8. 索引自动化 217
8.1. 索引相关概念 217
8.1.1. 索引项（Index Entry） 218
8.1.2.索引主项（索引第一层） 220
8.1.3.索引次项（索引第二层、第三层……） 220
8.2. 以“索引词条文件”批量标记索引项目 220
8.2.1. 索引词条文件的建立 221
8.2.2. 让 Word 自动化建立索引 222
8.2.3. 自定索引形貌 224
8.3. 逐一手工标记索引项 225
8.4. 索引佳例示范与讨论 227
8.5. 本书索引制作过程 229

8 索引自动化

8.1. 索引相关概念

索引是什么？精于看书的人都知道它并且善用它。这方面我就不多说了。

索引有多重要？那就不一定，取决于书籍的性质。通常科技类书一定要有索引，才有更高的工具书价值[1]。图书馆收藏图书时也会特别列出一条：该书有无索引。任何英法德俄……科技类书的版权页（扉页之后列有出版信息、ISBN 号码、版权声明、刷次日期……信息的那一页），都会写出该书有无索引。例如《*Design Patterns*》的版权页上写的是：

Includes bibliographical references and index.

并非任何大型文档都需要索引。厂商的技术报告、产品使用导引、业绩报告、财务报告等等，就其性质和价值和规模而言，也许并不需要索引。但至少书籍和论文一定要有索引。书籍的索引是个大工程，论文的索引相较之下轻松一些。

索引的完成必须有以下数个条件的配合：

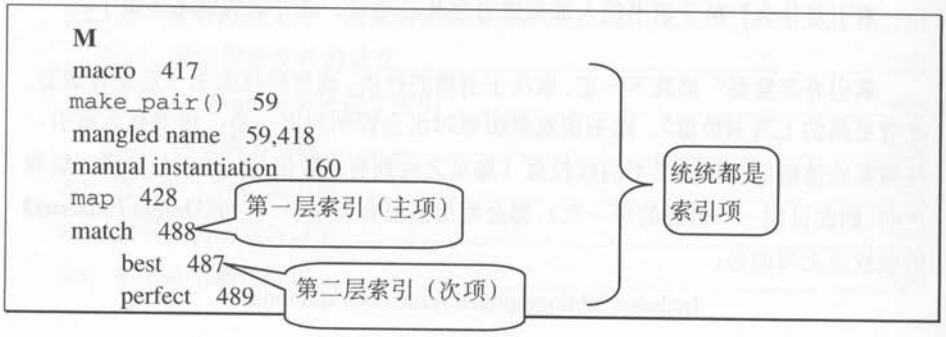
1. 排版软件必须有制作索引的功能。只要号称排版软件，都应该有这个能力！Word 有这个能力☺
2. 排版人要有操作上述功能的能力。制造索引的确是比较高级的操作，但相关信息太少也是让排版人“视索引为畏途”的主要原因。如果学习资源够多，排版

[1] 但是台湾许多科技类书（例如相当大宗的计算机技术书籍和电脑应用书籍）并不怎么重视索引。可能的原因包括：(1) 排版人不会制作索引、(2) 创作人嫌麻烦、(3) 创作人和排版人之间没有顺畅的沟通、(4) 因循旧习、(5) 中文并非拼音文字，中文索引有其先天缺陷（只能以笔划顺序为之）、(6) 以上皆是。

人制作起码的索引并不太困难。

3. 文档创作者必须对索引有点概念。这个不难，没吃过猪肉也该看过猪走路。但除了概念之外还必须有能力落实一份具体的“索引词条表”（对 Word 而言；详见 8.2 节），这比较是个挑战。
4. 目前的排版软件在索引制作方面尚未融入太多的人工智慧（只是想当然尔，我并未遍历群贤），所以制作出来的索引只是硬梆梆的页次列表而已。要让索引有高超的价值，文档创作者还必须在这种硬梆梆的初级品上面加工。8.4 节有一个索引佳例，可供大家借鉴与思考。

通常，最简单的索引长相如下：



为了做出这份索引，作者必须在书稿中标出上述每一个词条，告诉排版者“这个词条将成为索引的一分子”。这些词条就是 Word 所谓的索引项（index entry）。上图索引中的 "match" 词条下有第二层索引，其中 "match" 即是 Word 所谓的主索引项，第二层的 "best" 是次索引项，"perfect" 也是次索引项。

文档创作者必须告诉排版者以上所谓的索引项、主索引项、次索引项，排版者才能够在文档中设置它们。

8.1.1. 索引项（Index Entries）

索引项其实是一个域（见 9.3.9 节）。或者说，Word 以一个域表现一个索引项。

这个域的代码是 XE（想来应是 "indeX Entry" 的缩写），格式如下[2]：

```
{ XE "Text" [option] }
```

举个例子，如果您将 "generic" 标记（*mark*）为索引项（操作手法于 8.3 节详述），而且没有任何特殊设置，那么 Word 便在该词条之后（紧邻）放上一个这样的东西：

```
{ XE "generic" }
```

