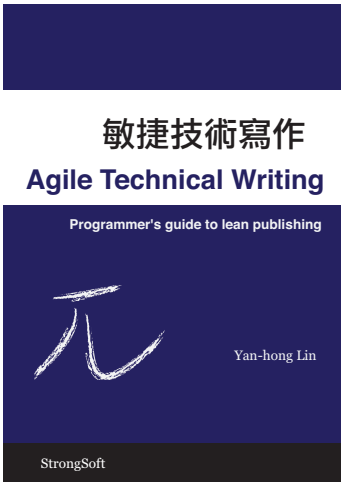


# 敏捷技術寫作
# Agile Technical Writing

***Release 1.0***

**Yan-hong Lin**

September 23, 2011

# CONTENTS

CHAPTER

ONE

# 前言

筆者第一次接觸電腦時，大約是在 386 PC 的時期，那時候學習 DOS 作業系統，文書處理使用 PE2 及漢書 (HE3) 軟體，這些文字模式下的工具軟體，操作起來相當容易上手，只要學會切換倚天中文系統 (ET3) 及輸入法，打開文書處理軟體，接下來就剩下「打字」的工作。那時候電腦對我來說，只是一部可以倒退刪除和移動游標的高級打字機。

即便是到了後期開始學會一些排版指令，可以讓點陣式印表機輸出的文件多一點字體變化，但是在使用電腦處理文書工作時，主要還是專注在「快速把內容建立完成」這件事，因此勤練打字，希望可以讓「產出」更有效率，讓打字輸入的速度跟得上大腦的思緒。

進入 Windows 視窗時代後，新的軟體每次都讓我感到驚艷；第一次使用 Office 軟體時，看到螢幕上的文字可以隨意變更字體、大小，效果馬上看得見，還可以插入圖片、修改色彩，甚至做出特效文字。也許這類軟體功能在十多年後的今天已經不足為奇，但是第一次看到的感動仍記憶猶新。

但回歸實際面，我發現新的軟體功能雖然強大，但真正需要用到的功能其實不到 1% 。剛開始覺得學習新事物很有趣，可是軟體改版速度之快，猶如火箭升空一般；從 Office 95, 2000, XP, 2003, 2007, 2010，改用 Linux 作業系統後開始接受 AbiWord 及 OpenOffice（新的名字是 LibreOffice ），買了 Mac 電腦後又學起 iWork 的 Pages, Keynote 。

天阿！我開始發現大部分在用電腦處理文書工作的時間，不管是交一份學校作業報告、撰寫專案設計書等，時間開始愈來愈不敷使用，時間被解決相容性問題吃掉一點、被新軟體介面不熟耗掉一些，每次都要依照不同規定摸索著如何把文件格式調整好，撰寫內容的過程中，也要不斷修改排版讓文件看起來比較順眼，思緒一直被打亂再重新開始；當不幸遇到惱人的詭異排版錯亂問題時，又必須停下工作先把問題克服。

我知道這些「文書處理常見問題」都有各式解決方法，也能透過範本、樣式設計，讓排版工作順利一些。我知道這些技巧，也花費不少時間學習更有效率地操作軟體，也經常幫助別人解決這類問題。

但是，這些跟「完成一份文件」有啥關係吶？

也許有些人認為完成一份文件本來就包含排版工作，但真的值得花那麼多時間搞排版嗎？

假設有位研究者，大腦裡已經裝有清楚完整的文件內容架構，要開始撰寫一篇論文，不妨想像一下過程有多少惱人的雜事：

1. 新買的筆電裝好 Word 2010 ，打開。媽呀！介面怎麼長得不一樣，開始去圖書館找來幾本書，拼老命把新軟體功能學會，至少也要懂些基本功夫。
2. 下載學校的論文格式規定，打開。該死！連一個樣式定義都沒有，只好仔細把全部的排版規格看過一次，自己一步一步把設定調整好，這時候好不容易學會的「文件樣式」技能終於派上用場，效率比別人快一點點。
3. 參考文獻用 EndNote 管理，好樣。但是！要把文獻搞到 Word 裡面，只能選擇 (1) 啟用「工人智慧」手動編輯模式 (2) 學習外掛自動化處理的功能。
4. 怎麼有些地方版面調來調去還是有問題？鬼打牆啦！沒關係，這種事情不是第一次，Google 搜尋一下總能找到解決辦法，要不然就找那位 Word 神人來幫忙，簡單啦！
5. 蝦密！？今年要改用新的論文格式？有沒有搞錯。沒關係，反正還有好幾天，而且採用一致的文件樣式，改一個樣式表就能套用到所有相同格式的段落。
6. 指導教授：寫得好，拿去投那個 XXX 研討會吧！真棒，這下能順利畢業了！不... 該死，研討會要求的格式怎麼跟學校差那麼多，還要繳交相容 XP/2000/2003 的.doc 格式，咦！轉換之後怎麼格式跑掉了，再次開啟工人智慧模式。

