給新貢獻人員的 **FreeBSD** 說明文件計畫入門書

給新貢獻人員的 **FreeBSD** 說明文件計畫入門書
Revision: f69fa39963
2017-07-17 06:35:59 +0000 by Ruey-Cherng Yu.

摘要

感謝您參與 FreeBSD 說明文件計劃，您的點滴貢獻，都相當寶貴。

本入手書內容包括：如何開始著手貢獻 FreeBSD 說明文件計劃 (FreeBSD Documentation Project, FDP) 的各項細節，以及會用到的一些工具、軟體，以及文件計畫的宗旨。

本入門書仍在持續撰寫中。任何修正或新增內容的建議都非常歡迎。

內容目錄

附表目錄

範例目錄

序

1. Shell 提示符號

下表顯示出一般使用者帳號與 root 的提示符號，在所有的文件例子中會用提示符號（Prompt），來提醒您該用哪種帳號才對。

帳號	提示符號
一般使用者	`%`
`root`	`#`

2. 書中所用的編排風格

下表為本書中所使用編排風格方式

代表意義	範例
指令	使用 `ls -l` 來列出所有的檔案。
檔名	編輯 `.login` 。
螢幕上會出現的訊息	`You have mail.`
輸入指令後，螢幕上會出現的對應內容。	`% date +"The time is %H:%M"` `The time is 09:18`
要參考的線上手冊	使用 su(1) 來切換帳號。
使用者名稱和群組名稱	只有 `root` 才可以做這件事。
語氣的強調。	使用者必須這樣做
打指令時，可替換的部份	要搜尋線上手冊的關鍵字，請輸入 `man -k ###`
環境變數。	`$HOME` 是指帳號的家目錄所在處。

3. 注意、提示、重要、警告與範例

出現在本文中的注意、警告、與範例。

注意

注意：表示需要注意的事項，其中包括您需要注意的事情，因為這些事情可能會影響到操作結果。

提示

提示：提供可能對您有用的資訊，例如簡化操作方式的技巧說明。

重要

重要：表示要特別注意的事情。一般來說，它們會包括操作指令時需要加的額外參數。

警告

警告：表示警告事項，比如如果您不則可能導致的損失。這些損失可能是對您或硬體造成實際傷害，也可能是無法估計的損害，例如一時疏忽而刪除重要檔案...。

範例 1. 範例的範本

這是舉例說明而已，通常包含應遵循的指令範例，或顯示某些特定動作所可能發生的結果。

4. 感謝

在此要感謝 Sue Blake, Patrick Durusau, Jon Hamilton, Peter Flynn, Christopher Maden 這些人的協助與閱讀初期草稿，並提供許多寶貴的潤稿意見與評論。

章 1. 概論

歡迎參與 FreeBSD 說明文件計劃 (FreeBSD Documentation Project, FDP)。維持優秀質量的文件對 FreeBSD 的成功來說十分重要，您的點滴貢獻都是十分寶貴的。

本文件描述：『 FDP 的架構有哪些』、『如何撰寫並提交文件』、『如何有效運用工具來協助撰稿』。

歡迎大家對 FDP 做出貢獻。唯一的成員要求就有貢獻的意願。

本入門書指出如何：

- 瞭解有哪些文件是由 FDP 所維護的。
- 安裝所需的說明文件工具和檔案。
- 修改說明文件。
- 提交修改以供審核並納入 FreeBSD 說明文件。

1.1. 快速上手

在編輯 FreeBSD 說明文件之前，有一些準備工作要做。首先，請訂閱 FreeBSD 文件計劃郵件論壇。有些團隊成員也會出現在 EFnet 的#bsddocs IRC 頻道。這些人可以幫忙解決文件相關的問題。

1. 安裝 textproc/docproj 套件或 Port。這個 meta-port 會安裝所有編輯和建置 FreeBSD 說明文件需要的軟體。

2. 在 ~/doc 安裝 FreeBSD 說明文件檔案庫的本地端工作副本 (請見 章 3, 工作副本)。

   ```
   % svn checkout https://svn.FreeBSD.org/doc/head    ~/doc
   ```

3. 設定文字編輯器：

   - 自動換行 (Word wrap) 設為 70 個字元。
   - Tab 定位點 (Tab stops) 設成 2。
   - 將行首每 8 個空白取代成 1 個 Tab。

   特定編輯器的設定方式列於 章 14, 編輯器設定。

4. 更新本地端工作副本

   ```
   % svn up ~/doc
   ```

5. 編輯需要修改的說明文件檔案。如果檔案需要大幅度的編修，請先諮詢郵件論壇。

   標籤 (Tag) 和 Entity 的使用方式可以參考 章 8, XHTML 標籤 和 章 9, DocBook 標籤。

6. 編輯完後，執行以下指令來檢查是否有問題：

   ```
   % igor -R filename.xml | less -RS
   ```

   檢查輸出並重新編輯檔案來修正顯示的錯誤，然後重新執行指令來找出剩下的問題。重複執行直到所有錯誤都解決完。

7. 永遠要送出修正前請先做建置測試 (Build-test)。在編輯的說明文件目錄最頂層執行 **make**，將會產生分頁的 HTML 格式 (Split HTML) 的文件。例如要建置 HTML 格式的英文版使用手冊，請在 en_US.ISO8859-1/books/handbook/ 目錄執行 make。

8. 修改並測試完後，產生“diff 檔”：

```
% cd ~/doc
% svn diff > bsdinstall.diff.txt
```

設一個可辨識的檔名。如上例中，是使用手冊的 bsdinstall 部份的修改。

9. 使用網頁版 問題回報 系統提交 diff 檔。如果使用網頁版，請輸入 [patch] ###### 的概要 。選擇 docs 分類和 doc-bug 類別。在訊息的主體中，輸入修正的簡短描述和其他相關的重要的細節。使用 [Browse...] 按鈕來附加 diff 檔。

1.2. FreeBSD 說明文件集

FDP 負責四類 FreeBSD 說明文件。

- 使用手冊 (Handbook)：使用手冊主要是給 FreeBSD 使用者提供詳盡的線上參考資料。
- 常見問答集 (FAQ)：主要是收集在各郵件論壇或論壇會常問到或有可能會問到的 FreeBSD 相關問題與答案。(簡單講，就是『問答集』格式) 通常會擺在這裡面的問答格式，不會放太長的詳細內容。
- 操作手冊 (Manual page)：英文版的系統手冊並不是由 FDP 所撰寫的，因為它們是屬於基礎系統 (Base system) 的部份。然而，FDP 可以修改這些文件，來讓這些文件寫得更清楚，甚至是勘正錯誤的地方。
- 網站：這是 FreeBSD 在網路上的主要部份，位於 http://www.FreeBSD.org/ 以及許多其他鏡像站 (Mirror)。這網站是許多人第一次接觸 FreeBSD 的地方

翻譯團隊負責翻譯使用手冊和網站到不同的語言。線上手冊目前並未翻譯

FreeBSD 網站、使用手冊、和 FAQ 的文件原始碼可以在 https://svn.FreeBSD.org/doc/ 的文件檔案庫取得。

線上手冊的原始碼則是在 https://svn.FreeBSD.org/base/ 的原始碼庫可以取得。

說明文件提交訊息可以用 svn log 察看。提交訊息也會封存在 http://lists.FreeBSD.org/mailman/listinfo/svn-doc-all 。

這些儲存庫的網頁版位於 https://svnweb.FreeBSD.org/doc/ 和 https://svnweb.FreeBSD.org/base/。

許多人會寫 FreeBSD 的教學文件或是 how-to 文章。有些保存在 FDP 的檔案中。其他一些說明文件則是作者希望放在他處。FDP 會盡力提供這些說明文件的連結。

章 2. 工具

有些工具軟體用來管理 FreeBSD 說明文件，並將他轉換成不同的輸出格式。有些則是在使用接下來章節的範例之前一定要安裝。有些工具是選擇性安裝的，但是裝了之後會更容易進行文件製作工作。

2.1. 必備工具

從 Port 套件集安裝 textproc/docproj。這個 meta-port 會安裝處理 FreeBSD 說明文件需要的所有應用程式。以下列出特定元件的進一步說明。

2.1.1. DTDs 與 Entities

FreeBSD 說明文件使用幾種文件類型定義 (DTDs) 與 XML entities 集合。這些都會經由 textproc/docproj Port 來安裝。

XHTML DTD (textproc/xhtml)
: XHTML 是全球資訊網的一種標記語言，也是整個 FreeBSD 網站所使用的格式。

DocBook DTD (textproc/docbook-xml)
: DocBook 設計來製作技術說明文件的標記語言版本。FreeBSD 說明文件是以 DocBook 來撰寫。

ISO 8879 entities (textproc/iso8879)
: 在 ISO 8879:1986 之中的 entity 被許多 DTD 所大量使用，包括了數學符號、拉丁字母符號(尖重音等音節符號也是)以及希臘符號。

2.2. 選用工具

以下應用程式並非必要，但有了可讓在說明文件的作業上更簡單或提升能力。

2.2.1. 軟體

Vim (editors/vim)
: 一個很受歡迎的編輯器，可以處理 XML 和他的衍生相關文件，例如 DocBook XML。

Emacs 或 XEmacs (editors/emacs 或 editors/xemacs)
: 這兩個編輯器都包含特別模式來編輯用 XML DTD 標記的文件。這個模式包含指令來減少打字量，並可以幫忙減少錯誤的發生。

章 3. 工作副本

工作副本 (Working copy) 指的是已下載到本地電腦的 FreeBSD 說明文件樹檔案庫，所有對工作副本的更改會經過測試後再以修補檔 (Patch) 的格式提交到主要檔案庫。

完整的說明文件樹副本會佔據 700 MB 的磁碟空間，要有空間能夠儲存暫存檔及各種輸出格式的測試版本需要 1 GB 的空間。

FreeBSD 說明文件檔案使用 Subversion 來管理，由於 Subversion 為 textproc/docproj 的必要應用程式之一，所以會隨著 textproc/docproj 一併安裝。

3.1. 說明文件與操作手冊

FreeBSD 說明文件不只有書籍與文章，還有所有指令與設定檔的操作手冊 (Manual page) 也是文件的一部份，其中也有一部份是 FDP 的地盤。相關的檔案庫有兩個：doc 中有書籍與文章，而 base 中有作業系統以及操作手冊。要編輯操作手冊則必須另外取出 (Checkout) base 檔案庫。

檔案庫中可能會含有數個版本的說明文件與原始碼。新的修改幾乎都只對最新版本 head 做更新。

3.2. 選擇一個目錄

FreeBSD 說明文件一般會儲存在 /usr/doc/ ，而系統原始碼及操作手冊則會存在 /usr/src/ 。這些目錄樹可改放在其他地方，有些使用者可能會為了避免與現有在主要目錄的資料搞混，把工作副本放在其他的地方。以下的例子會放在 ~/doc 以及 ~/src 兩個在使用者家目錄下的目錄。

3.3. 取出一份副本

從檔案庫下載工作副本的動作稱作 取出 (Checkout)，使用 svn checkout 來完成。本範例會取出主要說明文件樹最新版本的副本：

```
% svn checkout https://svn.FreeBSD.org/doc/head ~/doc
```

取出原始碼編輯操作手冊的動作也很相似：

```
% svn checkout https://svn.FreeBSD.org/base/head ~/src
```

3.4. 更新工作副本

在 FreeBSD 檔案庫中的文件與檔案每天都會更改，人們會修改檔案與提交變更的頻率非常快，即使取出 (Checkout) 只過小一段時間，本地的工作副本可能就與主要 FreeBSD 檔案庫有差異了。要更新本地版本以同步對主要檔案庫的變更可在有本地工作副本的目錄下使用 svn update ：

```
% svn update ~/doc
```

養成良好的習慣在編輯文件檔前先執行 svn update ，由於可能會有其他人最近才編輯完該檔案，若未更新本地工作副則不會有最新變更的內容，比其還要將舊版本地檔案與新版檔案庫檔案合併的動作來說，直接編輯最新版本的檔案要簡單多了。

3.5. 還原變更

有時才做完的變更可能就變的不需要了，或者作者剛想要重新撰寫。檔案可使以使用 svn revert 來“重設”成尚未被修改過的狀態，舉例來說，要清除所有對 chapter.xml 的修改然後還原到未修改的版本可：

```
% svn revert chapter.xml
```

3.6. 比對差異

在編輯一個檔案或數個檔案完成之後，需將本地工作副本與 FreeBSD 檔案庫的差異儲存到一個檔案然後提交。這些 差異 (Diff) 檔可透過將 `svn diff` 的輸出轉向到檔案來建立：

```
% cd ~/doc
% svn diff > doc-fix-spelling.diff
```

給檔案取一個有意義的名字來辨識這次修改的內容，上述範例為例則是要修正整個說明文件樹的拼寫。

若 diff 檔要使用網站的 "Submit a FreeBSD problem report" 介面來提交，請加上 `.txt` 副檔名來給認真又單純的網頁表單可以辨識其內容為純文字的線索。

請小心：`svn diff` 會產生所有在目前目錄及其子目錄的變更差異，若在該工作副本中有已經編輯過的檔案還沒有要提交，請列出需要比較差異的檔案清單：

```
% cd ~/doc
% svn diff disks/chapter.xml printers/chapter.xml   > disks-printers.diff
```

3.7. Subversion 參考文獻

以下範例會示範 Subversion 的基本用法，要取得更多資訊可至 Subversion Book 與 Subversion 說明文件。

章 4. 說明文件目錄結構

在 doc/ 樹中的檔案與目錄需要遵守特定結構是因為：

1. 讓自動轉換說明文件到各種格式更簡單。

2. 促進不同說明文件組織之間的一致性，以便更輕鬆的在不同說明文件之間切換。

3. 可以很容易的決定新的說明文件應該放在文件樹中的哪個位置。

除此之外，說明文件樹必須能容納各種不同語言與編碼的說明文件。很重要的一點是，說明文件樹結構不應强制任何特定默認或文化的偏好。

4.1. 最上層，doc/

在 doc/ 底下有兩種類型的目錄，兩種都有非常明確的目錄名稱與意義。

目錄	用途
share	含有未特定用於各說明文件翻譯與編碼的檔案。其子目錄更進一步將不同資訊的分類，例如，構成 make(1) 基礎設施的檔案放置於 share/mk ，而額外的 XML 支援檔（如 FreeBSD 延伸的 DocBook DTD) 則放在 share/xml 。
lang.*encoding*	每一個目錄代表可用的說明文件翻譯與編碼，例如 en_US.ISO8859-1/ 及 zh_TW.UTF-8/ 。名稱雖然長，但完整表達語言與編碼可以避免未來當一個翻譯團隊要以不同編碼提供同一個語言的說明文件造成的問題，這也同時避免往後要切換成萬國碼 (Unicode) 可能造成的問題。

4.2. *lang*.*encoding*/ 目錄

這些目錄中含有說明文件。在此階層說明文件分成三個分類，使用不同的目錄名稱來代表。

目錄	用途
articles	以 DocBook article （或同等級）標記的說明文件。非常短，且會分成幾個小節，通常取得時只會有一個 XHTML 檔案。
books	以 DocBook book（或同等級）標記的說明文件。有跟書籍一樣的長度，會分成數個章。通常取得時會包含一個大型的 XHTML 檔（供有較快連線速度的人使用，或者那些想直接在瀏覽器列印的人）與數個連結的較小的檔案。
man	供系統操作手冊 (Manual page) 翻譯使用。這個目錄會包含一個或多個 man*n* 目錄來對應已經翻譯的章節。

並非所有 *lang*.*encoding* 的目錄都會有這些子目錄，這要看該翻譯團隊已經完成了多少翻譯來決定。

4.3. 文件特定資訊

本節內含有關由 FDP 所管理的特定文件的特定注意事項。

4.3.1. 使用手冊 (Handbook)

books/handbook/

操作手冊是以使用 FreeBSD DocBook 擴充 DTD 的 DocBook XML 所撰寫。

使用手冊使用 DocBook book 來組織，整個手冊區分成數個部份 (part)，每個部份中內含數個章 (chapter)，而章 (chapter) 又更進一步的細分成數個節 (sect1) 與小節 (sect2, sect3) 以此類推。

4.3.1.1. 實體組織結構

在 handbook 目錄中有數個檔案及目錄。

注意

使用手冊的組織結構可能會隨時改變，本文件所詳述的組織結構可能會與現況不盡相同。有關使用手冊組織結構的問題可張貼到 FreeBSD documentation project mailing list。

4.3.1.1.1. Makefile

Makefile 定義了一些會影響 XML 原始碼要如何轉換至其他格式的變數，並列出產生使用手冊所需的各原始檔，接著會引用可處理在各種文件格式轉換的標準 doc.project.mk 程式碼。

4.3.1.1.2. book.xml

此為使用手冊的最上層文件，其中包含了使用手冊的 DOCTYPE 宣告以及用來描述使用手冊結構的元素。

book.xml 會使用 參數 Entities 來載入 .ent 副檔名的檔案。這些檔案 (稍後會詳述) 接著會定義使用手冊剩下篇幅會使用的 一般 Entities。

4.3.1.1.3. *directory* /chapter.xml

每個在使用手冊的章會儲存成名稱為 chapter.xml 的檔案，分別放在不同的目錄中。每個目錄均是以在 chapter 元素中 id 屬性中的值來命名。

例如，若有一章的檔案內容為：

```
<chapter id="kernelconfig">
...
</chapter>
```

那麼放置這個 chapter.xml 的目錄則會命名為 kernelconfig 。一般來說一章的所有內容會存放在這一個檔案。

當有 XHTML 版本的使用手冊之後，會用該檔產出 kernelconfig.html ，而這個名稱也是根據 id 的值而定，與目錄名稱無關。

在早期版本的使用手冊，檔案皆儲存在與 book.xml 相同的目錄中，而名稱會以 chapter 元素中的 id 屬性的值來命名。現在可在每個章節之中引用圖片，每個使用手冊章節的圖片會儲存在 share/images/books/handbook ，而在地化版本的圖片應放在與每個章節 XML 原始碼相同的目錄。命名空間會衝突是必然的，但以目錄多、檔案少與目錄少、檔案多的結構相比，目錄多、檔案少會較容易處理命名空間衝突的問題。

簡單來說會有很多個內含 chapter.xml 檔案的目錄，例如 basics/chapter.xml , introduction/chapter.xml 以及 printing/chapter.xml 。

重要

請勿以在使用手冊中的章節順序來命名章節或目錄，因為順序是會隨使用手冊重新組織後的內容改變的。重新組織結構應不需要去重新命名檔案，除非整個章節的階層被提升或下降。

`chapter.xml` 檔案並不是可以被單獨建置的完整 XML 文件，只能結合成整本使用手冊之後再一併建置。

章 5. 說明文件建置流程

本章內容涵蓋了說明文件建置流程以及如何使用 make(1) 來控制如何建置。

5.1. 繪製 Docbook 爲其他格式

從單一個 DocBook 原始檔可以產生各種不同類型的輸出，想要輸出的類型可用 FORMATS 變數來設定。已知的格式清單列在 KNOWN_FORMATS 中：

```
% cd ~/doc/en_US.ISO8859-1/books/handbook
% make -V KNOWN_FORMATS
```

表格 5.1. 常見輸出格式

FORMATS 值	檔案類型	說明
html	HTML，單檔	單一 book.html 或 article.html 。
html-split	HTML，多檔	多個 HTML 檔案，每個章或節一個檔案，供一般網站使用。
pdf	PDF	可攜的文件格式

預設輸出的格式會依文件而有所不同，但同常是 html-split 。其他格式可設定 FORMATS 為特定值來選擇。在 FORMATS 設定所需格式的清單可一次輸出多個格式。

範例 5.1. 建置單頁 HTML 輸出檔

```
% cd ~/doc/en_US.ISO8859-1/books/handbook
% make FORMATS=html
```

範例 5.2. 建置分頁 HTML 及 PDF 輸出檔

```
% cd ~/doc/en_US.ISO8859-1/books/handbook
% make FORMATS="html-split pdf"
```

5.2. FreeBSD 說明文件建置工具集

建置與安裝 FDP 說明文件會使用到以下工具。

- 主要的建置工具為 make(1)，特別是 Berkeley Make。
- 套件建置會使用 FreeBSD 的 pkg-create(8) 來處理。
- gzip(1) 用來建立壓縮版的文件，也支援 bzip2(1) 封存。tar(1) 則用在套件建置。
- install(1) 用來安裝說明文件。

