PUTEX 4.0 Big5 版使用手冊 (Rev 1.0)

靜宜大學資訊管理學系

蔡奇偉

cwtsay@pu.edu.tw

http://www.cs.pu.edu.tw/~tsay/putex *

中華民國九十三年九月十九日

*本計畫部份是由國科會提供經費贊助完成（計畫編號：NSC-86-2213-E-126-005）。

目錄

1 前言

這份手冊說明如何使用 PUTEX 4.0。筆者假設你曾使用過 TEX/LATEX 來排版文件，對 TEX/LATEX 已有基本的知識，所以本手冊不會論及 TEX/LATEX 的使用細節與內部原理，比方說：本文件不會告訴你如何產生 α, β, ... 等的希臘字母，也不會教你如何寫出類似 $\sqrt{x^2+1}$ 的數學公式，更不會談到 glue, box, line breaking 之類的概念。所以，如果你是一位 TEX 新手，筆者建議你先閱讀一些 TEX 或 LATEX 的書籍（如參考書目中的 [2, 3, 4, 5]），等具備了一些基本觀念後，再回來細讀此手冊。

PUTEX 構築在 Christian Schenk 先生的 MikTEX 系統上。PUTEX 本身只包括修改過的 TEX 程式與幾個輔助程式，其他如 TEX 字型、LATEX 樣式定義檔、DVI 驅動程式等，仍然得仰賴 MikTEX 系統的支援。因此安裝 PUTEX 之前，你必須先在電腦中先安裝好 MikTEX。

爲什麼取名爲 PUTEX 呢？這是因爲筆者服務於靜宜大學，PU 兩個英文字母是靜宜大學英文校名：Providence University 的字頭。因此，PUTEX 應該讀成 P·U·TEX。

1.1 如何取得 PUTEX?

PUTEX 計畫的 WWW 首頁是：

http://www.cs.pu.edu.tw/~tsay/putex/

你可以從以上網址下載最新版的 PUTEX 軟體、本手冊的最新版本、與瀏覽 TEX 相關的資訊。由於 PUTEX 必須在 MikTEX 的環境下才能執行，若你尚未安裝 MikTEX 的話，也請從上述的網站一併取得對應的 MikTEX 系統。

1.2 PUTEX 的特點

PUTEX 是一套在 Windows 系統下執行的中文 TEX 排版系統。PUTEX 改寫 D. E. Knuth 教授 TEX 程式的原始碼，使之能夠直接處理中文字元。由於不是採用前置處理（preprocessing）的方法來處理 TEX 檔案中的中文字元，PUTEX 比其他的中文 TEX 系統具有更高度的方便性、擴充性、與彈性。此外，PUTEX 還具有以下的功能：

- 支援所有廠牌（如華康、文鼎、全眞、或新研澤）、所有字貌的 TrueType 中文字型（如細明、楷書、行書、或勘亭流等）。

  註：PUTEX 並不直接支援英文 TrueType 字型。

- 可使用 TrueType 中文外字集字型。

- 具有標點符號避行首 / 行尾的排版功能，如「。」和「，」等標點符號將不出現在行首；而「（」和「【」等標點符號也不出現在行尾。

- 支援以中文來命名巨集指令。
- 可自由調整中文字的字間距（character spacing）。
- 可自由調整中文字與英文字之間的字間距。
- 可自由調整中文字基線（baseline）的位置。
- 支援中文直排的功能。
- 支援以中文數字表示法來顯示整數值的功能，讓你能夠輕鬆地製作出以「一、二、三、…」或以「壹、貳、參、…」爲標號的條列項目。
- 產生具有可攜性的 CDI（CJK Device Independent file）檔案。
- 提供 CDI 至 DVI（DeVice Independent file）的轉檔程式，所產生的 DVI 檔可以用 MikTEX 的 Yap 來預覽列印、用 dvips 轉換成 PostScript 檔、或用 dvipdfm 轉換成 PDF 檔。
- 提供 CDI 至 PDF 的轉檔程式。
- 提供製作中英文索引的 puidx 程式（註：目前只支援 Big5 碼）。

在底下的各節中，我們會進一步地闡明如何使用這些功能。

1.3 PUTEX 的軟、硬體需求

PUTEX 的軟、硬體最低需求如下：

軟體： Windows 作業系統與 MikTEX。TrueType 中文字型可依個人所需安裝。筆者建議你最少應有細明、楷書、和中黑這三種常用的字型。

硬體： Pentium 90Mz CPU, 16 MB RAM, 500 MB 以上的硬碟空間。

另外，你最好使用 15 吋（或以上）的顯示器，因爲你需要至少 800×600 的螢幕解析度，才能夠比較清晰地預覽 DVI 或 CDI 檔。

1.4 安裝 PUTEX

請以 WWW 瀏覽器閱讀檔案 `install.html`，並按照其中所述的步驟來進行安裝。安裝 PUTEX 之後，請務必依照附錄A 所述的方式來設定 `cfont.map` 這個字體對應檔，讓 PUTEX 能夠正確地利用 Windows 系統中的 TrueType 中文字型。

2 PUTEX 快速入門

PUTEX 的執行方式與英文版的 TEX 系統大致相同，即分爲以下三個步驟：

一、編輯 TEX 或 LATEX 格式的中文文件檔，如 `foo.tex`。

二、選擇下列適當的指令來產生 CDI 輸出檔：

- 如果 `foo.tex` 是 TEX 格式，則執行指令 `big5tex foo`
- 如果 `foo.tex` 是 LATEX 格式，則執行指令 `big5latex foo`

三、選擇下列的方式之一來預覽或列印 CDI 檔：

- 用 cdi2dvi 程式把 CDI 檔轉爲 DVI 檔，然後用 Yap 來預覽或列印。
- 把前述轉換所得的 DVI 檔，用 dvips 轉成 PostScript 檔然後用 gsview 來預覽或列印。
- 用 cdi2pdf 程式把 CDI 檔轉爲 PDF 檔，然後用 Adobe 公司的 Acrobat Reader 來預覽或列印。

以下我們就這三個步驟，做進一步的說明：

2.1 編輯中文 TEX/LATEX 文件

編寫內含中文的 TEX/LATEX 文件時，請注意以下幾點：

- 中文字元不可以直接鍵入數學式中，而必須含括在 `\mbox`（或 `\hbox`）之中。比方說：

  ```
  $x^2 > 0 \mbox{ 若 } x \ne 0$
  ```

  產生的結果爲：$x^2 > 0$ 若 $x \neq 0$。但是以下的輸入方式：

  ```
  $x^2 > 0 若 x \ne 0$
  ```

  則會產生錯誤訊息，而且其結果爲：$x^2 > 0x \neq 0$（其中的「若」字消失不見）。

- PUTEX 自 4.0 版起，撰寫中文 TEX 文件時，不必在第一行加入 `cfacedef.tex`，用英文 TEX 文件一樣的方式撰寫即可。

- 撰寫中文 LATEX 文件時，若想使用中文化的標題，請使用第 4 節所述的中文化文件類別（class）：twarticle, twreport, 或 twbook。比方說，本文件使用 twarticle，因此第一行是：

  ```
  \documentclass[11pt,a4paper]{twarticle}
  ```

- 若撰寫以英文爲主、中文只佔少部分的文件（譬如附中文摘要的英文文章），直接使用 LATEX 的文件類別即可，譬如：

```
\documentclass[11pt,a4paper]{article}
```

註：PUTEX 4 不再需要使用 twbase 套件。

2.2 編譯中文 TEX/LATEX 文件

假定 foo.tex 是一個 TEX 文件檔。你必須在「命令提示字元」視窗中（或 Windows 9x 的 DOS 視窗中）鍵入指令：

```
big5tex foo
```
（若 foo.tex 是 TEX 格式）

或

```
big5latex foo
```
（若 foo.tex 是 LATEX 格式）

來進行排版。如果一切正確，上述的指令會在目前的資料夾中產生名爲 foo.cdi 輸出檔。

由於 CDI 是一套擴充 DVI 格式的檔案規格，其中包含中文碼與中文字體等資訊，所以一般的 DVI 驅動程式無法解讀 CDI 檔。你必須用底下三節所介紹的方式來預覽或列印 CDI 檔。

假定 cdi2dvi 已設定完成，你只要鍵入以下兩行指令：

```
cdi2dvi foo
yap foo
```

即可在 Yap 的視窗中預覽到排版的結果。由於 Yap 具有自動更新的功能，在螢幕上保留 Yap 視窗，之後只要鍵入

```
cdi2dvi foo
```

無須再執行 Yap 程式，即可在 Yap 視窗中看到更新的結果。

cdi2dvi 執行時，會在工作目錄中新建一個名爲 cdifont 的子目錄。然後將產生的中文字型檔與 tfm (tex font metric) 檔置於其中。cdi2dvi 每次執行都會先刪除舊的 cdifont 子目錄，因此你不必擔心中文字型檔會日益增加，而佔據了大量硬碟空間。當你完稿後，可用底下的指令自行刪除 cdifont 目錄

```
del/s/q cdifont
```
（Windows NT/2000/XP）

或

`deltree cdifont` （Windows 95/98）

cdi2dvi 指令的語法如下：

`cdi2dvi` [*flags*] *cdi_file*[.cdi] [*dvi_file*[.dvi]]

其中，只有 *cdi_file* 不可缺外，其餘的參數與副檔名均可省略。若沒有指定 *dvi_file* 輸出檔，則 DVI 輸出檔將以 *cdi_file*.dvi 爲名。cdi2dvi 的指令選項如下：

-p *mode* *x*_resolution *y*_resolution

設定印表機的輸出模式名稱（*mode*）、水平解析度（*x*_resolution）、與垂直解析度（*y*_resolution）。

-s *mode* *x*_resolution *y*_resolution

設定螢幕的輸出模式名稱（*mode*）、水平解析度（*x*_resolution）、與垂直解析度（*y*_resolution）。

一般而言，你只要按照安裝手冊 [6] 的步驟設定好 cdi2dvi 之後，就不需要使用上述的指令選項。

2.3 轉換成 PostScript 檔

用 cdi2dvi 轉換所得的 DVI 檔可以再用 dvips 轉成 PostScript 格式。譬如：以下兩行指令即可把 foo.cdi 轉成 foo.ps：

```
cdi2dvi foo
dvips foo
```

2.4 使用 cdi2pdf 轉檔程式

cdi2dvi 產生的 DVI 檔，雖然可以用 dvipdfm 程式轉成 PDF 檔，但是在所得的 PDF 檔中，雙位元組的中文碼被換成單位元組的內碼，而且內嵌的中文字型是 bitmap 格式。

cdi2pdf 工具程式是筆者修改 dvipdfmx[1] 程式的成果。 cdi2pdf 把 CDI 檔直接換轉成 PDF 檔，並具有以下兩項優點：

- 保留原來的中文碼，因而在 Acrobat Reader 中可以使用文句搜尋的功能。
- 內嵌 TrueType 中文字型，在螢幕上瀏覽更美觀，而且字型資料只存使用到的字元，而不是整個字型檔，可有效地降低檔案大小。

[1] dvipdfm 的加強版，可處理 CJK TrueType 字型

按照安裝手冊 [6] 的步驟設定好 cdi2pdf 之後，只要輸入指令：

```
cdi2pdf foo
```

就可以把 `foo.cdi` 轉成 `foo.pdf`。

註：由於 Acrobat Reader 6 會把開啓的 PDF 檔設爲防寫，所以執行 cdi2pdf 之前，你必須先關閉預覽中的 PDF 檔，才不會造成無法寫入檔案的程式錯誤。

註：目前版本的 cdi2pdf 不支援斜體字和反白字等字體的設定。此外，由於 PDF 的規格限制：(1) 外字集字元可能無法正確地顯示 (2) 文件中的圖檔格式只能使用 JPG, PNG, 或 GIF，並不支援 EPS。

3 字元間距

這一節介紹 pTEX 設定字元間距的原則，以及利用雙字縮合（kerning）的技巧來排版出更美觀的文件。

3.1 空白字元的處理

空白字元在 TEX 文件中具有語法與語意兩種用途。在語法方面，空白字元用來分隔英文字（words）與分隔語元（tokens）。在語意方面，空白字元用來產生彈性寬度的空白間隔，並作爲可能的斷行點（line-breaking point）。底下兩點是 TEX 處理空白字元的基本原則：

- 除非空白字元被定義成產生空白寬度的指令（如在 `\verb` 指令或 `verbatim` 環境中），否則，連續多個空白字元會被刪減成一個空白字元。
- 跟隨在巨集指令之後的空白字元是用來隔斷指令名稱，排版時將全數刪除，而不會產生空白寬度。

pTEX 允許使用者自由調整中文的字間距和中英文的字間距。然而，這個功能增加了空白字元的處理複雜性，因爲空白字元不再單純地只產生英文的字間距，而可能產生以下三種空白間隔：

- 英文空白：英文字之間的空白間隔
- 中文空白：中文字之間的空白間隔
- 中英空白：中文字與英文字之間的空白間隔

絕大部分的情況下，pTEX 程式會正確地判斷出空白的種類。底下我們列出一些關於空白的規則，讓你能更精確地掌握空白字元的使用：

★ 中文字元之間不需要以空白字元隔開，pUfTEX 會自動在兩者之間插入中文空白。此外，在中文字元間加入一個或一百個空白字元並無差異，因爲它們所產生的排版效果與不加入空白字元完全相同。如果想增加某兩個中文字元的間距，你可以使用 \␣、\,、或 \hspace 之類的指令。

★ 中文字元與英文字元之間不需要以空白字元隔開，pUfTEX 會自動在兩者之間插入中英空白。此外，在兩者之間加入一個或一百個空白字元並無差異，因爲它們所產生的排版效果與不加入空白字元完全相同。

★ 中文字元與行間（inline）數學式之間不需要以空白字元隔開，pUfTEX 會自動在兩者之間插入中英空白。此外，在兩者之間加入一個或一百個空白字元並無差異，因爲它們所產生的排版效果與不加入空白字元完全相同。譬如底下兩行：