这东西是隐藏的，必须点击常用工具栏上的 ¶ 才能看到它。

XE 域的另一种格式是：

```
{ XE "Text:SubText" [option] }
```

其中的 `"Text:SubText"` 将在 8.1.3 节解释。

XE 域的选项（Options）

XE 域的选项（options）有以下数种。（阅读前请先建立一个观念：文档可拥有多份索引；【插入➔索引和目录(D)】+索引(X)即可做出一份索引）

- `\b` 在索引项所得页码上套用粗体格式。例如`{ XE "arrays" \b }` 会显示 **arrays 56**。如果索引项的相关样式（索引 1-索引 9）是粗体，则此选项会移除粗体格式。可与 `\i` 同时存在。
- `\f "Type"` 定义索引项的类型。这些类型只会被“也指定了相同类型”的 INDEX 域（9.3.8 节）取出制作成索引。例如 `{ XE "access" \f "a"}` 指定的类型是 a，那么只有 `{ INDEX \f "a" }` 才能抓出那个索引项制作成索引（也许我们可以把这样做出来的一份索引称为“a 类型”索引）。
- `\i` 在索引项所得页码上套用斜体格式。例如`{ XE "arrays" \i }` 会显示 *arrays 56*。如果索引项的相关样式（索引 1-索引 9）是斜体，则该选项会移除斜体格式。可与 `\b` 同时存在。
- `\r Bookmark` 将“标示有特定书签”的页面范围作为此索引项的页码。例如，`{XE "base class" \r BaseClass}` 会令该索引项在索引之中被显示诸如这样的型式：`base class, 20-25`。不过我个人从来没有正确试得这个选项功能！我的理解是：一个书签只可能置于（指向、代表、参照）某惟一一页，

[2] 这种表示法对于无编程技术背景的排版者恐怕比较陌生。中括号 [ ] 表示可有可无的、选择性的。option 表示选项，通常以 \r, \e, \n...型式来表现。

那么又如何能够使得索引项的页码呈现一个区间范围呢？不解⊗

- `\t "Text"` 以特定文字替换页码；该文字需以引号括住。例如，`{XE "derived classes" \t "请参阅 base classes" }` 会令该索引项在索引之中被显示为 derived classes, 请参阅 base classes。

8.1.2. 索引主项（索引第一层）

出现于索引第一层的就是主索引项，也就是 XE 域内那个以双引号括起来的词条：

```
{ XE "Text" [option] }
```

8.1.3. 索引次项（索引第二层、第三层……）

出现于索引第一层以下的，统统称为次索引项，也就是在 XE 域内，以双引号括起来的文字中，位于冒号之后的那个词条：

```
{ XE "Text:SubText" [option] }
```

若要进行第三层索引，可以这样设置：

```
{ XE "Text:SubText:SubText" [option] }
```

虽然 Word 允许用户制作多达 9 层索引，不过实用上以 2 层为主要，偶有 3 层。我个人鲜少看过 3 层以上的索引。

8.2. 以“索引词条文件”批量标记索引项目

面对索引项，Word 提供两种标记（*mark*）办法，一是批量作业，一是逐一作业。两相比较，当然是批量作业好：您可以先做一份完整书面资料，我称之为“索引词条文件”，并仔细推敲琢磨其中所列的 1,2,3...层索引词条的关系[1]。换句话说批量作业远比逐一作业更系统化、更组织化，比较不会迷乱。

批量作业有很高的运用价值，但其选项设置的弹性却输给逐一作业方式。两者我都会做介绍。

---

[1] 没有好的构思，别想做出好的索引；这不像制作目录那么容易而直观。

8.2.1. 索引词条文件的建立

首先，我们需要一个索引词条文件，记录每一笔主索引项和每一笔次索引项。这个文档必须是个双列表格——这是惟一条件，至于其内文字的格式细节如字体、字距、段落……都无所谓。第一列填写索引项目（索引词条），第二列依照 "Text:SubText" 格式填写主项与次项。如果打算制作第三层索引，则依照 "Text:SubText:SubText" 的格式填写。但是表格中无法加上 p.219 所说的选项。下面是个例子：

function templates	function:templates
class templates	class:templates

这个表格的意思是：我希望取得文档中的 "function templates" 词条的所有出现页码，并以这种方式排列：

function
templates, 4, 9, 12, 18, 28

我还希望取得文档中 "class templates" 词条的所有出现页码，并以这种方式排列：

class
templates, 4, 9, 12, 18, 28

下面是实际获得的结果（操作过程稍后介绍）：

class
templates, 9, 23, 24, 27, 28, 29, 32, 35, 37, 47, 51

function
templates, 9, 11, 15, 17, 21, 23, 24, 37, 45, 47, 53

这份结果被 Word 自动设为双栏如上— 双栏正是索引的惯常表现形式。但如果分栏位置不理想，用户必须手动调整（请参考 5.4 节）。

如果我把索引词条文件中的表格改成这样：

function templates	template:function
class templates	template:class

意思是希望取得文档中 "function templates" 词条的所有出现页码，并这样排列：

template
function, 4, 9, 12, 18, 28…

还希望取得文档中 "class templates" 词条的所有出现页码，并这样排列：

```
template
    class, 4, 9, 12, 18, 28…
```