5.3. 了解在說明文件樹中的 Makefile

在 FreeBSD 說明文件計劃樹底下的 Makefile 主要有三個類型。

- 子目錄的 Makefile 傳遞指令給在其所在目錄底下的目錄。
- 說明文件的 Makefile s 用來描述要如何在其所在目錄產生文件。
- Make 引用檔 會連結一些產生文件所需的程式，通常為 doc.*xxx*.mk。

5.3.1. 子目錄的 Makefile

這種 Makefile 通常採用的格式為：

```
SUBDIR =articles
SUBDIR+=books

COMPAT_SYMLINK = en

DOC_PREFIX?= ${.CURDIR}/..
.include "${DOC_PREFIX}/share/mk/doc.project.mk"
```

前四行非空白的行用來定義 make(1) 的變數 SUBDIR, COMPAT_SYMLINK 及 DOC_PREFIX 。

SUBDIR 敘述與 COMPAT_SYMLINK 敘述示範了如何指派數值到一個變數，覆蓋先前的值。

SUBDIR 的第二行敘述示範如何將數值附加到目前的變數值之後，SUBDIR 變數現在變成了 articles books。

DOC_PREFIX 指派式示範了如何只在變數尚未定義時才指派數值給變數。這個功能在當 DOC_PREFIX 不在 Makefile 所認為的地方時，使用者可以覆蓋這個值，並提供正確的值。

這所有的敘述實際代表什麼意思呢？ SUBDIR 會列出接下來建置流程應傳遞作業到那些子目錄。

COMPAT_SYMLINK 是一個用來指定語言官方編碼的相容性符號連結 (doc/en 會指向 en_US.ISO-8859-1)。

DOC_PREFIX 是到 FreeBSD 說明文件計劃樹根目錄的路徑。這並非每一次都可以很輕易找到，為了增加彈性，要改寫也很簡單。.CURDIR 是一個 make(1) 內建的變數，代表目前目錄的路徑。

最後一行會引用 FreeBSD 說明文件計劃的全專案 make(1) 系統檔 doc.project.mk ，用來轉換這些變數成為建置的指令。

5.3.2. 說明文件的 Makefile

這些 Makefile 用來設定 make(1) 變數來描述要如何建置在該目錄中的說明文件。

這裡有一個例子：

```
MAINTAINER=nik@FreeBSD.org

DOC?= book

FORMATS?= html-split html

INSTALL_COMPRESSED?= gz
INSTALL_ONLY_COMPRESSED?=

# SGML content
SRCS=  book.xml

DOC_PREFIX?= ${.CURDIR}/../../..

.include "$(DOC_PREFIX)/share/mk/docproj.docbook.mk"
```

`MAINTAINER` 變數讓提交者可以聲明文件在 FreeBSD 說明文件計劃中的所有權，並負責維護該文件。

`DOC` 是由此目錄會建立的主要文件的名稱（不需要 `.xml` 副檔名）。`SRCS` 會列出產生文件所需的各別檔案，此處也應引用要在重新建置 (Rebuild) 使用的重要的檔案。

`FORMATS` 用來指定此份文件建置時預設應採用的格式。`INSTALL_COMPRESSED` 是為在文件建置時預設要使用的壓縮技術清單。`INSTALL_ONLY_COMPRESS` 預設為空值的，若在建置時只想要產生壓縮後的文件則改成非空值。

`DOC_PREFIX` 以及 include 敘述句應不需再說明了。

5.4. FreeBSD 說明文件計劃 Make 引用檔

make(1) includes are best explained by inspection of the code. Here are the system include files:

- `doc.project.mk` is the main project include file, which includes all the following include files, as necessary.
- `doc.subdir.mk` handles traversing of the document tree during the build and install processes.
- `doc.install.mk` provides variables that affect ownership and installation of documents.
- `doc.docbook.mk` is included if `DOCFORMAT` is `docbook` and `DOC` is set.

5.4.1. `doc.project.mk`

By inspection:

```
DOCFORMAT?= docbook
MAINTAINER?= doc@FreeBSD.org

PREFIX?= /usr/local
PRI_LANG?= en_US.ISO8859-1

.if defined(DOC)
.if ${DOCFORMAT} == "docbook"
.include "doc.docbook.mk"
.endif
.endif

.include "doc.subdir.mk"
.include "doc.install.mk"
```

5.4.1.1. 變數

`DOCFORMAT` and `MAINTAINER` are assigned default values, if these are not set by the document make file.

`PREFIX` is the prefix under which the documentation building tools are installed. For normal package and port installation, this is `/usr/local` .

`PRI_LANG` should be set to whatever language and encoding is natural amongst users these documents are being built for. US English is the default.

注意

`PRI_LANG` does not affect which documents can, or even will, be built. Its main use is creating links to commonly referenced documents into the FreeBSD documentation install root.

5.4.1.2. 條件

The `.if defined(DOC)` line is an example of a make(1) conditional which, like in other programs, defines behavior if some condition is true or if it is false. `defined` is a function which returns whether the variable given is defined or not.

`.if ${DOCFORMAT} == "docbook"` , next, tests whether the `DOCFORMAT` variable is `"docbook"` , and in this case, includes `doc.docbook.mk` .

The two `.endif`s close the two above conditionals, marking the end of their application.

5.4.2. `doc.subdir.mk`

This file is too long to explain in detail. These notes describe the most important features.

5.4.2.1. 變數

- `SUBDIR` is a list of subdirectories that the build process should go further down into.
- `ROOT_SYMLINKS` is the name of directories that should be linked to the document install root from their actual locations, if the current language is the primary language (specified by `PRI_LANG`).
- `COMPAT_SYMLINK` is described in the Subdirectory Makefile section.

5.4.2.2. 目標與巨集

Dependencies are described by *`target`* `:` *`dependency1 dependency2 ...`* tuples, where to build `target` , the given dependencies must be built first.

After that descriptive tuple, instructions on how to build the target may be given, if the conversion process between the target and its dependencies are not previously defined, or if this particular conversion is not the same as the default conversion method.

A special dependency `.USE` defines the equivalent of a macro.

```
_SUBDIRUSE: .USE
.for entry in ${SUBDIR}
 @${ECHO} "===> ${DIRPRFX}${entry}"
 @(cd ${.CURDIR}/${entry} && \
 ${MAKE} ${.TARGET:S/realpackage/package/:S/realinstall/install/} DIRPRFX=
${DIRPRFX}${entry}/ )
.endfor
```

In the above, `_SUBDIRUSE` is now a macro which will execute the given commands when it is listed as a dependency.

What sets this macro apart from other targets? Basically, it is executed after the instructions given in the build procedure it is listed as a dependency to, and it does not adjust `.TARGET` , which is the variable which contains the name of the target currently being built.

```
clean: _SUBDIRUSE
 rm -f ${CLEANFILES}
```

In the above, `clean` will use the `_SUBDIRUSE` macro after it has executed the instruction `rm -f ${CLEANFILES}` . In effect, this causes `clean` to go further and further down the directory tree, deleting built files as it goes down, not on the way back up.

5.4.2.2.1. 已提供的目標

- `install` and `package` both go down the directory tree calling the real versions of themselves in the subdirectories (`realinstall` and `realpackage` respectively).

- `clean` removes files created by the build process (and goes down the directory tree too). `cleandir` does the same, and also removes the object directory, if any.

5.4.2.3. 更多條件

- `exists` is another condition function which returns true if the given file exists.
- `empty` returns true if the given variable is empty.
- `target` returns true if the given target does not already exist.

5.4.2.4. 在 `make (.for)` 中的迴圈結構

`.for` provides a way to repeat a set of instructions for each space-separated element in a variable. It does this by assigning a variable to contain the current element in the list being examined.

```
_SUBDIRUSE: .USE
.for entry in ${SUBDIR}
 @${ECHO} "===> ${DIRPRFX}${entry}"
 @(cd ${.CURDIR}/${entry} && \
 ${MAKE} ${.TARGET:S/realpackage/package/:S/realinstall/install/} DIRPRFX=
${DIRPRFX}${entry}/ )
.endfor
```

In the above, if `SUBDIR` is empty, no action is taken; if it has one or more elements, the instructions between `.for` and `.endfor` would repeat for every element, with `entry` being replaced with the value of the current element.

章 6. 網站

FreeBSD 網站是 FreeBSD 文件的一部份。網站的檔案儲存在文件樹目錄，此例中是 `~/doc`，的 `en_US.ISO8859-1/htdocs` 子目錄。

6.1. 環境變數

Several environment variables control which parts of the web site are built or installed, and to which directories.

提示

The web build system uses make(1), and considers variables to be set when they have been defined, even if they are empty. The examples here show the recommended ways of defining and using these variables. Setting or defining these variables with other values or methods might lead to unexpected surprises.

`DESTDIR`
: DESTDIR specifies the path where the web site files are to be installed.

  This variable is best set with env(1) or the user shell's method of setting environment variables, `setenv` for csh(1) or `export` for sh(1).

`ENGLISH_ONLY`
: Default: undefined. Build and include all translations.

  `ENGLISH_ONLY=yes` : use only the English documents and ignore all translations.

`WEB_ONLY`
: Default: undefined. Build both the web site and all the books and articles.

  `WEB_ONLY=yes` : build or install only HTML pages from the `en_US.ISO8859-1/htdocs` directory. Other directories and documents, including books and articles, will be ignored.

`WEB_LANG`
: Default: undefined. Build and include all the available languages on the web site.

  Set to a space-separated list of languages to be included in the build or install. The formats are the same as the directory names in the document root directory. For example, to include the German and French documents:

  ```
  WEB_LANG="de_DE.ISO8859-1 fr_FR.ISO8859-1"
  ```

`WEB_ONLY`, `WEB_LANG`, and `ENGLISH_ONLY` are make(1) variables and can be set in `/etc/make.conf`, `Makefile.inc`, as environment variables on the command line, or in dot files.

6.2. 建置並安裝網頁

Having obtained the documentation and web site source files, the web site can be built.

An actual installation of the web site is run as the `root` user because the permissions on the web server directory will not allow files to be installed by an unprivileged user. For testing, it can be useful to install the files as a normal user to a temporary directory.

In these examples, the web site files are built by user `jru` in their home directory, `~/doc`, with a full path of `/usr/home/jru/doc`.

提示

The web site build uses the `INDEX` from the Ports Collection and might fail if that file or `/usr/ports` is not present. The simplest approach is to install the Ports Collection.

範例 6.1. 建置完整網站與所有說明文件

Build the web site and all documents. The resulting files are left in the document tree:

```
% cd ~/doc/en_US.ISO8859-1/htdocs/
% make all
```

範例 6.2. 只建置英文版網站

Build the web site only, in English, as user `jru`, and install the resulting files into `/tmp/www` for testing:

```
% cd ~/doc/en_US.ISO8859-1/htdocs/
% env DESTDIR=/tmp/www make ENGLISH_ONLY=yes WEB_ONLY=yes all install
```

Changes to static files can usually be tested by viewing the modified files directly with a web browser. If the site has been built as shown above, a modified main page can be viewed with:

```
% firefox /tmp/www/data/index.html
```

Modifications to dynamic files can be tested with a web server running on the local system. After building the site as shown above, this `/usr/local/etc/apache24/httpd.conf` can be used with www/apache24:

```
# httpd.conf for testing the FreeBSD website
Define TestRoot "/tmp/www/data"

# directory for configuration files
ServerRoot "/usr/local"

Listen 80

# minimum required modules
LoadModule authz_core_module libexec/apache24/mod_authz_core.so
LoadModule mime_module libexec/apache24/mod_mime.so
LoadModule unixd_module libexec/apache24/mod_unixd.so
LoadModule cgi_module libexec/apache24/mod_cgi.so
LoadModule dir_module libexec/apache24/mod_dir.so

# run the webserver as user and group
User www
Group www

ServerAdmin you@example.com
ServerName fbsdtest

# deny access to all files
<Directory />
    AllowOverride none
    Require all denied
```

```
</Directory>

# allow access to the website directory
DocumentRoot "${TestRoot}"
<Directory "${TestRoot}">
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

# prevent access to .htaccess and .htpasswd files
<Files ".ht*">
    Require all denied
</Files>

ErrorLog "/var/log/httpd-error.log"
LogLevel warn

# set up the CGI script directory
<Directory "${TestRoot}/cgi">
    AllowOverride None
    Options None
    Require all granted
    Options +ExecCGI
    AddHandler cgi-script .cgi
</Directory>

Include etc/apache24/Includes/*.conf
```

Start the web server with

```
# service apache24 onestart
```

The web site can be viewed at http://localhost. Be aware that many links refer to the real FreeBSD site by name, and those links will still go to the external site instead of the local test version. Fully testing the local site will require temporarily setting DNS so `www.FreeBSD.org` resolves to `localhost` or the local IP address.

範例 6.3. 建置並安裝網站

Build the web site and all documents as user `jru`. Install the resulting files as `root` into the default directory, `/root/public_html` :

```
% cd ~/doc/en_US.ISO8859-1/htdocs
% make all
% su -
Password:
# cd /usr/home/jru/doc/en_US.ISO8859-1/htdocs
# make install
```

The install process does not delete any old or outdated files that existed previously in the same directory. If a new copy of the site is built and installed every day, this command will find and delete all files that have not been updated in three days:

```
# find /usr/local/www  -ctime 3 -delete
```

章 7. XML 入門

Most FDP documentation is written with markup languages based on XML. This chapter explains what that means, how to read and understand the documentation source, and the XML techniques used.

Portions of this section were inspired by Mark Galassi's Get Going With DocBook.

7.1. 概論

In the original days of computers, electronic text was simple. There were a few character sets like ASCII or EBCDIC, but that was about it. Text was text, and what you saw really was what you got. No frills, no formatting, no intelligence.

Inevitably, this was not enough. When text is in a machine-usable format, machines are expected to be able to use and manipulate it intelligently. Authors want to indicate that certain phrases should be emphasized, or added to a glossary, or made into hyperlinks. Filenames could be shown in a "typewriter" style font for viewing on screen, but as "italics" when printed, or any of a myriad of other options for presentation.

It was once hoped that Artificial Intelligence (AI) would make this easy. The computer would read the document and automatically identify key phrases, filenames, text that the reader should type in, examples, and more. Unfortunately, real life has not happened quite like that, and computers still require assistance before they can meaningfully process text.

More precisely, they need help identifying what is what. Consider this text:

> To remove `/tmp/foo` , use rm(1).
>
> ```
> % rm /tmp/foo
> ```

It is easy to see which parts are filenames, which are commands to be typed in, which parts are references to manual pages, and so on. But the computer processing the document cannot. For this we need markup.

"Markup" is commonly used to describe "adding value" or "increasing cost". The term takes on both these meanings when applied to text. Markup is additional text included in the document, distinguished from the document's content in some way, so that programs that process the document can read the markup and use it when making decisions about the document. Editors can hide the markup from the user, so the user is not distracted by it.

The extra information stored in the markup adds value to the document. Adding the markup to the document must typically be done by a person—after all, if computers could recognize the text sufficiently well to add the markup then there would be no need to add it in the first place. This increases the cost (the effort required) to create the document.

The previous example is actually represented in this document like this:

```
<para>To remove <filename>/tmp/foo</filename> , use &man.rm.1;.</para>

<screen>&prompt.user; <userinput>rm /tmp/foo</userinput></screen>
```

The markup is clearly separate from the content.

Markup languages define what the markup means and how it should be interpreted.

Of course, one markup language might not be enough. A markup language for technical documentation has very different requirements than a markup language that is intended for cookery recipes. This, in turn, would be very different from a markup language used to describe poetry. What is really needed is a first language used to write these other markup languages. A meta markup language.

This is exactly what the eXtensible Markup Language (XML) is. Many markup languages have been written in XML, including the two most used by the FDP, XHTML and DocBook.

Each language definition is more properly called a grammar, vocabulary, schema or Document Type Definition (DTD). There are various languages to specify an XML grammar, or schema.

A schema is a complete specification of all the elements that are allowed to appear, the order in which they should appear, which elements are mandatory, which are optional, and so forth. This makes it possible to write an XML parser which reads in both the schema and a document which claims to conform to the schema. The parser can then confirm whether or not all the elements required by the vocabulary are in the document in the right order, and whether there are any errors in the markup. This is normally referred to as "validating the document".

注意

Validation confirms that the choice of elements, their ordering, and so on, conforms to that listed in the grammar. It does not check whether appropriate markup has been used for the content. If all the filenames in a document were marked up as function names, the parser would not flag this as an error (assuming, of course, that the schema defines elements for filenames and functions, and that they are allowed to appear in the same place).

Most contributions to the Documentation Project will be content marked up in either XHTML or DocBook, rather than alterations to the schemas. For this reason, this book will not touch on how to write a vocabulary.

7.2. 元素、標籤與屬性

All the vocabularies written in XML share certain characteristics. This is hardly surprising, as the philosophy behind XML will inevitably show through. One of the most obvious manifestations of this philosophy is that of content and elements.

Documentation, whether it is a single web page, or a lengthy book, is considered to consist of content. This content is then divided and further subdivided into elements. The purpose of adding markup is to name and identify the boundaries of these elements for further processing.

For example, consider a typical book. At the very top level, the book is itself an element. This "book" element obviously contains chapters, which can be considered to be elements in their own right. Each chapter will contain more elements, such as paragraphs, quotations, and footnotes. Each paragraph might contain further elements, identifying content that was direct speech, or the name of a character in the story.

It may be helpful to think of this as "chunking" content. At the very top level is one chunk, the book. Look a little deeper, and there are more chunks, the individual chapters. These are chunked further into paragraphs, footnotes, character names, and so on.

Notice how this differentiation between different elements of the content can be made without resorting to any XML terms. It really is surprisingly straightforward. This could be done with a highlighter pen and a printout of the book, using different colors to indicate different chunks of content.

Of course, we do not have an electronic highlighter pen, so we need some other way of indicating which element each piece of content belongs to. In languages written in XML (XHTML, DocBook, et al) this is done by means of tags.

A tag is used to identify where a particular element starts, and where the element ends. The tag is not part of the element itself. Because each grammar was normally written to mark up specific types of information, each one will recognize different elements, and will therefore have different names for the tags.

For an element called *element-name* the start tag will normally look like *<element-name >*. The corresponding closing tag for this element is *</element-name >*.

範例 7.1. 使用元素（開始與結束標籤）

XHTML has an element for indicating that the content enclosed by the element is a paragraph, called `p`.

```
<p>This is a paragraph.  It starts with the start tag for
  the 'p' element, and it will end with the end tag for the 'p'
  element.</p>

<p>This is another paragraph.  But this one is much shorter.</p>
```

Some elements have no content. For example, in XHTML, a horizontal line can be included in the document. For these “empty” elements, XML introduced a shorthand form that is completely equivalent to the two-tag version:

範例 7.2. 使用沒有內容的元素

XHTML has an element for indicating a horizontal rule, called `hr`. This element does not wrap content, so it looks like this:

```
<p>One paragraph.</p>
<hr></hr>

<p>This is another paragraph.  A horizontal rule separates this
  from the previous paragraph.</p>
```

The shorthand version consists of a single tag:

```
<p>One paragraph.</p>
<hr/>

<p>This is another paragraph.  A horizontal rule separates this
  from the previous paragraph.</p>
```

As shown above, elements can contain other elements. In the book example earlier, the book element contained all the chapter elements, which in turn contained all the paragraph elements, and so on.

範例 7.3. 在元素中的元素；`em`

```
<p>This is a simple <em>paragraph</em> where some
  of the <em>words</em> have been <em>emphasized</em>.</p>
```

The grammar consists of rules that describe which elements can contain other elements, and exactly what they can contain.

重要

People often confuse the terms tags and elements, and use the terms as if they were interchangeable. They are not.

> An element is a conceptual part of your document. An element has a defined start and end. The tags mark where the element starts and ends.
>
> When this document (or anyone else knowledgeable about XML) refers to "the <p> tag" they mean the literal text consisting of the three characters <, p, and >. But the phrase "the p element" refers to the whole element.
>
> This distinction is very subtle. But keep it in mind.

Elements can have attributes. An attribute has a name and a value, and is used for adding extra information to the element. This might be information that indicates how the content should be rendered, or might be something that uniquely identifies that occurrence of the element, or it might be something else.

An element's attributes are written inside the start tag for that element, and take the form *attribute-name="attribute-value "*.

In XHTML, the `p` element has an attribute called `align`, which suggests an alignment (justification) for the paragraph to the program displaying the XHTML.

The `align` attribute can take one of four defined values, `left`, `center`, `right` and `justify`. If the attribute is not specified then the default is `left`.

範例 7.4. 使用元素的屬性

```
<p align="left"> The inclusion of the align attribute
  on this paragraph was superfluous, since the default is left.</p>

<p align="center"> This may appear in the center.</p>
```

Some attributes only take specific values, such as `left` or `justify`. Others allow any value.