公式 $x^2=4$ 的正解爲 $x=2$。
公式␣$x^2=4$␣的正解爲␣$x=2$。

（符號 ␣ 代表空白字元）產生的結果均爲：

公式 $x^2 = 4$ 的正解爲 $x = 2$。

★ pUfTEX 把正常中文字元的類別碼預設爲 letter（參見 5.7 節），因此你必須用至少一個空白字元將 TEX 指令與其後的中文字元隔開，否則該中文字元會被視爲指令名稱之一部份，而造成錯誤。比方說：

我愛 \TeX␣的排版能力

產生的結果爲

我愛 TEX 的排版能力

但是

我愛 \TeX 的排版能力

則會讓 pUfTEX 誤認「\TeX 的排版能力」爲一個指令，而造成指令未定義的錯誤。

★ pUfTEX 將中文符號字元的類別碼預設爲 other，因此 TEX 指令之後跟著中文符號時，你可以省略兩者間的空白。比方說以下三種輸入方式：

「我愛 \TeX。」、「我愛 \TeX␣。」、「我愛 \TeX\␣。」

都會產生相同的結果：「我愛 TEX。」然而，筆者建議你：

不要在中文標點符號之前使用空白指令 \␣。

否則會使得該標點符號的「避首 / 尾」功能失效（參見第 6.2 節）。

★ 請使用空白字元將 TEX/LATEX 指令與前後文字隔開，讓 pUfTEX 來決定這些空白的種類。

★ 若指令前後的空白字元無法產生正確的空白間距時，你可以將其改爲 \␣。通常這樣就可以解決大部分的問題。

★ 若改爲 \␣ 後仍無法獲得正確的空白，你可以用 pUTEX 的指令 \PUXspace（英文空白）、\PUXcspace（中文空白）、或 \PUXcespace（中英空白）來直接設定所需的空白。

★ 你可以在兩個字元間插入 \kern0pt 或 \hbox{} 來制止 pUTEX 在兩者之間加入空白間距。譬如：底下三行的輸入方式：

```
我喜歡在 WWW 上搜尋資料
我\hbox{}喜\hbox{}歡\hbox{}在\hbox{}WWW\hbox{}上搜尋資料
我\kern0pt喜\kern0pt歡\kern0pt在\kern0ptWWW\kern0pt上搜尋資料
```

所產生的結果分別爲：

我喜歡在 WWW 上搜尋資料
我喜歡在WWW上搜尋資料
我喜歡在WWW上搜尋資料

第二、三行的長度明顯地短於第一行的長度，因爲字元間並未加入空白間距。又比方說，\CDOTS 指令（定義在 chinesebasic 檔中）可用來產生中式連點「　　」，它的定義如下：

```
\newcommand*{\CDOTS}{\mbox{  \kern0pt  }}
```

如果兩個　之間不加 \hbox{} 的話，就會得到「　　」中間多出一個中文空白的寬度。

★ 在 \verb 指令與 verbatim 環境中的任何空白字元將會保留而不刪除，並產生英文空白字元的寬度。譬如：

```
\verb|姓名␣␣␣年齡␣␣␣住址␣␣␣Phone|
```

產生的結果爲：

姓名　年齡　住址　Phone

列完以上的通則後，底下我們舉一些輸入方式的範例。

例 **3–1**　如果巨集指令展開後的第一個語元是中文或英文字元（如 \TeX），則你不必在該巨集指令之前鍵入空白字元。譬如：假定指令 \A 的定義為：

```
\newcommand{\A}{排版}
```

則底下兩行輸入方式：

```
使用␣\TeX␣來␣\A␣真方便！
使用\TeX␣來\A␣真方便！
```

會產生相同的結果如下：

使用 TEX 來排版真方便！
使用 TEX 來排版真方便！

例 3–2 如果你使用字型設定群組，如 {\it ...} 或 {\itshape ...} 等等，則你不必在其群組之前後鍵入空白字元。譬如底下的兩行輸入方式：

```
\PUTeX␣處理空白的␣{\it 規則一}␣是 ...
\PUTeX␣處理空白的 {\it 規則一} 是 ...
```

會產生相同的結果如下：

PUTEX 處理空白的規則一是 ...
PUTEX 處理空白的規則一是 ...

在少數的狀況下，PUTEX 還是得要靠你的輔助才能正確地處理字元間的空白間隔。底下，我們示範用 \␣ 來取代空白字元 ␣ 以解決錯誤空白間隔的問題。

例 3–3 除了「\verb 指令中的第一個字元是中文字元」這個情況外，\verb 指令前後的空白字元都會產生正確的空白間隔。當 \verb 指令中的第一個字元是中文字元時，你必須用 \␣ 來取代 \verb 指令之前的空白字元 ␣。譬如：

```
我服務於␣\verb|Providence University|␣資管系。
我服務於\␣\verb| 靜宜大學 |␣資管系。
我服務於␣\verb| 靜宜大學 |␣資管系。
```

產生的結果分別為：

我服務於 Providence University 資管系。
我服務於靜宜大學資管系。
我服務於 靜宜大學資管系。

前兩行的結果是正確的；最後一行「於」與「靜」之間被誤認成英文空白，因此顯得稍寬而不美觀。

例 3–4 由於 \underline 指令是利用數學式的技巧來製作，因此 PUTEX 會將底線字與前後中文之間的空白視為中英空白。譬如：

```
你␣\underline{千萬不可以}␣說謊
```

產生的結果為：

你 千萬不可以 說謊

句中底線字與前後中文字的間距顯然不對。這時你可以用 \␣ 來取代空白字元 ␣：

```
你 \␣\underline{ 千萬不可以 }\␣ 說謊
```

而得到底下的正確結果：

你千萬不可以說謊

例 3–5 假定底線字的首尾是英文字元，則其前後的空白會正確地解釋成英式空白。譬如：

```
你 ␣\underline{never}␣ 說謊
```

將產生正確的結果：

你 never 說謊

例 3–6 有些指令（如 \index）會在前後「暗中」加入一些 TEX 內部資料結構，使得 PUTEX 無法正確地判斷空白的種類，即使用前述之 \␣ 的技巧也無法解決。比方說，在本手冊中，筆者定義如下的指令：

```
\newcommand*{\MikTeX}{Mik\TeX\index{MikTeX}}
```

讓指令\MikTeX 除了在本文中顯示出 MikTEX 以外，也自動地加入索引之中。以下三種輸入方式：

```
仍然得仰賴 \MikTeX␣系統的支援。
仍然得仰賴 \MikTeX\␣系統的支援。
仍然得仰賴 \MikTeX{}␣系統的支援。
```

產生的結果如下：

仍然得仰賴 MikTEX系統的支援。
仍然得仰賴 MikTEX系統的支援。
仍然得仰賴 MikTEX 系統的支援。

第一種輸入方式：前者 X 之後的空白被 TEX 吸收掉，使得兩字之間並無空白間隔，也使得 PUTEX 誤加入中文空白。第二種輸入方式：\␣ 被 PUTEX 解釋成中式空白。第三種則產生正確的結果，其中的 {}（括弧之間不可有空白字元）阻止 TEX 吸收掉其後的空白字元。

例 3–7 你可以定義巨集指令來避免多打空白字元。譬如有了以下的定義：

```
\newcommand{\pgref}[1]{ 第 \pageref{#1}␣頁 }
```

你就可以打

```
請參閱 \pgref{...} 之說明          %（「之」字前不需空白字元）
```

來取代

```
請參閱第 \pageref{...}␣頁之說明    %（「頁」字前需要空白字元）
```

3.2 雙字縮合

某些英文字元並置在一起時，由於字形的關係，會顯得間距過寬，譬如底下左邊所示的 A 和 V 兩個字母：

AV AV

雙字縮合（kerning）是把兩個字元縮近距離，以得到更美觀的效果，如上面右邊所示的 A 和 V 兩個字母。TEX 之所以是一個出色的排版軟體，其中的一個原因是：不需要使者者操心，它就能自動地處理英文字間的雙字縮合。

正常的中文字元都是方塊字，不太需要做雙字縮合，但是中文標點符號則不然，譬如底下的句子：

（「之」字前不需空白字元），如此即完成。

前頭的（ 和「 之間有點寬，後頭的 ）和，之間顯然過寬。你可以用 TEX 的 \kern 指令來縮短它們的寬度，譬如用以下的輸入：

```
（\kern-0.25em「之」字前不需空白字元）\kern-0.5em，如此即完成。
```

（em 是長度單位，約等於目前字型的大小），即可產生底下比較均勻美觀的結果：

（「之」字前不需空白字元），如此即完成。

不過，若每次都要如此繁瑣地輸入 \kern 指令，顯然費時費力又容易出錯。你可以在文件前端先定義好以下的中文標點符號指令：

```
\def\，{\kern -0.5em，}   % 把逗點往前縮 0.5em
\def\「{\kern -0.25em「}  % 把左引號往前縮 0.25em
```

當需要縮合時，就用這些指令來取代原來的標點。譬如底下我們用 \「 來取代左引號和用\，來取代逗點：

```
（\「之」字前不需空白字元）\，如此即完成。
```

同樣地可以得到前述較美觀的結果。

由於 pTeX 把中文標點符號的類別預設爲 other，所以前述的中文標點符號指令之後不需要以空白來隔斷，這讓你可以更輕鬆地使用它們。

筆者撰寫本文時，即在文件前端定義了若干中文標點符號指令，用在所有需要縮合的地方。有興趣的讀者可以檢視本文原始檔，看看它們的定義方式與使用時機。如果讀者手邊還保留有舊版的 pTeX 手冊，拿來和本手冊相對照，即可以發現新手冊標點符號的編排比較緊湊美觀。

註：由於中文標點符號指令指定的縮合距離與使用的中文字貌有關，沒有放諸四海皆準的數值，因此 pTeX 不把它們定爲標準指令，而必須由使用者自行定義之。在筆者構想出高效率的中文雙字縮合完整解決方案之前，你可以用前述的技巧來達成雙字縮合的需求。

4 LaTeX 中文化

爲了讓使用者能夠更方便地利用 LaTeX 來產生中文文件， Big5LaTeX 提供以下三種文件類別（document class）：

`twarticle`	中文化的 LaTeX article 文件類別。
`twreport`	中文化的 LaTeX report 文件類別。
`twbook`	中文化的 LaTeX book 文件類別。

此外，針對某些與 Big5LaTeX 不相容的套件（如 fancyvrb）， Big5LaTeX 提供 twpkgpatch 套件來解決不相容的問題。

4.1 chinesebasic 套件

這個套件包含中文化的共同設定，而且自動被前述的中文化文件類別載入。因此，以下所述的巨集指令，同樣適用於中文化文件類別。

註：爲了同時支援 Big5 和 GBK 中文碼，pTeX 4 修改之前版本所含的 twbase 套件並更名爲 chinesebasic 套件。

PUTEX 的 Logo

chinesebasic 定義了以下三個 Logo 指令：

- `\PUTeX` 產生 pTeX。
- `\PULaTeX` 產生 pLaTeX。
- `\PULaTeXe` 產生 pLaTeX 2_ε。

中式連點

chinesebasic 套件提供 \CDOTS 指令，其定義如下：

```
\newcommand*{\CDOTS}{\mbox{  \kern0pt  }}
```

比方說，若你輸入：

鼠、牛、虎、兔、\CDOTS、狗、豬

所得之結果將爲：

鼠、牛、虎、兔、　　、狗、豬

註：請勿輸入連續兩個「　」中文字元來產生上述的中式連點，理由如第 12 頁所述。

中式日期

chinesebasic 套件提供 \roctoday 指令。它的作用如同 LaTeX 的 \today 指令，用來輸出今天日期，不過 \roctoday 產生類似

中華民國八十七年五月二十日

的中式日期格式。此外， \Roctoday 指令則以大寫中文數字來顯示年月日，如：

中華民國捌拾柒年伍月貳拾日

最後， \rocyear 指令產生阿拉伯數字格式的民國年份，如：

今年是民國 ␣\rocyear␣年

產生的結果是：

今年是民國 93 年

以中文數字顯示 counter 的值

chinesebasic 套件提供的 \cnumber 或 \Cnumber 指令可用來將 counter 的數值以小寫或大寫中文數字的方式顯示。它們的作用有如 LaTeX 的 \arabic 指令。譬如：

```
\cnumber{\thesection}     % 以小寫中文數字顯示 counter \thesection 的值
\Cnumber{\thesection}     % 以大寫中文數字顯示 counter \thesection 的值
```

中式編號條列

chinesebasic 套件提供了兩個類似 enumerate 的中式編號條列環境。一個是以小寫中文數字來編號，另一個則是以大寫中文數字來編號。我們用以下的例子來說明它們的用法與所得的結果。

例 **4–1** 底下是使用「一二三」小寫中文數字編號條列環境的範例。左側是輸入方式，右側則為排版的結果。

輸入行

```
\begin{一二三}
  \item 首先，
    \begin{一二三}
      \item 第一條
        \begin{一二三}
          \item 第一項
            \begin{一二三}
              \item 第一點
            \end{一二三}
        \end{一二三}
    \end{一二三}
  \item 其次，
  \item 最後，
\end{一二三}
```

排版結果

一、 首先，

1. 第一條

(a) 第一項

i. 第一點

二、 其次，

三、 最後，

例 **4–2** 底下是使用「壹貳參」大寫中文數字編號條列環境的範例。左側是輸入方式，右側則為排版的結果。

輸入行

```
\begin{壹貳參}
  \item 首先，
    \begin{壹貳參}
      \item 第一條
        \begin{壹貳參}
          \item 第一項
            \begin{壹貳參}
              \item 第一點
            \end{壹貳參}
        \end{壹貳參}
    \end{壹貳參}
  \item 其次，
  \item 最後，
\end{壹貳參}
```