也許這一點都不惱人，因為造就了很多 Word 神人，研究發表得愈多，文書處理功力也愈高竿，都可以開班授課教軟體技能了。但是，到底是要發表研究成果？還是要學習 Word 軟體？到底要讓電腦幫我們工作，還是為了要拜託電腦給我們一份漂亮的文件，我們就得該死地幫電腦工作？

自從我開始接觸自由軟體，認識了 **企鵝** (Linux 作業系統的吉祥物)，就開始相信「**窗外有藍天白雲** 」，不走出窗外很容易得到憂鬱症。

由於長期使用電腦的關係，長時間使用滑鼠對手腕的傷害很大，我盡量改用可以減少動滑鼠、只要操作鍵盤就能完全大部分工作的軟體，作業系統的首選當然是 Linux (最近發現 Mac 也相當優)，只要熟悉 Bash Shell 的指令、VIM 文字編輯器的操作，就能以相當有效率的方式靠鍵盤 [1] 完成工作，包括系統管理、撰寫程式等等。

為了讓「文書處理」也能單靠著鍵盤完成，我發現「TeX」這套由大師 Knuth 教授發展的幕後排版系統。我開始閱讀李果正老師發表的 LaTeX 教學 [2] ，以及學習吳聰敏老師

[1] 推薦 Cherry 原廠的機械式鍵盤，手感一級棒
[2] http://edt1023.sayya.org/

等人開發的 cwTex 排版系統 [3] 。

網路上流傳這麼一段說法：

使用 LaTeX 時，你是詩人，LaTeX 是你的排版工人；使用 Word 時，你想當個詩人，可是你是個排版工人。 [4]

為什麼 Word 和 LaTex 會有如此差別呢？

Word 是一套功能強大、所視即所得的軟體，就因為正在編輯中的文件，就已經相當於完成的結果，甚至和列印輸出的成果十分接近。但是這項先進、介面豐富軟體的優點，讓我們很難專注在「完成內容」這件事。

即便許多人都清楚，要避免浪費時間做重複多餘的排版工作，最好的方法就是先用純文字編輯完大部份內容，最後在設計文件樣式，套用在各段落；要靠著預先設計一份良好的樣式，讓內容撰寫時可以不必調整排版設定不容易，因為要讓「標題」看起來有標題的效果，就必須把一段標題文字選取 **標題樣式** 。

我們也很難真正等到文件主要內容完成八、九成，才去開始動手進行排版工作，通常我們只要完成一個章或節，或是一個樣式比較獨特的段落，就會想看看整份文件的排版成果。

幕後排版軟體的好處，就是在編輯時不用去理會「排版之後的效果」，只要完成「文件大綱結構和內容」。

舉例來說，一段 LaTex 格式的文件原始碼可能長這樣：

```
\chapter{第一章標題}
\section{第一節標題}

這是第一節的文字內容。

這是第一節的第二段文字內容。

\section{第二節標題}

第二節的內容。

\begin{itemize}
\item 條列第一點。
\item 條列第二點。
\end{itemize}
```

[3] http://homepage.ntu.edu.tw/~ntut019/cwtex/cwtex.html
[4] 未發現出處

這份文件完全以純文字方式編輯，可以使用任何一種文字編輯器製作（如：Notepad, UltraEdit, Notepad++, TextMate, gedit, VIM, GNU Emacs 等），儲存成一份文字檔案（例如：doc1.tex ）。因為是純文字格式，章節、項目符號等效果都可以用文字標籤定義，所以文件製作的過程很少會需要碰到滑鼠，可以讓手指跟著大腦的思緒快速敲擊鍵盤將文字內容完成。