範例 7.5. 屬性的單引號

```
<p align='right'> I am on the right!</p>
```

Attribute values in XML must be enclosed in either single or double quotes. Double quotes are traditional. Single quotes are useful when the attribute value contains double quotes.

Information about attributes, elements, and tags is stored in catalog files. The Documentation Project uses standard DocBook catalogs and includes additional catalogs for FreeBSD-specific features. Paths to the catalog files are defined in an environment variable so they can be found by the document build tools.

7.2.1. 待辦事項...

Before running the examples in this document, install textproc/docproj from the FreeBSD Ports Collection. This is a meta-port that downloads and installs the standard programs and supporting files needed by the Documentation Project. csh(1) users must use `rehash` for the shell to recognize new programs after they have been installed, or log out and then log back in again.

1. Create `example.xml` , and enter this text:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>An Example XHTML File</title>
  </head>

  <body>
    <p>This is a paragraph containing some text.</p>

    <p>This paragraph contains some more text.</p>

    <p align="right"> This paragraph might be right-justified.</p>
  </body>
</html>
```

2. Try to validate this file using an XML parser.

   textproc/docproj includes the `xmllint` validating parser [22].

   Use `xmllint` to validate the document:

   ```
   % xmllint --valid --noout example.xml
   ```

   `xmllint` returns without displaying any output, showing that the document validated successfully.

3. See what happens when required elements are omitted. Delete the line with the `<title>` and `</title>` tags, and re-run the validation.

   ```
   % xmllint --valid --noout example.xml
   example.xml:5: element head: validity error : Element head content does not follow
    the DTD, expecting ((script | style | meta | link | object | isindex)* , ((title ,
    (script | style | meta | link | object | isindex)* , (base , (script | style | meta
    | link | object | isindex)*)?) | (base , (script | style | meta | link | object |
    isindex)* , title , (script | style | meta | link | object | isindex)*))), got ()
   ```

   This shows that the validation error comes from the *`fifth`* line of the *`example.xml`* file and that the content of the `<head>` is the part which does not follow the rules of the XHTML grammar.

   Then `xmllint` shows the line where the error was found and marks the exact character position with a ^ sign.

4. Replace the `title` element.

7.3. DOCTYPE 宣告

The beginning of each document can specify the name of the DTD to which the document conforms. This DOCTYPE declaration is used by XML parsers to identify the DTD and ensure that the document does conform to it.

A typical declaration for a document written to conform with version 1.0 of the XHTML DTD looks like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
```

That line contains a number of different components.

`<!`
: The indicator shows this is an XML declaration.

`DOCTYPE`
: Shows that this is an XML declaration of the document type.

`html`
: Names the first element that will appear in the document.

`PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"`
: Lists the Formal Public Identifier (FPI) for the DTD to which this document conforms. The XML parser uses this to find the correct DTD when processing this document.

  `PUBLIC` is not a part of the FPI, but indicates to the XML processor how to find the DTD referenced in the FPI. Other ways of telling the XML parser how to find the DTD are shown later.

`"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"`
: A local filename or a URL to find the DTD.

`>`
: Ends the declaration and returns to the document.

7.3.1. 正式公用識別碼 (FPI)

注意

It is not necessary to know this, but it is useful background, and might help debug problems when the XML processor can not locate the DTD.

FPIs must follow a specific syntax:

```
"Owner//Keyword Description//Language"
```

`Owner`
: The owner of the FPI.

  The beginning of the string identifies the owner of the FPI. For example, the FPI `"ISO 8879:1986//ENTITIES Greek Symbols//EN"` lists `ISO 8879:1986` as being the owner for the set of entities for Greek symbols. ISO 8879:1986 is the International Organization for Standardization (ISO) number for the SGML standard, the predecessor (and a superset) of XML.

  Otherwise, this string will either look like `-//Owner` or `+//Owner` (notice the only difference is the leading + or -).

  If the string starts with `-` then the owner information is unregistered, with a + identifying it as registered.

  ISO 9070:1991 defines how registered names are generated. It might be derived from the number of an ISO publication, an ISBN code, or an organization code assigned according to ISO 6523. Additionally, a registration authority could be created in order to assign registered names. The ISO council delegated this to the American National Standards Institute (ANSI).

  Because the FreeBSD Project has not been registered, the owner string is `-//FreeBSD`. As seen in the example, the W3C are not a registered owner either.

`Keyword`
: There are several keywords that indicate the type of information in the file. Some of the most common keywords are `DTD`, `ELEMENT`, `ENTITIES`, and `TEXT`. `DTD` is used only for DTD files, `ELEMENT` is usually used for DTD fragments that contain only entity or element declarations. `TEXT` is used for XML content (text and tags).

`Description`
: Any description can be given for the contents of this file. This may include version numbers or any short text that is meaningful and unique for the XML system.

Language
An ISO two-character code that identifies the native language for the file. `EN` is used for English.

7.3.1.1. `catalog` 檔案

With the syntax above, an XML processor needs to have some way of turning the FPI into the name of the file containing the DTD. A catalog file (typically called `catalog`) contains lines that map FPIs to filenames. For example, if the catalog file contained the line:

```
PUBLIC  "-//W3C//DTD XHTML 1.0 Transitional//EN"              "1.0/transitional.dtd"
```

The XML processor knows that the DTD is called `transitional.dtd` in the `1.0` subdirectory of the directory that held `catalog`.

Examine the contents of `/usr/local/share/xml/dtd/xhtml/catalog.xml`. This is the catalog file for the XHTML DTDs that were installed as part of the textproc/docproj port.

7.3.2. FPI 的替代方案

Instead of using an FPI to indicate the DTD to which the document conforms (and therefore, which file on the system contains the DTD), the filename can be explicitly specified.

The syntax is slightly different:

```
<!DOCTYPE html SYSTEM "/path/to/file.dtd">
```

The `SYSTEM` keyword indicates that the XML processor should locate the DTD in a system specific fashion. This typically (but not always) means the DTD will be provided as a filename.

Using FPIs is preferred for reasons of portability. If the `SYSTEM` identifier is used, then the DTD must be provided and kept in the same location for everyone.

7.4. 跳脫回 XML

Some of the underlying XML syntax can be useful within documents. For example, comments can be included in the document, and will be ignored by the parser. Comments are entered using XML syntax. Other uses for XML syntax will be shown later.

XML sections begin with a `<!` tag and end with a `>`. These sections contain instructions for the parser rather than elements of the document. Everything between these tags is XML syntax. The DOCTYPE declaration shown earlier is an example of XML syntax included in the document.

7.5. 註解

An XML document may contain comments. They may appear anywhere as long as they are not inside tags. They are even allowed in some locations inside the DTD (e.g., between entity declarations).

XML comments start with the string "`<!--`" and end with the string "`-->`".

Here are some examples of valid XML comments:

範例 7.6. XML 通用註解

```
<!-- This is inside the comment -->
```

```
<!--This is another comment-->

<!-- This is how you
     write multiline comments -->

<p>A simple <!-- Comment inside an element's content --> paragraph.</p>
```

XML comments may contain any strings except "`--`":

範例 7.7. 錯誤的XML 註解

```
<!-- This comment--is wrong -->
```

7.5.1. 待辦事項...

1. Add some comments to `example.xml` , and check that the file still validates using `xmllint` .
2. Add some invalid comments to `example.xml` , and see the error messages that `xmllint` gives when it encounters an invalid comment.

7.6. Entities

Entities are a mechanism for assigning names to chunks of content. As an XML parser processes a document, any entities it finds are replaced by the content of the entity.

This is a good way to have re-usable, easily changeable chunks of content in XML documents. It is also the only way to include one marked up file inside another using XML.

There are two types of entities for two different situations: general entities and parameter entities.

7.6.1. 一般 Entities

General entities are used to assign names to reusable chunks of text. These entities can only be used in the document. They cannot be used in an XML context.

To include the text of a general entity in the document, include `&`*`entity-name`*`;` in the text. For example, consider a general entity called `current.version` which expands to the current version number of a product. To use it in the document, write:

```
<para>The current version of our product is
  &current.version;.</para>
```

When the version number changes, edit the definition of the general entity, replacing the value. Then reprocess the document.

General entities can also be used to enter characters that could not otherwise be included in an XML document. For example, < and & cannot normally appear in an XML document. The XML parser sees the < symbol as the start of a tag. Likewise, when the & symbol is seen, the next text is expected to be an entity name.

These symbols can be included by using two predefined general entities: `<` and `&`.

General entities can only be defined within an XML context. Such definitions are usually done immediately after the DOCTYPE declaration.

範例 7.8. 定義一般 Entities

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!ENTITY current.version    "3.0-RELEASE">
<!ENTITY last.version       "2.2.7-RELEASE">
]>
```

The DOCTYPE declaration has been extended by adding a square bracket at the end of the first line. The two entities are then defined over the next two lines, the square bracket is closed, and then the DOCTYPE declaration is closed.

The square brackets are necessary to indicate that the DTD indicated by the DOCTYPE declaration is being extended.

7.6.2. 參數 Entities

Parameter entities, like general entities, are used to assign names to reusable chunks of text. But parameter entities can only be used within an XML context.

Parameter entity definitions are similar to those for general entities. However, parameter entries are included with `%entity-name ;`. The definition also includes the `%` between the `ENTITY` keyword and the name of the entity.

For a mnemonic, think “Parameter entities use the Percent symbol”.

範例 7.9. 定義參數 Entities

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!ENTITY % param.some "some">
<!ENTITY % param.text "text">
<!ENTITY % param.new  "%param.some more %param.text">

<!-- %param.new now contains "some more text" -->
]>
```

7.6.3. 待辦事項...

1. Add a general entity to `example.xml`.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!ENTITY version "1.1">
]>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>An Example XHTML File</title>
  </head>

  <!-- There may be some comments in here as well -->
```

```
  <body>
    <p>This is a paragraph containing some text.</p>

    <p>This paragraph contains some more text.</p>

    <p align="right"> This paragraph might be right-justified.</p>

    <p>The current version of this document is: &version;</p>
  </body>
</html>
```

2. Validate the document using `xmllint`.

3. Load `example.xml` into a web browser. It may have to be copied to `example.html` before the browser recognizes it as an XHTML document.

   Older browsers with simple parsers may not render this file as expected. The entity reference `&version;` may not be replaced by the version number, or the XML context closing `]>` may not be recognized and instead shown in the output.

4. The solution is to normalize the document with an XML normalizer. The normalizer reads valid XML and writes equally valid XML which has been transformed in some way. One way the normalizer transforms the input is by expanding all the entity references in the document, replacing the entities with the text that they represent.

   `xmllint` can be used for this. It also has an option to drop the initial DTD section so that the closing `]>` does not confuse browsers:

   ```
   % xmllint --noent --dropdtd example.xml > example.html
   ```

   A normalized copy of the document with entities expanded is produced in `example.html`, ready to load into a web browser.

7.7. 在引用檔使用 Entities

Both general and parameter entities are particularly useful for including one file inside another.

7.7.1. 在引用檔使用一般 Entities

Consider some content for an XML book organized into files, one file per chapter, called `chapter1.xml`, `chapter2.xml`, and so forth, with a `book.xml` that will contain these chapters.

In order to use the contents of these files as the values for entities, they are declared with the `SYSTEM` keyword. This directs the XML parser to include the contents of the named file as the value of the entity.

範例 7.10. 在引用檔使用一般 Entities

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!ENTITY chapter.1 SYSTEM "chapter1.xml">
<!ENTITY chapter.2 SYSTEM "chapter2.xml">
<!ENTITY chapter.3 SYSTEM "chapter3.xml">
<!-- And so forth -->
]>

<html xmlns="http://www.w3.org/1999/xhtml">
  <!-- Use the entities to load in the chapters -->
```

```
  &chapter.1;
  &chapter.2;
  &chapter.3;
</html>
```

警告

When using general entities to include other files within a document, the files being included (`chapter1.xml` , `chapter2.xml` , and so on) must not start with a DOCTYPE declaration. This is a syntax error because entities are low-level constructs and they are resolved before any parsing happens.

7.7.2. 在引用檔使用參數 Entities

Parameter entities can only be used inside an XML context. Including a file in an XML context can be used to ensure that general entities are reusable.

Suppose that there are many chapters in the document, and these chapters were reused in two different books, each book organizing the chapters in a different fashion.

The entities could be listed at the top of each book, but that quickly becomes cumbersome to manage.

Instead, place the general entity definitions inside one file, and use a parameter entity to include that file within the document.

範例 7.11. 在引用檔使用參數 Entities

Place the entity definitions in a separate file called `chapters.ent` and containing this text:

```
<!ENTITY chapter.1 SYSTEM "chapter1.xml">
<!ENTITY chapter.2 SYSTEM "chapter2.xml">
<!ENTITY chapter.3 SYSTEM "chapter3.xml">
```

Create a parameter entity to refer to the contents of the file. Then use the parameter entity to load the file into the document, which will then make all the general entities available for use. Then use the general entities as before:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!-- Define a parameter entity to load in the chapter general entities -->
<!ENTITY % chapters SYSTEM "chapters.ent">

<!-- Now use the parameter entity to load in this file -->
%chapters;
]>

<html xmlns="http://www.w3.org/1999/xhtml">
  &chapter.1;
  &chapter.2;
  &chapter.3;
</html>
```

7.7.3. 待辦事項...

7.7.3.1. 在引用檔使用一般 Entities

1. Create three files, `para1.xml` , `para2.xml` , and `para3.xml` .

   Put content like this in each file:

```
<p>This is the first paragraph.</p>
```

2. Edit `example.xml` so that it looks like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!ENTITY version "1.1">
<!ENTITY para1 SYSTEM "para1.xml">
<!ENTITY para2 SYSTEM "para2.xml">
<!ENTITY para3 SYSTEM "para3.xml">
]>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>An Example XHTML File</title>
  </head>

  <body>
    <p>The current version of this document is: &version;</p>

    &para1;
    &para2;
    &para3;
  </body>
</html>
```

3. Produce `example.html` by normalizing `example.xml` .

```
% xmllint --dropdtd --noent example.xml > example.html
```

4. Load `example.html` into the web browser and confirm that the `para`*n*`.xml` files have been included in `example.html` .

7.7.3.2. 在引用檔使用參數 Entities

注意

The previous steps must have completed before this step.

1. Edit `example.xml` so that it looks like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!ENTITY % entities SYSTEM "entities.ent"> %entities;
]>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>An Example XHTML File</title>
  </head>

  <body>
    <p>The current version of this document is: &version;</p>
```

```
    &para1;
    &para2;
    &para3;
  </body>
</html>
```

2. Create a new file called `entities.ent` with this content:

```
<!ENTITY version "1.1">
<!ENTITY para1 SYSTEM "para1.xml">
<!ENTITY para2 SYSTEM "para2.xml">
<!ENTITY para3 SYSTEM "para3.xml">
```

3. Produce `example.html` by normalizing `example.xml`.

```
% xmllint --dropdtd --noent example.xml > example.html
```

4. Load `example.html` into the web browser and confirm that the `para`*n*`.xml` files have been included in `example.html`.

7.8. 已標記小節

XML provides a mechanism to indicate that particular pieces of the document should be processed in a special way. These are called “marked sections”.

範例 7.12. 已標記的結構

```
<![KEYWORD [
  Contents of marked section
]]>
```

As expected of an XML construct, a marked section starts with `<!`.

The first square bracket begins the marked section.

`KEYWORD` describes how this marked section is to be processed by the parser.

The second square bracket indicates the start of the marked section's content.

The marked section is finished by closing the two square brackets, and then returning to the document context from the XML context with `>`.

7.8.1. 已標記關鍵字

7.8.1.1. `CDATA`

These keywords denote the marked sections content model, and allow you to change it from the default.

When an XML parser is processing a document, it keeps track of the “content model”.

The content model describes the content the parser is expecting to see and what it will do with that content.

The `CDATA` content model is one of the most useful.

`CDATA` is for “Character Data”. When the parser is in this content model, it expects to see only characters. In this model the `<` and `&` symbols lose their special status, and will be treated as ordinary characters.

注意

When using `CDATA` in examples of text marked up in XML, remember that the content of `CDATA` is not validated. The included text must be check with other means. For example, the content could be written in another document, validated, and then pasted into the `CDATA` section.

範例 7.13. 使用 `CDATA` 已標記小節

```
<para>Here is an example of how to include some text that contains
  many <literal>&lt;</literal>  and <literal>&amp;</literal>
  symbols.  The sample text is a fragment of
  <acronym>XHTML</acronym>.  The surrounding text (<para> and
  <programlisting> ) are from DocBook.</para>

<programlisting> <![CDATA[<p>This is a sample that shows some of the
  elements within <acronym>XHTML</acronym>.  Since the angle
  brackets are used so many times, it is simpler to say the whole
  example is a CDATA marked section than to use the entity names for
  the left and right angle brackets throughout.</p>

    <ul>
      <li>This is a listitem</li>
      <li>This is a second listitem</li>
      <li>This is a third listitem</li>
    </ul>

    <p>This is the end of the example.</p>]]></programlisting>
```

7.8.1.2. `INCLUDE` 與 `IGNORE`

When the keyword is `INCLUDE`, then the contents of the marked section will be processed. When the keyword is `IGNORE`, the marked section is ignored and will not be processed. It will not appear in the output.

範例 7.14. 在已標記小節中使用 `INCLUDE` 及 `IGNORE`

```
<![INCLUDE[
  This text will be processed and included.
]]>

<![IGNORE[
  This text will not be processed or included.
]]>
```

By itself, this is not too useful. Text to be removed from the document could be cut out, or wrapped in comments.

It becomes more useful when controlled by parameter entities, yet this usage is limited to entity files.

For example, suppose that documentation was produced in a hard-copy version and an electronic version. Some extra text is desired in the electronic version content that was not to appear in the hard-copy.

Create an entity file that defines general entities to include each chapter and guard these definitions with a parameter entity that can be set to either `INCLUDE` or `IGNORE` to control whether the entity is defined. After these conditional general entity definitions, place one more definition for each general entity to set them to an empty value. This technique makes use of the fact that entity definitions cannot be overridden but the first definition always takes effect. So the inclusion of the chapter is controlled with the corresponding parameter entity. Set to `INCLUDE`, the first general entity definition will be read and the second one will be ignored. Set to `IGNORE`, the first definition will be ignored and the second one will take effect.

範例 7.15. 使用參數 Entities 來控制已標記小節

```
<!ENTITY % electronic.copy "INCLUDE">

<![%electronic.copy;[
<!ENTITY chap.preface SYSTEM "preface.xml">
]]>

<!ENTITY chap.preface "">
```

When producing the hard-copy version, change the parameter entity's definition to:

```
<!ENTITY % electronic.copy "IGNORE">
```

7.8.2. 待辦事項...

1. Modify `entities.ent` to contain the following:

```
<!ENTITY version "1.1">
<!ENTITY % conditional.text "IGNORE">

<![%conditional.text;[
<!ENTITY para1 SYSTEM "para1.xml">
]]>

<!ENTITY para1 "">

<!ENTITY para2 SYSTEM "para2.xml">
<!ENTITY para3 SYSTEM "para3.xml">
```

2. Normalize `example.xml` and notice that the conditional text is not present in the output document. Set the parameter entity guard to `INCLUDE` and regenerate the normalized document and the text will appear again. This method makes sense if there are more conditional chunks depending on the same condition. For example, to control generating printed or online text.

7.9. 結論

That is the conclusion of this XML primer. For reasons of space and complexity, several things have not been covered in depth (or at all). However, the previous sections cover enough XML to introduce the organization of the FDP documentation.

章 8. XHTML 標籤

8.1. 簡介

This chapter describes usage of the XHTML markup language used for the FreeBSD web site.

XHTML is the XML version of the HyperText Markup Language, the markup language of choice on the World Wide Web. More information can be found at `http://www.w3.org/` .

XHTML is used to mark up pages on the FreeBSD web site. It is usually not used to mark up other documentation, since DocBook offers a far richer set of elements from which to choose. Consequently, XHTML pages will normally only be encountered when writing for the web site.

HTML has gone through a number of versions. The XML-compliant version described here is called XHTML. The latest widespread version is XHTML 1.0, available in both strict and transitional variants.

The XHTML DTDs are available from the Ports Collection in textproc/xhtml. They are automatically installed by the textproc/docproj port.

注意

This is not an exhaustive list of elements, since that would just repeat the documentation for XHTML. The aim is to list those elements most commonly used. Please post questions about elements or uses not covered here to the FreeBSD documentation project mailing list.

Inline Versus Block

In the remainder of this document, when describing elements, inline means that the element can occur within a block element, and does not cause a line break. A block element, by comparison, will cause a line break (and other processing) when it is encountered.

8.2. 正式公用識別碼 (FPI)

There are a number of XHTML FPIs, depending upon the version, or level of XHTML to which a document conforms. Most XHTML documents on the FreeBSD web site comply with the transitional version of XHTML 1.0.

