是一种免费软件。

PostScript 是 Adobe 公司的注册商标，因此其他人未经其许可不能使用这一商标，但是 Adobe 公司声明：如果某一家公司的产品真正支持 PostScript 技术，那么 Adobe 公司可以接受其使用“PostScript–compatible”(与 PostScript 兼容) 的说法。

Adobe 公司拥有 PostScript 运算符列表和《PostScript 技术规范》的版权。换句话说，只有该公司才能修改 PostScript 技术规范，使用该商标必须经过该公司同意才行。其他公司也可以开发基于 PostScript 的技术，或者修改而衍生出新的类似技术，但是由于商标属于 Adobe 公司专有，其他公司或者个人修改的或者新衍生的技术不能冠以 PostScript 的字样，Adobe 公司利用这一控制权来区别该公司和其他公司的产品。

不过，与其他专有软件公司相比，Adobe 公司在推广应用 PostScript 技术时采用的策略是相当宽松的，这一点可以从《PostScript Language Reference》第三版 1.5 节中的声明看出来，以下是摘出的内容：

- 利用 PostScript 语言编写程序
- 开发能生成含有 PostScript 语言命令输出的驱动程序
- 开发可解释用 PostScript 语言编写的程序的软件
- 为实现以上目的复制版权属于 Adobe 公司的运算符列表

唯一的条件是在复制版权属于 Adobe 公司的运算符列表时必须附带 Adobe 公司的版权声明，而且这一许可不适用于对其技术文档规范、其他专有出版物和专有软件的复制。

自由软件基金会的 Ghostscript

Adobe 公司是为数不多的让 Microsoft公司的商业并购策略碰壁失败的几个公司之一。在 PostScript 技术推出后，微软公司的老板 Bill Gates 很欣赏它，于是想收购 Adobe 公司，从而达到拥有 PostScript 技术的目的, 为了这一目的，Bill Gates 曾邀请 Adobe 公司的两位创始人 John Warnock 和 Chuck Geschke 到微软公司谈判。但是这两位创始人在耐心的讲解 PostScript 技术的要点之后，无意让微软公司收购。微软在碰了一鼻子灰后，决定转而支持 TrueType，这是由 Apple 公司开发的另一套矢量字库技术规范。TrueType 技术的规范可不像 PostScript 那么开放。

尽管我们无法得知 Adobe 公司的 PostScript 解释器是如何开发的（因为它是专有软件，我们无法看到其源代码），但是，由于 PostScript 技术规范是公开

的，而且其他软件人员可以根据这一规范开发可以解释 PostScript 文件的解释器，这一点是 Adobe 公司 PostScript 技术策略允许的，甚至是鼓励的，因为他们认为这一点有助于 Adobe 公司在印刷市场上巩固和扩大市场占有率。

这一招果然有效，市场上真的出现了编写 PostScript 解释器的公司和软件开发人员，其中具有代表性的程序是 Ghostscript。Ghostscript 是用 C 语言编写的，它实现了 PostScript 规范的绝大多数特性，作为一个 PostScript 解释器，它使自由软件社团可以充分利用现有的 PostScript 打印机和印刷技术来传播思想。

Ghostscript 有一个独特的技术特性，是 Adobe 公司原来没有实现的—它可以将 PostScript 文档在非 PostScript 打印机上打印出来。支持 PostScript 技术的打印机价格一般比较高，而 Ghostscript 则打破了这一技术限制，普通的激光打印机和喷墨打印机也可以打印出 PostScript 文件。

不过，这一软件包的许可证比较复杂，因为它有三个发行版本：一是 AFPL Ghostscript (以前称为 Aladdin Ghostscript)，按照 Aladdin Free Public License 发布；二是 GNU Ghostscript，按照 GNU GPL 许可证发布；三是商业版本的 Ghostscript。

Aladdin Free Public License 许可证的条款见本书附录，本书的第一部分有对这一许可证的详细解说。一般说来，GNU Ghostscript 的最新版本总是滞后 AFPL Ghostscript 最新版本，即与上一个 AFPL Ghostscript 版本相同，例如 GNU GhostScript 5.10 与 AFPL GhostScript 5.50 功能相当。

SGML/XML 与排版技术的结合

二十世纪九十年代初，基于 TEX 和 PostScript 排版和印刷技术已经占据了市场的主导地位，尽管当时还有其他竞争的技术，却都不足对它们形成真正的威胁。人们一度以为在这之后，这一领域似乎没有什么空间可以再开拓了。但是历史对持这种观点的人开了一个很大的玩笑，起源于二十世纪五十年代末期的符号表达式技术突然在二十世纪九十年代中期异军突起，以 XML 的新面孔卷土重来，再创出了生命的第二春，使得原来略显沉闷的排版和印刷技术又日趋活跃起来。

SGML 与符号表达式

为了搞清楚 XML 是怎样侵入 TEX 和 PostScript 传统领地的，我们先来看看 XML 的来龙去脉。XML 是在万维网联盟（W3C) 在 HTML 遇到麻烦之后提出的，而 HTML 则基本上是 SGML 的一个子集。SGML 规范最初是由 IBM

的研究员 Charles F. Goldfarb，Edward Mosher 和 Raymond Lorie 等三人于 1968 年制定的，后来在二十世纪八十年代成为 ISO 标准（ISO–8879：1986）。

SGML 的基本思想是使用标记来处理文档，而文档是创建出版物的原始素材。例如，对于书来说，尽管内容可能千差万别，但是却存在相同的结构：封面、扉页、版权页、目录、章节、编号列表、参考文献、索引、封底等。不同的出版物有不同的结构，例如字典和诗集的文档结构就完全不同。

从离散数学的角度来看，这类结构可以抽象成为一种树结构。所谓树结构，就是从根出发，向上长出主干，主干向上再长分支，分支再分叉，最终就是树叶。计算机软件里这类树结构到处都可以看到，UNIX 和 GNU 操作系统中的文件系统就是一个典型的例子。

一旦进入以结构的观点看待事物的视野，我们不得不提到法国的布尔巴基学派。第二次世界大战之后出现的布尔巴基学派所持有的观点就是利用结构来统一数学。他们将数学中的最基本结构归纳代数结构、拓扑结构和序结构这么三种。布尔巴基学派认为形形色色的数学结构都可以从这三种基本的结构出发来构建。

泛系理论在前人的基础上，提出了“辨异同、排泛序”的方法论。笔者认为这一方法论是认识世界和改造世界时采用的一条基本规律。“辨异同”是指找出事物之间的相同点，区别事物之间的差异，其基本的手段是使用抽象，而抽象的手段可以使用泛系中的形影关系和局整关系来刻画。