當你想預覽目前完成的文件，排版效果如何？只需要執行 LaTex 的幕後排版程式，例如以指令方式：

```
xelatex doc1.tex
```

就可以產生 doc1.pdf 檔案，使用 Acrobat Reader 等文件閱覽軟體開啟，就可以看到排版後的文件。

要如何調整排版效果，如字體大小、色彩等等呢？當然同樣也是可以透過文字標籤指令的方式，定義好的排版設定，還可以很容易地重複利用。

若熟悉 Linux 或其它 Unix 系統操作的朋友，這種操作模式肯定能讓文件撰寫工作更有效率及彈性，例如我們可以透過 ssh 登入一部裝有 LaTex 工具的遠端伺服器（或工作站），然後利用 VIM 或 Emacs 編輯器繼續撰寫文件內容，利用指令自動在背景進行幕後排版指令，並使用瀏覽器透過 Web 或 FTP 方式預覽排版後的 PDF 文件。這種純文字模式、單靠鍵盤就能完成排版工作的書寫方式，對電腦效能、網路頻寬的要求相當低，過程也相當有效率。

如果排版工作能夠如此輕鬆，那麼我們就可以開始改善文件撰寫的工作！

哪些人會需要撰寫文件呢？部落客發表網誌文章、商務人士計畫提案、工程師製作系統文件、研究人員發表論文，都需要大量撰寫文件。最近有一本暢銷新書《自由書寫術》就鼓勵讀者利用大量自由書寫 (free writing) 方式，捕捉各種想法，以找到簡報、企劃所需要的創意。

很多時候電腦並不能取代紙筆，隨手塗鴉的草圖可以快速描繪一個腦海裡的想法，很難用電腦軟體工具取代；但如果我們要書寫數量較多的文字時，特別是已經在紙上或腦海裡有了框架，需要開始填充大量文字，對於熟悉鍵盤輸入的寫作者，電腦仍是效率最好的工具。

當我們開始經常地用電腦寫作，就必須想辦法減少被「排版」浪費的時間。

這邊所述的 **排版** 並非正式書籍印刷或專用文件格式的那種專業排版，只是將版面調整成文件應有的基本構造。

舉例來說，一位部落客可能已經有既定的版面風格：「圖片置中、下方附加說明文字。」在發表第一篇文章時，這位部落客總共插入五張圖片，使用一般部落格平台提供的所視即所得編輯器，必須重複做了五次圖片置中、增加說明文字的動作，寫作者的手必須忙碌地在滑鼠及鍵盤間切換，思緒也可能因為這個排版動作而被打斷了一下。其他

包括 **引述文字** 、 **強調文字** 、 **程式碼區塊** 等都有固定的風格，但寫作者仍然每次都必須重新設定一次目前的段落，並且必須分辨在編輯器介面上該如何操作這些設定。這些干擾使得文字書寫變得沒有效率，即使是一篇並不像書本那樣要求排版格式的網誌文章。

對於計畫要出版書籍的作者，不但寫書時要排版，若平時就經常地書寫一些「材料」，順便發表在網路上分享，這些分散的文章可能放在部落格或其他文件平台，雖然不是正式發表的書籍內容，但仍要有一定的可閱讀性時，就浪費了更多排版時間。

但是，再一次地問，那些瑣碎的排版真的值得嗎？如果不做是否就會使可讀性變差？

還記得在過去 Web 還不是那麼盛行的年代，新聞群組 (New Groups) 及電子佈告欄 (BBS) 只支援純文字方式撰寫文章，但大量使用這些服務交換資訊、分享文件的網友，並沒有因為純文字而造成太大的不便，雖然不像 HTML 那樣美觀又支援多媒體，但仍然可以每天有效率地讀取大量資料。

例如一篇純文字撰寫的電子報可能是這樣：

```
┌--------- ■ 南 方 電 子 報 ■ ---- 2000/03/10 ┐
    讓商業邏輯下失去戰場的理想在網路發聲
└ 南方社區文化網路：http://www.south.nsysu.edu.tw ┘

====================【 編輯室手記 】====================

一九九五年三月九日，「南方」第一代工作人員 whitebeach
在簡陋的辦公室裡辛勤地 scan 文章。她對剛從醫學院下課的
ROACH 說：中山大學 BBS 為「南方」開設的留言版做好了。
於是 ROACH 寫一篇公告放上網路，whitebeach 開始把累積數
十萬字的文章，用原始的 Te ix 程式一一上傳到 BBS 上。
這就是「南方社區文化網路」的第一天。

===============【今日主題文章摘要】====================

◎網路時代的失語震撼（陳豐偉）

大約在四年前，我也曾借用楊照的概念，以「失語震撼」四個
字，形容台灣本土面對網路時代侵襲，卻因為不熟悉網路，無
法針對網路議題發言的文化人。當時，台灣的網路才剛開始大
眾化，大多數文化評論家根本不對網路發表意見。如果提到網
路，也常充斥著對網路負面、刻板的印象，嘲諷網路上的匿名
文化、譏笑網路上的言論空洞、浮濫。
```