```
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

8.3. 分節元素

An XHTML document is normally split into two sections. The first section, called the head, contains meta-information about the document, such as its title, the name of the author, the parent document, and so on. The second section, the body, contains content that will be displayed to the user.

These sections are indicated with `head` and `body` elements respectively. These elements are contained within the top-level `html` element.

範例 8.1. 一般的 XHTML 文件結構

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
   <title>The Document's Title </title>
  </head>

  <body>

    …

  </body>
</html>
```

8.4. 區塊元素

8.4.1. 標題

XHTML has tags to denote headings in the document at up to six different levels.

The largest and most prominent heading is `h1`, then `h2`, continuing down to `h6`.

The element's content is the text of the heading.

範例 8.2. `h1`, `h2`, 以及其他標題標籤

用法：

```
<h1>First section</h1>

<!-- Document introduction goes here -->

<h2>This is the heading for the first section</h2>

<!-- Content for the first section goes here -->

<h3>This is the heading for the first sub-section</h3>

<!-- Content for the first sub-section goes here -->

<h2>This is the heading for the second section</h2>

<!-- Content for the second section goes here -->
```

Generally, an XHTML page should have one first level heading (`h1`). This can contain many second level headings (`h2`), which can in turn contain many third level headings. Do not leave gaps in the numbering.

8.4.2. 段落

XHTML supports a single paragraph element, `p`.

範例 8.3. `p` 範例

用法：

```
<p>This is a paragraph.  It can contain just about any
  other element.</p>
```

8.4.3. 區塊引言

A block quotation is an extended quotation from another document that will appear in a separate paragraph.

範例 8.4. `blockquote` 範例

用法：

```
<p>A small excerpt from the US Constitution:</p>

<blockquote> We the People of the United States, in Order to form
  a more perfect Union, establish Justice, insure domestic
  Tranquility, provide for the common defence, promote the general
  Welfare, and secure the Blessings of Liberty to ourselves and our
  Posterity, do ordain and establish this Constitution for the
  United States of America.</blockquote>
```

8.4.4. 清單

XHTML can present the user with three types of lists: ordered, unordered, and definition.

Entries in an ordered list will be numbered, while entries in an unordered list will be preceded by bullet points. Definition lists have two sections for each entry. The first section is the term being defined, and the second section is the definition.

Ordered lists are indicated by the `ol` element, unordered lists by the `ul` element, and definition lists by the `dl` element.

Ordered and unordered lists contain listitems, indicated by the `li` element. A listitem can contain textual content, or it may be further wrapped in one or more `p` elements.

Definition lists contain definition terms (`dt`) and definition descriptions (`dd`). A definition term can only contain inline elements. A definition description can contain other block elements.

範例 8.5. `ul` 與 `ol` 範例

用法：

```
<p>An unordered list.  Listitems will probably be
  preceded by bullets.</p>

<ul>
```

```
  <li>First item</li>

  <li>Second item</li>

  <li>Third item</li>
</ul>

<p>An ordered list, with list items consisting of multiple
  paragraphs.  Each item (note: not each paragraph) will be
  numbered.</p>

<ol>
  <li><p>This is the first item.  It only has one paragraph.</p></li>

  <li><p>This is the first paragraph of the second item.</p>

    <p>This is the second paragraph of the second item.</p></li>

  <li><p>This is the first and only paragraph of the third
    item.</p></li>
</ol>
```

範例 8.6. 使用 dl 列定義清單

用法：

```
<dl>
  <dt>Term 1</dt>

  <dd><p>Paragraph 1 of definition 1.</p>

    <p>Paragraph 2 of definition 1.</p></dd>

  <dt>Term 2</dt>

  <dd><p>Paragraph 1 of definition 2.</p></dd>

  <dt>Term 3</dt>

  <dd><p>Paragraph 1 of definition 3.</p></dd>
</dl>
```

8.4.5. 已預排文字

Pre-formatted text is shown to the user exactly as it is in the file. Text is shown in a fixed font. Multiple spaces and line breaks are shown exactly as they are in the file.

Wrap pre-formatted text in the pre element.

範例 8.7. pre 範例

For example, the pre tags could be used to mark up an email message:

```
<pre>  From: nik@FreeBSD.org
  To: freebsd-doc@FreeBSD.org
```

```
  Subject: New documentation available

  There is a new copy of my primer for contributors to the FreeBSD
  Documentation Project available at

    &lt;URL:http://people.FreeBSD.org/~nik/primer/index.html&gt;

  Comments appreciated.

  N</pre>
```

Keep in mind that < and & still are recognized as special characters in pre-formatted text. This is why the example shown had to use `<` instead of <. For consistency, `>` was used in place of >, too. Watch out for the special characters that may appear in text copied from a plain-text source, like an email message or program code.

8.4.6. 表格

Mark up tabular information using the `table` element. A table consists of one or more table rows (`tr`), each containing one or more cells of table data (`td`). Each cell can contain other block elements, such as paragraphs or lists. It can also contain another table (this nesting can repeat indefinitely). If the cell only contains one paragraph then the pelement is not needed.

範例 8.8. `table` 的簡單用法

用法：

```
<p>This is a simple 2x2 table.</p>

<table>
  <tr>
    <td>Top left cell</td>

    <td>Top right cell</td>
  </tr>

  <tr>
    <td>Bottom left cell</td>

    <td>Bottom right cell</td>
  </tr>
</table>
```

A cell can span multiple rows and columns by adding the `rowspan` or `colspan` attributes with values for the number of rows or columns to be spanned.

範例 8.9. 使用 `rowspan`

用法：

```
<p>One tall thin cell on the left, two short cells next to
  it on the right.</p>

<table>
```

```
  <tr>
    <td rowspan="2"> Long and thin</td>
  </tr>

  <tr>
    <td>Top cell</td>

    <td>Bottom cell</td>
  </tr>
</table>
```

範例 8.10. 使用 **colspan**

用法：

```
<p>One long cell on top, two short cells below it.</p>

<table>
  <tr>
    <td colspan="2"> Top cell</td>
  </tr>

  <tr>
    <td>Bottom left cell</td>

    <td>Bottom right cell</td>
  </tr>
</table>
```

範例 8.11. **rowspan** 與 **colspan** 一起使用

用法：

```
<p>On a 3x3 grid, the top left block is a 2x2 set of
  cells merged into one.  The other cells are normal.</p>

<table>
  <tr>
    <td colspan="2" rowspan="2"> Top left large cell</td>

    <td>Top right cell</td>
  </tr>

  <tr>
    <!-- Because the large cell on the left merges into
         this row, the first <td> will occur on its
         right -->

    <td>Middle right cell</td>
  </tr>

  <tr>
    <td>Bottom left cell</td>

    <td>Bottom middle cell</td>
```

```
    <td>Bottom right cell</td>
  </tr>
</table>
```

8.5. 行內元素

8.5.1. 強調資訊

Two levels of emphasis are available in XHTML, `em` and `strong`. `em` is for a normal level of emphasis and `strong` indicates stronger emphasis.

`em` is typically rendered in italic and `strong` is rendered in bold. This is not always the case, and should not be relied upon. According to best practices, web pages only hold structural and semantical information, and stylesheets are later applied to them. Think of semantics, not formatting, when using these tags.

範例 8.12. `em` 與 `strong` 範例

用法：

```
<p><em>This</em> has been emphasized, while
  <strong> this</strong>  has been strongly emphasized.</p>
```

8.5.2. 標示等寬文字

Content that should be rendered in a fixed pitch (typewriter) typeface is tagged with `tt` (for “teletype”).

範例 8.13. `tt` 範例

用法：

```
<p>Many system settings are stored in
  <tt>/etc</tt>.</p>
```

8.5.3. 連結

注意

Links are also inline elements.

8.5.3.1. 連結在網站上的其他文件

A link points to the URL of a document on the web. The link is indicated with `a`, and the `href` attribute contains the URL of the target document. The content of the element becomes the link, indicated to the user by showing it in a different color or with an underline.

範例 8.14. 使用 <a href="...">

用法：

```
<p>More information is available at the
  <a href="http://www.&os;.org/">  &os; web site</a>.</p>
```

This link always takes the user to the top of the linked document.

8.5.3.2. 連結說明文件的特定章節

To link to a specific point within a document, that document must include an anchor at the desired point. Anchors are included by setting the `id` attribute of an element to a name. This example creates an anchor by setting the `id` attribute of a `p` element.

範例 8.15. 建立錨點

用法：

```
<p id="samplepara"> This paragraph can be referenced
  in other links with the name <tt>samplepara</tt>.</p>
```

Links to anchors are similar to plain links, but include a # symbol and the anchor's ID at the end of the URL.

範例 8.16. 連結到另一份文件中已命名的段落

The `samplepara` example is part of a document called `foo.html` . A link to that specific paragraph in the document is constructed in this example.

```
<p>More information can be found in the
  <a href="foo.html#samplepara">  sample paragraph</a> of
  <tt>foo.html</tt>.</p>
```

To link to a named anchor within the same document, omit the document's URL, and just use the # symbol followed by the name of the anchor.

範例 8.17. 連結到同一份文件已命名的段落

The `samplepara` example resides in this document. To link to it:

```
<p>More information can be found in the
  <a href="#samplepara"> sample paragraph</a> of this
  document.</p>
```

章 9. DocBook 標籤

9.1. 簡介

This chapter is an introduction to DocBook as it is used for FreeBSD documentation. DocBook is a large and complex markup system, but the subset described here covers the parts that are most widely used for FreeBSD documentation. While a moderate subset is covered, it is impossible to anticipate every situation. Please post questions that this document does not answer to the FreeBSD documentation project mailing list.

DocBook was originally developed by HaL Computer Systems and O'Reilly & Associates to be a Document Type Definition (DTD) for writing technical documentation [1]. Since 1998 it is maintained by the DocBook Technical Committee. As such, and unlike LinuxDoc and XHTML, DocBook is very heavily oriented towards markup that describes what something is, rather than describing how it should be presented.

The DocBook DTD is available from the Ports Collection in the textproc/docbook-xml port. It is automatically installed as part of the textproc/docproj port.

Formal Versus Informal

Some elements may exist in two forms, formal and informal. Typically, the formal version of the element will consist of a title followed by the informal version of the element. The informal version will not have a title.

Inline Versus Block

In the remainder of this document, when describing elements, inline means that the element can occur within a block element, and does not cause a line break. A block element, by comparison, will cause a line break (and other processing) when it is encountered.

9.2. FreeBSD 擴充項目

The FreeBSD Documentation Project has extended the DocBook DTD with additional elements and entities. These additions serve to make some of the markup easier or more precise.

Throughout the rest of this document, the term “DocBook” is used to mean the FreeBSD-extended DocBook DTD.

注意

Most of these extensions are not unique to FreeBSD, it was just felt that they were useful enhancements for this particular project. Should anyone from any of the other *nix camps (NetBSD, OpenBSD, Linux, ...) be interested in collaborating on a standard DocBook extension set, please contact Documentation Engineering Team <doceng@FreeBSD.org >.

[1] A short history can be found under http://www.oasis-open.org/docbook/intro.shtml#d0e41.

9.2.1. FreeBSD 元素

The additional FreeBSD elements are not (currently) in the Ports Collection. They are stored in the FreeBSD Subversion tree, as head/share/xml/freebsd.dtd.

FreeBSD-specific elements used in the examples below are clearly marked.

9.2.2. FreeBSD Entities

This table shows some of the most useful entities available in the FDP. For a complete list, see the `*.ent` files in `doc/share/xml` .

FreeBSD Name Entities		
`&os;`	FreeBSD	
`&os.stable;`	FreeBSD-STABLE	
`&os.current;`	FreeBSD-CURRENT	
Manual Page Entities		
`&man.ls.1;`	ls(1)	Usage: `&man.ls.1;` is the manual page for `<command>ls</command>`.
`&man.cp.1;`	cp(1)	Usage: The manual page for `<command>cp</command>` is `&man.cp.1;`.
`&man.`*`command`*`.`*`sectionnumber`*`;`	link to ***`command`*** manual page in section ***`sectionnumber`***	Entities are defined for all the FreeBSD manual pages.
FreeBSD Mailing List Entities		
`&a.doc;`	FreeBSD documentation project mailing list	Usage: `A link to the &a.doc;.`
`&a.questions;`	FreeBSD general questions mailing list	Usage: `A link to the &a.questions;.`
`&a.`*`listname`*`;`	link to ***`listname`***	Entities are defined for all the FreeBSD mailing lists.
FreeBSD Document Link Entities		
`&url.books.handbook;`	`https://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook`	Usage: `A link to the <link xlink:href="&url.books.handbook;/advanced-networking.html">Advanced Networking</link> chapter of the Handbook.`
`&url.books.`*`bookname`*`;`	relative path to ***`bookname`***	Entities are defined for all the FreeBSD books.
`&url.articles.committers-guide;`	`https://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/committers-guide`	Usage: `A link to the <link xlink:href="&url.articles.committers-`

		guide;">Committer's Guide</link> article.
&url.articles. *articlename* ;	relative path to ***articlename***	Entities are defined for all the FreeBSD articles.
Other Operating System Name Entities		
&linux;	Linux®	The Linux® operating system.
&unix;	UNIX®	The UNIX® operating system.
&windows;	Windows®	The Windows® operating system.
Miscellaneous Entities		
&prompt.root;	#	The root user prompt.
&prompt.user;	%	A prompt for an unprivileged user.
&postscript;	PostScript®	The PostScript® programming language.
&tex;	TeX	The TeX typesetting language.
&xorg;	Xorg	The Xorg open source X Window System.

9.3. 正式公用識別碼 (FPI)

In compliance with the DocBook guidelines for writing FPIs for DocBook customizations, the FPI for the FreeBSD extended DocBook DTD is:

```
PUBLIC "-//FreeBSD//DTD DocBook V4.2-Based Extension//EN"
```

9.4. 文件結構

DocBook allows structuring documentation in several ways. The FreeBSD Documentation Project uses two primary types of DocBook document: the book and the article.

Books are organized into `chapter`s. This is a mandatory requirement. There may be `part`s between the book and the chapter to provide another layer of organization. For example, the Handbook is arranged in this way.

A chapter may (or may not) contain one or more sections. These are indicated with the `sect1` element. If a section contains another section then use the `sect2` element, and so on, up to `sect5`.

Chapters and sections contain the remainder of the content.

An article is simpler than a book, and does not use chapters. Instead, the content of an article is organized into one or more sections, using the same `sect1` (and `sect2` and so on) elements that are used in books.

The nature of the document being written should be used to determine whether it is best marked up as a book or an article. Articles are well suited to information that does not need to be broken down into several chapters, and that is, relatively speaking, quite short, at up to 20-25 pages of content. Books are best suited to information that can be broken up into several chapters, possibly with appendices and similar content as well.

The FreeBSD tutorials are all marked up as articles, while this document, the FAQ, and the Handbook are all marked up as books, for example.

9.4.1. 開始撰寫書籍

The content of a book is contained within the `book` element. As well as containing structural markup, this element can contain elements that include additional information about the book. This is either meta-information, used for reference purposes, or additional content used to produce a title page.

This additional information is contained within `info`.

範例 9.1. 使用 **info** 的 **book** 樣板

```
<book>
  <info>
    <title>Your Title Here</title>

    <author>
      <personname>
        <firstname>Your first name</firstname>
        <surname>Your surname</surname>
      </personname>

      <affiliation>
 <address>
          <email>Your email address</email>
 </address>
      </affiliation>
    </author>

    <copyright>
      <year>1998</year>
      <holder role="mailto: your email address ">Your name</holder>
    </copyright>

    <releaseinfo> $FreeBSD$</releaseinfo>

    <abstract>
      <para>Include an abstract of the book's contents here.  </para>
    </abstract>
  </info>

  …

</book>
```

9.4.2. 開始撰寫文章

The content of the article is contained within the `article` element. As well as containing structural markup, this element can contain elements that include additional information about the article. This is either meta-information, used for reference purposes, or additional content used to produce a title page.

This additional information is contained within `info`.

範例 9.2. 使用 **info** 的 **article** 樣板

```
<article>
  <info>
    <title>Your title here</title>
```

```
    <author>
      <personname>
 <firstname> Your first name </firstname>
 <surname> Your surname </surname>
      </personname>

      <affiliation>
 <address>
   <email> Your email address </email> </address>
 </address>
      </affiliation>
    </author>

    <copyright>
      <year> 1998 </year>
      <holder role="mailto: your email address ">Your name </holder>
    </copyright>

    <releaseinfo> $FreeBSD$</releaseinfo>

    <abstract>
      <para> Include an abstract of the article's contents here.  </para>
    </abstract>
  </info>

  …

</article>
```

9.4.3. 標示章節

Use `chapter` to mark up your chapters. Each chapter has a mandatory `title`. Articles do not contain chapters, they are reserved for books.

範例 9.3. 簡單的章節

```
<chapter>
  <title> The Chapter's Title</title>

  ...
</chapter>
```

A chapter cannot be empty; it must contain elements in addition to `title`. If you need to include an empty chapter then just use an empty paragraph.

範例 9.4. 空白章節

```
<chapter>
  <title> This is An Empty Chapter</title>

  <para></para>
</chapter>
```

9.4.4. 章底下的小節

In books, chapters may (but do not need to) be broken up into sections, subsections, and so on. In articles, sections are the main structural element, and each article must contain at least one section. Use the `sect`*n* element. The *n* indicates the section number, which identifies the section level.

The first `sect`*n* is `sect1`. You can have one or more of these in a chapter. They can contain one or more `sect2` elements, and so on, down to `sect5`.

範例 9.5. 章中的小節

```
<chapter>
  <title>A Sample Chapter</title>

  <para>Some text in the chapter.</para>

  <sect1>
    <title>First Section</title>

    …
  </sect1>

  <sect1>
    <title>Second Section</title>

    <sect2>
      <title>First Sub-Section</title>

      <sect3>
 <title>First Sub-Sub-Section</title>

 …
      </sect3>
    </sect2>

    <sect2>
      <title>Second Sub-Section (1.2.2)</title>

      …
    </sect2>
  </sect1>
</chapter>
```

注意

Section numbers are automatically generated and prepended to titles when the document is rendered to an output format. The generated section numbers and titles from the example above will be:

- 1.1. First Section
- 1.2. Second Section
- 1.2.1. First Sub-Section
- 1.2.1.1. First Sub-Sub-Section

- 1.2.2. Second Sub-Section

9.4.5. 使用 part 元素來分部

parts introduce another level of organization between book and chapter with one or more parts. This cannot be done in an article.