排版結果

壹、 首先，

1. 第一條

(a) 第一項

i. 第一點

貳、 其次，

參、 最後，

標題指令名稱	原英文標題	中文標題	附註
\prefacename	Preface	序	
\partname	Part	篇	
\contentsname	Contents	目錄	
\listfigurename	List of Figures	圖形目錄	
\listtablename	List of Tables	表格目錄	
\indexname	Index	索引	
\appendixname	Appendix	附錄	
\abstractname	Abstract	摘要	
\tablename	Table	表格	
\figurename	Figure	圖	
\pagename	Page	頁	
\refname	References	參考文獻	article
\bibname	Bibliography	參考文獻	report & book

表格 1: 中文標題

4.2 標題中文化

中文化的文件類別：twarticle、twreport、和 twbook 會把英文標題改成中文。表格 1 列出所更改的標題。若你不滿意任何一個中文標題的名稱，你可以用 `\renewcommand` 指令重新定義之。譬如，若想將預設的中文標題「序」改成「自序」，你只要在文件前端鍵入如下的一行即可：

```
\renewcommand*{\prefacename}{自序}  % Preface
```

4.3 twarticle 文件類別

`twarticle.cls` 是 LaTeX `article.cls` 的中文化文件類別。你可以用下面的方式來使用它：

```
\documentclass[options]{twarticle}
```

`twarticle.cls` 與 `article.cls` 功能相同，另外也加進了 chinesebasic 套件的內容。

4.4 twreport 與 twbook 文件類別

twreport 和 twbook 分別是 LaTeX report 和 book 的中文化文件類別。你可以用下面的方式來使用它們：

```
\documentclass[options]{twreport}
```

或

```
\documentclass[options]{twbook}
```

由於 twreport 和 twbook 雷同，因此我們就以 twreport 爲例來說明兩者的使用方式。

twreport 除了擁有 report 所有的功能以外，另外加進了 chinesebasic 套件的內容與中文標題的設定。此外，

- twreport 修改了 \chapter 指令與 \appendix 指令，使前者產生「第一章」之類的章序號，後者產生「附錄 A」之類的附錄序號。
- 目錄裏阿拉伯數字格式的章序號也會改成「第一章」、「第二章」、之類的中式序號。
- twreport 定義了 \twchaptername 巨集指令，其內容是目前的中文章序號，如「第一章」等等。

4.4.1 文件類別的選項

一般而言，twreport 的章序號是使用小寫中文數字，章標題則是靠左對齊。不過，你可以用以下的文件類別選項來改變這些設定：

chapscnum	章序號使用俗體中文數字，如「第廿一章」。
chapucnum	章序號使用大寫中文數字，如「第貳拾壹章」。
chapfcnum	章序號使用正式中文數字，如「第壹拾壹章」。
chapacnum	章序號使用全形阿拉伯中文數字，如「第２１章」。
rightchhd	章標題靠右對齊。
centerchhd	章標題置中對齊。

比方說，以下的文件類別選項設定：

```
\documentclass[chapucnum,rightchhd]{twreport}
```

會使得章序號改用大寫中文數字，如「第貳拾壹章」，以及使得章標題靠右對齊。

註：目錄及頁首中的章序號也會使用所設定的章序號中文數字格式。

4.4.2 改變章序號方式

如果想讓 \chapter 產生「第一講」，而非原本的「第一章」之類的章序號，你只要重新定義指令 \chaptername 即可：

```
\renewcommand*{\chaptername}{講}
```

4.4.3 改變章標題的字型

twreport 內部利用指令 \TWchapHeadingFont 和 \TWchapContentFont 來分別設定章標題及其在目錄中所使用的字型。它們預設的定義如下：

```
\newcommand*{\TWchapHeadingFont}{\normalfont\Huge\bfseries}
\newcommand*{\TWchapContentFont}{\large\bfseries}
```

使得章標題裏的中文在兩個地方都使用中黑體字型。如果你想改變章標題的中文字型，可以重新定義以上的兩個指令。譬如：

```
\renewcommand*{\TWchapHeadingFont}{\normalfont\Huge\bfseries\PUXFbr}
```

將使得章標題的中文改用粗圓體字型。同樣地，以下的定義：

```
\newcommand*{\TWchapContentFont}{\Large\bfseries\PUXFmk}
```

將使得在目錄裏的章標題使用比較大的字型，以及中文改用中楷體字型。

註：如果你對以上的 \PUXFbr 與 \PUXFmk 指令感到「霧煞煞」，沒關係，因為它們是 PUTEX 用來變更中文字貌的指令。我們會在 5.2 節介紹它們。

4.4.4 中文參照號碼

twreport 提供指令 \twref 讓你產生中文參照號碼。比方說，你可以用 \label 指令來定義某一章的標籤，如：

```
\chapter{製作中文索引}\label{chap:makeidx}
```

然後利用以下的指令：

```
\newcommand{\chapref}[1]{第 \twref{chap:#1}\chaptername}
```

使得 \chapref{makeidx} 產生出如「第八章」而非「第 8 章」的中文參照號碼。

由於指令 \twref 只是將參照號碼的第一個數字改成中文數字，所以應用在節或數學公式時，會得到「八 .1 節」或「公式九 .3」的結果。基於這個限制，指令 \twref 比較適用於章號或頁碼等只由一個數字所組成的參照號碼。

4.5 twpkgpatch 套件

Big5LaTeX 只修改了 LaTeX 的 \verb 指令與 verbatim 環境的定義，使其中的空白字元能夠正確地解讀，其他所有的 LaTeX 指令都忠實地保留下來。因此 Big5LaTeX 與 LaTeX 相容度極高，幾乎所有的 LaTeX 的套件都可以一成不變地使用在 Big5LaTeX 之中。然而，有些套件提供類似 \verb 指令與 verbatim 的環境（如 fancyvrb ）就會與 Big5LaTeX 不相容。因此 Big5LaTeX 提供 twpkgpatch 套件來解決這類的問題。

以 fancyvrb 套件爲例，你可以用下面的方式來解決不相容的問題：

```
\usepackage{fancyvrb}
\usepackage[fancyvrb]{twpkgpatch}
```

即載入 fancyvrb 套件之後，再以 fancyvrb 爲選項載入 twpkgpatch 套件。

註：到目前爲止，筆者只發現 fancyvrb 套件與 Big5LaTeX 不相容。如果你發現到其他不相容的套件時，請通知筆者，我會盡力加以解決。這種狀況應該是極少發生的吧：）

4.6 一些 LaTeX 的中文化技巧

在這一小節中，我們示範 LaTeX 中文化的一些技巧。

4.6.1 設定節標題的中文字型

標準的 LaTeX 採用 cmbx（粗體字）爲節標題的英文字貌，而 Big5LaTeX 檔又將 cmbx 對應至中黑體，所以節標題的中文字將以中黑體出現。你可以利用 *LaTeX Companion* 書中所述的技巧來變更節標題使用的中英字貌 [2, pp 27–31]。

例 **4–3** 若想用粗圓體取代中黑體作為節標題的中文字貌，你可以採用以下的設定方式：

```
\newcommand{\seccface}{\PUXfacematch\PUXFbr}  % 節標題的中文字貌
\renewcommand{\section}{\@startsection
  {section}%                                  % the name
  {1}%                                        % the level
  {0mm}%                                      % the indent
  {-1\baselineskip}%                          % the beforeskip
  {0.5\baselineskip}%                         % the afterskip
  {\bfseries\Large\seccface}}%                % the style
\renewcommand{\subsection}{\@startsection
  {subsection}%                               % the name
  {2}%                                        % the level
  {0mm}%                                      % the indent
  {-\baselineskip}%                           % the beforeskip
```

```
  {0.5\baselineskip}%                          % the afterskip
  {\bfseries\large\seccface}}%                 % the style
\renewcommand{\subsubsection}{\@startsection
  {subsubsection}%                             % the name
  {3}%                                         % the level
  {0mm}%                                       % the indent
  {-\baselineskip}%                            % the beforeskip
  {0.5\baselineskip}%                          % the afterskip
  {\bfseries\normalsize\seccface}}%            % the style
```

4.6.2 中式節序號

如果想用中式數字作爲節序號，你可以參考 [2, p.24] 所述的技巧。比方說，以下三個巨集的重新定義：

```
\renewcommand{\thesection}{\cnumber{section}}
\renewcommand{\thesubsection}{\thesection.\cnumber{subsection}}
\renewcommand{\thesubsubsection}{\thesubsection.\cnumber{subsubsection}}
```

會使得

```
\section{ 簡介 }
\subsection{ 零與一 }
\subsubsection{ 簡史 }
```

產生類似如下的結果：

一　簡介
一.一　零與一
一.一.一　簡史

4.6.3 設定中式的頁首標題

你可以利用以下的 LaTeX 指令來加入頁首標題：

```
\pagestyle{headings}
```

或利用 fancyhdr 套件（原名爲 `fancyheading.sty`[2, p. 224]）來設定中式的頁首標題。譬如底下是一個適用於 twreport 與 twbook 的設定方式：

```
\usepackage{fancyhdr}
\pagestyle{fancyplain}
\renewcommand{\chaptermark}[1]{%
   \markboth{\twchaptername\ \ \ #1}{}}
\renewcommand{\sectionmark}[1]{\markright{\thesection\ \ #1}}
\lhead[\fancyplain{}{\bfseries\thepage}]%
    {\fancyplain{}{\bfseries\rightmark}}
\rhead[\fancyplain{}{\bfseries\leftmark}]%
    {\fancyplain{}{\bfseries\thepage}}
\cfoot{}
```

以上的設定將會產生類似下面所示的頁首標題：

11.1 基本概念 （奇數頁） **391**

392 （偶數頁） **第十一章 動畫製作**

4.6.4 中文的定理標題

你可以仿照下面的定義方式，把定理類（Theorem-like, [3, p. 79]）環境的標題中文化：

```
\newtheorem{thm}{定理}
```

有此定義之後，下列四行的輸入：

```
\begin{thm}
假定 $a$, $b$, $c$, 和 $n$ 均爲正整數。當 $n\geq 3$ 時，$a^n + b^n \neq c^n$
成立。此爲 {\PUXFmb 費瑪最後定理 }。
\end{thm}
```

可得到如下的結果：

定理 1 假定 a, b, c, 和 n 均爲正整數。當 $n \geq 3$ 時，$a^n + b^n \neq c^n$ 成立。此爲**費瑪最後定理**。

你可以用中文字貌匹配指令（參見 5.2.2 節）來改變定理類環境所使用的中文字貌。譬如：若再定義一個環境，將 thm 內的中文改用 `\PUXFmfs`（中仿宋）：

```
\newenvironment{nthm}{\begin{thm}\PUXfacematch\PUXFmfs}{\end{thm}}
```

則

```
\begin{nthm}
假定 $a$, $b$, $c$, 和 $n$ 均爲正整數。當 $n\geq 3$ 時，$a^n + b^n \neq c^n$
成立。此爲 {\bf 費瑪最後定理 }。
\end{nthm}
```

將可得到如下的結果：

定理 2 假定 a, b, c, 和 n 均為正整數。當 $n \geq 3$ 時，$a^n + b^n \neq c^n$ 成立。此為**費瑪最後定理**。

其中大部分的中文均改成了「中仿宋體」。

4.6.5 其他

由於中文字筆畫遠比英文字母來得複雜，因此 LaTeX 預設的單行間距往往使得中文文件的頁面過於擁擠。你可以重新定義 \baselinestretch 來增加行距以避免這個問題。筆者建議以下的設定：

```
\renewcommand{\baselinestretch}{1.2}
```

或

```
\renewcommand{\baselinestretch}{1.25}
```

最後，本手冊的範例均以反白的例字爲標誌，並以節爲範圍來編號。此範例環境的設定方式如下所示：

```
\PUXcfacedef\PUXeg=eg 中楷 s=v
\newfont{\egfont}{CFONTeg11}
\newfont{\egcntfont}{CFONTtb10}
\newcounter{example}[section] \setcounter{example}{1}
\newenvironment{example}{\bigskip\noindent%
  \refstepcounter{example}%
  {\egfont 例 }\ \ {\bf\small \thesection--\theexample}%
  \quad\small\egcntfont}{\medskip}
```

5 PUTEX 基本指令與內部參數

針對中文排版的特性，PUTEX 增加了一些基本指令和內部參數。在這一節中，我們介紹提供給一般使用者的指令與參數，下一節介紹提供給 TEX 格式檔（format）或 LaTeX 套件製作者的指令與參數。

名稱	用途	型態	參閱
\puxCharSet	設定文件的字元集	L	6.1 節
\puxCJKinput	控制是否讀入 CJK 字元	L	5.1 節
\puxgCEspace	調整所有的中英字間距	G	5.4.4 節
\puxgCfaceDepth	調整所有的中文字貌深度	G	5.3.1 節
\puxgCspace	調整所有的中文字間距	G	5.4.1 節
\puxgRotateCtext	逆時鐘旋轉中文字 90 度	O	5.9 節
\puxCJKcharOther	設定中文字元的類性	L	5.7 節
\puxXspace	保留空白字元	L	7 節

表格 2: pTEX 新增的參數（依字母序排列）

爲了避免與現有 TEX/LATEX 指令的名稱起衝突，我們採用底下的字頭規則來命名 pTEX 指令與參數：

- 指令的名稱均以大寫英文 PUX（代表 pTEX eXtension）起頭。
- 區域性（local）參數的名稱以小寫英文 pux 起頭。這些參數的作用範圍僅及於設定的群組（group）之內。
- 全域性（global）參數的名稱以小寫英文 puxg 起頭。這些參數的作用範圍涵蓋整個文件範圍。

在表格 2 和 3 中，我們按照字母序列出這些參數與指令的名稱、用途、型態、以及本手冊中的參照章節。其中型態欄中的字母含意如下：

G 指令或參數具有全域性的影響力。其效力將維持至重新定義爲止。

L 指令或參數具有區域性的影響力。其效力不會影響外部群組。

D 指令或參數的影響力與範疇無關（don't care）。

O 參數僅能變更其值一次（once）。

除此之外，pTEX 也擴充了 TEX 的 \font 指令的語法與功能，使其能夠以類似定義英文字型的方式來定義中文字型。這樣做的目的是爲了讓 pTEX 也能夠使用 LATEX 2ε 的 NFSS2 規格。細節請參見第 5.2.4 節