即便沒有黑體或楷體、加大字型的標題，我們還是可以很容易閱讀這篇純文字格式的電子報。

在電子佈告欄，我們可以用很多方式讓文章達到資訊傳遞的目的：

```
【工商服務】二手電腦書特價出清

*電話： 123-4567
*店址： xxx 市 ooo 路 000 號

**** 大特價 **** 即日起 9 月 30 日止
1. 快快樂樂學 LaTeX          $99
2. 快快樂樂學 Markdown       $199
```

瞧！文字及符號本身就具有良好的「格式化」的能力。

本書並非要鼓吹大家放棄多采多姿多媒體文件，筆者自己也很喜愛使用 iWork Pages 軟體製作文件；但是在平日書寫時我們仍要以「敏捷」為目標，讓文字撰寫更有效率，這本書就是希望提供一些工具和方法，讓「高效書寫」與「高品質排版」可以魚與熊掌兼得。

CHAPTER

TWO

# 請協助本書的發展

本書以「DRM FREE」方式發行，您可以將取得的檔案用於任何支援該格式的閱讀裝置。

您可以從網站取得本書最新版本。

如有任何問題或建議，請與作者聯絡：Yan-hong Lin <lyhcode@gmail.com>

CHAPTER

THREE

# ABOUT AGILE TECHNICAL WRITING

Agile Technical Writing: Programmer's guide to lean publishing

這是一本開放源碼電子書，以敏捷技術寫作方式撰稿，利用 Git 版本控制工具進行協作，並使用 GitHub 平台。

CHAPTER

FOUR

# 五花八門的文件撰寫格式

## 4.1 WYSIWYG

1. Word 軟體
2. 線上文件編輯器

### 4.1.1 HTML

HTML (Hypertext Markup Language) 就是撰寫網頁使用的格式，它能夠直接利用任何一種文字編輯軟體撰寫，只要使用作業系統提供的瀏覽器開啟檔案，就可以看到排版效果。

簡單的文件，直接以 HTML 原始碼撰寫，本身就可以得到排版效果：

1. 純文字編輯
2. 使用瀏覽器即可看到排版結果

例如一個標題加上段落文字：

```
<h1> 這是大標題 </h1>
<p> 這是一個段落，內容不重要。</p>
```

我們也可以強調一段文字：

```
<p> 這是一個段落，並且 <strong> 強調這段文字 </strong>。</p>
```

許多線上服務，可以讓你測試這些 HTML 的效果：

1. http://sandbox.coreyworrell.com/
2. http://htmlsandbox.com/

只要搭配 CSS 樣式表，就能讓同一份 HTML 原始碼可以得到不同風格的排版效果。

例如在 CSS Zen Garden 網站中，就提供許多設計師製作的範例，示範了如何透過 CSS 讓同一份網頁文件，呈現出風格迥異的樣貌。

- CSS Zen Garden http://www.csszengarden.com/

以 HTML 撰寫的文件，當我們複製其中一段文字、貼到其它文書處理軟體時，原本的部份格式可能保留下來。

使用 HTML 發佈文件具有幾項優點：

1. W3C 標準已受到廣泛支援。
2. 可攜性佳；閱讀者只需要使用大多數電腦皆有內建的瀏覽器軟體，就能看到排版後的文件效果。
3. 撰寫容易，只需使用一般文字編輯器。
4. 可以將瀏覽器處理過的文件，複製需要的內容到其它排版軟體加工，部份效果可能會保留。
5. 若有需要，也有大量所視即所得工具可以輔助編輯。

建議讀者可以了解一些常用的 HTML 標籤語法，因為稍後要介紹的許多文件格式，都可以透過工具自動轉換成 HTML 格式，若能認識這些標籤，對工具的應用就更容易掌握。