```
<part>
  <title>Introduction</title>

  <chapter>
    <title>Overview</title>

    ...
  </chapter>

  <chapter>
    <title>What is FreeBSD?</title>

    ...
  </chapter>

  <chapter>
    <title>History</title>

    ...
  </chapter>
</part>
```

9.5. 區塊元素

9.5.1. 段落

DocBook supports three types of paragraphs: formalpara , para, and simpara.

Almost all paragraphs in FreeBSD documentation use para. formalpara includes a title element, and simpara disallows some elements from within para. Stick with para.

範例 9.6. para 範例

用法：

```
<para>This is a paragraph.  It can contain just about any
  other element.</para>
```

輸出結果：

This is a paragraph. It can contain just about any other element.

9.5.2. 區塊引言

A block quotation is an extended quotation from another document that should not appear within the current paragraph. These are rarely needed.

Blockquotes can optionally contain a title and an attribution (or they can be left untitled and unattributed).

範例 9.7. `blockquote` 範例

用法：

```
<para>A small excerpt from the US Constitution:</para>

<blockquote>
  <title>Preamble to the Constitution of the United States</title>

  <attribution>Copied from a web site somewhere</attribution>

  <para>We the People of the United States, in Order to form a more
    perfect Union, establish Justice, insure domestic Tranquility,
    provide for the common defence, promote the general Welfare, and
    secure the Blessings of Liberty to ourselves and our Posterity, do
    ordain and establish this Constitution for the United States of
    America.</para>
</blockquote>
```

輸出結果：

A small excerpt from the US Constitution:

> Preamble to the Constitution of the United States
>
> We the People of the United States, in Order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common defence, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our Posterity, do ordain and establish this Constitution for the United States of America.
>
> —Copied from a web site somewhere

9.5.3. 提示、注意、警告、注意事項及重要資訊

Extra information may need to be separated from the main body of the text. Typically this is “meta” information of which the user should be aware.

Several types of admonitions are available: `tip`, `note`, `warning`, `caution`, and `important`.

Which admonition to choose depends on the situation. The DocBook documentation suggests:

- Note is for information that should be heeded by all readers.
- Important is a variation on Note.
- Caution is for information regarding possible data loss or software damage.
- Warning is for information regarding possible hardware damage or injury to life or limb.

範例 9.8. `tip` 與 `important` 範例

用法：

```
<tip>
  <para>&os; may reduce stress.</para>
</tip>
```

```
<important>
  <para>Please use admonitions sparingly.  Too many admonitions
    are visually jarring and can have the opposite of the
    intended effect.</para>
</important>
```

輸出結果：

提示

FreeBSD may reduce stress.

重要

Please use admonitions sparingly. Too many admonitions are visually jarring and can have the opposite of the intended effect.

9.5.4. 範例

Examples can be shown with `example`.

範例 9.9. **example** 原始碼

用法：

```
<example>
  <para>Empty files can be created easily:</para>

  <screen>&prompt.user; <userinput>touch file1 file2 file3</userinput></screen>
</example>
```

輸出結果：

範例 9.10. **example** 的結果

Empty files can be created easily:

```
% touch file1 file2 file3
```

9.5.5. 清單與步驟

Information often needs to be presented as lists, or as a number of steps that must be carried out in order to accomplish a particular goal.

To do this, use `itemizedlist` , `orderedlist` , `variablelist` , or `procedure` . There are other types of list elements in DocBook, but we will not cover them here.

`itemizedlist` and `orderedlist` are similar to their counterparts in HTML, `ul` and `ol`. Each one consists of one or more `listitem` elements, and each `listitem` contains one or more block elements. The `listitem` elements are analogous to HTML's `li` tags. However, unlike HTML, they are required.

範例 9.11. `itemizedlist` 與 `orderedlist` 範例

用法：

```
<itemizedlist>
  <listitem>
    <para>This is the first itemized item.</para>
  </listitem>

  <listitem>
    <para>This is the second itemized item.</para>
  </listitem>
</itemizedlist>

<orderedlist>
  <listitem>
    <para>This is the first ordered item.</para>
  </listitem>

  <listitem>
    <para>This is the second ordered item.</para>
  </listitem>
</orderedlist>
```

輸出結果：

- This is the first itemized item.
- This is the second itemized item.

1. This is the first ordered item.
2. This is the second ordered item.

An alternate and often useful way of presenting information is the `variablelist` . These are lists where each entry has a term and a description. They are well suited for many types of descriptions, and present information in a form that is often easier for the reader than sections and subsections.

A `variablelist` has a `title`, and then pairs of `term` and `listitem` entries.

範例 9.12. `variablelist` 範例

用法：

```
<variablelist>
  <varlistentry>
    <term>Parallel</term>

    <listitem>
```

```
      <para>In parallel communications, groups of bits arrive
 at the same time over multiple communications
 channels.</para>
    </listitem>
  </varlistentry>

  <varlistentry>
    <term>Serial</term>

    <listitem>
      <para>In serial communications, bits arrive one at a
 time over a single communications
 channel.</para>
    </listitem>
  </varlistentry>
</variablelist>
```

輸出結果：

Parallel
: In parallel communications, groups of bits arrive at the same time over multiple communications channels.

Serial
: In serial communications, bits arrive one at a time over a single communications channel.

A `procedure` shows a series of `step`s, which may in turn consist of more `step`s or `substep`s. Each `step` contains block elements and may include an optional title.

Sometimes, steps are not sequential, but present a choice: do this or do that, but not both. For these alternative choices, use `stepalternatives` .

範例 9.13. `procedure` 範例

用法：

```
<procedure>
  <step>
    <para>Do this.</para>
  </step>

  <step>
    <para>Then do this.</para>
  </step>

  <step>
    <para>And now do this.</para>
  </step>

  <step>
    <para>Finally, do one of these.</para>

    <stepalternatives>
      <step>
 <para>Go left.</para>
      </step>

      <step>
 <para>Go right.</para>
```

```
      </step>
    </stepalternatives>
  </step>
</procedure>
```

輸出結果：

1. Do this.
2. Then do this.
3. And now do this.
4. Finally, do one of these:
   - Go left.
   - Go right.

9.5.6. 顯示檔案範本

Fragments of a file (or perhaps a complete file) are shown by wrapping them in the `programlisting` element.

White space and line breaks within `programlisting` are significant. In particular, this means that the opening tag should appear on the same line as the first line of the output, and the closing tag should appear on the same line as the last line of the output, otherwise spurious blank lines may be included.

範例 9.14. `programlisting` 範例

用法：

```
<para>When finished, the program will look like
  this:</para>

<programlisting> #include &lt;stdio.h&gt;

int
main(void)
{
    printf("hello, world\n");
}</programlisting>
```

Notice how the angle brackets in the `#include` line need to be referenced by their entities instead of being included literally.

輸出結果：

When finished, the program will look like this:

```
#include <stdio.h>

int
main(void)
{
    printf("hello, world\n");
}
```

9.5.7. 標註

A callout is a visual marker for referring to a piece of text or specific position within an example.

Callouts are marked with the `co` element. Each element must have a unique `id` assigned to it. After the example, include a `calloutlist` that describes each callout.

範例 9.15. `co` 與 `calloutlist` 範例

```
<para>When finished, the program will look like
  this:</para>

<programlisting> #include &lt;stdio.h&gt; <co xml:id="co-ex-include"/>

int <co xml:id="co-ex-return"/>
main(void)
{
    printf("hello, world\n"); <co xml:id="co-ex-printf"/>
}</programlisting>

<calloutlist>
  <callout arearefs="co-ex-include">
    <para>Includes the standard IO header file.</para>
  </callout>

  <callout arearefs="co-ex-return">
    <para>Specifies that <function>main()</function> returns an
      int.</para>
  </callout>

  <callout arearefs="co-ex-printf">
    <para>The <function>printf()</function> call that writes
      <literal>hello, world</literal> to standard output.</para>
  </callout>
</calloutlist>
```

輸出結果：

When finished, the program will look like this:

```
#include <stdio.h> ❶

int ❷
main(void)
{
    printf("hello, world\n"); ❸
}
```

❶ Includes the standard IO header file.
❷ Specifies that `main()` returns an int.
❸ The `printf()` call that writes `hello, world` to standard output.

9.5.8. 表格

Unlike HTML, DocBook does not need tables for layout purposes, as the stylesheet handles those issues. Instead, just use tables for marking up tabular data.

In general terms (and see the DocBook documentation for more detail) a table (which can be either formal or informal) consists of a `table` element. This contains at least one `tgroup` element, which specifies (as an attribute)

the number of columns in this table group. Within the tablegroup there is one `thead` element, which contains elements for the table headings (column headings), and one `tbody` which contains the body of the table.

Both `tgroup` and `thead` contain `row` elements, which in turn contain `entry` elements. Each `entry` element specifies one cell in the table.

範例 9.16. `informaltable` 範例

用法：

```
<informaltable pgwide="1">
  <tgroup cols="2">
    <thead>
      <row>
        <entry>This is Column Head 1</entry>
        <entry>This is Column Head 2</entry>
      </row>
    </thead>

    <tbody>
      <row>
 <entry>Row 1, column 1</entry>
 <entry>Row 1, column 2</entry>
      </row>

      <row>
 <entry>Row 2, column 1</entry>
 <entry>Row 2, column 2</entry>
      </row>
    </tbody>
  </tgroup>
</informaltable>
```

輸出結果：

This is Column Head 1	This is Column Head 2
Row 1, column 1	Row 1, column 2
Row 2, column 1	Row 2, column 2

Always use the `pgwide` attribute with a value of `1` with the `informaltable` element. A bug in Internet Explorer can cause the table to render incorrectly if this is omitted.

Table borders can be suppressed by setting the `frame` attribute to `none` in the `informaltable` element. For example, `informaltable frame="none"` .

範例 9.17. 表格使用 `frame="none"` 範例

輸出結果：

This is Column Head 1	This is Column Head 2
Row 1, column 1	Row 1, column 2

This is Column Head 1	This is Column Head 2
Row 2, column 1	Row 2, column 2

9.5.9. 給使用者遵循的範例

Examples for the user to follow are often necessary. Typically, these will consist of dialogs with the computer; the user types in a command, the user gets a response back, the user types another command, and so on.

A number of distinct elements and entities come into play here.

`screen`
: Everything the user sees in this example will be on the computer screen, so the next element is `screen`.

  Within `screen`, white space is significant.

`prompt`, `&prompt.root;` and `&prompt.user;`
: Some of the things the user will be seeing on the screen are prompts from the computer (either from the operating system, command shell, or application). These should be marked up using `prompt`.

  As a special case, the two shell prompts for the normal user and the root user have been provided as entities. To indicate the user is at a shell prompt, use one of `&prompt.root;` and `&prompt.user;` as necessary. They do not need to be inside `prompt`.

注意

`&prompt.root;` and `&prompt.user;` are FreeBSD extensions to DocBook, and are not part of the original DTD.

`userinput`
: When displaying text that the user should type in, wrap it in `userinput` tags. It will be displayed differently than system output text.

範例 9.18. `screen`, `prompt` 與 `userinput` 範例

用法：

```
<screen>&prompt.user; <userinput>ls -1</userinput>
foo1
foo2
foo3
&prompt.user; <userinput>ls -1 | grep foo2</userinput>
foo2
&prompt.user; <userinput>su</userinput>
<prompt>Password: </prompt>
&prompt.root; <userinput>cat foo2</userinput>
This is the file called 'foo2'</screen>
```

輸出結果：

```
% ls -1
foo1
foo2
```

```
foo3
% ls -1 | grep foo2
foo2
% su
Password:
# cat foo2
This is the file called 'foo2'
```

注意

Even though we are displaying the contents of the file `foo2`, it is not marked up as `programlisting` . Reserve `programlisting` for showing fragments of files outside the context of user actions.

9.6. 行內元素

9.6.1. 強調資訊

To emphasize a particular word or phrase, use `emphasis` . This may be presented as italic, or bold, or might be spoken differently with a text-to-speech system.

There is no way to change the presentation of the emphasis within the document, no equivalent of HTML's `b` and `i`. If the information being presented is important, then consider presenting it in `important` rather than `emphasis` .

範例 9.19. `emphasis` 範例

用法：

```
<para>&os; is without doubt <emphasis> the</emphasis>
  premiere &unix;-like operating system for the Intel
  architecture.</para>
```

輸出結果：

FreeBSD is without doubt the premiere UNIX®-like operating system for the Intel architecture.

9.6.2. 縮寫

Many computer terms are acronyms, words formed from the first letter of each word in a phrase. Acronyms are marked up into `acronym` elements. It is helpful to the reader when an acronym is defined on the first use, as shown in the example below.

範例 9.20. `acronym` 範例

用法：

```
<para>Request For Comments (<acronym> RFC</acronym> ) 1149
  defined the use of avian carriers for transmission of
```

```
  Internet Protocol (<acronym> IP</acronym> ) data.  The
  quantity of <acronym> IP</acronym>  data currently
  transmitted in that manner is unknown.</para>
```

輸出結果：

Request For Comments (RFC) 1149 defined the use of avian carriers for transmission of Internet Protocol (IP) data. The quantity of IP data currently transmitted in that manner is unknown.

9.6.3. 引言

To quote text from another document or source, or to denote a phrase that is used figuratively, use `quote`. Most of the markup tags available for normal text are also available from within a `quote`.

範例 9.21. `quote` 範例

用法：

```
<para>However, make sure that the search does not go beyond the
  <quote> boundary between local and public administration</quote> ,
  as <acronym> RFC</acronym>  1535 calls it.</para>
```

輸出結果：

However, make sure that the search does not go beyond the “boundary between local and public administration”, as RFC 1535 calls it.

9.6.4. 鍵盤按鍵、滑鼠按鍵及組合鍵

To refer to a specific key on the keyboard, use `keycap`. To refer to a mouse button, use `mousebutton` . And to refer to combinations of key presses or mouse clicks, wrap them all in `keycombo` .

`keycombo` has an attribute called `action`, which may be one of `click`, `double-click` , `other`, `press`, `seq`, or `simul`. The last two values denote whether the keys or buttons should be pressed in sequence, or simultaneously.

The stylesheets automatically add any connecting symbols, such as `+`, between the key names, when wrapped in `keycombo` .

範例 9.22. 鍵盤按鍵、滑鼠案件及組合鍵範例

用法：

```
<para>To switch to the second virtual terminal, press
  <keycombo action="simul"> <keycap> Alt</keycap>
    <keycap> F1</keycap> </keycombo> .</para>

<para>To exit <command> vi</command>  without saving changes, type
  <keycombo action="seq"> <keycap> Esc</keycap> <keycap> :</keycap>
    <keycap> q</keycap> <keycap> !</keycap> </keycombo> .</para>

<para>My window manager is configured so that
  <keycombo action="simul"> <keycap> Alt</keycap>
    <mousebutton> right</mousebutton>
  </keycombo>  mouse button is used to move windows.</para>
```

輸出結果：

To switch to the second virtual terminal, press Alt+F1.

To exit `vi` without saving changes, type Esc : q !.

My window manager is configured so that Alt+right mouse button is used to move windows.

9.6.5. 應用程式、指令、選項與引用

Both applications and commands are frequently referred to when writing documentation. The distinction between them is that an application is the name of a program or suite of programs that fulfill a particular task. A command is the filename of a program that the user can type and run at a command line.

It is often necessary to show some of the options that a command might take.

Finally, it is often useful to list a command with its manual section number, in the "command(number)" format so common in Unix manuals.

Mark up application names with `application` .

To list a command with its manual section number (which should be most of the time) the DocBook element is `citerefentry` . This will contain a further two elements, `refentrytitle` and `manvolnum` . The content of `refentrytitle` is the name of the command, and the content of `manvolnum` is the manual page section.

This can be cumbersome to write, and so a series of general entities have been created to make this easier. Each entity takes the form `&man.`*`manual-page`*`.`*`manual-section`*`;`.

The file that contains these entities is in `doc/share/xml/man-refs.ent` , and can be referred to using this FPI:

```
PUBLIC "-//FreeBSD//ENTITIES DocBook Manual Page Entities//EN"
```

Therefore, the introduction to FreeBSD documentation will usually include this:

```
<!DOCTYPE book PUBLIC "-//FreeBSD//DTD DocBook V4.1-Based Extension//EN" [

<!ENTITY % man PUBLIC "-//FreeBSD//ENTITIES DocBook Manual Page Entities//EN">
%man;

…

]>
```

Use `command` to include a command name "in-line" but present it as something the user should type.

Use `option` to mark up the options which will be passed to a command.

When referring to the same command multiple times in close proximity, it is preferred to use the `&man.`*`command`*`.`*`section`*`;` notation to markup the first reference and use `command` to markup subsequent references. This makes the generated output, especially HTML, appear visually better.

範例 9.23. 應用程式、指令、選項範例

用法：

```
<para><application> Sendmail</application>  is the most
  widely used Unix mail application.<para>
```

```
<para><application> Sendmail</application>  includes the
  <citerefentry>
    <refentrytitle> sendmail</refentrytitle>
    <manvolnum> 8</manvolnum>
  </citerefentry> , &man.mailq.1;, and &man.newaliases.1;
  programs.</para>

<para>One of the command line parameters to <citerefentry>
    <refentrytitle> sendmail</refentrytitle>
    <manvolnum> 8</manvolnum>
  </citerefentry> , <option> -bp</option> , will display the current
  status of messages in the mail queue.  Check this on the command
  line by running <command> sendmail -bp</command> .</para>
```

輸出結果：

Sendmail is the most widely used Unix mail application.

Sendmail includes the sendmail(8), mailq(1), and newaliases(1) programs.

One of the command line parameters to sendmail(8), `-bp`, will display the current status of messages in the mail queue. Check this on the command line by running `sendmail -bp` .

注意

Notice how the `&man.command.section;` notation is easier to follow.

9.6.6. 檔案、目錄、副檔名、裝置名稱

To refer to the name of a file, a directory, a file extension, or a device name, use `filename` .

範例 9.24. `filename` 範例

用法：

```
<para>The source for the Handbook in English is found in
  <filename> /usr/doc/en_US.ISO8859-1/books/handbook/</filename> .
  The main file is called <filename> book.xml</filename> .
  There is also a <filename> Makefile</filename>  and a
  number of files with a <filename> .ent</filename>  extension.</para>

<para><filename> kbd0</filename>  is the first keyboard detected
  by the system, and appears in
  <filename> /dev</filename> .</para>
```

輸出結果：

The source for the Handbook in English is found in `/usr/doc/en_US.ISO8859-1/books/handbook/` . The main file is called `book.xml` . There is also a `Makefile` and a number of files with a `.ent` extension.

`kbd0` is the first keyboard detected by the system, and appears in `/dev` .

9.6.7. Port 名稱

FreeBSD Extension

These elements are part of the FreeBSD extension to DocBook, and do not exist in the original DocBook DTD.

To include the name of a program from the FreeBSD Ports Collection in the document, use the `package` tag. Since the Ports Collection can be installed in any number of locations, only include the category and the port name; do not include `/usr/ports`.

By default, `package` refers to a binary package. To refer to a port that will be built from source, set the `role` attribute to `port`.

範例 9.25. `package` 範例

用法：

```
<para>Install the <package>net/wireshark</package> binary
  package to view network traffic.</para>

<para><package role="port">net/wireshark</package> can also be
  built and installed from the Ports Collection.</para>
```

輸出結果：

Install the net/wireshark binary package to view network traffic.

net/wireshark can also be built and installed from the Ports Collection.

9.6.8. 主機、網域、IP 位址、使用名稱、群組名稱及其他系統項目

FreeBSD Extension

These elements are part of the FreeBSD extension to DocBook, and do not exist in the original DocBook DTD.

Information for “system items” is marked up with `systemitem`. The `class` attribute is used to identify the particular type of information shown.

`class="domainname"`
: The text is a domain name, such as `FreeBSD.org` or `ngo.org.uk`. There is no hostname component.

`class="etheraddress"`
: The text is an Ethernet MAC address, expressed as a series of 2 digit hexadecimal numbers separated by colons.

`class="fqdomainname"`
: The text is a Fully Qualified Domain Name, with both hostname and domain name parts.

`class="ipaddress"`
: The text is an IP address, probably expressed as a dotted quad.

`class="netmask"`
The text is a network mask, which might be expressed as a dotted quad, a hexadecimal string, or as a `/` followed by a number (CIDR notation).

`class="systemname"`
With `class="systemname"` the marked up information is the simple hostname, such as `freefall` or `wcarchive`.

`class="username"`
The text is a username, like `root`.

`class="groupname"`
The text is a groupname, like `wheel`.

範例 9.26. `systemitem` 與類別 (Class) 範例

用法：

```
<para>The local machine can always be referred to by the
  name <systemitem class="systemname">  localhost</systemitem> , which will have the
 IP
  address <systemitem class="ipaddress">  127.0.0.1</systemitem> .</para>

<para>The <systemitem class="domainname">  FreeBSD.org</systemitem>
  domain contains a number of different hosts, including
  <systemitem class="fqdomainname">  freefall.FreeBSD.org</systemitem>  and
  <systemitem class="fqdomainname">  bento.FreeBSD.org</systemitem> .</para>

<para>When adding an <acronym> IP</acronym>  alias to an
  interface (using <command> ifconfig</command> )
  <emphasis> always</emphasis>  use a netmask of
  <systemitem class="netmask">  255.255.255.255</systemitem>  (which can
  also be expressed as
  <systemitem class="netmask">  0xffffffff</systemitem> ).</para>

<para>The <acronym> MAC</acronym>  address uniquely identifies
  every network card in existence.  A typical
  <acronym> MAC</acronym>  address looks like
  <systemitem class="etheraddress">  08:00:20:87:ef:d0</systemitem> .</para>

<para>To carry out most system administration functions
  requires logging in as <systemitem class="username">  root</systemitem> .</para>
```

輸出結果：

The local machine can always be referred to by the name `localhost`, which will have the IP address `127.0.0.1`.

The `FreeBSD.org` domain contains a number of different hosts, including `freefall.FreeBSD.org` and `bento.FreeBSD.org`.

When adding an IP alias to an interface (using `ifconfig`) always use a netmask of `255.255.255.255` (which can also be expressed as `0xffffffff`).

The MAC address uniquely identifies every network card in existence. A typical MAC address looks like `08:00:20:87:ef:d0`.

To carry out most system administration functions requires logging in as `root`.

9.6.9. 統一資源識別碼 (URI)

Occasionally it is useful to show a Uniform Resource Identifier (URI) without making it an active hyperlink. The `uri` element makes this possible:

範例 9.27. `uri` 範例

用法：