将看似不同的事物等同起来考虑和对待是人类的高级智慧之一。庄子是中国古代哲学家中第一个具有这样的思想的人物。庄子的齐物论将事物之间的异同关系提高到哲学理论的层次，后人认为庄子将什么东西都看成是一样的，当然是曲解了庄子的思想而走向了极端。在那么远的年代庄子就能达到我们今天很多人所不能的认识水平，庄子的基本思想是卓越的。

在相似的事物之间看出差别来是人类的另一种高级智慧，莱布尼茨说过世界上没有两片树叶是完全相同的。根据事物之间的异同关系，就可以对我们研究的系统进行层次划分和聚类。层次与层次之间，或者次层次内部的事物之间往往可以呈现出某种序关系。

由于人类的认识能力有限，我们认识的事物永远只是事物全部属性的一个部分，因此人类无时无刻地都在使用“抽象”这个工具。人类认识事物有一个特点，就是使用特有地符号系统。早期的人类使用口头语言我们无法得知是什么声音，但是那个时期各种象形文字，我们却可以从考古发现中找到很多。象形文字最终发展成为今天的书面语言符号系统。语言文字作为一种特定的符号，用于记录和交流人的思想。对于事物异同关系的运筹，往往是通过符号系统的操作进行的。如果这类操作可以变成数字计算在计算机上进行，那么计算机就可以作为一种信息处理工具。

为了更好地使用符号系统来记录和交流思想，人们开始分离思想地逻辑结构和表达形式，相应地出现了所谓的“逻辑标记”和“表象标记”。逻辑标记用来描述文档的结构，例如一本书具有封面、扉页、版权页、目录、章节、编号列表、参考文献、索引、封底等等，都可以逻辑标记加以标记和区分。我们可以很容易地将书的结构抽象成为一棵“树”。显然，不同主题的图书具有不同的编辑结构，字典和诗集就具有不同的主题，因此具有不同的编辑结构。编辑结构确定的对象中总是由“字符”来表达（准确地讲，应该是“字汇”，一首诗歌总是一些字词的集合，尽管写出这些字词的组合需要作者具有诗人的气质、天才和灵感）。

即使相同的字符，却有不同的字形，例如汉字的常见字形就有宋体、仿宋体、楷体和黑体等。给定的字体还有字号、颜色、粗细等差别和属性。一些字体的组合就构成了页面对象（例如段落就是由一些字形的集合形成的），页面对象组合起来就形成版面。传统的出版行业中，逻辑结构（包括主题结构，编辑结构和字符）是作者和出版社编辑决定的，而表象结构（包括版面设计、页面对象和字形）是印刷厂的排字工人和排字机决定和完成的工作。

对于大型而且复杂的出版物（例如大百科全书），出版社的编辑总是使用约定的标记对作者的手稿进行标记，而出版社和印刷厂之间则约定对某种类型的出版物(document type)采用相应的样式表（style sheet）。这一分工的好处和缺点都是明显的：对于大型的出版物，出版社编辑们提交地文档具有高度地一致性和灵活性，而印刷厂则可以根据出版社的订单要求快速统一地对文档进行格式化，例如对于大百科全书或者年鉴这样的出版物，每隔一段时间出版社都会出修订版本或者新的版本，出版社如果希望以另外样式出版，则只要明确提出新的样式表即可，印刷厂可以立即完成格式化的工作，从而大大地提高了生产效率。缺点是出版社和印刷厂需要拥有一些编辑和熟练的技术工人从事为文档打标记和格式化的工作。用计算机的术语说，就是可以对文档进行“批处理”。

这一局面在计算机、特别是个人计算机进入出版业之后发生了变化。忽然间，“所见即所得”式的字处理软件变成了作者的时尚，用户们喜欢直接操作的系统，因为他们可以直接看到自己的劳动成果。这类字处理软件只允许作者直接操作表象标记，而忽略了对逻辑标记的使用。对于小型的文档，例如一封信，这种“所见即所得”的文档是可以满足需要的，但是对于大型的文档创作，这为印刷厂带来了灾难，因为每一次作者改动作品某一个地方，往往导致要对整个出版物重新格式化。

后期出现的字处理软件已经认识到这些问题，因此，“所见即所得”的字处理器后来逐渐增加了脚本语言、样式表和指向数据库的连接功能。

二十世纪八十年代末期，当英国人贝纳斯－李发明万维网（WWW）时，其出发点就是让人们在互联网上可以方便的交换文件。李在考察了多种技术方案之后，决定选择 SGML 技术的思想，并在 SGML 技术的基础上，针对网

络的特点，取出了 SGML 的一个很小的子集定义了 HTML 规范。李定义的 WWW 和 HTML 是极具创新的，因为他将两个表面看似毫不相干的领域联系在一起，他看到了传统出版和网络技术的共同点，就是都要完成文档的传输和表达：传统出版通过纸张完成从作者到读者的文档传输和表达，而万维网则是从一台计算机通过互联网向另外一台计算机传输文件并将文件表达出来。既然处理的关系转化是相同的，那么就可以采用相同的技术来处理不同的对象。

从技术实现上看，李的创意主要有三个：一是采用面向非连接的 HTTP 协议。HTTP 与 telnet 不同，文件的传输过程中互联网两端的计算机可能出现交互(术语称为“会话”)，但是在会话之间的时间间隔里，两台计算机不处于 telnet 那样的连接状态，这一点是李的首创。李对于网络上文件的定位机制发明了一套新的办法，这构成了他的第二个创新，即引入了一种在 Internet 上一种通用的定位机制，称为 URL (Universal Resource Locator)。通过这一机制，可以在网络上定位一个文件(或者称为资源，因为任何资源都可以存放在文件中)的地址，相当于“网络指南针”。第三个创新就是在文档内部使用了超文本连接技术，尽管这一技术不是李最先发明的，但是李将它作了新的处理，使得利用这一技术可以将分散在网络上的任意多个文件都整合成为一个整体显示在用户的计算机上，而这一实现机制又是通过 HTTP 协议和 URL 协作来完成的。因此，李的这三个创意是相互配合的，或者说是三位一体的。

遗憾的是，这三个极具创新的东西中的 HTML 却在由他人实现的过程中被大打折扣。李在《编织万维网》这部自传体的回忆录中提到了万维网在诞生和发展壮大过程中的一些人和事情，其中包括当时还是大学生的马克·安德森。马克在得知李的思想后，随即与一伙人开始编写一个能够在网络上获取并显示 HTML 文档的程序，这个程序就是现在大家熟悉的 Web 浏览器。[8]