实际获得的结果（当然）也由 Word 自动以双栏形式排列。

索引词条文件对中文也适用！

请注意，这样制造出来的索引，排列次序和索引词条文件中的表格内容次序无关，而是和索引词条的中文笔划多寡或英文字典次序有关，并且由 Word 自动排序。

8.2.2. 让 Word 自动化建立索引

首先如上一节所说建好索引词条文件，然后点击【插入➔索引和目录(D)】+索引，获得以下画面。

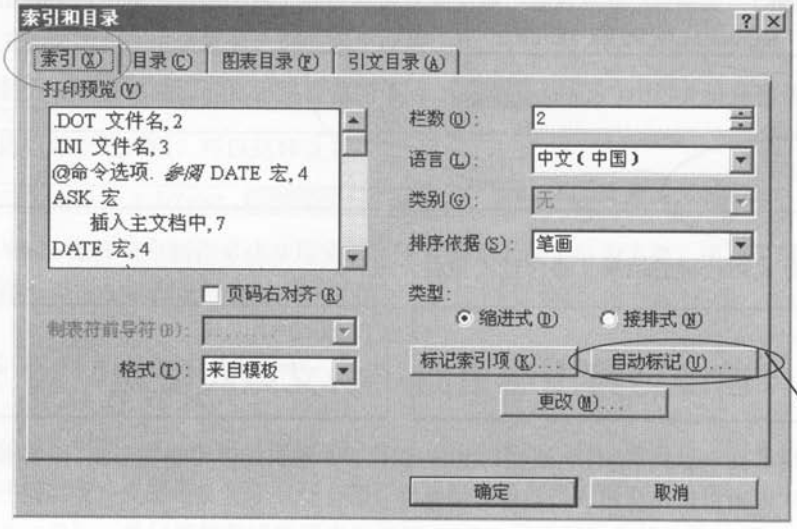
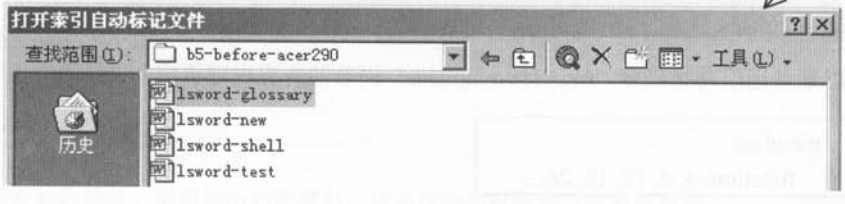


图8-1 索引和目录

点击右侧的自动标记(U)，于是 Word 向您索求索引词条文件：



选定索引词条文件后，将光标移至文档中您希望放置索引的地点，再次点击【插入➔索引和目录(D)】，如图 8-1，并点击右下角的确定，Word 便会自动为我们制造出一份索引（它会在文档中插入一个 INDEX 域，详见 9.3.8 节）。此时 Word 会自动显示所有的隐藏符号，为的是秀出域 XE 让您看见。

假设我以本节所说的批量标记方式，为“页面视图”这个词条建立对应的 XE 索引项{ XE "视图:页面视图" }，而后任何时间选 ，将隐藏符号全部显露出来，便可看见类似这样的画面：

> ▪ 2.3.3. **页面视图**{ XE "**视图:页面视图**" }
>
> 关于页面视图{ XE "视图:页面视图" }之开启、呈现、关闭（切换至其他视图），请见图 2-1 及其说明。如果当时也处于文档结构视图中（带出了结构图），那么页面视图就出现于右视窗，否则就出现于全视窗。
>
> 页面视图{ XE "视图:页面视图" }最棒的特点是，忠实呈现所有页面元素（扉页、题献、序言、目录、前言、正文、页眉、页码、附录、参考文献、书目、索引）于制版和印刷时的实际表现。想当然尔这会耗用大量电脑运算。以往硬件设备不足，专家有时候会建议您选择性地使用其他视图来进行编辑。如今速度问题已经罕见[11]，因此我建议完全在页面视图下进行所有工作。
>
> ▪ **文档结构图 + 页面视图**{ XE "视图:页面视图" } = **创作者的好帮手**