```
<para>This URL shows only as text:
  <uri>https://www.FreeBSD.org</uri>.  It does not
  create a link.</para>
```

輸出結果：

This URL shows only as text: `https://www.FreeBSD.org` . It does not create a link.

To create links, see 節 9.8, “連結”.

9.6.10. 郵件地址

Email addresses are marked up as `email` elements. In the HTML output format, the wrapped text becomes a hyperlink to the email address. Other output formats that support hyperlinks may also make the email address into a link.

範例 9.28. 有超連結的 `email` 範例

用法：

```
<para>An email address that does not actually exist, like
  <email>notreal@example.com</email>, can be used as an
  example.</para>
```

輸出結果：

An email address that does not actually exist, like <notreal@example.com >, can be used as an example.

A FreeBSD-specific extension allows setting the `role` attribute to `nolink` to prevent the creation of the hyperlink to the email address.

範例 9.29. 沒有超連結的 `email` 範例

用法：

```
<para>Sometimes a link to an email address like
  <email role="nolink"> notreal@example.com</email> is not
  desired.</para>
```

輸出結果：

Sometimes a link to an email address like <notreal@example.com > is not desired.

9.6.11. 說明 Makefile

FreeBSD Extension

These elements are part of the FreeBSD extension to DocBook, and do not exist in the original DocBook DTD.

Two elements exist to describe parts of Makefile s, buildtarget and varname.

buildtarget identifies a build target exported by a Makefile that can be given as a parameter to make. varname identifies a variable that can be set (in the environment, on the command line with make, or within the Makefile) to influence the process.

範例 9.30. buildtarget 與 varname 範例

用法：

```
<para>Two common targets in a <filename>Makefile</filename>
  are <buildtarget>all</buildtarget> and
  <buildtarget>clean</buildtarget>.</para>

<para>Typically, invoking <buildtarget>all</buildtarget> will
  rebuild the application, and invoking
  <buildtarget>clean</buildtarget> will remove the temporary
  files (<filename>.o</filename> for example) created by the
  build process.</para>

<para><buildtarget>clean</buildtarget> may be controlled by a
  number of variables, including <varname>CLOBBER</varname>
  and <varname>RECURSE</varname>.</para>
```

輸出結果：

Two common targets in a Makefile are all and clean.

Typically, invoking all will rebuild the application, and invoking clean will remove the temporary files (.o for example) created by the build process.

clean may be controlled by a number of variables, including CLOBBER and RECURSE.

9.6.12. 實際文字 (Literal)

Literal text, or text which should be entered verbatim, is often needed in documentation. This is text that is excerpted from another file, or which should be copied exactly as shown from the documentation into another file.

Some of the time, programlisting will be sufficient to denote this text. But programlisting is not always appropriate, particularly when you want to include a portion of a file “in-line” with the rest of the paragraph.

On these occasions, use literal.

範例 9.31. literal 範例

用法：

```
<para>The <literal>maxusers 10</literal>  line in the kernel
  configuration file determines the size of many system tables, and is
  a rough guide to how many simultaneous logins the system will
  support.</para>
```

輸出結果：

The `maxusers 10` line in the kernel configuration file determines the size of many system tables, and is a rough guide to how many simultaneous logins the system will support.

9.6.13. 顯示使用者必填的項目

There will often be times when the user is shown what to do, or referred to a file or command line, but cannot simply copy the example provided. Instead, they must supply some information themselves.

`replaceable` is designed for this eventuality. Use it inside other elements to indicate parts of that element's content that the user must replace.

範例 9.32. `replaceable` 範例

用法：

```
<screen>&prompt.user; <userinput>man <replaceable>command</replaceable> </
userinput></screen>
```

輸出結果：

```
% man command
```

`replaceable` can be used in many different elements, including `literal`. This example also shows that `replaceable` should only be wrapped around the content that the user is meant to provide. The other content should be left alone.

用法：

```
<para>The <literal>maxusers <replaceable>n</replaceable> </literal>
  line in the kernel configuration file determines the size of many system
  tables, and is a rough guide to how many simultaneous logins the system will
  support.</para>

<para>For a desktop workstation, <literal>32</literal>  is a good value
  for <replaceable>n</replaceable> .</para>
```

輸出結果：

The `maxusers` *n* line in the kernel configuration file determines the size of many system tables, and is a rough guide to how many simultaneous logins the system will support.

For a desktop workstation, 32 is a good value for *n*.

9.6.14. 顯示 GUI 按鈕

Buttons presented by a graphical user interface are marked with `guibutton`. To make the text look more like a graphical button, brackets and non-breaking spaces are added surrounding the text.

範例 9.33. `guibutton` 範例

用法：

```
<para>Edit the file, then click
  <guibutton> [ Save ]</guibutton>  to save the
  changes.</para>
```

輸出結果：

Edit the file, then click [Save] to save the changes.

9.6.15. 引用系統錯誤

System errors generated by FreeBSD are marked with `errorname` . This indicates the exact error that appears.

範例 9.34. `errorname` 範例

用法：

```
<screen> <errorname> Panic: cannot mount root</errorname> </screen>
```

輸出結果：

```
Panic: cannot mount root
```

9.7. 圖片

重要

Image support in the documentation is somewhat experimental. The mechanisms described here are unlikely to change, but that is not guaranteed.

To provide conversion between different image formats, the graphics/ImageMagick port must be installed. This port is not included in the textproc/docproj meta port, and must be installed separately.

A good example of the use of images is the `doc/en_US.ISO8859-1/articles/vm-design/` document. Examine the files in that directory to see how these elements are used together. Build different output formats to see how the format determines what images are shown in the rendered document.

9.7.1. 圖片格式

The following image formats are currently supported. An image file will automatically be converted to bitmap or vector image depending on the output document format.

These are the only formats in which images should be committed to the documentation repository.

EPS (Encapsulated Postscript)
: Images that are primarily vector based, such as network diagrams, time lines, and similar, should be in this format. These images have a `.eps` extension.

PNG (Portable Network Graphic)
: For bitmaps, such as screen captures, use this format. These images have the `.png` extension.

PIC (PIC graphics language)
: PIC is a language for drawing simple vector-based figures used in the pic(1) utility. These images have the `.pic` extension.

SCR (SCReen capture)
: This format is specific to screenshots of console output. The following command generates an SCR file `shot.scr` from video buffer of `/dev/ttyv0`:

```
# vidcontrol -p  < /dev/ttyv0  > shot.scr
```

This is preferable to PNG format for screenshots because the SCR file contains plain text of the command lines so that it can be converted to a PNG image or a plain text depending on the output document format.

Use the appropriate format for each image. Documentation will often have a mix of EPS and PNG images. The `Makefile`s ensure that the correct format image is chosen depending on the output format used. Do not commit the same image to the repository in two different formats.

重要

The Documentation Project may eventually switch to using the SVG (Scalable Vector Graphic) format for vector images. However, the current state of SVG capable editing tools makes this impractical.

9.7.2. 圖片檔案位置

Image files can be stored in one of several locations, depending on the document and image:

- In the same directory as the document itself, usually done for articles and small books that keep all their files in a single directory.
- In a subdirectory of the main document. Typically done when a large book uses separate subdirectories to organize individual chapters.

  When images are stored in a subdirectory of the main document directory, the subdirectory name must be included in their paths in the `Makefile` and the `imagedata` element.
- In a subdirectory of `doc/share/images` named after the document. For example, images for the Handbook are stored in `doc/share/images/books/handbook`. Images that work for multiple translations are stored in this upper level of the documentation file tree. Generally, these are images that can be used unchanged in non-English translations of the document.

9.7.3. 圖片標籤

Images are included as part of a `mediaobject`. The `mediaobject` can contain other, more specific objects. We are concerned with two, the `imageobject` and the `textobject`.

Include one `imageobject`, and two `textobject` elements. The `imageobject` will point to the name of the image file without the extension. The `textobject` elements contain information that will be presented to the user as well as, or instead of, the image itself.

Text elements are shown to the reader in several situations. When the document is viewed in HTML, text elements are shown while the image is loading, or if the mouse pointer is hovered over the image, or if a text-only browser is being used. In formats like plain text where graphics are not possible, the text elements are shown instead of the graphical ones.

This example shows how to include an image called `fig1.png` in a document. The image is a rectangle with an A inside it:

```
<mediaobject>
  <imageobject>
    <imagedata fileref="fig1"/>  ❶
  </imageobject>

  <textobject>
    <literallayout class="monospaced">  +---------------+ ❷
|       A       |
+---------------+</literallayout>
  </textobject>

  <textobject>
    <phrase>A picture</phrase> ❸
  </textobject>
</mediaobject>
```

❶ Include an `imagedata` element inside the `imageobject` element. The `fileref` attribute should contain the filename of the image to include, without the extension. The stylesheets will work out which extension should be added to the filename automatically.

❷ The first `textobject` contains a `literallayout` element, where the `class` attribute is set to `monospaced`. This is an opportunity to demonstrate ASCII art skills. This content will be used if the document is converted to plain text.

Notice how the first and last lines of the content of the `literallayout` element butt up next to the element's tags. This ensures no extraneous white space is included.

❸ The second `textobject` contains a single `phrase` element. The contents of this phrase will become the `alt` attribute for the image when this document is converted to HTML.

9.7.4. 圖片 `Makefile` 項目

Images must be listed in the `Makefile` in the `IMAGES` variable. This variable must contain the names of all the source images. For example, if there are three figures, `fig1.eps`, `fig2.png`, `fig3.png`, then the `Makefile` should have lines like this in it.

```
…
IMAGES= fig1.eps fig2.png fig3.png
…
```

or

```
…
IMAGES=  fig1.eps
IMAGES+= fig2.png
IMAGES+= fig3.png
…
```

Again, the `Makefile` will work out the complete list of images it needs to build the source document, you only need to list the image files you provided.

9.7.5. 在子目錄中的圖片與章節

Be careful when separating documentation into smaller files in different directories (see 節 7.7.1, “在引用檔使用一般 Entities”).

Suppose there is a book with three chapters, and the chapters are stored in their own directories, called `chapter1/chapter.xml` , `chapter2/chapter.xml` , and `chapter3/chapter.xml` . If each chapter has images associated with it, place those images in each chapter's subdirectory (`chapter1/` , `chapter2/` , and `chapter3/`).

However, doing this requires including the directory names in the `IMAGES` variable in the `Makefile` , and including the directory name in the `imagedata` element in the document.

For example, if the book has `chapter1/fig1.png` , then `chapter1/chapter.xml` should contain:

```
<mediaobject>
  <imageobject>
    <imagedata fileref="chapter1/fig1"/>    ❶
  </imageobject>

  …

</mediaobject>
```

❶ The directory name must be included in the `fileref` attribute.

The `Makefile` must contain:

```
…
IMAGES=  chapter1/fig1.png
…
```

9.8. 連結

注意

Links are also in-line elements. To show a URI without creating a link, see 節 9.6.9, “統一資源識別碼 (URI)”.

9.8.1. `xml:id` 屬性

Most DocBook elements accept an `xml:id` attribute to give that part of the document a unique name. The `xml:id` can be used as a target for a crossreference or link.

Any portion of the document that will be a link target must have an `xml:id` attribute. Assigning an `xml:id` to all chapters and sections, even if there are no current plans to link to them, is a good idea. These `xml:id`s can be used as unique reference points by anyone referring to the HTML version of the document.

範例 9.35. 在章與節上加 `xml:id` 的範例

```
<chapter xml:id="introduction">
  <title>Introduction</title>

  <para>This is the introduction.  It contains a subsection,
    which is identified as well.</para>

  <sect1 xml:id="introduction-moredetails">
    <title>More Details</title>

    <para>This is a subsection.</para>
  </sect1>
```

```
</chapter>
```

Use descriptive values for `xml:id` names. The values must be unique within the entire document, not just in a single file. In the example, the subsection `xml:id` is constructed by appending text to the chapter `xml:id`. This ensures that the `xml:id`s are unique. It also helps both reader and anyone editing the document to see where the link is located within the document, similar to a directory path to a file.

9.8.2. 使用 `xref` 交叉參照

`xref` provides the reader with a link to jump to another section of the document. The target `xml:id` is specified in the `linkend` attribute, and `xref` generates the link text automatically.

範例 9.36. `xref` 範例

Assume that this fragment appears somewhere in a document that includes the `xml:id` example shown above:

```
<para>More information can be found
  in <xref linkend="introduction"/>  .</para>

<para>More specific information can be found
  in <xref linkend="introduction-moredetails"/>   .</para>
```

The link text will be generated automatically, looking like (emphasized text indicates the link text):

> More information can be found in Chapter 1, Introduction.
>
> More specific information can be found in Section 1.1, “More Details”.

The link text is generated automatically from the chapter and section number and `title` elements.

9.8.3. 連結在網站上的其他文件

The link element described here allows the writer to define the link text. When link text is used, it is very important to be descriptive to give the reader an idea of where the link goes. Remember that DocBook can be rendered to multiple types of media. The reader might be looking at a printed book or other form of media where there are no links. If the link text is not descriptive enough, the reader might not be able to locate the linked section.

The `xlink:href` attribute is the URL of the page, and the content of the element is the text that will be displayed for the user to activate.

In many situations, it is preferable to show the actual URL rather than text. This can be done by leaving out the element text entirely.

範例 9.37. `link` 到 FreeBSD 說明文件網頁範例

Link to the book or article URL entity. To link to a specific chapter in a book, add a slash and the chapter file name, followed by an optional anchor within the chapter. For articles, link to the article URL entity, followed by an optional anchor within the article. URL entities can be found in `doc/share/xml/urls.ent` .

Usage for FreeBSD book links:

```
<para>Read the <link
    xlink:href="&url.books.handbook;/svn.html#svn-intro">   SVN
    introduction</link>, then pick the nearest mirror from
  the list of <link
    xlink:href="&url.books.handbook;/svn.html#svn-mirrors">   Subversion
    mirror sites</link>.</para>
```

輸出結果：

Read the SVN introduction, then pick the nearest mirror from the list of Subversion mirror sites.

Usage for FreeBSD article links:

```
<para>Read this
  <link xlink:href="&url.articles.bsdl-gpl;">   article
    about the BSD license</link>, or just the
  <link xlink:href="&url.articles.bsdl-gpl;#intro">   introduction</link>.</para>
```

輸出結果：

Read this article about the BSD license, or just the introduction.

範例 9.38. `link` 到 FreeBSD 網頁範例

用法：

```
<para>Of course, you could stop reading this document and go to the
  <link xlink:href="&url.base;/index.html">  FreeBSD home page</link>  instead.</para>
```

輸出結果：

Of course, you could stop reading this document and go to the FreeBSD home page instead.

範例 9.39. `link` 到外部網頁範例

用法：

```
<para>Wikipedia has an excellent reference on
  <link
    xlink:href="http://en.wikipedia.org/wiki/GUID_Partition_Table">    GUID
    Partition Tables</link>.</para>
```

輸出結果：

Wikipedia has an excellent reference on GUID Partition Tables.

The link text can be omitted to show the actual URL:

```
<para>Wikipedia has an excellent reference on
  GUID Partition Tables: <link
    xlink:href="http://en.wikipedia.org/wiki/GUID_Partition_Table">    </link>.</para>
```

The same link can be entered using shorter notation instead of a separate ending tag:

```
<para>Wikipedia has an excellent reference on
  GUID Partition Tables: <link
```

```
    xlink:href="http://en.wikipedia.org/wiki/GUID_Partition_Table"/>    .</para>
```

The two methods are equivalent. Appearance:

Wikipedia has an excellent reference on GUID Partition Tables: `http://en.wikipedia.org/wiki/GUID_Partition_Table` .

章 10. 樣式表

XML is concerned with content, and says nothing about how that content should be presented to the reader or rendered on paper. Multiple style sheet languages have been developed to describe visual layout, including Extensible Stylesheet Language Transformation (XSLT), Document Style Semantics and Specification Language (DSSSL), and Cascading Style Sheets (CSS).

The FDP documents use XSLT stylesheets to transform DocBook into XHTML, and then CSS formatting is applied to the XHTML pages. Printable output is currently rendered with legacy DSSSL stylesheets, but this will probably change in the future.

10.1. CSS

Cascading Style Sheets (CSS) are a mechanism for attaching style information (font, weight, size, color, and so forth) to elements in an XHTML document without abusing XHTML to do so.

10.1.1. DocBook 文件

The FreeBSD XSLT and DSSSL stylesheets refer to `docbook.css` , which is expected to be present in the same directory as the XHTML files. The project-wide CSS file is copied from `doc/share/misc/docbook.css` when documents are converted to XHTML, and is installed automatically.

章 11. 翻譯

本章節是供要翻譯 FreeBSD 說明文件（常見問答集（FAQ）、使用手冊（Handbook）、教學（Tutorial）、操作手冊 (Manual page) 等）到各種語言的常見問答集（FAQ）。

本文件 主要 是以 FreeBSD 德文說明文件計劃的翻譯常見問答集為母本而來的，原始撰稿者為 Frank Gründer <elwood@mc5sys.in-berlin.de >，並由 Bernd Warken <bwarken@mayn.de > 再翻譯回英文版。

本常見問答集是由文件工程團隊 Documentation Engineering Team <doceng@FreeBSD.org > 所維護。

問： i18n 與 l10n 代表的是什麼意思？

答： i18n 指的是國際化 (Internationalization) 而 l10n 指的是在地化 (Localization)。這些都是為了書寫方便而用的簡寫。

i18n 就是開頭為 “i” 後面有 18 個字母，最後接 “n”。同樣地，l10n 是開頭為 “l” 後面有 10 個字母，最後接 “n”。

問： 有給翻譯人員參與討論的郵遞論壇 (Mailing list) 嗎？

答： 有的，不同的語系翻譯人員都各自有自屬的郵遞論壇。這份 翻譯計劃清單 有列出各翻譯計劃的詳細 mailing lists 及相關網站。此外，有一般翻譯討論的<freebsd-translators@freebsd.org >郵件論壇。

問： 需要更多人一起參與翻譯嗎？

答： 當然囉，越多人參與翻譯，那麼就能夠越快翻完，而且英文版說明文件若有增減、更新的話，各翻譯版也可以儘快同步囉。

不一定得是專業譯者，才能參與翻譯的。

問： 有要求哪些語言能力呢

答： 理論上，必須要對英文非常熟稔，而且很明顯地，對想翻譯的語言必須要能運用自如。

英文也並非一定要會的。比如說，可以把西班牙文 (Spanish) 的 FAQ 翻譯為匈牙利文（Hungarian）。

問： 該學會哪些程式的使用呢？

答： 強烈建議在自己機器上也建立 FreeBSD Subversion 檔案庫的備份（至少要有說明文件的部分），這可以執行：

```
% svn checkout https://svn.FreeBSD.org/doc/head/ head
```

svn.FreeBSD.org 是公共的 SVN 伺服器。可以從 Subversion 鏡像站 清單檢查認證的伺服器。

注意

這需要安裝 devel/subversion 套件。

你可以很自在地使用 svn。他可以讓你察看說明文件檔案不同版本之間的修改差異。

例如你要看 en_US.ISO8859-1/books/fdp-primer/book.xml 版本r33733 和 r33734 的差異，請執行：