前面已经提到，SGML 采用了树形结构来标记文档的逻辑结构，表象的标记则由样式语言（例如成了 ISO 标准的 DSSSL 语言）定义。HTML 是 SGML 的一个很小的子集，在提出之初，李没有为 HTML 定义表象标记的规范，因此文件中图形和图象怎样显示由用户端的软件处理。马克纠集了一伙同学开始开发这么一个用户端的浏览器，这一浏览器的名字叫 Mosaic。Mosaic 开始可以采用直观的图形用户界面来访问和显示万维网上的文件。相比之下，李的浏览器仿佛一下子成了古董。由于马克这伙人的工作创造了一个新的不倚赖于计算机操作系统的计算平台，因此一些内行的风险投资家开始注意到这一软件的重要性，最终马克被一位风险投资家相中，以他为基础成立了 Netscape 通信公司。

[8] 李本人是一位天才的程序设计大师，他原来在 NeXT 计算机上用 Objective C 搞过很多程序开发(包括最初的 Web 服务器和一个 Web 浏览器，但是那个浏览器只能以文本的形式显示文件。万维网联盟目前的浏览器是 Amaya，在此之前为 Arena，但是 Arena 已经停止开发。Amaya 浏览器是一个成熟的软件包，尽管仍然还在发展中，当然它早就可以显示图形和图象了。Amaya 不仅是一个浏览器，而且是一个编辑器。

早在 Netscape 成立之前，李对马克的印象就不是太好，他认为马克这帮人做起事来骄傲自大，目中无人，最让李扫兴的事就是马克等人在浏览器中加入了大量的表象标记，这样“引诱”网页的设计者们向 HTML 倾注了大量的格式化标签，从而偏离了 Web 原来源自 SGML 的技术发展方向。由于当时李还是一个不知名的人物，工业界根本听不见他的意见。

随着二十世纪九十年代初期和中期的 Web 规模的大爆炸，一些头脑清醒的人开始意识到 HTML 被浏览器软件开发商创造的表象标记“污染”之后产生的危机，这时李也已经从欧洲来到了美国，成立了“万维网联盟”（W3C），变成了名人，在工业界的影响力今非昔比。大器晚成的李现在已经是可以在网络界呼风唤雨的重量级的人物，而曾经红红火火的网景公司已经不再有什么动力，马克则基本上已经淡出了这一市场。最终万维网联盟决定推出 XML，让 XML 成为 WWW 的规范语言。

另一个值得一提的是奥莱理公司。这家出版公司二十世纪八十年代很早就与达文波特等公司一起根据 SGML 起草了一份供出版社使用的逻辑标记规范，这一套规范后来演变成为现在的 DocBook，很多出版社和软件文档的作者在使用 DocBook。DocBook 有很多竞争对手，奥莱理的失策在于商业运作方面，当人们蜂拥而至在 Web 上淘金时，这家公司却满足于出版市场的业绩，技术上不思进取，最终失掉了发展的大好机会，以被挤出了软件市场龟缩回出版市场而告终。

让我们来看看 XML 文件是一个什么样子。

```
<?xml version=‘‘1.0’’ encoding=‘‘utf-8’’?>

<!DOCTYPE html PUBLIC ‘‘-//W3C/DTD XHTML 1.0 Transitional//EN’’
    ‘‘http://www.w3c.org/TR/xhtml1/DTD/xhtml1-transitional.dtd’’>

<html>
<head>
<title>Free Software: New Game Rules</title>
</head>
<body>
<p>This is a free book under GNU Verbatim copyleft license.</p>
<p>You can copy this book verbatim.</p>
<p>The book covers the license and games under the copyleft.</p>
</body>
</html>
```

尽管 XML 现在风靡全球，但是它在技术上却是符号表达方式的另一种实现形式。符号表达式的英文名称是“Symbolic Expression”，或者简写成为“S–Expression”。Lisp 的设计思想受原子论的影响很深，在 Lisp 中，所有的东西 (程序与数据）都是由为数不多的几种原子复合构成的，最基本的原子只有两

类：一是数字，一是符号。零个或者多个原子按照先后顺序排列起来构成列表，不含任何原子的列表称为空表。由原子和列表构成的列表就称为符号表达式。

我们再回到对序关系的讨论。集合论现在是数学理论的基础，对于序关系的定义，集合论给出了清晰的解释。有两种最基本的序关系：偏序与全序。对于一个集合，定义了某种关系，它具有自返性、反对称性和传递性，那么这个集合上就存在偏序关系。对偏序关系进一步进行限定可以得到全序关系。例如，一维实数系里所有的数都可以按照大小关系进行排列，因此一维实数数系是全序的。显然，对于具有全序关系的有限集合，总存在着一个最开始的元素，然后其他的元素全部可以按照这一关系进行排序，直到穷尽集合中所有的元素。从 1 开始的某个有限的自然数的集合是计算机最容易处理的全序关系，因为 1是自然数的最前元，其他的自然数都可以按照大小关系一个个地排列起来，直到所有的元素都按照大小关系排序完毕。这时我们可以将具有这样的全序关系的集合按照序关系从头到尾排列成为一个线性的列表。符号表达式可以用来描述这种线性列表。这种线性列表可以视为最基本的泛序。

但是，符号表达式是具有非常强大的数据结构描述能力，它描述的结构不仅仅局限于线性列表。在数据结构的理论中，树的概念是非常基本的，它是一种非常重要的非线性数据结构。很早人们就发现，符号表达式可以非常方便地描述非线性的树结构。这里，布尔巴基学派的拓扑结构开始进入符号表达式中，因为树上的节点可以指向另一棵树，从而建立两棵树的连通性。

在所有的树结构中，二叉树又具有非常重要的意义。所谓二叉树，就是一棵树从根开始的每一个节点发展下来只有两个分叉。通过二叉树，我们可以将非线性的树结构转化成为线性的列表。二叉树是将树结构的符号表达式转化成为具有线性列表的桥梁。

对于以符号表达式来表示的线性列表，第一个元素往往具有特别的意义，因为波兰表达式规定第一个元素可以对后续的元素进行逻辑或者数字运算操作。这里，布尔巴基学派的代数结构开始进入了符号表达式。丘奇等人建立了完整的 λ 演算理论，清晰地刻画了代数结构中函数与参数的表达形式，其得意门生麦卡锡得到了启发，将具有严格定义的 λ 表达式带进了 Lisp 的实现中。