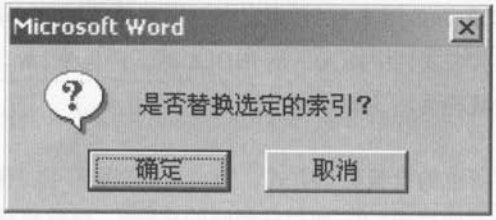
上图以框线框起的都是 Word 为我们加上的 XE 索引项。这些索引项虽然会推挤正常的内容，但实际打印时它们并不会出现，所以不会对页面带来不该有的影响。您注意到了吗，每一段落中的相同词条只有第一个会被冠以 XE 索引项。

重复建立索引，可以吗？

通常，整份文档定稿之后我们才会开始制作索引。但有时因为操之过急、有时因为疏漏，有时的确有所需要（稍后我会讨论“一份文档拥有两份索引”的情况），必须重新或再次制作一份索引。怎么办？

这时候我们必须重复上一页的所有动作：重新点击一次自动标记(U)（也就是重新载入“索引词条文件”以便重新标记索引项）、重新点击一次确定（用以生

成索引）。按下确定按钮时，Word 会询问您：



如果按下是(Y)，旧索引便获得了更新。但是请注意，虽然新制作出来的索引会反映索引词条文件中的新增项目，但索引词条文件中被删除的项目仍会出现在新制作出来的索引中（那当然不好）— 因为原先标记的索引项（XE 域）还在。

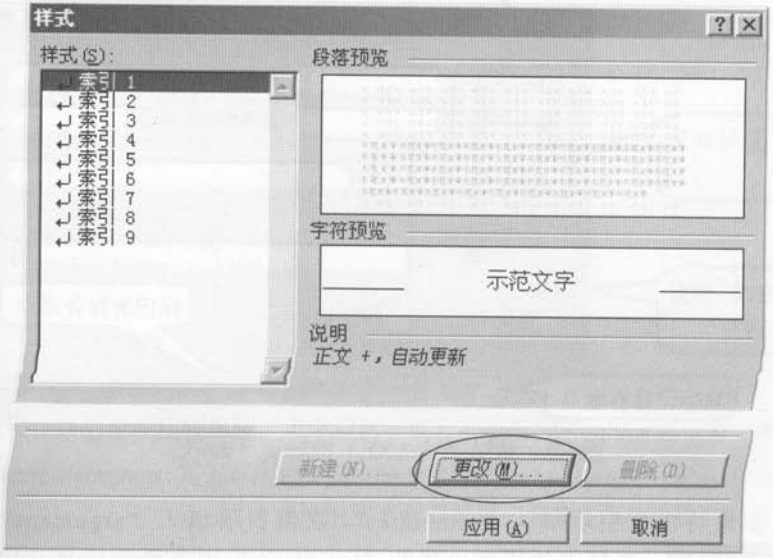
如果按下否(N)，旧索引会保留不动，另做出一份新索引。

一份文档带有两份索引，可以吗？

当然可以。有些书籍的确带有两份索引。例如《*More Effective C++*》就有一份 General Index 和一份 Index of Example Classes, Functions, and Templates。

8.2.3. 自定索引形貌

Word 以索引 1、索引 2、索引 3……样式来表现各层索引。如果您需要改变索引的长相（通常您会需要☺），请在任何时候点击【插入➔索引和目录(D)】，获得如图 8-1的画面，然后在其左侧格式(T):方块中选择来自模板，而后点击该图右下角的更改(M)，获得以下画面：



再点击下方的更改(M)，获得如图 5-2的画面，这就可以进行索引 1 至索引 9 样式的修改了。

8.3. 逐一手工标记索引项

逐一加入索引项，并不是个好差事，在工作流程上会把事情弄得很杂乱，又没有留下一份可以回顾检讨的书面资料。如果您大事底定（使用先前介绍的“索引项目批量标记”方式），最后才临时需要加上一两个索引，倒是可以这么办（手工加入）。

在文档中手工标记（mark）索引项的做法是：圈选希望作为索引项的文字，例如 "templates"，或是将输入点移到打算插入索引项的地点，然后点击【插入➔索引和目录(D)】+标记索引项(K)或按下 Alt+Shift+X，出现一个 modeless 对话框[1]，此时您所选取的文字会自动出现于图 8-2对话框的主索引项栏（可改动它，因为索引项不见得就必须是主索引项）。

[1] 所谓 modeless 对话框，就是那种“不会阻碍程序其他部分之操作”的对话框。例如【编辑➔查找...】、【编辑➔替换...】所产生的对话框都是。另一种所谓的 model 对话方块，就是那种“会阻碍程序其他部分之操作”的对话方块，例如【文件➔打开...】、【文件➔另存为...】对话框都是。