```
% svn diff -r33733:33734 en_US.ISO8859-1/books/fdp-primer/book.xml
```

問： 要怎麼找出來還有誰要跟我一起翻譯的呢？

答： 說明文件計劃翻譯頁 列了目前已知的各翻譯者成果，如果已經有其他人也在做跟你一樣的翻譯工作，那麼請不要重複浪費人力，請與他們聯繫看看還有哪些地方可以幫上忙的。

若上面並未列出你母語的翻譯，或是也有人要翻譯但還未公開宣布的話，那麼就寄信到 FreeBSD 說明文件計劃郵遞論壇 。

問： 都沒人翻譯為我所使用的語言，該怎麼辦？

答： 恭喜啊，你剛好踏上 “FreeBSD #### 說明文件翻譯計劃” 的啓程之路，歡迎登船。

首先呢，先判斷是否有妥善規劃時間，因為你只有一個人在翻而已，因此，相關翻譯成果的公布、與其他可能會幫忙的志工們聯繫這些工作都是你的職責所在。

寫信到文件計劃郵遞論壇 (Documentation Project mailing list) 向大家宣布你正準備要翻譯，然後文件計劃的翻譯部分就會更新相關資料。

若你的國家已經有人提供 FreeBSD 的鏡像站 (Mirror) 服務的話，那麼就先跟他們聯繫，並詢問你是否在上面可以有網頁空間來放相關計劃資料，以及是否可以有提供電子郵件帳號或郵遞論壇服務。

然後，就開始翻文件囉，一開始翻譯的時候，先找些篇幅較短的文件會比較容易些 —— 像是 FAQ 啦，或是如何上手之類的說明文章。

問： 已經翻好一些說明文件了，該寄到哪呢？

答： 這要看情況而定。若你是在翻譯團隊內做的話 （像是日本團隊、德國團隊），他們會有自己內部流程來決定翻譯文件怎麼送，這些大致流程會在他們網頁上面有寫。

若你是某語系的唯一翻譯者（或你是負責某翻譯計劃，並想把成果回饋給 FreeBSD 計劃），那麼你就應該把自己的翻譯成果寄給 FreeBSD 計劃。（細節請看下個問題）

問： 我是該語系的唯一翻譯者，該怎麼把翻譯成果寄出去呢？

or

我們是翻譯團隊，該怎麼把我們成員翻譯成果寄出去呢？

答： 首先，請先確定你的翻譯成果組織條理分明，並可正確編譯，也就是說： 把它擺到現有說明文件樹內是可以正確編譯成功的。

目前，FreeBSD 說明文件都是放在最上層的 head/ 目錄內。而該目錄下的則根據其 ISO639 所定義的語系代碼來做分類命名的（在 1999/1/20 之後的 FreeBSD 版本中的 /usr/share/misc/iso639 ）。

若你這個語系可能會有不同編碼方式（像是：中文）那麼就應該會像下面這樣，來依你所使用的編碼方式細分。

最後，你應該建立好各文件的目錄了。

舉例來說，假設有瑞典文 (Swedish) 版的翻譯，那麼應該會長像：

```
head/
    sv_SE.ISO8859-1/
                    Makefile
                    htdocs/
                          docproj/
                    books/
                          faq/
                              Makefile
                              book.xml
```

sv_SE.ISO8859-1 是依照 *## (Lang).## (Encoding)* 的規則來建立的譯名。請注意：其中有兩個 Makefile 檔，它們是用來建置說明文件的。

然後請用 tar(1) 與 gzip(1) 來把你的說明文件壓縮起來，並寄到本計劃來。

```
% cd doc
% tar cf swedish-docs.tar sv_SE.ISO8859-1
```

```
% gzip -9 swedish-docs.tar
```

接著，把 swedish-docs.tar.gz 放到網頁空間上，若你沒有自己網頁空間的話 (ISP 不提供)，那麼可以該檔寄到文件工程團隊 Documentation Engineering Team <doceng@FreeBSD.org > 來。

還有，記得用 Bugzilla 提交一個回報來通知大家你已經提交說明文件了，還有，若有人可以幫忙檢閱、複審文件的話，對翻譯品質較好，因為這也有助於提升翻譯品質的流暢度。

最後，會有人 （可能是文件計劃管理者，或是文件工程團隊 Documentation Engineering Team <doceng@FreeBSD.org > 成員） 會檢閱你的翻譯文件，並確認是否可正常編譯。此外，他們會特別注意下列幾點：

1. 你的檔案是否都有用 RCS tag（像是 "ID" 之類的）？

2. sv_SE.ISO8859-1 是否可以順利make all 編譯呢？

3. make install 是否結果有正確

若有問題的話，那麼檢閱者會叮嚀你，來讓這些翻譯成果可以正確使用。

若沒問題的話，那麼就會很快把你的翻譯成果提交。

問： 可以加入某語系或某國家才有的東西到翻譯內容內嗎？

答： 我們希望不要這麼做。

舉例來說，假設你正準備把使用手冊 (Handbook) 翻譯為韓文版，並希望把韓國零售商也加到你翻譯的韓文版使用手冊內。

我們想不出來有啥原因，為什麼不把這些資訊提供給英文版呢？(或是德文、西班牙文、日文等 ...) 因為，有可能英語讀者跑去韓國時，會想買 FreeBSD 相關產品。此外，這也可以提升 FreeBSD 的可見度，很顯然的，這並不是件壞事啊。

若你有某國才有的資料，請提供給英文版使用手冊以作為修訂（用 Bugzilla），然後再把英文版的修訂部分翻為你要翻譯的使用手冊吧。

謝謝。

問： 要怎麼把該語系特有的字元寫進去翻譯內容呢？

答： 說明文件內所有的非 ASCII (Non-ASCII) 的字元，都要使用 SGML entities 才能寫進去。

簡單來說，長相一開頭會是 and 符號 (&)，然後是該 Entity 名稱，最後接上分號 (;)。

這些 Entity 名稱都是 ISO8879 所制訂的，其在 Port 樹內的 textproc/iso8879。

以下舉一些例子：
Entity 名稱: é
外觀: é
說明: 小 “e”，並帶尖、重音 (Acute accent)
Entity 名稱: É
外觀: É
說明: 大 “E”，並帶尖、重音 (Acute accent)
Entity 名稱: ü
外觀: ü
說明: 小 “u”，並帶日耳曼語系中的母音變化 (Umlaut)

在裝了 iso8879 這個 Port 之後，就可以在 /usr/local/share/xml/iso8879 找到這些的詳細列表。

問： 如何稱呼讀者呢？

答： 在英文文件內，讀者都是以 “you” 來稱呼，而有些語言並沒有正式/非正式的區隔。

若你所要翻的語言可以區別這些差異，那麼請用該語系在一般技術說明文件上所使用的稱呼吧。如果容易造成困惑的話，那麼請改用較中性的稱呼來取代。

問： 翻譯成果內要不要附上一些其他訊息呢？

答： 要。

每份英文版原稿的開頭，通常會有像下面的內容：

```
<!--
     The FreeBSD Documentation Project

     $FreeBSD$
-->
```

實際上的內容可能稍有不同，但每份原稿都會附上 $FreeBSD$ 這一行以及 The FreeBSD Documentation Project 宣告。請注意：$FreeBSD$ 開頭的這行是會由 Subversion 隨著每次異動而自動更改的，所以新檔案的話請保持原狀（也就是只要寫 $FreeBSD$ 就好了）。

翻譯文件中，必須都要有 $FreeBSD$ 這行，並且把 FreeBSD Documentation Project 這行改為 The FreeBSD *language* Documentation Project 。

此外，還必須加上第三行來指出你所翻譯的，到底是以英文版原稿的哪一版本為母本所做的翻譯。

因此呢，西班牙文版 (Spanish) 的檔案開頭應該是長像這樣：

```
<!--
     The FreeBSD Spanish Documentation Project

     $FreeBSD$
     Original revision: r38674
-->
```

章 12. PO 翻譯

12.1. 簡介

GNU gettext 系統提供翻譯者一個簡單的方法來建立和維護文件的翻譯。翻譯的字串從原始文件題取出來到 PO (Portable Object) 檔。字串的翻譯用另外的編輯器輸入。翻譯的字串可以直接使用，或是編譯成原始文件的完整翻譯版本。

12.2. 快速上手

我們會假設您已做過在 節 1.1, “快速上手” 中的步驟，除此之外還必須打開 textproc/docproj Port 中的 TRANSLATOR 選項。如果沒有打開這個選項，請打開選項後重新安裝 Port。

```
# cd /usr/ports/textproc/docproj
# make config
# make clean deinstall install clean
```

這個範例示範如何建立 Leap Seconds 短文的西班牙文翻譯。

過程 12.1. 安裝 PO 編輯器

- 編輯翻譯檔案需要 PO 編輯器。這個範例使用 editors/poedit。

  ```
  # cd /usr/ports/editors/poedit
  # make install clean
  ```

過程 12.2. 初始設定

第一次建立新的翻譯時，目錄結構和 Makefile 必須建立或是從英文版複製過來。

1. 建立新翻譯的目錄。英文文章原始碼位於 ~/doc/en_US.ISO8859-1/articles/leap-seconds/ 。西班牙文翻譯將會放在 ~/doc/es_ES.ISO8859-1/articles/leap-seconds/ 。除了語系目錄的名稱外，其他路徑相同。

   ```
   % svn mkdir --parents ~/doc/es_ES.ISO8859-1/articles/leap-seconds/
   ```

2. 從原始文件處將 Makefile 複製到翻譯目錄。

   ```
   % svn cp ~/doc/en_US.ISO8859-1/articles/leap-seconds/Makefile \
       ~/doc/es_ES.ISO8859-1/articles/leap-seconds/
   ```

過程 12.3. 翻譯

翻譯文件公有兩個步驟：將可翻譯的字串從原始文件提去出來，然後翻譯這些字串。重複這些步驟，直到翻譯者認為文件的翻譯部份已經足夠用來產生可讀的翻譯文件。

1. 從英文的原始文件提取字串到 PO 檔：

   ```
   % cd ~/doc/es_ES.ISO8859-1/articles/leap-seconds/
   % make po
   ```

2. 使用 PO 編輯器將翻譯輸入 PO 檔。有幾個不同的編輯器可以使用。這裡用的是 editors/poedit 的 poedit 。

   PO 檔名是兩個字元的語系碼後面接底線和兩個字元的區域碼。以西班牙語來說，檔名是 es_ES.po 。

   ```
   % poedit es_ES.po
   ```

過程 12.4. 產生翻譯文件

1. 產生翻譯文件

```
% cd ~/doc/es_ES.ISO8859-1/articles/leap-seconds/
% make tran
```

產生的文件名稱與英文原始文件名稱相符，文章通常是 `article.xml` ，書籍是 `book.xml` 。

2. 可以轉換成 HTML 來檢查產生的檔案，並用瀏覽器來察看。

```
% make FORMATS=html
% firefox article.html
```

12.3. 建立新翻譯

建立新翻譯文件的第一步是找到或建立一個目錄來放它。FreeBSD 將翻譯文件放在子目錄，用語系和區域以 *## (lang)_## (REGION)* 來命名。*## (lang)* 是小寫的兩個字元碼，接著是底線和兩個字元的大寫 *REGION* 碼。

表格 12.1. 語系名稱

語言	地區	翻譯目錄名稱	PO 檔名稱	字元集
英文	美國	en_US.ISO8859-1	en_US.po	ISO 8859-1
孟加拉文	孟加拉	bn_BD.UTF-8	bn_BD.po	UTF-8
丹麥文	丹麥	da_DK.ISO8859-1	da_DK.po	ISO 8859-1
德文	德國	de_DE.ISO8859-1	de_DE.po	ISO 8859-1
希臘文	希臘	el_GR.ISO8859-7	el_GR.po	ISO 8859-7
西班牙文	西班牙	es_ES.ISO8859-1	es_ES.po	ISO 8859-1
法文	法國	fr_FR.ISO8859-1	fr_FR.po	ISO 8859-1
匈牙利文	匈牙利	hu_HU.ISO8859-2	hu_HU.po	ISO 8859-2
義大利文	義大利	it_IT.ISO8859-15	it_IT.po	ISO 8859-15
日文	日本	ja_JP.eucJP	ja_JP.po	EUC JP
韓文	韓國	ko_KR.UTF-8	ko_KR.po	UTF-8
蒙古文	蒙古	mn_MN.UTF-8	mn_MN.po	UTF-8
荷蘭文	荷蘭	nl_NL.ISO8859-1	nl_NL.po	ISO 8859-1
挪威文	挪威	no_NO.ISO8859-1	no_NO.po	ISO 8859-1
波蘭文	波蘭	pl_PL.ISO8859-2	pl_PL.po	ISO 8859-2
葡萄牙文	巴西	pt_BR.ISO8859-1	pt_BR.po	ISO 8859-1
俄文	俄羅斯	ru_RU.KOI8-R	ru_RU.po	KOI8-R
賽爾維亞	賽爾維亞文	sr_YU.ISO8859-2	sr_YU.po	ISO 8859-2
土耳其文	土耳其	tr_TR.ISO8859-9	tr_TR.po	ISO 8859-9
中文	中國	zh_CN.UTF-8	zh_CN.po	UTF-8
中文	台灣	zh_TW.UTF-8	zh_TW.po	UTF-8

翻譯位於主要說明文件目錄的子目錄，這裡假設如 節 1.1, “快速上手” 所示，是 `~/doc/` 。例如德文位於 `~/doc/de_DE.ISO8859-1/` ，法文位於 `~/doc/fr_FR.ISO8859-1/` 。

每個語系目錄包含不同文件類型的子目錄，通常是 `articles/` 和 `books/` 。

將目錄名稱組合起來就是文章或書的完整路徑。例如，NanoBSD 文章的法語翻譯在 ~/doc/fr_FR.ISO8859-1/articles/nanobsd/ 。而使用手冊的蒙古文翻譯在~/doc/mn_MN.UTF-8/books/handbook/ 。

當翻譯到一個新語系時必須建立一個新的語系目錄。如果語系目錄已經存在，那只需要有 articles/ 或 books/ 的子目錄。

FreeBSD 說明文件的編譯是由同一個目錄的 Makefile 控制。簡單的文章可以從原始的英語目錄直接複製 Makefile 過來。書籍的翻譯流程結合多個獨立的 book.xml 和 chapter.xml 成為一個檔案，所以書籍翻譯的 Makefile 必須複製並修改。

範例 12.1. 建立 Porter 手冊的西班牙語翻譯

建立 Porter 手冊 的西班牙文翻譯。原文是位於 ~/doc/en_US.ISO8859-1/books/porters-handbook/ 的書籍。

1. 西班牙文 books 目錄 ~/doc/es_ES.ISO8859-1/books/ 已經存在，所以只要建立 Porter 手冊的子目錄：

```
% cd ~/doc/es_ES.ISO8859-1/books/
% svn mkdir porters-handbook
A         porters-handbook
```

2. 從原始文件的目錄複製 Makefile ：

```
% cd ~/doc/es_ES.ISO8859-1/books/porters-handbook
% svn cp ~/doc/en_US.ISO8859-1/books/porters-handbook/Makefile .
A         Makefile
```

修改 Makefile 內容以產生單一的 book.xml ：

```
#
# $FreeBSD$
#
# Build the FreeBSD Porter's Handbook.
#

MAINTAINER=doc@FreeBSD.org

DOC?= book

FORMATS?= html-split

INSTALL_COMPRESSED?= gz
INSTALL_ONLY_COMPRESSED?=

# XML content
SRCS=  book.xml

# Images from the cross-document image library
IMAGES_LIB+=    callouts/1.png
IMAGES_LIB+=    callouts/2.png
IMAGES_LIB+=    callouts/3.png
IMAGES_LIB+=    callouts/4.png
IMAGES_LIB+=    callouts/5.png
IMAGES_LIB+=    callouts/6.png
IMAGES_LIB+=    callouts/7.png
IMAGES_LIB+=    callouts/8.png
IMAGES_LIB+=    callouts/9.png
IMAGES_LIB+=    callouts/10.png
IMAGES_LIB+=    callouts/11.png
```

```
IMAGES_LIB+=    callouts/12.png
IMAGES_LIB+=    callouts/13.png
IMAGES_LIB+=    callouts/14.png
IMAGES_LIB+=    callouts/15.png
IMAGES_LIB+=    callouts/16.png
IMAGES_LIB+=    callouts/17.png
IMAGES_LIB+=    callouts/18.png
IMAGES_LIB+=    callouts/19.png
IMAGES_LIB+=    callouts/20.png
IMAGES_LIB+=    callouts/21.png

URL_RELPREFIX?= ../../../..
DOC_PREFIX?= ${.CURDIR}/../../..

.include "${DOC_PREFIX}/share/mk/doc.project.mk"
```

現在文件結構已經準備好讓翻譯者執行 `make po` 開始翻譯。

範例 12.2. 建立 PGP 金鑰文章的法語翻譯。

建立 PGP 金鑰文章 的法文翻譯。原文是位於 `~/doc/en_US.ISO8859-1/articles/pgpkeys/` 的文章。

1. 法文文章目錄 `~/doc/fr_FR.ISO8859-1/articles/` 已經存在，所以只要建立 PGP 金鑰文章的子目錄：

   ```
   % cd ~/doc/fr_FR.ISO8859-1/articles/
   % svn mkdir pgpkeys
   A         pgpkeys
   ```

2. 從原始文件的目錄複製 `Makefile` ：

   ```
   % cd ~/doc/fr_FR.ISO8859-1/articles/pgpkeys
   % svn cp ~/doc/en_US.ISO8859-1/articles/pgpkeys/Makefile .
   A         Makefile
   ```

   檢查 `Makefile` 的內容。因為這是簡單的文章，此例的 `Makefile` 不用修改。第二行的 `$FreeBSD...$` 版本字串將會在檔案提交時被版本控制系統替換掉。

   ```
   #
   # $FreeBSD$
   #
   # Article: PGP Keys

   DOC?= article

   FORMATS?= html
   WITH_ARTICLE_TOC?= YES

   INSTALL_COMPRESSED?= gz
   INSTALL_ONLY_COMPRESSED?=

   SRCS=   article.xml

   # To build with just key fingerprints, set FINGERPRINTS_ONLY.

   URL_RELPREFIX?= ../../../..
   DOC_PREFIX?=    ${.CURDIR}/../../..

   .include "${DOC_PREFIX}/share/mk/doc.project.mk"
   ```

文章結構處理好後，可以執行建立 `make po` 建立 PO 檔。

12.4. 翻譯

gettext 系統大幅減少翻譯者要追蹤的事情。字串從原始文件提取到 PO 檔。再用 PO 檔編輯器輸入字串的翻譯。

FreeBSD PO 翻譯系統不會覆蓋掉 PO 檔。所以提取步驟可以在任何時候重複執行來更新 PO 檔。

用 PO 檔編輯器來編輯檔案。此例是用 editors/poedit，因為它很簡單而且系統需求低。其他的 PO 檔編輯器提供一些特點，能使翻譯工作更輕鬆。`Port` 套件集裡有數個編輯器，包括 devel/gtranslator 。

保留 PO 檔是很重要的。它包含所有的翻譯成果。

範例 12.3. 翻譯 Porter 手冊到西班牙文

輸入 Porter 手冊的西班牙文內容

1. 切換到西班牙文 Porter 手冊的目錄並更新 PO 檔。產生的 PO 檔如 表格 12.1, “語系名稱” 所示，名叫 `es_ES.po` 。

```
% cd ~/doc/es_ES.ISO8859-1/books/porters-handbook
% make po
```

2. 使用 PO 檔編輯器輸入翻譯：

```
% poedit es_ES.po
```

12.5. 給翻譯者的提示

12.5.1. 保留 XML 標籤

保留在英文原文的 XML 標籤。

範例 12.4. 保留 XML 標籤

英文原文：

```
If <acronym> NTP</acronym>  is not being used
```

西班牙文翻譯：

```
Si <acronym> NTP</acronym>  no se utiliza
```

12.5.2. 保留空白

保留要翻譯字串前後的空白，翻譯過的版本也需要有這些空白。

12.5.3. 不要翻譯的標籤

有些標籤的內容要一字不差地保留，不要翻譯。

- <citerefentry>
- <command>
- <filename>
- <literal>
- <manvolnum>
- <orgname>
- <package>
- <programlisting>
- <prompt>
- <refentrytitle>
- <screen>
- <userinput>
- <varname>

12.5.4. $FreeBSD$ 字串

在檔案中使用到的 $FreeBSD$ 版本字串都需要特別處理，例如在 範例 12.1, “建立 Porter 手冊的西班牙語翻譯” ，使用這些字串的用意並非要展開成版本。英文的說明文件會使用 $ Entity 來避免在檔案中用到錢字符號：

```
&dollar;FreeBSD&dollar;
```

版本控制符號不會把 $ entities 看成金錢符號，所以不會把字串展開成版本字串。

當 PO 檔建立之後，在範例中使用到的 $ Entity 會被取代成實際的錢字符號，這會使的 $FreeBSD$ 字串在提交時錯誤的被版本控制系統展開成版本字串。

在英文文件上使用的方法也可以用在翻譯上，翻譯時在 PO 編輯器用 $ 來取代金錢符號：

```
&dollar;FreeBSD&dollar;
```

12.6. 編譯翻譯的文件

原文的翻譯版本可以在任何時候被建立。未翻譯的部份會以英文呈獻。大部份 PO 編輯器有指標可以顯示翻譯完成度。這讓翻譯者更容易看翻譯好的字串是否足夠來編譯最終的文件。

範例 12.5. 編譯西班牙文 Porter 手冊

編譯和預覽之前範例翻譯的西班牙文版 Porter 手冊

1. 編譯翻譯好的文件。因為原文是書籍，所以產生的文件是 book.xml 。