5.1 關閉中文字元的讀入

傳統 TEX 使用碼值 0 – 127 的 ASCII 字元。中文字元由兩個 bytes 所組成，而且第一個 byte 的值一定大於 128。pTEX 即根據這兩個規則來分辨出輸入檔中單位元組的英文字元與雙

名稱	用途	型態	參閱
\PUXacnumber	轉換成中文數字	D	5.5 節
\PUXcadcode	設定中文字元的類別碼	L	5.7 節
\PUXcespace	插入中英文空白間距	D	5.4.7 節
\PUXcfacecespace	調整字貌的中英字間距	G	5.4.5 節
\PUXcfacecspace	調整字貌的中文字間距	G	5.4.2 節
\PUXcfacedef	定義中文字貌	G	5.2.1 節
\PUXcfacedepth	調整中文字貌深度	G	5.3.2 節
\PUXcfontcespace	調整字型的中英字間距	G	5.4.6 節
\PUXcfontcspace	調整字型的中文字間距	G	5.4.3 節
\PUXchar	輸入中文字元內碼	D	5.6 節
\PUXchardef	定義中文字元指令	L	5.6 節
\PUXcjknumber	轉換成中文數字	D	5.5 節
\PUXcnumber	轉換成中文數字	D	5.5 節
\PUXcspace	插入中文空白間距	D	5.4.7 節
\PUXdumpfontinfo	輸出字型資訊	D	7 節
\PUXexspace	插入英文空白間距	D	5.4.7 節
\PUXfacematch	匹配中英文字貌	L	5.2.2 節
\PUXfcnumber	轉換成中文數字	D	5.5 節
\PUXfontmatch	匹配中英文字型	L	5.2.5 節
\PUXlocalnames	定義特殊字元表	L	5.7 節
\PUXrangecadcode	設定中文字元區間的類別碼	L	5.7 節
\PUXrangetypecode	設定中文字元區間的型態碼	L	6.2 節
\PUXspace	插入英文空白間距	D	5.4.7 節
\PUXscnumber	轉換成中文數字	D	5.5 節
\PUXsetdefaultcface	設定預設的中文字貌	L	5.2.2 節
\PUXtypecode	設定中文字元的型態碼	L	6.2 節
\PUXucnumber	轉換成中文數字	D	5.5 節

表格 3: pTEX 新增的指令（依字母序排列）

位元組的中文字元。然而，許多歐洲國家的字元集通常含有 256 個字元，使用到 129 – 255 的碼值，因此文件中若含有這些國家的字元時，就會造成 PUTEX 解讀中文字元的困難。又有少部分 LATEX 套件內含碼值超過 128 的字元，也會造成 PUTEX 處理上的錯誤，如 url.sty 與使用到它的 hyperref.sty。爲了克服這類問題，PUTEX 提供以下控制中文字元讀取方式的參數：

\puxCJKinput[=]0 | 1

（方括號內的項目表示可省略，0 | 1 表示 0 或 1 兩者擇一。）若設爲 1（預設值），則正常解讀中文字元；若設爲 0，則取消解讀中文字元，而將其視爲兩個 bytes。

例 **5–1** 如果你檢視本文的 LATEX 原始檔，可以發現以下幾行：

```
\puxCJKinput=0    % 關閉中文輸入，避免讀入 url.sty 時發生錯誤
\usepackage[dvipdfm,
  pdftitle={PUTeX User's Guide},
  pdfauthor={Chey-Woei Tsay},
  pdfpagemode=UseOutlines,
  bookmarks,bookmarksopen,
  pdfstartview=FitH,
  colorlinks, linkcolor=blue,citecolor=blue,
  urlcolor=red,
  ]
  {hyperref}
\puxCJKinput=1    % 重新開啓中文輸入
```

例 **5–2** 當中文輸入處於關閉的狀態時，你可以用已經定義成產生中文字句的巨集或 \PUXchar 指令（後面會介紹）來輸入少量的中文，如下面所示的方式：

```
\newcommand{\manual}{手冊}
\puxCJKinput=0
PUTeX \PUXchar"A8CF\PUXchar"A5CE\manual
\puxCJKinput=1
```

會產生如下的結果

PUTeX 使用手冊

註：此處的「使」字必須用「\PUXchar"A8CF」而不能用「\PUXchar‘\使」來產生。

最後，我們要特別提醒讀者：當你關閉中文輸入後，在同一群組使用中文之前記得要重新開啓，否則就無法正確讀入中文字元了。

5.2 中文字貌與字型

與其他的中文 TEX 系統相比，PUTEX 提供最強大卻最容易使用的中文字型處理技術。你可以使用任何中文 TrueType 字型，而不再受限於少少幾套的中文字型。你也不需要在硬碟中安裝頗佔空間的中文 bitmap 字型。此外，你只需使用幾行指令就能在 TEX/LATEX 中輕鬆地定義與使用各式各樣的中文字型。

在說明如何設定中文字型之前，我們先區別**字貌**（font face）與**字型**（font）這兩個術語。簡單地說：所謂字貌即字的外觀，而字型等同於字貌再加上大小宣告。PUTEX 的中文字貌是由底下的屬性來定義：

字體（**font look**） 區別出字的形狀。以中文爲例，細明體、楷書體、和**中黑體**是三種不同的字體。

字重（**font weight**） 用以設定字的粗細。中文 TrueType 字型可以定義出 9 種粗細程度。

字樣（**font shape**） 用以設定字的樣式，如正體（normal）、斜體（italic）、反白體（reversed）、與旋轉體（rotated）。

5.2.1 定義中文字貌

PUTEX 的 `\PUXcfacedef` 指令用來定義中文字貌。它的語法如下：

`\PUXcfacedef`*faceid* [=] *ename cname* [*attributes*]

（方括號內的項目表示可以省略），其中各參數的意義如下：

\faceid 代表此中文字貌的指令。你可以用此指令來改變目前使用的中文字貌（參見 5.2.3 節）。

ename 指定中文字貌的英文代名（identifier）。此名稱必須全由（大小寫）英文字母組成，不得使用其他的字元。也不得重複使用。

cname 指定中文字體的邏輯名稱（如細明、中楷等）。此名稱可用中文來命名。你必須在 cfonts.map 檔中定義其所對應的眞實中文字體（參見附錄A），否則此字貌將使用預設的眞實中文字體（通常爲新細明體），而無法列印出你所期望的字體。

attributes 這一個參數選項列指定以下的字貌屬性（可以任意組合和以任意的順序宣告）：

`t=100|···|900` 此屬性設定字貌的粗細（字重）。屬性值所對應的粗細程度如表格 4 所示。若沒有宣告此屬性的話，則字貌預設爲正常的粗細（即 $t = 400$）。

`d=`*n* 設定字貌深度（depth）爲字型大小的 $n/1000$。

100	特細	200	中細	300	細
400	正常	500	稍粗	600	中粗
700	粗	800	特粗	900	超粗

表格 4: 字重屬性值對應的粗細程度

`s=i` 設定字貌爲斜體字。

`s=r` 設定逆時針旋轉 90 度的字貌。

`s=v` 設定爲反白字貌。

底下是一些定義中文字貌的例子：

```
\PUXcfacedef\PUXFbody=body bodyface d=170   % bodyface 中文字貌，字深 0.17
\PUXcfacedef\PUXFtmb=tmb 細明 t=700         % 細明粗體中文字貌
\PUXcfacedef\PUXFtmi=tmi 細明 s=i           % 細明斜體中文字貌
\PUXcfacedef\PUXFtmv=tmv 細明 s=v           % 細明反白體中文字貌
\PUXcfacedef\PUXFtmr=tmr 細明 s=r           % 細明旋轉體中文字貌
\PUXcfacedef\PUXFmrrv=mrrv 中圓 s=r s=v     % 中圓旋轉反白體中文字貌
```

爲了方便起見，Big5TEX/LATEX 內建了若干常見的中文字貌。這些字貌均採用預設的屬性值。表格 5 列出這些字貌的邏輯名稱與英文名稱。

細明	tm	中明	mm	粗明	bm	特明	sm
細黑	tb	中黑	mb	粗黑	bb	特黑	sb
細圓	tr	中圓	mr	粗圓	br	特圓	sr
細楷	tk	中楷	mk	粗楷	bk	特楷	sk
細仿宋	tfs	中仿宋	mfs	粗仿宋	bfs	特仿宋	sfs
細隸書	tli	中隸書	mli	粗隸書	bli	特隸書	sli
細行書	tsn	中行書	msn	粗行書	bsn	特行書	ssn
細疊圓	tdr	中疊圓	mdr	粗疊圓	bdr	特疊圓	sdr
勘亭流	kd	古印	gn	綜藝	je	魏碑	wb
POP1	po	顏體	yt	儷中宋	lms	少女	girl
`default`	default						

表格 5: Big5TEX/LATEX 內建的中文字貌

爲了方便記憶，除了字貌 `default` 以外，筆者用下面兩個規則來命名字貌的英文名稱：

1. 如果字貌集有粗細之別（如細明、中明、粗明、和特明），則第一個字母用 t、m、b、和 s 四個字母分別代表細（thin）、中（medium）、粗（bold）、和特（super）四種粗細。如果無粗細之別，則不加此指示粗細的字母。

2. 在代表粗細的字母之後，使用一個或兩個英文字母來代表中文字體，如 m 代表明體、b 代表**黑體**、k 代表*楷書*、r 代表**圓體**、 fs 代表*仿宋體*等等。

此外，*若字貌的英文名稱爲 xx*，則指令命名爲 \PUXF*xx*（PUXF 中的字母 F 代表 Face 之意）。比方說：若英文名稱是 mm 的字貌，其字貌指令則命名爲 \PUXFmm。

當你加入自定的中文字貌，命名時，請務必避開這些內建字貌名稱，否則會造成重複定義的錯誤。譬如：

\PUXcfacedef\a=tm 細明	錯誤！與內建的字貌 tm 同名
\PUXcfacedef\b=mk 中楷	錯誤！與內建的字貌 mk 同名
\PUXcfacedef\PUXFmk=mk 中楷	錯誤！與內建的字貌 mk 同名

5.2.2 定義中英字貌的匹配規則

中/英字貌的匹配規則可說是 PUfTEX 的核心觀念之一，因爲透過這項技巧，中文字型變得更容易設定與使用。比方說： LATEX 英文字貌預設的使用方式爲：cmr 用於正常字、cmtt 用於定寬字、cmbx 用於粗體字、以及 cmti 用於斜體字。如果我們設定以下四個匹配規則：

- 英文用 cmr 字貌時，中文則用 \PUXFtm 字貌。
- 英文用 cmtt 字貌時，中文則用 \PUXFtm 字貌。
- 英文用 cmbx 字貌時，中文則用 \PUXFmb 字貌。
- 英文用 cmti 字貌時，中文則用 \PUXFmk 字貌。

就可以輕易地獲得以下的中文字型變化效果：

- 正常的中文字使用細明體。
- 出現在 \tt 環境的中文使用細明體。
- 出現在 \bf 環境的中文使用中黑體。
- 出現在 \em 和 \it 環境的中文使用中楷體。
- 改變英文字型大小的指令（如 \small 和 \Large）也同時改變中文字型的大小。

上述中/英字貌的匹配規則必須利用 PUfTEX 的 \PUXfacematch 指令來達成。此指令的第一種格式如下：

\PUXfacematch *eface_name* *cface_id*

其中，參數 *eface_name* 是英文字貌的名稱， *cface_id* 是中文字貌指令。指令的作用將使得出現在英文字貌 *eface_name* 環境中的中文字元，均採用 *cface_id* 所代表的中文字貌，且其字型大小與英文字型相同。比方說，底下的指令將英文字貌 cmr 匹配成 \PUXFtm（細明）：

```
\PUXfacematch cmr \PUXFtm
```

用白話一點的句子來說就是：「當英文使用 cmr 字型時，中文就使用細明體。」

爲了方便製作中文 TEX/LATEX 文件，Big5TEX/LATEX 預先定義了下列中/英字貌匹配規則：

```
\PUXfacematch cmr  \PUXFtm     % 正常字爲細明體
\PUXfacematch cmtt \PUXFtm     % 定寬字爲細明體
\PUXfacematch cmbx \PUXFmb     % 粗體字爲中黑體
\PUXfacematch cmti \PUXFmk     % 斜體字爲中楷體
\PUXfacematch cmbxti \PUXFmk   % 粗斜體用中楷體
```

這些規則讓使用者可以用字體變化指令（如 \rm, \em, \it, \bf, 等）來選擇中文字型；也可以用字型大小變化指令（如 \small, \large, \huge 等）來改變中文的大小。舉例來說，以上的字貌匹配可以得到下表所示的 LATEX 文字變化效果：

輸入	列印結果
正常文字 normal	正常文字 normal
{\bf bold（粗體字）}	**bold（粗體字）**
{\it italic（斜體字）}	*italic（斜體字）*
{\em emphasize（強調字）}	*emphasize（強調字）*
{\small 小字}	小字
{\Large 大字}	大字
{\Huge 巨字}	巨字

例 **5–3** 如果你想把正常中文字改用細圓體，只需將 cmr 的匹配中文字貌改成 \PUXFtr 即可。比方說，你可以在 LATEX 文件中鍵入：

```
\documentclass...
\PUXfacematch cmr \PUXFtr      % 正常中文字改用細圓體
```

或在 TEX 文件的第一行鍵入：

```
\PUXfacematch cmr \PUXFtr      % 正常中文字爲細圓體
```

例 5–4 你可以自行定義其它的中英字貌匹配。假定你想將 cmbxti（粗斜體英文字貌）與粗黑體中文字貌匹配，可以在 LaTeX 文件中鍵入：

```
\documentclass...
\PUXfacematch cmbxti \PUXFbb
```