符号表达式在 Lisp 中随处可见，为了说明这一点，我们举上面的这一段 XML 代码的例子。如果使用 Lisp 的符号表达式来写应该是这个样子：

```
(?xml (@ (version "1.0")) (@ (encoding "utf-8")))
(!DOCTYPE html PUBLIC "-//W3C/DTD XHTML 1.0 Transitional//EN"
  "http://www.w3c.org/TR/xhtml1/DTD/xhtml1-transitional.dtd")

(html
 (head
  (title "Free Software: New Game Rules"  ))
 (body
  (p "This is a free book under copyleft license")
  (p "You can copy this book verbatim")
  (p "The book covers the license and games under the copyleft")))
```

从上面两段代码的比较中，我们可以看出 Lisp 样式的符号表达式比 XML 的表达方式在形式更加紧凑一些。如果你习惯了 Lisp 中成对的小括号，那么你可能更喜欢使用 Lisp 的符号表达式来编写网页。

数学家对其他工程技术人员从事的工作可以从本质上抽象为“辨异同、排泛序”。基于异同关系的“辨异同”可以描述事物之间的相同与差异，基于序关系的“排泛序”可以帮助我们理清数据和算法。

XML 公布之后，旋即被应用到各行各业。原因在于需要根据各行各业的特点来定义符号表达式中第一个函数元素的语义。例如 XSL 是基于 XML 制定的样式语言，它就对样式表语言中各种操作进行了完整的定义，现在，人们可以根据 XSL来指定各种各样的样式表，并且利用自己喜欢的语言去实现 XSL的操作。

XSL–FO 是使用 Java 语言编写的格式化对象，Java 语言已经含有丰富的类库，其中 Java 2D 可以为平面作图提供全部所需要的函数和类库。XSL–FO 可以将含有样式表的 XML文档格式化成为 sosofo，也就是一连串的格式化对象的序列，现在在浏览器上或者在纸张上打印出来。

忽然间，基于 XML 的 XSLT 成为了一种新的格式化工具，有人甚至断定它可能替代原有的 TEX 和 PostScript 技术。笔者认为这是有一定可能性的。不过，目前基于 XML 的格式化工具还远远没有达到像 TEX 或者 PostScript 技术那样的成熟度。一个更加理想的方案是将这些技术结合起来，利用 XML 为文档打标记，然后让 TEX 或者 PostScript 作为格式化工具的后台引擎。这已经不是一个简单的技术性设想，而是排版技术界许多人正在努力的现实。

Schema

为什么说基于 XML 的排版技术目前仍然尚不够成熟呢？

具体的技术原因很多。突出的一点在于 XML 文档总体上分成两个部分：元数据和数据。元数据是说明数据的数据。我们中国人在学习外语时，我们的母语（汉语）扮演了元语言的角色。对于我们不懂的外语，我们总是找一本具有汉语说明的教科书来说明外语的意思，包括语义、语法等各个方面。XML也与此类似，以往需要通过 DTD（文档类型定义）来描述。DTD 优点类似我国历史上科举制度的八股文，它用来说明文档的结构，以及数据的类型。这一想法原意是好的，但是问题出在不同的人在写作 DTD 时采用了不同的写法，导致了大量的 DTD 之间转换困难。

为了解决这一问题，W3C 提出了 Schema 规范。按照 W3C 的解释，只要按照这一规范编写元数据，那么，XML 文档就具有相当高的移植性。但是 Schema 规范发布的时间不长，深入人心尚有待时日。

李发明的万维网之前有两个技术上的梦想：第一个梦想是人们可以通过万维网实现信息共享，万维网成为人们相互合作的强大工具。应该说这一梦想目前已经实现了，现在任何一位作者(只要他愿意)都可以通过万维网让作品与全球的读者分享。万维网的另一个梦想是将人与人之间的合作延伸到计算机与计算机之间，目前这一梦想还没有实现。李的基本思路是实现一种语义的万维网 (Semantic Web)。关于语义万维网的看法和观点有多种，简单地讲就是让计算机具有更多的智能。这一个梦想实现起来的难度比实现第一个要大很多，目前万维网联盟正在努力朝这一方向迈进，不过我们还难以看出万维网联盟已经理出了一个什么头绪。

W3C 规范的版权

W3C 提出的万维网模型是构建在 Internet 之上的，而 Internet 的技术规范采用了一种称为 RFC 的方式制订，它与标准化组织制订标准的过程不同。标准化组织一旦制订出一项标准，则往往具有某种强制力。但是 RFC 的方式则由某个技术专家或者技术组织提出一份草案，供公众提出修改意见，定型后供大家共同遵守。初看起来这种技术文件好像不如标准的力度大，但是以往的一些实践表明，在很多情形下工业界对 RFC 的遵守程度大大高于对 ISO 之类的标准的遵守程度，著名的 Internet Email 规范是定义在 RFC–821 中的，它定义的 SMTP 协议在互联网上击败了由 ISO 制订的 X.400 电子邮件标准。

WWW 工作协议称为 HTTP，最初工作于 UNIX 平台的第 80 号端口上[9]，使用第 80 号端口利用 TCP 协议进行数据传输，每个 TCP 分组交换数据包中就是按照 HTTP 组装的 HTML 或者 XML 数据。自从“万维网联盟”(W3C)成

[9] 互联网是诞生在 UNIX 平台之上的，笔者认为每一个从事计算科学和计算机工程技术的人应该理解和学会使用 UNIX 系统，就目前而言 GNU/Linux 已经完全实现了它的所有功能，而且做得更多更好。

立之后，它变成了制订所有 Web 规范的一个组织，而是是各种技术规范的版权所有者。

简言之，W3C 组织的宗旨就是让 WWW 成为信息交换的一个平台，它制订的规范是可以自由使用的，万维网联盟将万维网的协议、规范等置于了公众领域(public domain)。这一点体现了李的宽阔胸襟：当大多数人想如何使用万维网为自己挣钱时，李考虑的却是如何让万维网为大家更好地使用和服务。不过因为没有采用 GPL 之类的自由软件许可证协议，在普及万维网时难免出现一些以前发生过的类似法律问题。随着 WWW 成为新的计算平台，一些大的公司开始支持 W3C，这一变化既有好的影响 (—W3C 获得了工业界更加广泛的支持)，也有一些消极因素—因为这些公司支持 W3C 行动的背后往往隐藏着各自的商业目的，而万维网联盟采用的 RFC 类型的规范制订方法很大程度上倚赖于工业界的行动的总体结果，如果某天 RFC 的制定者改变了主意，决定更改版权的授权方式，则社团可能要收到相当大的影响。以往这种情况不是没有发生过，Gopher 和 Java 语言规范的许可证就出现过这种情形。

有鉴于此，自由软件社团经常注意着 W3C 的举动。实际上，在自由软件基金会和万维网联盟之间就曾经传出一些不和谐的声言。例如 2003 年，在自由软件基金会的网站上就发表了一份声明，号召自由软件社团对 W3C 一项提案提出质疑，现摘录如下：

Our Position

The Free Software Foundation, represented by its General Counsel, Prof. Eben Moglen of Columbia University Law School, participated in the W3 Consortium Patent Policy Working Group from November 2001 through the current Last Call draft. The Foundation regards the current Last Call draft, which proposes the adoption of a "royalty–free" or "RF" patent policy, as a significant step in the direction of protecting the World Wide Web from patent–encumbered standards. But the proposed policy is not an adequate final outcome from the Foundation's point of view.