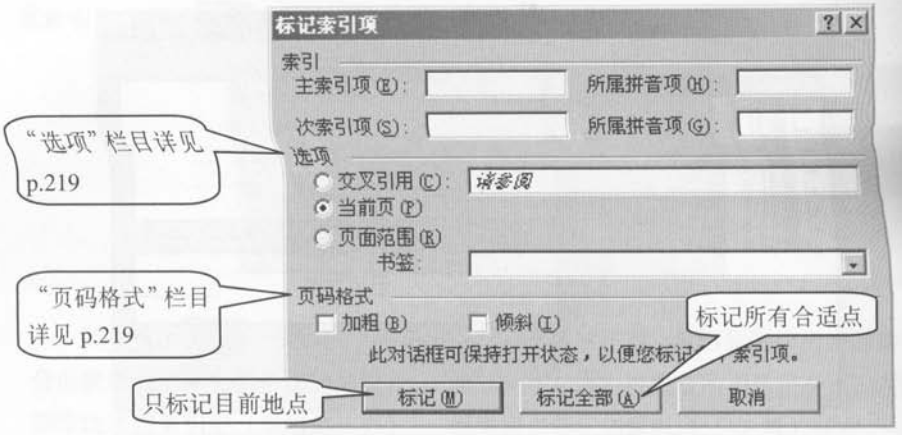


图8-2 索引项目（XE）的标记（*mark*）

如果将主索引项填入 "templates"，次索引项填入 "argument"，按下 标记全部(A)，文档中的所有出现的 "templates" 词条便会被 Word 加上隐藏的 `{ XE "templates:argument" }`。此时如果立刻再一次对 "templates" 进行 标记全部(A)，Word 会回复“无标记的索引项”[2]；如果只是进行 标记(M)（并将主索引项填为 "jjhou"），那么被圈选（欲作为索引项目）的文字后方会被加上 `{ XE "jjhou" }`。如果随后又在文档某处加了一个 "templates" 词条，然后又再一次对 "templates" 进行 标记全部(A)，Word 会在新添加的那个 "templates" 词条后方加上隐藏的 `{ XE "templates:argument" }`。

最后，回到 索引和目录 对话框，按下 确定 钮，产生类似这样的结果：

```
templates
    argument, 3, 4, 5, 7, 9, 10, 11…（后略）
```

Word 进行标记（*mark*）动作时，只考虑文字内容（区分大小写），不考虑其样式。如果 标记全部(A) 的是 "temp"，那么文档中所有 "templates" 的前半段 "temp" 也都会被标记，变成 `temp{ XE "temp" }plates`。此外，标记全部(A) 会对目录文字也进行标记，这通常不是我们所要的，必须事后清理☹

也可以使用 9.2.1 节所说的方法（Ctrl-F9），并参考 9.3.9 节的描述，加入 XE 域，作为索引项。

[2] 这个信息会不会有误呢？会不会应该是“找不到未标记的索引项”？

8.4. 索引佳例示范与讨论

下面是摘录自《*C++ Primer*》的一小部分索引：

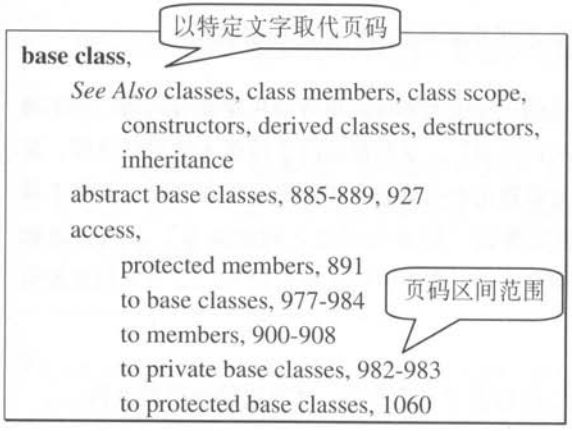
```
generic algorithms,
    ...
    heap generic algorithms, 1194-1197
        make_heap(), 1194
        pop_heap(), 1195
        push_heap(),1195
        sort_heap(),1195
```

这种索引词条的组织比较单纯，上下层次分明：generic algorithms 之下有一类称为 heap generic algorithms，其下有四种 algorithms 分别是 `make_heap()`，`pop_heap()`，`push_heap()`，`sort_heap()`。欲做出这个索引效果，只要在 Word 的“索引词条文件”中安排双栏内容如下：

make_heap()	generic algorithms:heap generic algorithms:make_heap()
pop_heap()	generic algorithms:heap generic algorithms:pop_heap()
push_heap()	generic algorithms:heap generic algorithms:push_heap()
sort_heap()	generic algorithms:heap generic algorithms:sort_heap()

以及为 "heap generic algorithms" 词条加上 `XE \r` 选项。

下面是摘录自《*C++ Primer*》的另一小部分索引：



以上索引告诉读者：

- 关于 base class，可参考 classes, class members, class scope...