或在 TeX 文件的第一行鍵入：

```
\PUXfacematch cmbxti \PUXFbb
```

若中文字元出現在未設定中文字型也未定義匹配的英文字貌環境中，Big5TeX/LaTeX 會自動選擇 default 字貌，換句話說， default 就是所謂的預設中文字貌。你可以用指令：

\PUXsetdefaultcface*cface_id*

改選 *cface_id* 爲預設的中文字貌。譬如：

```
\PUXsetdefaultcface\PUXFmk
```

使得字貌 \PUXFmk（中楷）取代 default 成爲預設的中文字貌。

匹配英文 PostScript 字貌

底下，我們示範如何匹配英文 PostScript 字貌，也介紹 \PUXfacematch 的第二種格式。此處我們假定你熟悉 LaTeX 的 NFSS 與 PSNFSS，若非如此的話，請先參考 The LaTeX Companion 書中的說明 [2]。

如果想改用 PostScript Times-Roman 來取代 LaTeX 預設的 Computer Modern 英文字貌，你可以採用 times.sty 套件（置於\texmf\tex\latex\psnfss 目錄）。其主要的內容如下：

```
\renewcommand{\sfdefault}{phv}
\renewcommand{\rmdefault}{ptm}
\renewcommand{\ttdefault}{pcr}
```

我們得知：正體字、斜體字、與粗體字是靠 ptm 所產生，接下來參考 ot1ptm.fd 字型定義檔的內容，它也是置於 \texmf\tex\latex\psnfss 目錄中。由其中的三行定義：

```
\DeclareFontShape{OT1}{ptm}{m}{n}{<-> ptmr7t}{}
\DeclareFontShape{OT1}{ptm}{m}{it}{<-> ptmri7t}{}
\DeclareFontShape{OT1}{ptm}{b}{n}{<-> ptmb7t}{}
```

可知上述三種字體分別採用 ptmr7t、ptmri7t、與 ptmb7t 字貌。加入以下四行就可以使得細明匹配正體英文、中楷匹配斜體英文、以及中黑匹配粗體英文：

```
\usepackage{times}
\PUXfacematch ptmr7t \PUXFtm
\PUXfacematch ptmri7t \PUXFmk
\PUXfacematch ptmb7t \PUXFmb
```

爲了簡化上述顯然有點複雜的步驟，pTEX 4 新增底下第二種格式的 \PUXfacematch 指令：

\PUXfacematch*cface_id*

此指令把目前英文字型的字貌與中文字貌 *cface_id* 匹配成對，並依據此新匹配規則變更目前的中文字型。

例 **5–5** 我們可以使用第二型的 \PUXfacematch 指令來設定上述 PostScript 字貌的匹配，而不需要去查英文字貌的名稱。譬如：在 LaTeX 文件的前端，我們可以做以下的匹配設定：

```
\usepackage{times}
\rmfamily\upshape\mdseries\PUXfacematch\PUXFtm % 羅馬體 → 細明（一般文字）
\rmfamily\upshape\bfseries\PUXfacematch\PUXFbm % 粗羅馬體 → 粗明（標題文字）
\rmfamily\itshape\mdseries\PUXfacematch\PUXFmk % 斜體 → 中楷
\rmfamily\itshape\bfseries\PUXfacematch\PUXFbk % 粗斜體 → 粗楷
\ttfamily\upshape\mdseries\PUXfacematch\PUXFtm % 定寬字 → 細明
\sffamily\upshape\mdseries\PUXfacematch\PUXFmb % 黑體字 → 中黑
\sffamily\upshape\bfseries\PUXfacematch\PUXFbb % 粗黑體字 → 粗黑
% 其他英文字貌對應至預設的中文字貌
```

（上面 LaTeX 字型選擇指令的意義，請參考 [3, p.66]。）

結束這一小節之前，我們用下面的例子來說明 \PUXfacematch 指令的區域性效果：

例 **5–6** 以下的輸入：

```
{\rm 細明☆ {\rm 還是細明☆ \PUXfacematch\PUXFmk 變成中楷 } ☆又回到細明 }
```

產生的結果為：

細明☆還是細明☆變成中楷☆又回到細明

因為這種區域性效果，所以上一個例中，

```
\rmfamily\upshape\mdseries\PUXfacematch\PUXFtm % 羅馬體用細明（一般文字）
```

不能用底下的方式取代：

```
\textrm{\PUXfacematch\PUXFtm} % 匹配只在 {...} 中有效，而沒有全域性的效果
{\rmfamily\upshape\mdseries\PUXfacematch\PUXFtm} % 理由同上
```

5.2.3 改變目前使用的中文字貌

在 \PUXcfacedef 指令中所定義的中文字貌指令（如 \PUXFmk）可以用來改變目前群組內的中文字貌（中文字型的大小則依目前的英文字型而定）。字貌指令是一種局部性指令，它的效力涵蓋所處的群組與其內部群組，但不包括外部群組。在其效力範圍內，中文字貌指令會取消中/英字貌的匹配設定。

例 **5–7** 以下的輸入方式：

```
細明體、{\PUXFmk 中楷體}、{\PUXFmb 中黑體}
```

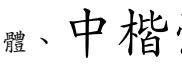
產生的結果為：

細明體、中楷體、**中黑體**

例 **5–8** 底下的兩種輸入方式：

```
正常、{\small\PUXFmk 中楷體}、{\huge\PUXFmk 中楷體}
正常、{\PUXFmk\small 中楷體}、{\PUXFmk\huge 中楷體}
```

會產生相同的結果：

正常、中楷體、 

例 **5–9** 底下的輸入方式：

```
{\PUXFtm 正常、{\em 強調字}、{\bf 粗體字}}
```

產生的結果為：

正常、強調字、粗體字

（都是細明體）而不是：

正常、*強調字*、**粗體字**

因為字貌指令 \PUXFtm 使得群組內的字貌匹配失效。如果想獲得上一行的效果，你可以採用底下的輸入方式：

```
{\PUXFtm 正常、{\PUXFmk 強調字}、{\PUXFmb 粗體字}}
```

5.2.4 定義中文字型

在 TEX 中，我們用 \font 指令來定義一個英文字型。譬如：

```
\font\tenrm cmr10
```

這一行指令定義 \tenrm 爲 cmr10 的字型。大部份的 TEX 字型名稱是由字貌名稱與字型大小所組成。以 cmr10 爲例，cmr 爲其字貌名稱，而 10 表示其字型大小爲 10pt。PUTEX 即根據此觀察來擴充 \font 指令的功能，使之能用於定義中文的字型。其用法如下：

\font*cfontid* CFONT*fn*

其中，*cfontid* 是使用者自訂的中文字型指令名稱；CFONT*fn* 則是由三個部份所組成：

CFONT	這個開頭英文字告知 PUTEX：此處將定義中文字型，而非 TEX 字型。
f	是一個已定義之中文字貌的英文名稱。
n	是一個用來宣示字型大小的正整數。

底下是定義中文字型的一些範例（此處採用表格 5 的中文字貌）：

```
\font\a CFONTtm10               % \a 是 10pt 的細明體
\font\b CFONTtm24               % \b 是 24pt 的細明體
\font\c CFONTmk18               % \c 是 18pt 的中楷體
\font\d CFONTmfs12 at 14pt      % \d 是 14pt 的中仿宋體
\font\e CFONTsb12 scaled 2000   % \e 是 24pt 的特黑體
```

如果用 LATEX 的 \newfont 指令來定義字型的話，以上的例子可改寫成：

```
\newfont{\a}{CFONTtm10}               % \a 是 10pt 的細明體
\newfont{\b}{CFONTtm24}               % \b 是 24pt 的細明體
\newfont{\c}{CFONTmk18}               % \c 是 18pt 的中楷體
\newfont{\d}{CFONTmfs12 at 14pt}      % \d 是 14pt 的中仿宋體
\newfont{\e}{CFONTsb12 scaled 2000}   % \e 是 24pt 的特黑體
```

5.2.5 設定中英文字型的匹配

PUTEX 的字貌匹配技術雖然簡化了中文字型的設定與使用，但是完全不考慮字型大小的因素，有時也會帶來一些困擾。PUTEX 因而提供 \PUXfontmatch 指令讓使用者可以做更精確細微的字型匹配。\PUXfontmatch 指令也有兩種形式，其語法分別如下：

1. \PUXfontmatch*cfaceid*

2. \PUXfontmatch *efont* *cfont*

第一種形式是用來匹配目前使用之英文字型。匹配的中文字型將以 *cfaceid* 爲其字貌，並以英文字型的大小爲其大小。第二種形式中的 *efont* 是一個英文字型的名稱、 *cfont* 是一個中文字型的名稱。它的作用是讓出現於 *efont* 英文字型環境內的中文採用 *cfont* 中文字型。

例 **5–10** 由於 \PUXfontmatch 是一個區域性指令，因此以下的輸入：

```
{\em 強調☆ {\em\PUXfontmatch\PUXFtm 強調}☆強調}
```

所得到的結果為：

強調☆強調☆*強調*

而非

強調☆強調☆強調

例 **5–11** 假定你做如下的字型匹配設定：

```
\font\A cmr10
\font\B CFONTmk10
\PUXfontmatch \A \B
```

這會使得

```
{\A cmr10 font 和 中楷體 10pt 字型}
```

產生底下的排版結果：

cmr10 font 和中楷體 10pt 字型

其中的中英文字皆為 10pt 大小。

例 **5–12** 你可以匹配大小不同的中英字型。譬如：

```
\font\A cmr10
\font\B CFONTmk20
\PUXfontmatch \A \B
```

將使得

```
{\A cmr10 font 和 中楷體 20pt 字型}
```

產生以下的結果：

cmr10 font 和中楷體 20pt 字型

其中的英文字為 10pt 大小，而中文字為 20pt 大小。

5.3 調整中文字貌深度

基線（baseline）是一條用來排列文字的虛擬水平線。字型深度（depth）決定基線穿越文字的位置。由於 TrueType 中文字型的深度通常爲字型大小的 0.2，而 TEX 字型的深度則爲字型大小的 0.25，兩者的差距有時會造成中英文並置時，高矮不協調。因此 PUTEX 提供參數 \puxgCfaceDepth 讓你調整所有中文字的深度；提供指令 \PUXcfacedepth 讓你調整特定中文字貌的文字深度。調整深度時，你應該先設定 \puxgCfaceDepth的值，然後才用 \PUXcfacedepth 設定個別中文字貌的文字深度。

由於 PUTEX 將深度視爲字貌的屬性，因此你在文件前端設定好字貌深度後，請勿在文件中加以改變，否則會產生不正確的排版結果。

5.3.1 全域性地調整中文字深度

PUTEX 的參數 \puxgCfaceDepth 是用來調整中英文並置對齊的高度。其格式如下：

\puxgCfaceDepth[=]n

整數值 n 用來設定深度。n 值愈大，則中文字位置就愈低，反之則愈高。假定字型大小爲 s，深度的計算公式如下：

$$\text{depth} = s \times n/1000$$

\puxgCfaceDepth 參數的預設值是 200（即採用 TrueType 的深度值 0.2）。此設定尙能獲得良好的排版結果。如果你不滿意此預設值，你可以改變它。一般而言，範圍應在 150 至 200 之間。

5.3.2 調整中文字貌深度

PUTEX 的 \PUXcfacedepth 指令可用來調整個別中文字貌的文字深度。其語法如下：

\PUXcfacedepth*cfaceid*=n

其中的 *cfaceid* 是一個中文字貌指令。 \PUXcfacedepth 是一個全域性的指令。設定中文字貌 *cfaceid* 的深度後，其效果將維持至重新設定 *cfaceid* 的深度爲止。

例 **5–13** 華康隸書體中文與 TEX cmr 字型並置時，會顯得過高。你可以用以下的指令：

```
\PUXcfacedepth\PUXFmli=250
```

將隸書體的深度調整成 0.25，使其位置往下移。

5.4 調整文字的間距

pUTEX 提供一些參數讓你調整中文的字間距以及中文與英文的間距。調整的對象可以是所有的中文字貌、單一的中文字貌、或單一的中文字型。然而，你應該按照

一、 調整所有文字的間距

二、 調整單一中文字貌的文字間距

三、 調整單一中文字型的文字間距

這個順序下達指令，否則會造成調整無效。比方說：若你先調整某一字型的文字間距，然後再調整該字型所屬字貌的文字間距，則後者的設定將取代前者的設定，而使得前者的設定無效。

所有調整文字間距的指令都是全域性，設定之後，效果將維持至重新變更其值爲止。爲了保持文字間距的協調與統一，你最好只在文件前端設定間距來調整文件整體的緊密程度。若要局部調整，你應該使用 `\hspace`、`\␣`、或 `$\,$` 之類的 LaTeX 指令。

5.4.1 調整所有中文字的間距

參數 `\puxgCspace` 是用來調整所有中文字的間距。它的語法如下：

`\puxgCspace`[=]w [`plus` m `minus` n]

其中的 w, m, 和 n 是三個整數值，分別稱爲空白間距的自然寬度（natural width）、加量（stretch）、與減量（shrink）。w 值愈大，間距就愈寬，反之則愈窄；m 是空白間距可延伸的建議最大值（因爲可能會超過此值，所以不稱爲最大值）；n 是空白間距可縮短的最大值。想進一步探究這些術語與概念的讀者，請參考 [4, p.69]。

假定 s 是目前中文字型的大小，則該字型的空白寬度計算公式如下：

$$\begin{aligned} \text{width} &= s \times w/1000 \\ \text{stretch} &= s \times m/1000 \\ \text{shrink} &= s \times n/1000 \end{aligned}$$

爲了與舊版的 pUTEX 相容，若不設定加量與減量，計算公式則變成：

$$\begin{aligned} \text{width} &= s \times w/1000 \\ \text{shrink} &= s \times |w|/3000 \\ \text{stretch} &= \begin{cases} 250s/2000 & \text{if } w < 250 \\ s \times w/2000 & \text{if } w \geq 250 \end{cases} \end{aligned}$$