The proposed policy permits W3C members participating in W3 technical working groups to commit their patent claims "royalty–free" for use by implementers of the standard, but with "field of use" restrictions that would be incompatible with section 7 of the GNU General Public License. Such "field of use" restrictions, in other words, would prevent implementation of W3C standards as Free Software.

Section 7 of the GNU GPL is intended to prevent the distribution of software which appears to be Free (because it is released under a copyright license

guaranteeing the freedoms to use, copy, modify, and redistribute) but which cannot, in fact, be modified and redistributed because of patent license restrictions that limit the use of patent claims practiced by the software to a particular purpose. Though other Free Software licenses may not happen to contain provisions equivalent to GPL's Section 7, this does not imply that programs released under those licenses will be Free Software if the patent claims contributed "royalty–free" to the standard those programs implement are limited to a particular field of use.

As an example, W3 members may contribute patent claims to a standard describing the behavior of web servers providing particular functionality. A Free Software program implementing that standard would be available for others to copy from, in order to add functionality to browsers, or non–interactive web clients. But if, as the present proposed policy permits, the patent–holder has licensed the practicing of its patent claims "royalty–free" only "in order to implement the standard", reuse of the relevant code in these latter environments would still raise possible patent infringement problems.

For this reason, the proposed policy does not actually protect the rights of the Free Software community to full participation in the implementation and extension of web standards. The goal of our participation in the policy making process at W3C has not been achieved. The Foundation urges all those who care about the right of Free Software developers to implement all future web standards to send comments to the W3C urging that the policy be amended to prohibit the imposition of "field of use" restrictions on patent claims contributed to W3C standards. The address to which such comments should be emailed is <www–patentpolicy–comment@w3.org>. The deadline for receipt of comments is Tuesday 31 December 2002. Further Non–Legal Explanation of Position

Many in the community have requested some additional explanation of FSF's objections to the policy. We have added them below

FSF's objections center around Section 3 of the W3C's proposed patent policy. Item 3 of that section says that the royalty–free license "may be limited to implementations of the Recommendation, and to what is required by the Recommendation". That is a "field of use" restriction.

The problem is the interaction of such a "field of use" restriction with Section 7 of GPL. Under Section 7, the "field of use" restriction is a "conditions are imposed on you [the distributor of GPL'ed software] that contradict the conditions of this License". The "conditions of this license" require, for

example, that those receiving distributions of GPL'ed software have the right to run the program for any purpose (Section 0), the right to modify it for any purpose (Section 2), etc. Any of these "purposes" could easily practice the teachings of the patent beyond what the "field of use" restriction allows.

Here's a detailed step–by–step example that shows how this problem could play out:

1. Programmer P downloads the Konqueror web browser, receiving it under terms of GPL.

2. P learns of a new web standard that requires exercising a technique for parsing URLs that is patented by Corporation C. C has licensed the patent under an RF, non–exclusive license, but with a "field of use" restriction that says the license can be used to "implement the standard". The standard, as it turns out, covers only what browsers must do with URLs, and says nothing about the server side or clients that aren't user browsers.

3. P implements this technique in Konqueror, and seeks to redistribute the modified version on his website so that other users can benefit from Konqueror now complying with the standard. If he does, he is bound by the GPL under copyright law, because he is redistributing a modified version.

4. However, he knows full well of a condition on that code that contradicts the GPL (violating Section 7) – namely, he knows that C's patent license prohibits folks from taking his URL parsing code and putting it into, say, a search engine. Therefore, under GPL Section 7, he is prohibited from redistribution.

5. You might think that he can simply assign his copyright to the existing copyright holder of Konqueror let distribution happen from that source. They could distribute under GPL, but they would be granting a self–contradicting license. Nothing prohibits someone from distributing copyrighted works under licenses that make no sense and are self–contradictory. However, it is certainly true that those who receive distribution of the works are stuck and can't undertake further distribution or modification themselves.

Thus, regardless of who makes the changes, the result either shuts down distribution or forces the original developer to abandon GPL. Both outcomes

are very unfortunate. This is why we encourage you to write to comment on the Last Call Draft.

自由软件基金会显然希望 WWW 这么一项伟大的技术发明成为自由发布信息和传播信息的工具，而不希望这一工具本身受任何专有软件或者专利的约束，与自由软件的精神背道而驰。在目前的状态下，Web 现在是自由的，但是这不能保证未来的局面也是这样。

商业模式

基于自由软件的排版服务的商业模式从一个方面讲已经相当成熟，从另外一些方面看又是一个新兴的行业，一个主要原因在于传统的出版行业经过几百年的发展，业已是一个相当成熟的行业，基于 TEX、PostScript 和新兴的 WWW 的技术既与之结合得很好，又几乎彻底地改变它原来的面貌，并开始左右着这一工业将来的走势。在这种局面下，成熟的和有待于探索的商业模式都是非常丰富的。

模式一：培训技术人员

正如前面提到的，TEX（以及 METAFONT、METAPOST）、PostScript 和 XML 都是非常复杂而灵活的技术。尽管这些技术学习起来入门容易，但是不经过刻苦学习和积累经验，一般人是难以成为一名资深专家的。如果在刻苦学习的前提下，配合专家的导引，那么学习的曲线可以大大缩短。自由软件基金会中国研究院推出的“黑客道”培训课程中有一门课程，名称是“软件文档创作”，就有这方面的内容，有志趣的读者可以参加这方面的培训[10]。与教育有关的商业模式我们已经在本书中详细探讨过了，在此不再赘述。