- 关于范围更窄化的 abstract base class，可看 p885-p889, p927
- 欲访问 base class 的 protected members，可看 p891
- 欲访问 base class，可看 p977-p984
- 欲访问 base class 的 members，可看 p900-p908
- 欲访问 private base class，可看 p982-p983
- 欲访问 protected base class，可看 p1060

这些词条并不像上个例子有那么严明的上下层次。其中显现的“以 *See Also...* 取代页码”、“页码区间范围”等等，都是 Word 可以做出来的效果（见 p.219 的 XE 域选项），所以技术层次不是问题（虽然……呃……我在 p.219 说过，我自己才疏学浅，尚未能够正确试出 \r 选项），关键在于如何完善设计出“词条与词条之间的关系”，并正确插入对应的索引项目。

再举一个例子，仍然摘录自《*C++ Primer*》索引：

constructors, 691-702
    for base classes, 908-919
        in multiple inheritance, 970-971
        in single inheritance, 908-914
        in virtual inheritance, 998-1002
        memberwise initialization, 943-948
    as conversion functions, 780-782

这个例子和上个例子一样，各词条之间并非有严明的上下层次关系。

从后两个例子可以看出来，显然，排版人不可能单方面决定索引词条，更不可能单方面设计出“词条与词条之间的关系”。这些都超出了排版人的责任范围。至于文档作者，虽然可以不太困难地整理出索引词条，却也不可能在对索引制作工具毫无所悉的情况下设计出正确而完善的“词条与词条之间的关系”，因为诸如 for,to,with,in,as,of 等修饰词都必须在“指定文档中的主索引项（8.1.2 节）和次索引项（8.1.3 节）”时逐一设置。

好的索引，绝对是排版者和文档作者充分合作下，才有可能呈现的产物。

8.5. 本书索引制作过程

我个人专注于电脑技术的钻研和写作已逾十年，在海峡两岸各有多本书籍出版，这样的经历或足以让我谈谈台湾和大陆的计算机书籍。两岸计算机书籍不论应用类或技术类，只要是学术圈以外的，鲜少有索引，包括我自己的作品（惭愧！）。其原因或可归纳为 p.217 脚注 1 的六大项。

虽然我曾经做过索引[3]，却是 100%人工苦哈哈做出来的。撰写本书之前，我从未藉由 Word 的帮助制造索引（顶多只用它来寻找词条出现页码）。老实说，几近三个星期的摸索，才让我对本书的索引制作有了足够的概念和进展。这一节记录的便是这份经验。

索引的精细有着级别上的区分。最普通的一级是单纯将词条（**索引项目**）出现页码陈列出来。再上层楼则是对这些页码做筛选工作，只留适当者。更好一些的是为某些词条加上 "*See Also*"（“参阅”）。再好一些则是在某些词条（经过筛选后的）出现页码中加上诸如 as, with, in, of, to, for…与其他词条间的关系描述。

数十本计算机书籍的翻译经验使我深信，索引的制作肯定是一种 know how，国外大出版社肯定有专业团队负责这个工作[4]。以我的最大部头译作《*C++ Primer* 3/e》（by Stanly B. Lippman & Josée Lajoie）为例，该书总页数 1237，索引占了 39 页，估计涵盖 3,500 个词条，其中对于各词条的描述、词条间的关系，都相当详尽。作为一本工具书，又是一本大书，这样精细的索引非常有价值。

我以做出上述水准的索引为目标，并在此将整个实作心得分享给您。

- 首先确定索引的层级数目，进而决定各层索引的表现方式

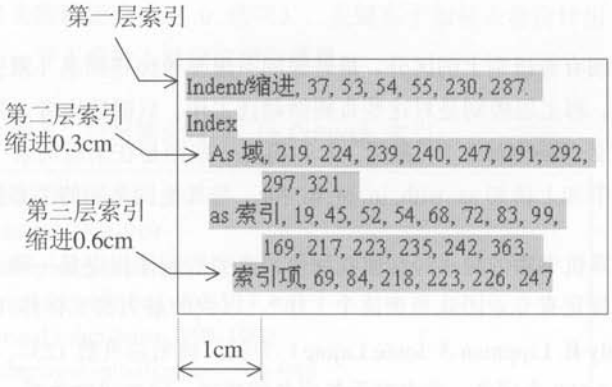
  本书采用 3 层索引，每一层索引都在 1.0cm 处折行，每一次层索引退缩 0.3cm。我以下表的一个**自定样式**和 3 个**内建样式**完成上述要求，视觉效果请见本书 p.385 索引。

---

[3] 1998 以前我的著作没有索引！1998 以后的惟一著作《*STL* **源码剖析**》虽有索引表，却是以笨功夫（100%人工）做出来的。近 5 年来我把大部分心力放在翻译上头，虽然每一本译作都保有索引，却是直接拿原文书索引来使用。这么做需要三个条件配合：(1) 中英页页对译 (2) 保留诸多英文术语 (3) 补充中英术语对照表。