但是並不建議平時以 HTML 大量書寫：

1. HTML 文件可以簡單，但也可以很複雜，若要轉換成其他文件格式並不容易。
2. 在未使用瀏覽器開啟的純文字顯示下，並不容易閱讀。

## 4.2 Wiki

Wiki 並非一種文件格式，而是諸多線上文件協作平台的統稱。

Wiki 相當適合用來整理資料，它雖然是為多人協作目的發展，但是作為個人的筆記、知識管理工具也相當適用。

由於架設 Wiki 的軟體種類繁多，各自支援不同的語法；但其實各種 Wiki 文件格式都大同小異，只需要熟悉平時慣用的平台語法即可。

首先要介紹的是廣泛使用的 Mediawiki。

Mediawiki [1] 就是知名網站「維基百科 (Wikipedia [2] ) 」使用的文件格式，它的設計讓多人可以線上協同編輯文件內容，可以利用工具比較出不同修訂版本之間的差異，並且容易更正、復原等。在純文字下也具有一定的可讀性，不難看出文件的結構。

範例：

```
== 大標題 ==

這裡是一個文字段落。

=== 第一章 ===

內容

==== 第一節 ====

=== 第二章 ===
```

只要熟悉 Mediawiki 的基本語法，就可以在 Mediawiki 架設的協作平台上暢行無阻，有許多學校和組織都利用它架設內容管理系統、知識文件平台。

由於 Mediawiki 是為打造 Wikipedia 而發展，功能十分強大，也支援各種擴充，例如增加內嵌影片的語法等。但這也造成 Mediawiki 的語法比較複雜。

## 4.3 Markdown

Markdown [3] 的目標是實現「易讀易寫」的 **純文字** 文件格式。 [4]

使用 Markdown 撰寫文件時，「內容」是主角，搭配簡單常用的符號或空格，讓文件不用經過其他工具處理，直接純文字格式發佈，也能保有良好的可讀性。

範例：

```
大標題
======

小標題
------

段落文字

> 引言 1 ...
```

[1] http://mediawiki.org/
[2] http://wikipedia.org/
[3] http://daringfireball.net/projects/markdown/
[4] http://markdown.tw/

```
> 引言 2 ...

* 項目 1
* 項目 2
```

由於語法簡單易用，且接近一般人在撰寫純文字電子郵件時的習慣，Markdown 日漸受到各類平台採用，例如開放源碼的專案代管系統 GitHub [5] ，就以 Markdown 作為預設的說明文件撰寫格式。

Markdown 非常適合用在軟體專案的 README 等純文字說明文件。

## 4.4 reStructuredText

test2 test

[5] http://github.github.com/github-flavored-markdown/

CHAPTER

FIVE

# 研究論文

CHAPTER

SIX

# 範例：運用自由書寫撰寫研究論文

## 6.1 研究背景

### 6.1.1 動機

### 6.1.2 目的

## 6.2 文獻探討

CHAPTER

SEVEN

# 使用 SPHINX 工具製作高品質文件

## 7.1 認識 Sphinx 軟體

Wikipedia 對 Sphinx 一詞的解釋：

斯芬克斯最初源於古埃及的神話，它被描述為長有翅膀的怪，通常為雄性，當時的傳說中有三種斯芬克司——人面獅身的 Androsphinx，羊頭獅身的 Criosphinx（阿曼的聖物），鷹頭獅身的 Hieracosphinx。亞述人和波斯人則把斯芬克司描述為一隻長有翅膀的公牛，長著人面、絡腮鬍子，戴有皇冠。到了希臘神話裡，斯芬克司卻變成了一個雌性的邪惡之物，代表著神的懲罰。「Sphinx」源自希臘語「Sphiggein」，意思是「拉緊」，因為希臘人把斯芬克司想像成一個會扼人致死的怪物。[1]

本文介紹的 Sphinx [2] 是文件製作工具，讓寫作者可以輕鬆產生高品質、外觀漂亮的文件；在 Google 搜尋 Sphinx 可以發現另外一個與它同名的專案 [3] ，但功能可是完全不相同。

Sphinx 作者 Georg Brandl 採用 Python 程式語言開發，並且以 BSD 授權方式，發佈 Sphinx 的原始碼；您可以免費取得、使用 Sphinx 製作自己的文件。Sphinx 受到 Python 及其他開發社群的喜愛，被廣泛使用在製作各類文件。[4]

[1] Sphinx from Wikipedia, http://en.wikipedia.org/wiki/Sphinx
[2] Sphinx (Python Documentation Generator), http://sphinx.pocoo.org/
[3] Sphinx (Open Source Search Engine), http://sphinxsearch.com/
[4] Projects using Sphinx, http://sphinx.pocoo.org/examples.html