`\puxgCspace` 的預設值是 50（不設定加量與減量）。

例 **5–14** 若你想把空白寬度設成約為字型寬的 1/5，可以宣告 \puxgCspace=200（因為 200/1000 = 1/5）。若你希望僅以中文字型本身的周圍空白來分隔文字，可以宣告 \puxgCspace=0。

例 **5–15** 假定 s 是中文字型的大小。底下設定的效果如註解中的說明：

```
\puxgCspace=50 plus 0 minus 10   % 空白寬度設在 0.04s 到 0.05s 之間
\puxgCspace=50 plus 10 minus 0   % 空白寬度設為最少 0.05s
\puxgCspace=50 plus 0 minus 0    % 空白寬度固定為 0.05s
```

我們不建議你設定固定的空白寬度（如上例的第三項），因爲這會造成 TEX 程式不容易找到適當的斷行點，使得許多文字行不是太長就是太短。以下是筆者建議設定中文字間距的三個步驟：

1. 先把 w 值設定成最適當的中文字間距寬度。
2. 選擇 n 的值使得 $w - n$ 是能容許的最小中文字間距。
3. 若想讓 TEX 在斷行時把多餘空白主要分配給中文間距，則把 m 值設大一點，否則，設小一點。

然而在大部分的情況下，使用舊式的設定方式（省略加量與減量）就可以得到不錯的效果。

5.4.2 調整中文字貌的文字間距

\PUXcfacecspace 指令用來調整個別中文字貌中文間距。其語法如下：

\PUXcfacecspace*cfaceid*[=]*w* [plus *m* minus *n*]

其中的 *cfaceid* 是一個中文字貌指令，前餘的參數 w, m, 和 n 如前一小節所述。

例 **5–16** 由於標楷體的字體比較小，使得標楷體的文字間距看起來寬了些，因而你可以用指令：

```
\PUXcfacecspace\PUXFmk=-50
```

將空白寬度設成約為字型大小的 −0.05 倍以縮短文字間距。

5.4.3 調整中文字型的文字間距

\PUXcfontcspace 指令用來調整個別中文字型的字間距。它有以下兩種型式：

\PUXcfontcspace*cfontid*[=]*w* [plus *m* minus *n*]
\PUXcfacecspace\font[=]*w* [plus *m* minus *n*]

第一式調整中文字型 *cfontid* 的文字間距；第二式調整目前中文字型的文字間距（此處，\font 指令被解讀成目前中文字型，而不是 TEX \font 指令的原意）。

不同於字貌，字型的大小是固定的，所以設定字型的文字間距時，\PUXcfontcspace 指令中的參數 w, m, 和 n 必須加上長度單位。下表列出常用的長度單位：

pt	point (1 pt = 1/72.27 in)
in	inch (1 in = 72.27 pt)
cm	公分 (2.54 cm = 1 in)
mm	公厘 (10 mm = 1 cm)
ex	目前英文字型字母 x 的高度（約等於字型高度的一半）
em	目前英文字型字母 M 的寬度（約等於字型寬度）

（更完整的表列請參見 [4, p.56]。）

例 **5–17** 以下的設定：

```
\font\A CFONTmk12
\PUXcfontcspace\A=0.72pt plus 20pt minus 6pt
```

將 12pt 中楷體字型 \A 的中文字間距的自然寬度設為 0.72 pt、加量為 20 pt、和減量為 6 pt。

5.4.4 調整中文與英文的字間距

PUTEX 的參數 \puxgCEspace 是用來調整所有中文字與英文字的間距。它的語法如下：

\puxgCEspace[=]*w* [plus *m* minus *n*]

請參考 5.4.1 節關於 w, m, 和 n 的說明。

假定 s 是目前中文字型的大小，則該字型的中英空白寬度計算公式如下：

$$\begin{aligned} \text{width} &= s \times w/1000 \\ \text{stretch} &= s \times m/1000 \\ \text{shrink} &= s \times n/1000 \end{aligned}$$

爲了與舊版的 pUTEX 相容，若不設定加量與減量，計算公式則變成：

$$\begin{aligned} \text{space} &= s \times w/1000 \\ \text{shrink} &= s \times |w|/3000 \\ \text{stretch} &= s \times |w|/2000 \end{aligned}$$

\puxgCEspace 的預設值是 150（不設定加量與減量）。

例 **5–18** 若你想把空白寬度變為中文字型寬度的 1/5，可以宣告 \puxgCespace=200。若你希望僅以中文字型本身的周圍空白來分隔中英文字，可以宣告 \puxgCespace=0。

5.4.5 調整中文字貌的中英字間距

指令 \PUXcfacecespace 可用來調整個別中文字貌的中英字間距。其語法如下：

\PUXcfacecespace*cfaceid*[=]*n* [plus *m* minus *n*]

其中的 *cfaceid* 是一個中文字貌指令。

5.4.6 調整中文字型的中英字間距

指令 \PUXcfontcespace 可用來調整個別中文字型的中英字間距。它有以下兩種型式：

\PUXcfontcespace*cfontid*[=]*w* [plus *m* minus *n*]
\PUXcfontcespace\font[=]*w* [plus *m* minus *n*]

第一式調整中文字型 *cfontid* 的中英字間距；第二式調整目前中文字型的中英字間距（此處，\font 指令被解讀成目前中文字型，而不是 TEX \font 指令的原意）。情參考 5.4.3 節中關於參數 w, m, 和 n 的說明。

例 **5–19** 以下的設定：

```
\font\A CFONTmk12
\PUXcfontcespace\A=0.72pt plus 20pt minus 6pt
```

將 12 pt 中楷體字型 \A 的中英字間距的自然寬度設為 0.72 pt、加量為 20 pt、和減量為 6 pt。

5.4.7 插入空白

若 pUTEX 無法產生正確的空白，或你想自行插入某種空白間距，這時可以用以下的指令：

\PUXspace 英文空白（如同英文字之間的空白 ␣）

\PUXexspace	英文空白（如同英文字之間的空白指令 \␣）
\PUXcspace	中文空白
\PUXcespace	中英空白
\PUXchar"A140	中文空白字元（參見 5.6 節）

5.5 中式數字

pUTEX 提供一些指令把整數值轉換成中文數字串。譬如：\PUXcnumber 123 把 123 轉換成字串「一百二十三」。表格 6 列出這些指令的名稱。其中，俗體與小寫的差異在於：俗體使用「廿」和「卅」來取代「二十」和「三十」。正式與大寫的差異在於：正式不省略十進位數字（如表格 6 中數值 12 所示）。阿拉伯則是採用阿拉伯數字的 Big5 全形字。\PUXcjknumber 則是把每個阿拉伯數字一一地換成中文數字。它的完整形式如下：

\PUXcjknumber n [offset[=]l]

若省略 offset 部分，則 l 的預設值是 0。這個指令假定特殊字元表自第 l 開始存放了 0, 1, ..., 9 阿拉伯數字的本地對應字元（請參閱 6.3 節關於特殊字元表的說明）。以中文爲例，0 的中文對應字元是「〇」、1 對應至「一」、　　、等等。然後，\PUXcjknumber 把每一位數轉換成本地的對應字元。

指令	格式	12	123	1230531
\PUXcnumber	小寫	十二	一百二十三	一百二十三万〇五百三十一
\PUXscnumber	俗體	十二	一百廿三	一百廿三万〇五百卅一
\PUXucnumber	大寫	拾貳	壹佰貳拾參	壹佰貳拾參萬零伍佰參拾壹
\PUXfcnumber	正式	壹拾貳	壹佰貳拾參	壹佰貳拾參萬零伍佰參拾壹
\PUXacnumber	阿拉伯	１２	１２３	１２３０５３１
\PUXcjknumber	中文	一二	一二三	一二三〇五三一

表格 6: 中式數字指令、格式、與範例

例 **5–20** 以下顯示各式中文數字的產生方法：

輸入	結果
\PUXcnumber 123	一百二十三
\PUXscnumber 123	一百廿三
\PUXucnumber 123	壹佰貳拾參
\PUXfcnumber 123	壹佰貳拾參
\PUXacnumber 123	１２３

PUTEX 4 另新增一個轉換數字的指令，可用來把數字 1, 2, 3, ⋯ 轉成「甲、乙、丙、　　」或「子、丑、寅、　　」。此指令的格式如下：

\PUXnameseq n min[=]a max[=]b offset[=]l

其作用是列印出特殊字元表中第 $l+n-a$ 的字元，其中， $n, a, b,$ 和 l 的值必須介於 0 到 255 之間，而且 $a \le n \le b$，否則會產生錯誤的結果。

例 5–21　由於特殊字元表自位置 54 開始存放甲、乙、丙、　　等 10 個代表天干的字元，因此：

```
\PUXnameseq 1 min=1 max=10 offset=54 得到字元「甲」
\PUXnameseq 2 min=1 max=10 offset=54 得到字元「乙」
      ⋮
\PUXnameseq 10 min=1 max=10 offset=54 得到字元「癸」
```

例 5–22　由於特殊字元表自位置 64 開始存放子、丑、寅、　　等 12 個代表地支的字元，因此：

```
\PUXnameseq 1 min=1 max=12 offset=64 得到字元「子」
\PUXnameseq 2 min=1 max=12 offset=64 得到字元「丑」
      ⋮
\PUXnameseq 12 min=1 max=12 offset=64 得到字元「亥」
```

5.6 字元碼指令

PUTEX 仿照 TEX 基本指令 \char 的功能 [4, p.43]，提供 \PUXchar 指令讓使用者能夠用字元碼的方式輸入中文字元。你可以選用以下的方式輸入字元碼：

```
\PUXchar"hhhh        % hhhh 是十六進位的字元碼
\PUXchar'oooooo      % oooooo 是八進位的字元碼
\PUXchar`\c          % c 是字元
```

不論那一種方式，輸入的字元碼都必須大於 255，否則會產生錯誤訊息。此外，由 \PUXchar 指令輸入的字元，其類別碼都是 12 (other)（參見 5.7 節）。

例 5–23　\PUXchar 指令可用來輸入不易從鍵盤鍵入的字元，譬如：

```
\PUXchar"A1F0   產生  ♀
\PUXchar"A1F1   產生  ⚷
\PUXchar`\ㄅ    產生  ㄅ
\PUXchar"A1C0   產生  ㊣
\PUXchar"C740   產生
\PUXchar"C741   產生
```

註：有些中文字體（如新細明體和標楷體）並未包含所有的 Big5 符號字元。

例 **5–24** 中文空白字元的十六進位 Big5 碼為 A140，所以你可以用 \PUXchar"A140 的方式插入一個中文空白字元。譬如輸入：姓\PUXchar"A140名，所得的結果為「姓　名」。

你也可以用 \PUXchar 指令來輸入外字集的中文字元。我們以底下的範例來說明其步驟。

例 **5–25** 菜的「 」字並不在標準的 Big5 字碼集之內。不過，華康外字集提供此字，且其字碼定為 8E4D。假定你打算使用華康中黑體外字集中的「 」字，你可以先做類似以下的定義：

```
\PUXcfacedef\PUXFxmb=xmb 外中黑
\newcommand{\九}{{\PUXFxmb\PUXchar"8E4D }}
```

並在 cfonts.map 檔中加上一行：

```
 外中黑        華康中黑體外字集
```

此後，你只要輸入「\九␣菜」就可以產生「 菜」一詞。

PUTEX 也仿照 TEX 基本指令 \chardef 的功能 [4, p. 44]，提供 \PUXchardef 指令。其格式如下：

```
\PUXchardef\cmd="hhhh
```

其作用是將指令 *cmd* 定義成字元碼爲 hhhh 的中文字元。

例 **5–26** 底下的指令定義：

```
\PUXchardef\正="A1C0
\PUXchardef\空="A140
```

讓你可以使用指令「\正」來產生㊣、指令「\空」來插入一個中文空白字元「 」。

5.7 設定中文字元的類別碼

TEX 讀入字元時，會根據字元的類別碼（catcode）[4, p.37] 來執行預定的動作。通常英文字母的類別碼是 11（代表 letter）、阿拉伯數字和一般標點符號的類別碼是 12（代表 other）、其他特殊的符號，如 \ { } $ ··· 等等則被分配特定的類別碼。letter 和 other 類的字元通常不引發特殊處理，而直接進入排版階段。兩者不同處在於：letter 字元可用於指令的命名，而 other 字元則不行。PUTEX 把正常中文字元的類別預設成 letter，其他符號字元則預設成 other。

TEX 的基本指令 \catcode 可用來區域性地改變某字元的類別碼，藉此改變該字元的處理方式。PUTEX 提供對應的指令 \PUXcatcode：

`\PUXcatcode` n[=]c

把碼值爲 n 的中文字元的類別碼（區域性地）設成 c。針對設定一連串字元的類別碼，pUTEX 提供以下的指令：

`\PUXrangecatcode` m [`to`] n [`=`] c

把字碼從 m 到 n 的字元的類別碼都設成 c。

例 **5–27** 中文句點（。）的十六進位碼值為 A143。你可以用底下前四種方式來設定其類別碼為 letter：

```
\PUXcatcode‘\。=11       % 用字元碼值（註：‘ 是位於鍵盤左上角的反引號）
\PUXcatcode"A143=11      % 用十六進位碼值
\PUXcatcode’120503=11    % 用八進位碼值
\PUXcatcode 41283=11     % 用十進位碼值
\PUXcatcode 。=11        % 錯誤！
\PUXcatcode’\。=11       % 錯誤！不能用單引號，必須用反引號
```

當字元的類別碼被設成 13 時，此字元變成指令，所以又稱爲**主動字元**（active character）。我們以下面的例子來示範利用主動字元來變更中文標點符號的排版方式。