模式二：新型编辑人员

人类社会步入网络时代后，从技术层面上讲，在作家与读者之间共享知识和信息已经不需要出版社的介入了，因为互联网上可以在作家和读者之间立即形成直接关系，例如您现在阅读的这本书的 HTML 和 PostScript/PDF 文件就可以直接从笔者的网站上（http://www.rons.net.cn）下载到您的计算机上，使用 Web 浏览器 (例如 Mozilla) 阅读，或者使用 Ghostscript 的 Ghostview 或者 PDF 文件阅读器来看，或者使用打印机将文件打印到纸张上来阅读。作家与读者的关系又回到了出版工业诞生以前的形态。作家的作品可以公布在互联网络上，供读者利用 Web 或者 FTP 协议下载，作家也可以利用电子邮件来传播作品，或者利用邮件列表软件包来建立自己的读者群体，

[10] 参见本书“黑客道与教育传统的复兴”一章。

作者还可以在 Web 网站上开设讨论区收集读者的反馈意见，发布更新版本，等等。知识和信息的传播成本大大地下降了。

不过，现实中的情况比这要复杂许多。一个问题是计算机和计算机网络对于信息的搜寻、分类、重新加工等非常方便，但是在计算机屏幕上长时间阅读却是件令人痛苦的事情。我们当中的绝大多数人还是喜欢阅读纸媒体的图书，通过阅读书来获取知识和信息。纸媒体的图书可以随时随地阅读，可以在书上自由地圈点或标记，这些好处在计算机屏幕上是不能得到的。因此我们还是需要有纸媒体的书。作家往往没有这种兴趣花费时间和金钱去印刷，而出版社可以承揽这种业务。

即使将来人们发明类似于纸张的新材料，使我们在阅读这种记载在新材料上的知识或者信息时感到非常方便和舒适，还有一大问题需要考虑。

由于 SGML/XML 的流行，出版社中编辑人员的工作性质发生了很大变化，由于越来越多的文档将采用 XML来出版，而许多作者没有时间或者精力来处理打标记之类枯燥乏味的工作，因此需要有这方面的专门编辑人员来从事这类工作。

模式三：排版服务

由于信息不对称性的存在，因此不大可能每一个人都有时间和精力去学习和掌握使用 TEX、PostScript 或者 XML 等技术的细节，而在发布信息时，信息的格式表达可能是非常重要的东西（很多场合下，格式本身就是在传递某种信息），因此在信息发布者和信息接受者之间形成了隔离地带，这一隔离地带可以靠专业的技术人员的服务来填补。

模式四：对硬件的技术支持

对于从事印刷的企业，由于 PostScript 技术设备的广泛采用，因此需要内行的技术人员来参与生产设备的操作、管理，其中软件人员的作用是不可忽视的，在生产中，合理地使用软件人员，可以大幅度地提高设备的利用率，创作更多的经济利润。

在生产中创造更多利润的一个简单诀窍就是尽可能地让设备满负荷运行，扣除设备维护、检修等时间之外，如果设备总是处在正常的工作状态，那么设备的利用率就高，其价值体现也就越充分。有些用户总是害怕计算机太辛苦，尽量少用计算机以延长计算机的使用寿命，（不知当初购买计算机的目的是什么?）而计算机硬件的发展是非常快速的，因此正确的使用观念是在有效的时间内尽量多地使用计算机，让它为你创作更多的利润，除了收回成本之外，更让它为你带来更多的回报。

目前有一个普遍存在的问题,很多计算机用户收到商业广告传递的隐形误导,在购买计算机设备时盲目追求高档次硬件配置,而忽略了软件的作用,结果造成很大的浪费。今天个人计算机用户们主要使用电脑进行文字处理、上网通信、电子表格、游戏等简单信息处理,这些简单的信息处理任务对于 32位的个人计算机来说是很容易快速完成的。举个例子,在朋友之间发送电子邮件时,如果没有图片,那么使用最简单的文本就足够表达思想进行沟通了,没有必要使用格式化的格式来加大电子邮件文件的尺寸。减小文件尺寸一般意味着电子邮件可以在网络上更快得到传送。即使被淘汰的 386 档次的个人计算机,也可以变废为宝,通过安装自由软件,改造成为服务器。

笔者曾经为一位朋友在他的 386 计算机上安装了一套电子邮件服务器,通过增加一块便宜的网卡,这台电脑就具备了每天可以为他投递十万份电子邮件的能力[11]! 这一测试结果让这位朋友大感意外,他根本没有想到一度被他遗弃的老旧电脑居然还有这么大的潜力可挖掘。如果你细心观察周围的计算机使用情况,就不难发现绝大多数硬件的计算潜力没有发挥出来。

这位朋友淘汰这台计算机的原因是它不能运行当时刚刚发行的 MS–Windows 95 操作系统,运行 MS–Windows 95 需要至少 8MB 内存,即使具有 8MB 内存,计算机的性能也还是不够理想,因为运行具有图形用户界面的应用程序需要使用大量的内存,而他的那台电脑已经无法扩展内存了,于是他又买了一台新的电脑。像 MS–Windows 这样的专有软件,因为没有系统的源代码,因此无法剪裁定制,以使之可以运行在原有的计算机硬件上。

如果说一台个人计算机被闲置造成的浪费还比较小的话,那么一套工业设备闲置的浪费长期累积下来造成的浪费就相当惊人了。笔者曾经看到某家工厂花费数百万元巨资购买了 VAX 机器,但是购买回来之后一直在机房里睡大觉。这家工厂舍得花钱购买硬件,但是却舍不得花很少的钱开发应用软件,这种做法真是令人费解。

绝大多数的 PostScript 硬件设备都是昂贵的,因此软件人员的技术支持是具有实实在在的意义的。前面提到过,不同的人员生成的 PostScript 代码在运行时效率的弹性很大,有时哪怕是作出小的改动也会大大提高程序的运行效率,对于昂贵的 PostScript 设备而言,如果在固定的单位时间内可以打印输出更多的文件,那么硬件产生的效益就会增加很多。

关于对硬件的技术支持,下面的“按需打印”一节还会谈到。

模式五:开发中文字库

前面提到过,汉语排版的数学模型比西文要简单,但是这不等于其开发工作的工作量很小,实际上,由于汉字的数量众多,因此汉字字库开发的工作量

[11] 当然,我已经告诉他不要利用这一电脑对外发送任何电子邮件垃圾。

是相当惊人的。光有一种字库在实际中是不够的，至少需要四种字库才能使排版效果比较美观。

另外，中国是一个多民族的国家，除了广泛使用的汉语之外，兄弟民族的语言文字种类也很多，使用的人口也不少。例如藏语、蒙语、彝语等就是与汉语差别很大的语种，其字形设计和排版规则等与汉语就完全不同。在网络时代更有大量的技术问题亟待解决，这些都为自由软件的开发人员提供了广阔的发展空间。