## 7.2 安裝 Sphinx 軟體

作業系統環境需求：

1. Python 2.4+

## 7.3 快速建立 Sphinx 文件專案

使用 sphinx-quickstart 可以快速建立新的文件專案。:

首先：:

```
mkdir MyBook
cd MyBook
```

輸入指令：:

```
sphinx-quickstart
```

執行畫面：:

```
Welcome to the Sphinx 1.0.7 quickstart utility.

Please enter values for the following settings (just press Enter to
accept a default value, if one is given in brackets).

Enter the root path for documentation.
> Root path for the documentation [.]:
```

按 Enter 鍵（直接使用目前的資料夾位置）。:

```
You have two options for placing the build directory for Sphinx output.
Either, you use a directory "_build" within the root path, or you separate
"source" and "build" directories within the root path.
> Separate source and build directories (y/N) [n]:
```

按 Enter 鍵（使用預設值）。:

```
Inside the root directory, two more directories will be created; "_templates"
for custom HTML templates and "_static" for custom stylesheets and other static
files. You can enter another prefix (such as ".") to replace the underscore.
> Name prefix for templates and static dir [_]:
```

按 Enter 鍵（使用預設值）。:

```
The project name will occur in several places in the built documentation.
> Project name: MyBook
> Author name(s): John
```

輸入書 (或文件) 名及作者姓名，這項設定將會顯示在 HTML, PDF 等。:

```
Sphinx has the notion of a "version" and a "release" for the
software. Each version can have multiple releases. For example, for
Python the version is something like 2.5 or 3.0, while the release is
something like 2.5.1 or 3.0a1.  If you don't need this dual structure,
just set both to the same value.
> Project version: 1.0a
> Project release [1.0a]:
```

輸入自訂的版本編號，同樣會顯示在 HTML, PDF 等。:

```
The file name suffix for source files. Commonly, this is either ".txt"
or ".rst".  Only files with this suffix are considered documents.
> Source file suffix [.rst]:
```

按 Enter 鍵（採用預設.rst 副檔名）。:

```
One document is special in that it is considered the top node of the
"contents tree", that is, it is the root of the hierarchical structure
of the documents. Normally, this is "index", but if your "index"
document is a custom template, you can also set this to another filename.
> Name of your master document (without suffix) [index]:
```

按 Enter 鍵（主文件採用預設檔名 index ，會產生 index.rst 這個檔案）。:

```
Sphinx can also add configuration for epub output:
> Do you want to use the epub builder (y/N) [n]:

Please indicate if you want to use one of the following Sphinx extensions:
> autodoc: automatically insert docstrings from modules (y/N) [n]:
> doctest: automatically test code snippets in doctest blocks (y/N) [n]:
> intersphinx: link between Sphinx documentation of different projects (y/N) [n]:
> todo: write "todo" entries that can be shown or hidden on build (y/N) [n]:
> coverage: checks for documentation coverage (y/N) [n]:
> pngmath: include math, rendered as PNG images (y/N) [n]:
> jsmath: include math, rendered in the browser by JSMath (y/N) [n]:
> ifconfig: conditional inclusion of content based on config values (y/N) [n]:
> viewcode: include links to the source code of documented Python objects (y/N) [n]:

A Makefile and a Windows command file can be generated for you so that you
only have to run e.g. `make html' instead of invoking sphinx-build
directly.
> Create Makefile? (Y/n) [y]:
> Create Windows command file? (Y/n) [y]:
```

連續按 Enter 回答後續的問題，這些設定可以在日後有需要時調整。

最後，將會出現以下訊息，恭喜你已經成功建立新的 Sphinx 文件專案。

## 7.4 編輯文件

預設:

```
.. Zen of Technical Writing documentation master file, created by
   sphinx-quickstart on Wed Sep 21 09:53:39 2011.
   You can adapt this file completely to your liking, but it should at least
   contain the root `toctree` directive.

Welcome to Zen of Technical Writing's documentation!
====================================================

Contents:

.. toctree::
   :maxdepth: 2

Indices and tables
==================

* :ref:`genindex`
* :ref:`modindex`
* :ref:`search`
```

## 7.5 使用 watchr

**安裝::** gem install watchr

> Successfully installed watchr-0.7 1 gem installed Installing ri documentation for watchr-0.7... Installing RDoc documentation for watchr-0.7...

CHAPTER

EIGHT

# INDICES AND TABLES

- genindex
- modindex
- search