例 **5–28** 圖 1 顯示兩種中文標點的排版方式：(a) 標點置中並佔一個全形字元的寬度；(b) 標點往左移若干寬度，句點和逗點往下移。如果你檢視本件的原始檔，可以看到 (a) 和 (b) 的文字輸入完全相同，惟一差別在於：(b) 增加了以下的設定：

```
\PUXcatcode‘\，=\active
\def，{\kern-0.25em\raise-.75ex\hbox{\PUXchar‘\，}\kern-0.2em}
\PUXcatcode‘\。=\active \def。{\kern-0.25em\raise-.75ex\hbox{\PUXchar‘\。}}
\PUXcatcode‘\；=\active \def；{\kern-0.2em\PUXchar‘\；}
\PUXcatcode‘\：=\active \def：{\kern-0.2em\PUXchar‘\：}
\PUXcatcode‘\「=\active \def「{\kern-0.5em\PUXchar‘\「}
\PUXcatcode‘\」=\active \def」{\kern-0.25em\PUXchar‘\」}
```

其中，我們用 `\PUXcatcode` 把中文標點，。；「 」設成主動字元，然後再定義成指令來改變它們的排版位置。舉例來說：

```
\def，{\kern-0.25em\raise-.75ex\hbox{\PUXchar‘\，}\kern-0.2em}
```

使得逗號往左移 0.25em、往下移 0.75ex、然後再讓後面跟著的字元往左移 0.2em。由於逗號已經是主動字元，所以在定義中必須改用「`\PUXchar‘\，`」，否則會造成自我遞迴定義的錯誤。

例 **5–29** 我們可以利用主動字元來轉換字元。舉例來說，以下的設定：

我說道：「爸爸，你走吧。」他往車外看了看，說：「我買幾個橘子去。你就在此地，不要走動。」我看那邊月臺的柵欄外有幾個賣東西的等著顧客。走到那邊月臺，須穿過鐵道，須跳下去又爬上去。父親是一個胖子，走過去自然要費事些。我本來要去的，他不肯，只好讓他去。我看見他戴著黑布小帽，穿著黑布大馬褂，深青布棉袍，蹣跚地走到鐵道邊，慢慢探身下去，尚不大難。可是他穿過鐵道，要爬上那邊月臺，就不容易了。他用兩手攀著上面，兩腳再向上縮；他肥胖的身子向左微傾，顯出努力的樣子。這時我看見他的背影，我的淚很快地流下來了。我趕緊拭幹了淚，怕他看見，也怕別人看見。我再向外看時，他已抱了朱紅的橘子往回走了。過鐵道時，他先將桔子散放在地上，自己慢慢爬下，再抱起桔子走。到這邊時，我趕緊去攙他。他和我走到車上，將橘子一股腦兒放在我的皮大衣上。於是撲撲衣上的泥土，心裡很輕鬆似的，過一會說：「我走了，到那邊來信！」我望著他走出去。他走了幾步，回過頭看見我，說：「進去吧，裏邊沒人。」等他的背影混入來來往往的人裡，再找不著了，我便進來坐下，我的眼淚又來了。

節錄自朱自清 — 背影 —

(a) 現代式的標點

我說道:「爸爸,你走吧。」他往車外看了看,說:「我買幾個橘子去。你就在此地,不要走動。」我看那邊月臺的柵欄外有幾個賣東西的等著顧客。走到那邊月臺,須穿過鐵道,須跳下去又爬上去。父親是一個胖子,走過去自然要費事些。我本來要去的,他不肯,只好讓他去。我看見他戴著黑布小帽,穿著黑布大馬褂,深青布棉袍,蹣跚地走到鐵道邊,慢慢探身下去,尚不大難。可是他穿過鐵道,要爬上那邊月臺,就不容易了。他用兩手攀著上面,兩腳再向上縮; 他肥胖的身子向左微傾,顯出努力的樣子。這時我看見他的背影,我的淚很快地流下來了。我趕緊拭幹了淚,怕他看見,也怕別人看見。我再向外看時,他已抱了朱紅的橘子往回走了。過鐵道時,他先將桔子散放在地上,自己慢慢爬下,再抱起桔子走。到這邊時,我趕緊去攙他。他和我走到車上,將橘子一股腦兒放在我的皮大衣上。於是撲撲衣上的泥土,心裡很輕鬆似的,過一會說:「我走了,到那邊來信!」我望著他走出去。他走了幾步,回過頭看見我,說:「進去吧,裏邊沒人。」等他的背影混入來來往往的人裡,再找不著了,我便進來坐下,我的眼淚又來了。

節錄自朱自清 — 背影 —

(b) 復古式的標點

圖 1: 兩種標點方式

```
\PUXcatcode`\（=\active \def「{『}
\PUXcatcode`\）=\active \def」{』}
```

會使得輸入：（TEX）產生結果：『TEX』，因為（和）分別被『和』給置換掉了。

例 **5–30** 在 verb 與 verbatim 環境中，PUTEX 仍會在中英文字元之間加入中英空白。有時，這會造成字元間距過大，譬如：

```
\begin{verbatim}
    \newcommand{\manual}{手冊}
\end{verbatim}
```

產生的結果為：

```
\newcommand{\manual}{ 手冊 }
```

「手冊」一詞與左右花括號的間距顯然過寬。編排本文件時，筆者為了解決這個問題，特別挑選不常用到的中文符號⊥，在文件前端加入以下的巨集定義：

```
\PUXcatcode`\⊥=\active \def⊥{\kern0pt}
```

然後在「手冊」一詞的前後端加上⊥，如下所示：

```
\begin{verbatim}
    \newcommand{\manual}{⊥手冊⊥}
\end{verbatim}
```

這樣就可以得到如下所示比較美觀的結果：

```
\newcommand{\manual}{手冊}
```

這個技巧是利用到以下兩個事實：

- verbatim 環境只把所有英文字元的類別改成 other，而沒有更動中文字元的類別。
- ⊥ 被定義成 \kern0pt 使得 PUTEX 不會加入中英空白（參閱第 12 頁）。

例 **5–31** 製作 Big5TEX/LATEX 的格式檔（format）時，筆者用以下的指令，把 Big5 字元集裏的所有符號字元的類別碼都設成 12 (other)：

```
\PUXrangecatcode"A140 to "A1FE=12
\PUXrangecatcode"A240 to "A258=12
\PUXrangecatcode"A262 to "A2AE=12
\PUXrangecatcode"C6A1 to "C6E6=12
\PUXrangecatcode"C874 to "C8FE=12
\PUXrangecatcode"F9DD to "F9FE=12
```

除了上述的類別碼指令外，pTEX 提供另一個區域性參數來控制中文字元類別碼：

\puxCJKcharOther[=]n

若 $n = 0$（預設值），則中文字元的類別碼依照前述類別碼指令的設定；若 $n \neq 0$，則所有中文字元的類別碼一律爲 12 (other)。

5.8 中文指令名稱

由於正常中文字元預設的類別爲 letter，所以你可以用中文來命名指令。譬如：在 TEX 中，若定義如下的中文指令：

\def\校名{靜宜大學}

指令「\校名」即代表「靜宜大學」一詞。在 LATEX 中，可用如下的方法來定義同樣的中文指令：

\newcommand{\校名}{靜宜大學}

你可以將參數 \puxCJKcharOther 的值設成 1 來關閉中文指令的功能（不建議如此做）。

5.9 中文直排

中式直排是利用將中文字逆時針旋轉 90 度的技巧來達成。當然，你也必須在 LATEX 中修訂紙張的尺寸定義、列印時選擇橫式（landscape）列印方式，如此才能列印出正確的中文直排效果。你可以在定義字貌時，用設定屬性 s =r 的方式（參見 5.2.1 節），來定義旋轉字體。不過，pTEX 提供的參數 \puxgRotateCtext 可以讓你更簡便地旋轉字貌來製作中文直排。它的設定方式如下：

\puxgRotateCtext=n

若 n 爲 0 時，表示不自動旋轉字貌。這是 \puxgRotateCtext 預設的參數值。若 n 爲任何不等於 0 的整數值時，pTEX 會自動地將所有非旋轉字體的字貌改成旋轉字體的字貌，所有旋轉字體的字貌改成非旋轉字體的字貌。由於 \puxgRotateCtext 是一種旗標（flag）參數，如果你已經將 \puxgRotateCtext 的值設成一個非 0 的數值，你將無法再把它改回 0 值。

6 用於製作格式檔或 LATEX 套件的指令

這一節介紹的的指令主要是用於製作格式檔或 LATEX 套件，一般的使用者通常不會直接用到它們。如果你屬於一般的使用者，可以略過此節的內容。

6.1 設定文件字元集

pTEX 正蛻變成 CJK (Chinese-Janpenese-Korean) TEX 系統，爲了辨識文件所使用的字元集（character set），pTEX 4 新增底下的（區域性）指令：

\puxCharSet[=]*n*

n 是一個整數值，用來代表文件所使用的字元集。目前已定義的字元集如下：

n=0: UCS2 (two-byte Unicode)

n=1: Big5 (Taiwan and Hong Kong)

n=2: GBK (China and Singapore)

n=3: Shift-JIS (Japan)

n=4: KS_C_5601 (Korea)

雖然目前 pTEX 4 的格式檔只提供 Big5 和 GBK 的版本，cdi2dvi 也只支援這兩種字元集，不過，其他國家人士可以根據本手冊所述，製作出來格式檔，就可以用 cdi2pdf 來預覽和列印。當然，在筆者未完成英文版手冊之前，這位仁兄可得先學會閱讀繁體中文 :)

未來（pTEX 5?），我們希望能利用這個參數來提供多字元集文件的編排，譬如在 Big5 文件中加入 GBK 編碼或 UCS2 編碼的文件。

6.2 設定中文字元的型態碼

在中文排版的慣例上，句點「。」、逗點「，」等標點符號應該避免出現在一行的最前端，這些字元稱爲*避首字元*。左括弧「（」或「【」等則應該避免出現在一行的最後端，這些字元稱爲*避尾字元*。pTEX 支援這項避行首/行尾字元的排版功能。Big5TEX/LATEX 依據微軟公司的建議 [1, pp 241–242]，支援表格 7 所列出的避行首 / 行尾字元。此表格除了涵蓋大部分的中文標點符號外，也包含了若干英式標點符號（其中字碼爲二位數者）。

爲了支援其他字元集的避行首/行尾功能，pTEX 4 新增以下的指令：

\PUXtypecode *n*[=] 0 | 1 | 2

把碼值爲 *n* 的中文字元的型態碼（typecode）區域性地設成 0（正常）、1（避尾）、或 2（避首）。針對設定一連串字元的型態碼，pTEX 提供以下的指令：

\PUXrangetypecode *m* [to] *n* [=] 0 | 1 | 2

把字碼從 *m* 到 *n* 的字元的型態碼都設成 0, 1, 或 2。

例 **6–1** 如果你用以下的指令：

```
\PUXtypecode`\。=0
```

把句點的型態碼改為正常，則會取消它的避首功能（即允許句點出現在行首）。

21	!	A147	：	A156	—	A16E	》
29	)	A148	？	A157	｜	A170	ﹽ
2C	,	A149	！	A158	—	A172	〉
2E	.	A14A	：	A159	｜	A174	﹀
3A	:	A14B		A15A	＿	A176	」
3B	;	A14C	‥	A15B	｝	A178	﹂
3F	?	A14D	，	A15C	﹌	A17A	』
5D	]	A14E	、	A15E	）	A17C	﹄
7D	}	A14F	．	A160	﹀	A17E	）
A141	，	A150	·	A162	}	A1A2	｝
A142	、	A151	；	A164	﹀	A1A4	〕
A143	。	A152	：	A166	〕	A1A6	’
A144	·	A153	？	A168	﹀	A1A8	”
A145	•	A154	！	A16A	】	A1AA	″
A146	；	A155	｜	A16C	﹀	A1AC	‵

(a) 避行首的字元碼與符號

— 28	(	A165	〔	A173	﹏	A1A3	〔
5B	[	A167	﹏	A175	「	A1A5	‘
7B	{	A169	【	A177	﹁	A1A7	“
A15D	（	A16B	﹁	A179	『	A1A9	〝
A15F	﹏	A16D	《	A17B	﹁	A1AB	‶
A161	｛	A16F	ﹽ	A17D	（		
A163	﹏	A171	〈	A1A1	｛		

(b) 避行尾的字元碼與符號

表格 7: Big5TEX/LaTeX 的避行首 / 尾標點符號

區間	字元	個數	說明
0–9	〇一二三四五六七八九	10	CJK 數字
10–24	〇一二三四五六七八九十百千万億	15	中文小寫數字
25–39	零壹貳參肆伍陸柒捌玖拾佰仟萬億	15	中文大寫數字
40–49	０１２３４５６７８９	10	全形阿拉伯數字
50–53	負－廿卅	4	負號與特殊數字
54–63	甲乙丙丁戊己庚辛壬癸	10	天干
64–75	子丑寅卯辰巳午未申酉戌亥	12	地支
76–89	年時分秒日月火水木金土星期曜	14	日期與時間
90–93	春夏秋冬	4	四季

表格 8: 特殊字元表的預設內容

6.3 特殊字元表

爲了讓 pTEX 的一些指令（如產生中文數字的指令）能夠在不同的字元集下正常運作，pTEX 內部有一個可含 256 個 CJK 字元的特殊字元表（table of local names）。其中，前 128 個（0–127）保留給格式檔製作者來設定，而且必須按照規定填入適當的字元，才能讓某些 pTEX 指令產生預期的效果；後 128 個（128–255）則可讓一般使用者來運用。

底下的指令讓你存入一個 CJK 字元到特殊字元表中：

`\PUXlocalnames` n [=] c

其中，n, $0 \le n \le 255$, 是存入位置，c 是存入的 CJK 字元碼。譬如底下的指令在特殊字元表中第 54 格中存入 Big5 字元「甲」：

```
\PUXlocalnames54="A5D2  % the first char of Heavenly Stems（甲）
```