[4] 我很羡慕国外作家可以不必一肩扛起索引的重责大任，也希望有朝一日了解作家和索引团队之间的合作模式——我相信作家无法完全置身于索引事外。

类型	样式名称	细目描述	说明
↵	!索引文字	正文 + 字体: 8.5 磅, 行距 最小值 11 磅	
↵	索引 1	!索引文字 + 缩进: 悬挂 1 厘米, 段落间距 段前 2 磅 段后 0 磅, 自动更新	搭配第 1 层索引
↵	索引 2	!索引文字 + 缩进: 左 0.3 cm 悬挂 0.7 cm, 段落间距 段前 0 磅 段后 0 磅, 自动更新	搭配第 2 层索引
↵	索引 3	!索引文字 + 缩进: 左 0.6 cm 悬挂 0.4 cm, 段落间距 段前 0 磅 段后 0 磅, 自动更新	搭配第 3 层索引



■ 利用 p.220 的 XE 域 /t 选项，制作 *See Also*

假设我希望，当读者检索 "视图→大纲视图" 或 "视图→页面视图" 时，这份索引能够带引读者去看 "Outline" 或 "Print Layout" 这两个词条。欲达到这个目的，只要在文档任何地点（稍后续论）写下两笔 XE \t 即可：

```
{ XE "视图:大纲视图" \t "参阅 Outline" }
{ XE "视图:页面视图" \t "参阅 Print Layout" }
```

至于最终真正记录页码的 "Outline" 和 "Print Layout" 两个词条，其 XE 索引项目既可采用 8.2 节的批量作业方式也可采用 8.3 节的手工逐一作业方式加入文档之中。稍后我将采用批量作业方式。

一开始所有这类 XE \t 都被我放在“索引”章前。为了便于检视和检讨，每一笔 XE \t 都独立一行（尾端有**换行符**）。但是这么一来，虽然这些设置本身并不影响页码（只要它们被隐藏起来，↵），**换行符**却总是有作用，会影响其后

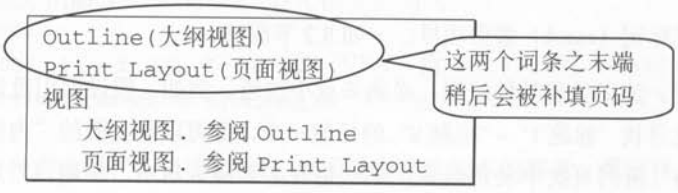
文档（本例为“索引”章）的页码。如果把换行符都拿掉，就不会影响页码，却又造成不易检阅与检讨。为此，我把所有 XE \t 放到“索引”章之后（也就是整份文档最后），这就无论如何不影响其他内容的编页了。最终完成全部索引后，切断索引的域关系（按 Ctrl+Shift+F9），再删除所有 XE \t，就可以让整份文档干干净净。不过也很可能您会想要保留这些 XE \t 设置，以备将来修改，那么就别忘了在删除它们之前先复制一份储存入其他文件。

■ 利用 p.220 的 XE 域 /t 选项，模拟应有的词条关系

接续以上动作。为提醒自己已经设了两个“稍后必须做出对应之 XE 索引项目”的词条，我又加上两个 XE \t：

```
{ XE "Outline(大纲视图)" \t }
{ XE "Print Layout(页面视图)" \t }
```

此时如果点击【插入➔索引和目录(D)】+索引(X)+确定，可获得索引如下，对于稍后制作“索引词条文件”有助忆效果：



这种“以 XE \t 模拟日后索引之实际所含项目”的手法，可对索引的预先整体掌握带来帮助。这么做的另一个主要原因是，将来（以批量手法）真正为每个词条加上 XE 索引项目时会给文档带来“破坏性”的影响[5]，因此在正式插入 XE 之前，先以“较单纯且位置得以集中”的 XE \t 模拟索引全貌，很有好处。

XE \t 也可以模拟多层索引，例如：

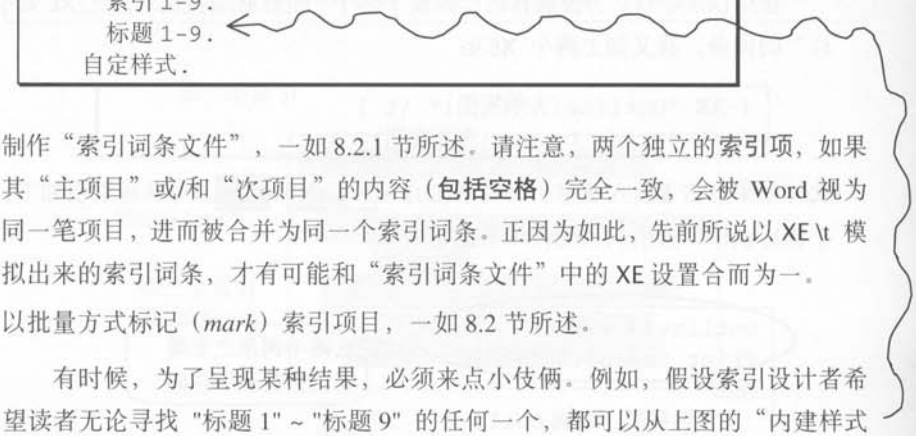
```
{ XE "Style:as Word 对象" \t }
{ XE "Style:as 样式" \t }
{ XE "Style:as 样式:内建样式:目录 1-9" \t }
XE "Style:as 样式:内建样式:索引 1-9" \t
```