考虑到中文字库在信息技术中的基础性地位，而现在没有真正自由版本的中文字库，“一、百、万”工程知难而进，以汉字字库设计作为工程的起点，笔者希望社团内部有更多的人开发自由字库，因为我们的确需要很多很多的字库才能是我们的文档和出版物看起来更加美观，更加富有表现力。

模式六：按需打印服务

“按需打印”（Print–On–Demand，POD）是最近几年出现的新技术。这里的“按需”是指按照需要的拷贝数量打印文档和出版出版物。有专家进行过统计，80% 以上的文档的印刷数量少于 500册, 有些技术会议的会刊资料可能连 100 本都用不到。真正可以达到 3,000 册的文档可以按照书出版发行的情况并不多见。为了解决这一矛盾，POD 技术应运而生。下面就是这么一种 POD 设备的图片。

没有联网的 POD 生产线

已经联网的 POD 生产线

图 4 按需打印生产线

目前大多数 POD 生产流水线还没有与互联网连接上，即使联网的 POD 设备，由于网络连接速度较慢，其优势还没有充分发挥出来。随着高速宽带互联网络的普及，在网络上传递大尺寸的文件将非常快，因此任何人可以通过网络投递打印任务给 POD 服务商，并从最近的 POD 站点完成打印任务，提交给用户。

当然这是中远期的设想，目前绝大多数 POD 生产厂家来自德国和瑞士等国，其设计和制造技术是相当先进的，因为这两个国家的精密机械设备的制造工艺水平是世界一流的，展览会上经常可以看到其设备，但遗憾的是目前展出的设备成本太高。如何进一步降低成本，是按需打印技术目前需要认真解决的问题。基于 POD 技术的研究和开发，可以开发一种集机械、精细化工、光学、电子和自动化控制技术于一体的新制造业。

模式七：支持自由软件社团

上个世纪八十年代末期和九十年代初期，TEX 曾经在国内科技界流行过一阵子，但是由于中文支持跟不上，加上 MS–Windows 的 MS–Word 之类的专有文字处理软件包的流行，TEX 渐渐地淡出了，其用户数量越来越少。

笔者自 1995 年开始初次接触 TEX，并立即迷恋上这一系统，并试图在这一软件包上加上中文支持，笔者从 1999 年开始在国内开始推广应用 TEX，这一行动后来演变成为向所有的中文 TEX 用户提供技术支持。2000 年正式组建了“中文 TEX 用户俱乐部”(Chinese TEX User Group)。这是迄今为止世界上的第一个中文 TEX 用户俱乐部，因此它有时也被称为“中国 TEX 用户俱乐部”，英文简称 CTUG。

CTUG 是一个依靠“武汉荣世数据通信有限责任公司”支持办起来的开放组织，开设初期的一些经费是该公司提供的，该公司业务中有一项就是排版。通过对 CTUG 的支持，该公司从 CTUG 的会员中物色到了几位高水平的 TEX 排版人才。这里展示了商业公司如何支持自由软件社团组织建设的一种成功的商业模式。今天，CTUG 在会员们的共同努力下，规模已经变得较大，而其赞助商因为获得了高水平的人才，其服务对象已经扩展到国内多个地区及海外多个国家，对 CTUG 回馈的资助比初期更多了，回馈的资金主要用来供 CTUG 开发自由的汉字字库。

2002 年 8 月，“国际数学家大会”（ICM）在中国举行，这是 ICM 历史上第一次在欧美之外的国家举办。会议期间，来自各国数学家几乎清一色地使用 TEX 来撰写论文、交流思想、展开讨论。会议上的投影展示使用的文件都是使用由 TEX 生成的 PDF 文件（基于 PostScript 技术的文件格式），中国数学界的众多学者通过这次会议重新开始认识到国际数学界的公认写作标准，很多人表示将学习和采用 TEX 来发表文章，而停止使用专有的字处理软件。中国 TEX 用户俱乐部派人参加了这一国际数学界的盛会，并在 ICM2002 的展览会上向国内外数学家们展示了自己，吸纳了许多新会员。

CTUG 目前正在进行的开发项目除了设计自由的汉字字库之外，还有利用 Scheme 语言重新编写一个类似 TEX 的排版系统，称为 NeoTEX。原有的 TEX 系统是编译型的系统，每一次排版(除非一切正确无误)都会包括三个阶段：编辑 ⇒ 编译 ⇒ 修正，每一次这三个步骤循环下来，一直到用户得到满意的排

版结果为止。这一模式对于专家型的用户工作得很好，但是对于一般的用户就显得不那么直观方便了。

在笔者启动这一开发工程之前，早就有人注意到这一问题并进行了卓有成效的探索，例如荷兰的黑客 Joris van der Höven 就受到 GNU Emacs 和 TEX 两个软件的启发，开发了 GNU TEX$_{MACS}$，这是一个很了不起的软件包，他已经很好地揉合了 Richard Stallman 和 Donald Knuth 两位天才程序设计大师的思想，而且走得更远。他在 GNU TEX$_{MACS}$ 中体现的创作意图是在“所见即所得”的图形用户界面下编辑结构化的文档，目前程序的输出可以达到印刷级别的高级排版质量，TEX 的字库也可以投入直接使用。它下一步的目标是继续完善其自动化生成数学公式的功能，成为一个计算机代数系统的界面。

最有趣的地方就是 GNU TEX$_{MACS}$ 使用了 Scheme/Guile 语言来作为新软件功能的扩展语言，在这一点上，它与著名的图形处理软件包 GIMP 的设计思想是非常相似。[12] Scheme 是解释型的函数型编程语言，Scheme 解释器可以一边阅读输入，一边求值计算并输出结果，因此可以将整个排版系统设计成为一个交互式的系统。

NeoTEX 的基本设计思想与 GNU TEX$_{MACS}$ 是大致一致的。不过，NeoTEX 将是 MING/OS 上的一个构件，而且是一个非常重要而基本的构件。通过 NeoTEX，计算机用户可以向计算机和计算机网络提出问题，观控计算机和计算机网络上的资源，因此 NeoTEX 是一个网络浏览器软件包。

在技术实现上，NeoTEX 沿用 *NTS* (New Typesetting System) 系统中的基本算法，在 `WEB2C` 代码的基础上，将其 C 代码用面向对象的 Objective–C 改写为多组尾递归的原子性例程，可以作为 Scheme 的原子函数使用(这一工程称为 `WEB2SCM`)，而输出的默认格式为以 DVI 为基础，结合 Scheme 的符号表达式和 Scheme 的绘图函数而设计的 .nts 格式。操作系统底层的显示硬件的驱动程序是利用直接汇编语言代码编写的，而且是 Scheme 的原子例程的一个组成部分。尽管在设计这些原子例程时大量借鉴了 X Window 系统和 MIT Scheme 中绘图函数的设计思路，但是它仍是一个原创性的作品，运行 NeoTEX 并不需要这些软件包。