你可以用 `\the\PUXlocalnames` n 的方式讀出第 n 格的字元，譬如：

`\the\PUXlocalnames54` 產生字元「甲」

表格 8 列出所有 pTEX 格式檔必須填入的字元與其位置。其中，位置 0–9 的數字字元是供指令 `\PUXcjknumber` 來轉換各國當地的數字寫法。位置 10–53 的字元則是專門用來產生中文格式的數字（參閱 5.5 節）。「日月火水木金土」這樣的安排是按照日文星期的順序。

例 **6–2** 一般的使者用可以使用特殊字元表 128–255 的位置。譬如：若你加入下面的字元：

```
\PUXlocalnames128=‘\    \PUXlocalnames129=‘\    \PUXlocalnames130=‘\
\PUXlocalnames131=‘\    \PUXlocalnames132=‘\    \PUXlocalnames133=‘\
\PUXlocalnames134=‘\    \PUXlocalnames135=‘\    \PUXlocalnames136=‘\
\PUXlocalnames137=‘\
```

就可以用 5.5 節所介紹的 \PUXnameseq 指令把數值 1, 2, ..., 10 對應到 ，，...， ：

\PUXnameseq n min=1 max=10 offset=128 % $1 \le n \le 10$

6.4 數字處理指令

爲了便利其他國家轉換數字格式，pUTEX 4 提供底下的指令：

\PUXsplitnumber n

執行之後，數字 n 的資訊會存入內部暫存器： \puxnumdigit, \puxsign, 和 \puxdigit。譬如，若輸入以下的指令：

```
\PUXsplitnumber 39765
number of digits = \the\puxnumdigits
sign = \the\puxsign          % 1: positive, -1: negative
digit 0 = \the\puxdigit0,    % the right-most digit
digit 1 = \the\puxdigit1,    digit 2 = \the\puxdigit2,
digit 3 = \the\puxdigit3,    digit 4 = \the\puxdigit4,
digit 5 = \the\puxdigit5,    digit 6 = \the\puxdigit6,
digit 7 = \the\puxdigit7,    digit 8 = \the\puxdigit8,
digit 9 = \the\puxdigit9
```

會得到以下的結果：

number of digits = 5
sign = 1
digit 0 = 5, digit 1 = 6, digit 2 = 7, digit 3 = 9, digit 4 = 3,
digit 5 = 0, digit 6 = 0, digit 7 = 0, digit 8 = 0, digit 9 = 0

7 其他 PUTEX 的基本指令

參數 \puxXspace 的用途是爲了讓 pUTEX 能夠正確地處理 LATEX 的 \verb 指令與 verbatim 環境中的空白字元。一般使用者可以不用理會它的存在。

你可以在 \end{document} 之前，加入 pUTEX 的 \PUXdumpfontinfo 指令，將文件中所使用的英文字型、中文字貌、中文字型、字型匹配、和字貌匹配等資訊輸出至 log 檔案

中。你除了使用這些資訊於偵錯目的外，也可利用其中 TEX 字型的資訊來設計字型字貌的匹配規則。底下是 \PUXdumpfontinfo 所輸出的部分結果：

```
Tex fonts
0: nullfont dsize= 0.0pt at 0.0pt matched CJK font=5001
1: cmex10 dsize= 10.0pt at 10.0pt matched CJK font=5001
2: line10 dsize= 10.0pt at 10.0pt matched CJK font=5001
...
Chinese faces
0: id=nullcface name=nullcjkface charset=0 weight=400 .....
1: id=default name=default charset=1 weight=400 style=0 w=1.0 h=1.0 d=0.2
2: id=tm name= 細明 charset=1 weight=400 style=0 w=1.0 h=1.0 d=0.2
3: id=mm name= 中明 charset=1 weight=400 style=0 w=1.0 h=1.0 d=0.2
...
Chinese fonts
5001:face= nullcface dsize= 0.0pt at 0.0pt
5002:face= tm dsize= 5.0pt at 5.0pt
5003:face= tm dsize= 7.0pt at 7.0pt
5004:face= tm dsize= 10.0pt at 10.0pt
...
English/Chinese font faces matching table
0: eface=cmr cface_id=tm cface_num=2
1: eface=cmtt cface_id=tm cface_num=2
2: eface=cmbx cface_id=mb cface_num=11
3: eface=cmti cface_id=mk cface_num=7
4: eface=cmbxti cface_id=mk cface_num=7
```

8 製作中英文索引

你可以利用 PUTEX 的 puidx 程式 來製作中英文索引。puidx 使用的中英文排序法則如下：

- 中文以筆畫爲序，英文則以字母爲序。
- 若中文字元與英文字元的字序相同時，則英文字元先於中文字元。
- 中英文合併排序。譬如： A < 一 < B < 丁 < C < 大 < D ⋯ 等等。

根據上述的法則，當中文詞首字元的筆畫序與英文字首的字母序相同時，兩者將置於同一個大項之中（如本手冊的索引所示）。比方說：以一劃中文字開頭的中文詞將與以字母 A（或 a）開頭的英文字置於第一大項之中、以兩劃中文字開頭的中文詞將與以字母 B（或 b）開頭的英文字置於第二大項之中、　　、等等依此類推。

puidx 脫胎於 LaTeX 的 makeindex 程式，除了底下三點的差異外，保留了其他所有 makeindex 的功能：

- puidx 不支援德文的排序，即指令選項 `-g` 無效。
- puidx 更改 headings_flag 的作用。當 headings_flag 爲正值時，索引中的每一大項將以英文字母與中文筆畫數做爲標題（如本手冊的索引所示）；當 headings_flag 的值爲 0 時，每一大項不具有標題；當 headings_flag 爲負值時，索引中的每一大項只以中文筆畫數做爲標題而不會出現英文字母（適用於製作純中文索引）。
- puidx 能用來製作中英文索引，而 makeindex 無法處理中文索引。

當你參考 makeindex 的使用說明時 [2, 3, 5]，只要記住這三點，就可以完全按照 makeindex 的方式來使用 puidx。

最後，我們介紹 putex 所具有的一項便利功能（同樣適用於 makeindex）。假定 `foo.tex` 是一個中文 LaTeX 文件檔。若你將索引的樣式定義存在同目錄中的 `foo.mst` 檔中，則執行以下指令時

```
puidx foo
```

putex 會自動地讀取 `foo.mst` 中的樣式定義。利用這個技巧，你就不需要如 [2] 所述執行以下比較複雜的指令：

```
puidx -s foo.mst foo
```

比方說：本手冊的檔名爲 `guide140.tex`。底下是 `guide40.mst` 的內容：

```
preamble "\\begin{theindex}\\small\n"
heading_prefix "{\\bigskip\\large\\bfseries "
heading_suffix "\\medskip}\\nopagebreak\n\n"
headings_flag 1
delim_0 "\\dotfill "
delim_1 "\\dotfill "
delim_2 "\\dotfill "
```

利用以上樣式設定產生出來的索引就如最後兩頁所示。

9 致謝

筆者除了感謝國科會的資助外，也要特別感謝靜宜大學資管系的謝易霖與劉皓朋兩位同學。他們撰寫了第一版的 cdi2dvi，讓 PuTeX 使用者能夠方便地利用 dvips。

參考文獻

[1] N. Kano. *Developing International Software for Windows 95 and Windows NT*. Microsoft Press. 1995. 2nd. Ed. Addison-Wesley. 1995.

[2] F. Mittelbach, M. Goossens. *The LaTeX Companion*. 2nd Ed. Addison-Wesley. 2004.

[3] H. Kopka and P. W. Daly. *A Guide to LaTeX2*. 4th Ed. Addison-Wesley. 2004.

[4] D. E. Knuth. *The TeXBook*. Addison-Wesley. 1986.

[5] L. Lamport. *LaTeX User's Guide and Reference Manual*. 2nd. Ed. Addison-Wesley. 1994.

[6] 蔡奇偉 . *PUTEX* 安裝說明 .
`http://www.cs.pu.edu.tw/~tsay/putex/install.html`.

A 邏輯 –實體字體對應檔

TrueType 中文字體種類繁多，各廠牌字型的命名方式也各不相同。爲了使 pUTEX 文件與其 CDI 輸出檔具有可攜性（portability），pUTEX 在內部採用邏輯字體名稱而非眞實的字體名稱， CDI 輸出檔也記錄邏輯字體名稱。

當預覽、列印、或轉換 CDI 檔時， cdi2dvi 必須先將邏輯字體轉換成眞實的字體，才能夠產生出正確的字型圖像。 cdi2dvi 利用 pUTEX 系統檔 `cfonts.map` 中所定義的邏輯 –實體字體對應關係來做兩者之間的轉換。cfonts.map 檔案的格式非常簡單，其規則如下：

- 行中字元 % 之後的文字均視爲註解文字。
- 非註解或空白的文字行稱爲定義行。定義行用來宣告邏輯 –實體字體間的對應關係。
- 定義行由兩欄文字所組成，第一欄爲邏輯字體的名稱，第二欄爲實體字體的名稱。兩欄之間必須以一個或一個以上的空白字元（或 Tab 字元）隔開。
- 邏輯字型名稱必須是字貌定義指令 `\PUXcfacedef` 中的中文邏輯字體名稱（參見 5.2.1 節）。
- 實體字體的名稱必須與 Windows「控制台」中「字型」視窗所顯示的中文字體名稱完全相同。
- 第一個定義行的實體字體是預設的實體字體。當某一邏輯字體無對應的實體字體時， CDI 驅動程式會自動將其對應至此預設的實體字體。
- 每一個邏輯字體只能定義一個實體字體。不過一個實體字體卻可能對應至多個邏輯字體。換句話說，邏輯字型與實體字體間的對應關係是一種「多對一」的函數。

以下是預設的 `cfonts.map` 內容（由於筆者採用文鼎字型，所以實體字體均爲文鼎字型的名稱）。

如果你使用別種廠牌字型，請修改其內容。否則將無法產生出正確的字型。

```
default     新細明體        % 預設的實體字型
細明        文鼎中明
中明        文鼎中明
粗明        文鼎粗明
特明        文鼎特明
細楷        文鼎細楷
中楷        標楷體
粗楷        文鼎粗楷
細黑        文鼎細黑
中黑        文鼎中黑
```

外中黑	文鼎中黑體外字集
粗黑	文鼎粗黑
特黑	文鼎特黑
細圓	文鼎細圓
中圓	文鼎中圓
粗圓	文鼎粗圓
中疊圓	文鼎疊圓體
空疊圓	文鼎空疊圓
細隸書	文鼎中隸
中隸書	文鼎中隸
粗隸書	文鼎粗隸
中行書	文鼎中行書
古印體	文鼎古印體
中仿宋	文鼎中仿
粗仿宋	文鼎粗仿
勘亭流	文鼎勘亭流
綜藝	文鼎新藝體
POP1	文鼎 POP-4

B 預設的中文字貌

底下是製作 Big5TEX 與 Big5LATEX 格式檔時，所定義的中文字貌與匹配規則：

```
\PUXcfacedef\PUXFbigfive=default default
\PUXsetdefaultcface\PUXFbigfive
\PUXcfacedef\PUXFtm=tm 細明
\PUXcfacedef\PUXFmm=mm 中明
\PUXcfacedef\PUXFbm=bm 粗明
\PUXcfacedef\PUXFsm=sm 特明
\PUXcfacedef\PUXFtk=tk 細楷
\PUXcfacedef\PUXFmk=mk 中楷
\PUXcfacedef\PUXFbk=bk 粗楷
\PUXcfacedef\PUXFsk=sk 特楷
\PUXcfacedef\PUXFtb=tb 細黑
\PUXcfacedef\PUXFmb=mb 中黑
\PUXcfacedef\PUXFbb=bb 粗黑
\PUXcfacedef\PUXFsb=sb 特黑
\PUXcfacedef\PUXFtr=tr 細圓
\PUXcfacedef\PUXFmr=mr 中圓
\PUXcfacedef\PUXFbr=br 粗圓
\PUXcfacedef\PUXFsr=sr 特圓
\PUXcfacedef\PUXFtli=tli 細隸書
\PUXcfacedef\PUXFmli=mli 中隸書
\PUXcfacedef\PUXFbli=bli 粗隸書
\PUXcfacedef\PUXFsli=sli 特隸書
\PUXcfacedef\PUXFtfs=tfs 細仿宋
\PUXcfacedef\PUXFmfs=mfs 中仿宋
\PUXcfacedef\PUXFbfs=bfs 粗仿宋
\PUXcfacedef\PUXFsfs=sfs 特仿宋
\PUXcfacedef\PUXFtsn=tsn 細行書
\PUXcfacedef\PUXFmsn=msn 中行書
\PUXcfacedef\PUXFbsn=bsn 粗行書
\PUXcfacedef\PUXFssn=ssn 特行書
\PUXcfacedef\PUXFtdr=tdr 細疊圓
\PUXcfacedef\PUXFmdr=mdr 中疊圓
\PUXcfacedef\PUXFbdr=bdr 粗疊圓
\PUXcfacedef\PUXFsdr=sdr 特疊圓
\PUXcfacedef\PUXFsdr=edr 空疊圓
\PUXcfacedef\PUXFkd=kd 勘亭流
\PUXcfacedef\PUXFgn=gn 古印
\PUXcfacedef\PUXFje=je 綜藝
\PUXcfacedef\PUXFwb=wb 魏碑
\PUXcfacedef\PUXFpo=po POP1
\PUXcfacedef\PUXFyt=yt 顏體
```

```
\PUXcfacedef\PUXFlms=lms 儷中宋
\PUXcfacedef\PUXFgirl=girl 少女
%
% 定義中英文字貌匹配
\PUXfacematch cmr \PUXFtm
\PUXfacematch cmtt \PUXFtm
\PUXfacematch cmbx \PUXFmb
\PUXfacematch cmti \PUXFmk
\PUXfacematch cmbxti \PUXFmk   % 粗斜體用中楷體
```

索引

Q · 十七劃

R · 十八劃

S · 十九劃

T · 二十劃

U · 二十一劃

V · 二十二劃

W · 二十三劃

Y · 二十五劃