5 我的意思是它会星罗棋布地改变文件结构，并且在**又做了一些操作**之后便**难以全面回复文件原貌**。

```
XE "Style:as 样式:内建样式:标题 1-9" \t
XE "Style:as 样式:内建样式" \t
XE "Style:as 样式:自定样式" \t
```

这便做出下面这样的索引：

```
style
  as Word 对象.
  as 样式.
    内建样式.
      目录 1-9.
      索引 1-9.
      标题 1-9.
    自定样式.
```



- 制作“索引词条文件”，一如 8.2.1 节所述。请注意，两个独立的**索引项**，如果其“主项目”或/和“次项目”的内容（**包括空格**）完全一致，会被 Word 视为同一笔项目，进而被合并为同一个索引词条。正因为如此，先前所说以 XE \t 模拟出来的索引词条，才有可能和“索引词条文件”中的 XE 设置合而为一。
- 以批量方式标记（*mark*）索引项目，一如 8.2 节所述。

  有时候，为了呈现某种结果，必须来点小伎俩。例如，假设索引设计者希望读者无论寻找 "标题 1" ~ "标题 9" 的任何一个，都可以从上图的“内建样式→标题 1-9”所列页次中获得线索，那么也许必须插入所有 "标题 1"、"标题 2" …… "标题 9" 的对应**索引项**，再将取得之页码整理于上述所列的 "标题 1-9" 词条。

- 点击【插入➔索引和目录(D)】+索引(X)+确定，获得一份由 Word 为我们自动化生成的索引。
- 逐一修改每个词条的 XE 内容。请注意，也许您会想要按下 Ctrl+Shift+F9 切断索引的**域**关系，而后直接修改索引内容，然而我想最好还是保留**域**关系，因此应该修改的是 XE 设置内容。如果这样，这份索引便可重制、可更新（见 9.2.2 节）。但如果采用第一种做法，索引内容将成为“飘零的落花”，其成分从此只是一般文字，不复能够进行任何自动化工程。
- 先前当我们以批量方式针对某些词条插入 XE **索引项**，讨厌的是，竟连出现于目录中的那些词条也被自动标记 XE **索引项**。这些 XE 的出现，并非因为**目录项**的对应标题有了变化（固然那是可能的）而连带影响目录内容，而是因为目录内容受了“批量插入 XE”的影响。但是您知道，目录页码有什么必要被记录到索

引去呢？任何词条在目录中都只是“出现”，毫无解释或提供额外信息的可能。因此有必要去除那些 XE 设置。很简单，只要在目录区更新域（如 9.2.2 节）即可，这将使目录区内的 XE 索引项消失。致于“目录项所对应之标题被插入了 XE 索引项”，该 XE 索引项并不会被抓进目录区来（算它聪明☺）。

- 显示 XE 或不显示 XE（利用 钮），会影响整个文档的页码，进而影响索引区内所收集的页码。因此，如果您在显示 XE 之后，更新索引区内的域，您会发现整个索引区记录的页码全都改头换面了。别紧张，那只是反映当时现况罢了。显示 XE 时，文档画面可能会变得很糟糕，像这样：

```
#001 Dim { XE "Dim As (VBA 指令)" } startParag → '区块中的第一个段落（代表第一行程序代码）
#002 Dim { XE "Dim As (VBA 指令)" } endParag → '区块中的最后一个段落（代表最后一行程序代码）
#003 → '注：采前闭后开区间 []，所以 endParag 其实是代表最后一行的下一段落
#004 Dim { XE "Dim As (VBA 指令)" } nLineNum → → → '行号（数值）
#005 Dim { XE "Dim As (VBA 指令)" } sLineNum As String { XE "String (VBA 型别)" } → '行号（文字）
#006 Dim { XE "Dim As (VBA 指令)" } i
#007
#008 For { XE "For To (VBA 指令)" } startParag = 1 To ActiveDocument.Paragraphs { XE "Paragraphs (Word 对象)" }.Count
```

不过没有关系，一旦隐藏 XE，画面又会恢复原本的清爽，像这样：

```
#001 Dim startParag     '区块中的第一个段落（代表第一行程序代码）
#002 Dim endParag       '区块中的最后一个段落（代表最后一行程序代码）
#003     '注：采前闭后开区间 []，所以 endParag 其实是代表最后一行的下一段落
#004 Dim nLineNum                   '行号（数值）
#005 Dim sLineNum As String         '行号（文字）
#006 Dim i
#007
#008 For startParag = 1 To ActiveDocument.Paragraphs.Count
```