以今天的眼光看，DVI 文件格式是类似 Java 的 bytecode 的中间格式，只要有底层硬件支持的接口，那么 DVI 文件就可以在硬件上输出。.nts 文件将 DVI 对象化，以模块的方式封装了 DVI 代码, 并对 DVI 的指令序列作了扩充，文件格式中还含有元数据。这种新的格式融合了 XML 和 PDF 两种文件格式的优点，来自 XML 的优点是通过符号表达式体现的[13]，包括文本、图形、声音、

[12] 关于 GNU TEX$_{MACS}$的详情，可以访问其网络主页 http://www.texmacs.org。

[13] 基于符号表达式来表示排版和生成图形的思想曾早在采用 Lisp 的 Interleaf 和 AutoCAD 系统中就出现过，但是这些排版系统却是专有软件。

图像、电影等文件的嵌入、合成、链接等；而来自 PDF 的特点在于格式可以体现作者的表达[14]。

当然，除了 .nts 格式，其他可能的输出格式还有 PDF、PostScript、SVG 等。关于 NeoTEX 与 MING/OS 之间的关系和其他相关信息，读者可以参考本书的《“明”操作系统的设计思想》一章。

模式八：数字空间与电子图书馆

目前学术界有一个突出的趋势，即越来越多的学术著作是以“数字方式诞生的”，正如本文前面提到的，很多作者熟悉利用 TEX、METAPOST 等自由软件，他们在写作时直接使用这些软件来搞创作，得到的电子文件可以直接放置在 Web 网站上，而不是使用容易分类和扫描的纸张。

美国麻省理工学院 (MIT)一向以勇于探索宏伟思想著称，它已经开始着手创办一所几乎涵盖全院学者和研究人员全部脑力劳动成果的长期“数字图书馆”。据美国《环球报》2002 年 11 月 4 日报道，这个图书馆称为 DSpace，是麻省理工学院与惠普公司的合作项目，它将保存浩瀚无垠的数字信息，收藏的内容从课堂讲座录音到实验、大脑扫描、海底勘探乃至星际太空探测，无所不包。麻省理工学院希望牵头成立一个全球性质的大学联合会，利用 DSpace 技术建成图书系统，实现全球联网计算机共享学术信息。

截止到 2002 年 11 月，德国注册的普通公共图书馆有 11,322家，科学图书馆有 1,268家，2001 年支出达 14 亿欧元，名列世界第一；美国共有 9,000 个公共图书馆，全球排名第二；英国共有 5,000 家公共图书馆以及面向学生，研究人员和官员开发的资料档案馆；法国的市立图书馆有 2,795 家，从业人员 22,748 人；瑞典也有 1,500 家公共图书馆，2002 年的费用增加到 3.3 亿欧元。中国的邻居俄罗斯和日本的公共图书馆的数量和藏书量也都名列世界前茅。可以设想一旦这么多的图书馆实现互联，我们可以在互联网上可以检索查阅的资料将有多么丰富！

公共图书馆的费用一般由政府承担，因此其最终来源是纳税人交纳的税款。通过向社会提供信息服务，可以极大地丰富全社会的文化生活，有助于提高全社会的生产力，自由软件的哲学要求信息可以自由共享，通过互联的图书馆可以将传统的资源与新的数字化资源联系在一起，并创造大量的就业机会，其前景是非常光明的。

[14] 关于这一文件格式的详细情况，可以参见笔者在《Free Software》Vol.1，Issue 5 上发表的文章。

出版行业在中国

中国古代有闻名于世的四大发明：指南针、纸张、印刷术和火药，其中前三项技术都是与信息相关的—指南针提供关于方向和位置的信息，纸张用来记录文字和图形信息，而印刷术用于大量复制信息。直到十五世纪以前，中国古代文明之所以在长达两千年的时间内在世界范围内遥遥领先，信息技术的高度发达恐怕是一个很重要原因。

但是，这一切都些都已经是很遥远的过去的辉煌。自从明清开始，统治阶级的愚民政策和弱民政策禁锢了人的思想，历次人头落地的文字狱迫使读书人只能在钻进故纸堆里梦幻功名，再也没有人真正能够潜心研究科学技术上的问题。中国的科学技术之所以在近代历史上落后于西方，这中间的道理是可以一目了然的。

印刷术和纸张最早都诞生在中国，但是我们不得不遗憾地承认我们的印刷和出版技术已经大大落后了。不仅是在技术层面上，我们难以提出新的思想，而且整个社会对于创新也缺乏有效的激励机制，很多地方还存在种种限制创新的人为障碍。

新中国成立之后，特别是从二十世纪八十年代以来，中国开始走上了民主法治的道路，人民生活水平和民主权利空前提高，这是有目共睹的事实。但是，这并不意味着宪法中规定的人权全部都真正完全实现了。现实生活中，中国大陆目前实施着历史上(也许是世界上)最全面的商业许可证制度，公民的一切行为几乎都要经过政府部门的审批。[15]

可以严格地讲，目前的出版制度没有完全体现出宪法规定的对公民权利的保护。根据宪法，每一个公民都有言论自由、出版自由，这是一项基本的人权。而现实情况是并非任何一个公民都可以从事自由的出版活动。

国父孙中山先生曾说，*当今世界发展潮流，浩浩荡荡，顺之者昌，逆之者亡*。清朝统治者因为实施防民之口甚于防川的愚民和弱民政策，最终导致亡国。后人哀之而不鉴之，亦使后人而复哀后人也。

以笔者之管见，互联网络的兴起有可能为开启民智的一个契机。在加入了WTO 的今天，中国与世界已经紧密联系在一起，网络技术正在加速这一整体化进程，建立在自由软件哲学和技术基础上新型出版业在中国的前途可能是无限辉煌的。

May 01, 2002 第一版初稿；

[15] 当然，有些商业行为需要从业者需要具备一定的资质条件。为了保护公众的利益，政府实施相应的资质量审批是应该的，也是必要的。不过这不应该是限制公民言论和出版等自由的理由。

Oct 01, 2002 第一次修改；
Dec 18, 2002 第二次修改；
Dec 27, 2002 第三次修改；
Mar 08, 2003 第四次修改；
Apr 20, 2003 第五次修改；
May 09, 2003 第一版定稿。
Aug 12, 2003 第 1.1 版。
Aug 27, 2003 第 1.2 版。