```
% cd ~/doc/es_ES.ISO8859-1/books/porters-handbook
% make tran
```

2. 轉換翻譯好的 book.xml 成 HTML 並用 Firefox 來瀏覽。這和英文版是相同的步驟，其他 FORMATS 也可以這樣做。請見 表格 5.1, “常見輸出格式” 。

```
% make FORMATS=html
% firefox book.html
```

12.7. 提交新翻譯

準備要提交的新翻譯。這包含新增檔案到版本控制系統，對檔案設定額外的屬性，並建立 diff 來提交。

範例中產生的 diff 檔可以被附加到 文件問題回報 (Documentation bug report) 或 程式碼審查 (Code review) 。

範例 12.6. NanoBSD 文章的西班牙文翻譯

1. 增加 FreeBSD 版本字串註解到 PO 檔的第一行：

```
#$FreeBSD$
```

2. 增加 Makefile 、PO 檔和產生的 XML 翻譯到版本控制系統：

```
% cd ~/doc/es_ES.ISO8859-1/articles/nanobsd/
% ls
Makefile article.xml es_ES.po
% svn add Makefile article.xml es_ES.po
A         Makefile
A         article.xml
A         es_ES.po
```

3. 在這些檔案設定 Subversion svn:keywords 屬性為 FreeBSD=%H 讓 $FreeBSD$ 字串可以在提交時被展開成為路徑、修訂、日期以及作者：

```
% svn propset svn:keywords FreeBSD=%H Makefile article.xml es_ES.po
property 'svn:keywords' set on 'Makefile'
property 'svn:keywords' set on 'article.xml'
property 'svn:keywords' set on 'es_ES.po'
```

4. 設定檔案的 MIME 類型。書籍和文章是 text/xml ，PO 檔是 text/x-gettext-translation 。

```
% svn propset svn:mime-type text/x-gettext-translation es_ES.po
property 'svn:mime-type' set on 'es_ES.po'
% svn propset svn:mime-type text/xml article.xml
property 'svn:mime-type' set on 'article.xml'
```

5. 從 ~/doc/ 建立這些新檔案的 diff，讓檔名顯示完整的路徑。這可以幫助提交者辨識目標語系目錄。

```
% cd ~/doc
svn diff es_ES.ISO8859-1/articles/nanobsd/ > /tmp/es_nanobsd.diff
```

範例 12.7. Explaining-BSD 文章的韓文 UTF-8 翻譯

1. 增加 FreeBSD 版本字串註解到 PO 檔的第一行：

```
#$FreeBSD$
```

2. 增加 Makefile 、PO 檔和產生的 XML 翻譯到版本控制系統：

```
% cd ~/doc/ko_KR.UTF-8/articles/explaining-bsd/
% ls
Makefile article.xml ko_KR.po
% svn add Makefile article.xml ko_KR.po
A         Makefile
A         article.xml
A         ko_KR.po
```

3. 在這些檔案設定 Subversion svn:keywords 屬性為 FreeBSD=%H 讓 $FreeBSD$ 字串可以在提交時被展開成為路徑、修訂、日期以及作者：

```
% svn propset svn:keywords FreeBSD=%H Makefile article.xml ko_KR.po
property 'svn:keywords' set on 'Makefile'
property 'svn:keywords' set on 'article.xml'
property 'svn:keywords' set on 'ko_KR.po'
```

4. 設定檔案的 MIME 類型。因為這些檔案使用 UTF-8 字元集，這也需要指定。為了防止版本控制系統將這些檔案誤認為二進位資料，fbsd:notbinary 屬性也需要設定。

```
% svn propset svn:mime-type 'text/x-gettext-translation#charset=UTF-8' ko_KR.po
property 'svn:mime-type' set on 'ko_KR.po'
% svn propset fbsd:notbinary yes ko_KR.po
property 'fbsd:notbinary' set on 'ko_KR.po'
% svn propset svn:mime-type 'text/xml#charset=UTF-8' article.xml
property 'svn:mime-type' set on 'article.xml'
% svn propset fbsd:notbinary yes article.xml
property 'fbsd:notbinary' set on 'article.xml'
```

5. 從 ~/doc/ 建立這些新檔案的 diff。

```
% cd ~/doc
svn diff ko_KR.UTF-8/articles/explaining-bsd > /tmp/ko-explaining.diff
```

章 13. 寫作風格

13.1. 叮嚀

Technical documentation can be improved by consistent use of several principles. Most of these can be classified into three goals: be clear, be complete, and be concise. These goals can conflict with each other. Good writing consists of a balance between them.

13.1.1. 要明瞭

Clarity is extremely important. The reader may be a novice, or reading the document in a second language. Strive for simple, uncomplicated text that clearly explains the concepts.

Avoid flowery or embellished speech, jokes, or colloquial expressions. Write as simply and clearly as possible. Simple text is easier to understand and translate.

Keep explanations as short, simple, and clear as possible. Avoid empty phrases like “in order to”, which usually just means “to”. Avoid potentially patronizing words like “basically”. Avoid Latin terms like “i.e.” or “cf.”, which may be unknown outside of academic or scientific groups.

Write in a formal style. Avoid addressing the reader as “you”. For example, say “copy the file to `/tmp`” rather than “you can copy the file to `/tmp`”.

Give clear, correct, tested examples. A trivial example is better than no example. A good example is better yet. Do not give bad examples, identifiable by apologies or sentences like “but really it should never be done that way”. Bad examples are worse than no examples. Give good examples, because even when warned not to use the example as shown, the reader will usually just use the example as shown.

Avoid weasel words like “should”, “might”, “try”, or “could”. These words imply that the speaker is unsure of the facts, and create doubt in the reader.

Similarly, give instructions as imperative commands: not “you should do this”, but merely “do this”.

13.1.2. 要完整

Do not make assumptions about the reader's abilities or skill level. Tell them what they need to know. Give links to other documents to provide background information without having to recreate it. Put yourself in the reader's place, anticipate the questions they will ask, and answer them.

13.1.3. 要簡潔

While features should be documented completely, sometimes there is so much information that the reader cannot easily find the specific detail needed. The balance between being complete and being concise is a challenge. One approach is to have an introduction, then a “quick start” section that describes the most common situation, followed by an in-depth reference section.

13.2. 準則

To promote consistency between the myriad authors of the FreeBSD documentation, some guidelines have been drawn up for authors to follow.

使用美式英語拼寫
: There are several variants of English, with different spellings for the same word. Where spellings differ, use the American English variant. “color”, not “colour”, “rationalize”, not “rationalise”, and so on.

注意

The use of British English may be accepted in the case of a contributed article, however the spelling must be consistent within the whole document. The other documents such as books, web site, manual pages, etc. will have to use American English.

不要使用縮寫式
: Do not use contractions. Always spell the phrase out in full. “Don't use contractions” is wrong.

  Avoiding contractions makes for a more formal tone, is more precise, and is slightly easier for translators.

使用逗號串行
: In a list of items within a paragraph, separate each item from the others with a comma. Separate the last item from the others with a comma and the word “and”.

  For example:

  > This is a list of one, two and three items.

  Is this a list of three items, “one”, “two”, and “three”, or a list of two items, “one” and “two and three”?

  It is better to be explicit and include a serial comma:

  > This is a list of one, two, and three items.

避免多餘的語句
: Do not use redundant phrases. In particular, “the command”, “the file”, and “man command” are often redundant.

  For example, commands:

  Wrong: Use the `svn` command to update sources.

  Right: Use `svn` to update sources.

  Filenames:

  Wrong: ... in the filename `/etc/rc.local` ...

  Right: ... in `/etc/rc.local` ...

  Manual page references (the second example uses `citerefentry` with the `&man.csh.1;` entity):.

  Wrong: See `man csh` for more information.

  Right: See csh(1).

在句子之間空兩個空白
: Always use two spaces between sentences, as it improves readability and eases use of tools such as Emacs.

  A period and spaces followed by a capital letter does not always mark a new sentence, especially in names. “Jordan K. Hubbard” is a good example. It has a capital `H` following a period and a space, and is certainly not a new sentence.

For more information about writing style, see Elements of Style, by William Strunk.

13.3. 風格指南

由於說明文件是由衆多作者所維護，為了保持寫作風格的一貫性，請遵守下列撰寫風格慣例。

13.3.1. 大小寫

Tag 的部份都是用小寫字母，譬如是用 para，而非PARA。

而 SGML 內文則是用大寫字母表示，像是： <!ENTITY…> 及 <!DOCTYPE…> ，而不是 <!entity…> 及 <!doctype…> 。

13.3.2. 縮寫

縮寫字 (Acronym) 通常在書中第一次提到時，必須同時列出完整拼法，比如："Network Time Protocol (NTP)"。定義縮寫字之後，應該儘量只使用該縮寫字(而非完整詞彙，除非使用完整詞彙可以更能表達語意)來表達即可。通常每本書只會第一次提到時，才會列出完整詞彙，但若您高興也可以在每章第一次提到時又列出完整詞彙。

所有縮寫要包在acronym 標籤內。

13.3.3. 縮排

無論檔案縮排設定為何，每個檔案的第一行都不縮排。

未完的標籤會以多兩個空白來增加縮排，結尾的標籤則少兩個空白來縮減縮排。若已達 8 個空白，則以 tab 取代之。此外，在 tab 前面不要再用空白，也不要在每行後面加上空白。每個 tag 的內文若超過一行的話，則接下來的就多兩個空白以做縮排。

舉個例子，這節所用的寫法大致是下面這樣：

```
<chapter>
  <title>...</title>

  <sect1>
    <title>...</title>

    <sect2>
      <title>Indentation</title>

      <para>The first line in each file starts with no indentation,
 <emphasis>regardless</emphasis>  of the indentation level of
 the file which might contain the current file.</para>

      ...
    </sect2>
  </sect1>
</chapter>
```

有長屬性的標籤也是遵循一樣的原則。遵守縮排規則可以幫助編輯和作者了解哪些內容在標籤內：

```
<para>See the <link
    linkend="gmirror-troubleshooting">  Troubleshooting</link>
  section if there are problems booting.  Powering down and
  disconnecting the original <filename>ada0</filename>  disk
  will allow it to be kept as an offline backup.</para>

<para>It is also possible to journal the boot disk of a &os;
  system.  Refer to the article <link
    xlink:href="&url.articles.gjournal-desktop;">  Implementing UFS
    Journaling on a Desktop PC</link>  for detailed
  instructions.</para>
```

When an element is too long to fit on the remainder of a line without wrapping, moving the start tag to the next line can make the source easier to read. In this example, the `systemitem` element has been moved to the next line to avoid wrapping and indenting:

```
<para>With file flags, even
  <systemitem class="username"> root</systemitem> can be
  prevented from removing or altering files.</para>
```

Configurations to help various text editors conform to these guidelines can be found in 章 14, 編輯器設定.

13.3.4. 標籤風格

13.3.4.1. 標籤間距

同一縮排階層的標籤要以空一行來做區隔，而不同縮排階層的則不必。比如：

```
<article lang='en'>
  <articleinfo>
    <title>NIS</title>

    <pubdate>October 1999</pubdate>

    <abstract>
      <para>...
 ...
 ...</para>
    </abstract>
  </articleinfo>

  <sect1>
    <title>...</title>

    <para>...</para>
  </sect1>

  <sect1>
    <title>...</title>

    <para>...</para>
  </sect1>
</article>
```

13.3.4.2. 分隔標籤

像是 `itemizedlist` 這類的標籤事實上本身不含任何文字資料，必須得由其他標籤來補充內文。這類的標籤會獨用一整行。

另外，像是 `para` 及 `term` 這類的標籤並不需搭配其他標籤，就可附上文字資料，並且在標籤後面的同一行內即可立即寫上這些內文。

當然，這兩類的標籤結尾時也是跟上面道理相同。

不過，當上述這兩種標籤混用時，會有很明顯的困擾。

當第一類標籤的後面接上第二類標籤的話，那麼要把這兩類標籤各自分行來寫。後者標籤的段落，也是需要做適當縮排調整。

而第二類標籤結尾時，可以與第一類標籤的結尾放在同一行。

13.3.5. 空白變更

在提交修改時，請別在修改內容的同時也一起更改編排格式。

如此一來，像是翻譯團隊才能迅速找出你改了哪些內容，而不用費心思去判斷該行的改變，是由於格式重排或者內容異動。

舉例說明，若要在某段加上兩個句子，如此一來該段落的某行勢必會超出 80 縱列，這時請先 commmit 修改。接著，再修飾過長行落的換行，然後再次 commit 之。而第二次的 commit 紀錄，請明確說明這只是 whitespace-only（修改空白而已）的更改，如此一來，翻譯團隊就可以忽略第二次 commit 了 。

13.3.6. 不斷行空白

請避免一些情況下的斷行：造成版面醜醜的、或是須連貫表達的同一句子。斷行的情況會隨所閱讀的工具不同而有所不同。尤其是透過純文字瀏覽器來看 HTML 說明文件時會更明顯看到類似下面這樣不好的編排段落：

```
Data capacity ranges from 40 MB to 15
GB.  Hardware compression …
```

請使用   以避免同句子之間的斷行，以下示範如何使用不斷行空白：

- 在數字與單位之間：

  ```
  57600&amp;nbsp;bps
  ```

- 在程式名稱與版號之間：

  ```
  &amp;os;&amp;nbsp;9.2
  ```

- 多個單字的名稱之間（在套用到如 “The FreeBSD Brazilian Portuguese Documentation Project” 這種由三到四個字所組成的名稱時請小心）：

  ```
  Sun&amp;nbsp;Microsystems
  ```

13.4. 詞彙表

以下詞彙表列出使用在 FreeBSD 文件的正確拼法和大小寫。若找不到要找的詞彙，請詢問 FreeBSD documentation project mailing list 。

字詞	XML 程式碼	備註
CD-ROM	<acronym> CD-ROM </acronym>	
DoS (Denial of Service)	<acronym> DoS</acronym>	
email		
file system		
IPsec		
Internet		
manual page		
mail server		
name server		
Ports Collection		
read-only		
Soft Updates		
stdin	<varname> stdin</varname>	
stdout	<varname> stdout</varname>	
stderr	<varname> stderr</varname>	

字詞	XML 程式碼	備註
Subversion	`<application> Subversion </application>`	不要用大寫 SVN 來表示 Subversion 應用程式。若要表示指令，請使用 `<command> svn</command>` 。
UNIX®	`&unix;`	
userland		指那些要會用在使用者空間 (User space) 而非核心的東西。
web server		

章 14. 編輯器設定

Adjusting text editor configuration can make working on document files quicker and easier, and help documents conform to FDP guidelines.

14.1. Vim

Install from editors/vim or editors/vim-lite, then follow the configuration instructions in 節 14.1.2, “設置”.

14.1.1. 使用

Press P to reformat paragraphs or text that has been selected in Visual mode. Press T to replace groups of eight spaces with a tab.

14.1.2. 設置

Edit `~/.vimrc`, adding these lines to the end of the file:

```
if has("autocmd")
    au BufNewFile,BufRead *.sgml,*.ent,*.xsl,*.xml call Set_SGML()
    au BufNewFile,BufRead *.[1-9] call ShowSpecial()
endif " has(autocmd)

function Set_Highlights()
    "match ExtraWhitespace /^\s* \s*\|\s\+$/
    highlight default link OverLength ErrorMsg
    match OverLength /\%71v.\+/
    return 0
endfunction

function ShowSpecial()
    setlocal list listchars=tab:>>,trail:*,eol:$
    hi def link nontext ErrorMsg
    return 0
endfunction " ShowSpecial()

function Set_SGML()
    setlocal number
    syn match sgmlSpecial "&[^;]*;"
    setlocal syntax=sgml
    setlocal filetype=xml
    setlocal shiftwidth=2
    setlocal textwidth=70
    setlocal tabstop=8
    setlocal softtabstop=2
    setlocal formatprg="fmt -p"
    setlocal autoindent
    setlocal smartindent
    " Rewrap paragraphs
    noremap P gqj
    " Replace spaces with tabs
    noremap T :s/        /\t/<CR>
    call ShowSpecial()
    call Set_Highlights()
    return 0
endfunction " Set_SGML()
```

14.2. Emacs

Install from editors/emacs or editors/emacs-devel.

14.2.1. 檢驗

Emacs's nxml-mode uses compact relax NG schemas for validating XML. A compact relax NG schema for FreeBSD's extension to DocBook 5.0 is included in the documentation repository. To configure nxml-mode to validate using this schema, create `~/.emacs.d/schema/schemas.xml` and add these lines to the file:

```
<locatingRules xmlns="http://thaiopensource.com/ns/locating-rules/1.0">
  <documentElement localName="section" typeId="DocBook">
  <documentElement localName="chapter" typeId="DocBook">
  <documentElement localName="article" typeId="DocBook">
  <documentElement localName="book" typeId="DocBook">
  <typeId id="DocBook" uri="/usr/local/share/xml/docbook/5.0/rng/docbook.rnc">
</locatingRules>
```

14.2.2. 使用 Flycheck 和 Igor 自動化校對

The Flycheck package is available from Milkypostman's Emacs Lisp Package Archive (MELPA). If MELPA is not already in Emacs's packages-archives, it can be added by evaluating

```
(add-to-list 'package-archives '("melpa" . "http://stable.melpa.org/packages/") t)
```

Add the line to Emacs's initialization file (one of `~/.emacs` , `~/.emacs.el` , or `~.emacs.d/init.el`) to make this change permanent.

To install Flycheck, evaluate

```
(package-install 'flycheck)
```

Create a Flycheck checker for textproc/igor by evaluating

```
(flycheck-define-checker igor
  "FreeBSD Documentation Project sanity checker.

See URLs http://www.freebsd.org/docproj/ and
http://www.freshports.org/textproc/igor/."
  :command ("igor" "-X" source-inplace)
  :error-parser flycheck-parse-checkstyle
  :modes (nxml-mode)
  :standard-input t)

  (add-to-list 'flycheck-checkers 'igor 'append)
```

Again, add these lines to Emacs's initialization file to make the changes permanent.

14.2.3. FreeBSD 說明文件特定的設定

To apply settings specific to the FreeBSD documentation project, create `.dir-locals.el` in the root directory of the documentation repository and add these lines to the file:

```
;;; Directory Local Variables
;;; For more information see (info "(emacs) Directory Variables")

((nxml-mode
  (eval . (turn-on-auto-fill))
  (fill-column . 70)
  (eval . (require 'flycheck))
  (eval . (flycheck-mode 1))
  (flycheck-checker . igor)
  (eval . (add-to-list 'rng-schema-locating-files "~/.emacs.d/schema/schemas.xml"))))
```

14.3. nano

Install from editors/nano or editors/nano-devel.

14.3.1. 設置

Copy the sample XML syntax highlight file to the user's home directory:

```
% cp /usr/local/share/nano/xml.nanorc ~/.nanorc
```

Add these lines to the new `~/.nanorc` .

```
syntax "xml" "\.([jrs]html?|xml|xslt?)$"
# trailing whitespace
color ,blue "[[:space:]]+$"
# multiples of eight spaces at the start a line
# (after zero or more tabs) should be a tab
color ,blue "^([TAB]*[ ]{8})+"
# tabs after spaces
color ,yellow "( )+TAB"
# highlight indents that have an odd number of spaces
color ,red "^(([ ]{2})+|(TAB+))*[ ]{1}[^ ]{1}"
# lines longer than 70 characters
color ,yellow "^(.{71})|(TAB.{63})|(TAB{2}.{55})|(TAB{3}.{47}).+$"
```

Process the file to create embedded tabs:

```
% perl -i'' -pe 's/TAB/\t/g' ~/.nanorc
```

14.3.2. 使用

Specify additional helpful options when running the editor:

```
% nano -AKipwz -r 70 -T8  chapter.xml
```

Users of csh(1) can define an alias in `~/.cshrc` to automate these options:

```
alias nano "nano -AKipwz -r 70 -T8"
```

After the alias is defined, the options will be added automatically:

```
% nano chapter.xml
```

章 15. 他山之石

This document is deliberately not an exhaustive discussion of XML, the DTDs listed, and the FreeBSD Documentation Project. For more information about these, you are encouraged to see the following web sites.

15.1. FreeBSD 說明文件計劃

- FreeBSD 說明文件計劃網頁
- FreeBSD 使用手冊

15.2. XML

- W3C's XML 網頁 SGML/XML 網頁

15.3. HTML

- 全球資訊網協會
- The HTML 4.0 規格表

15.4. DocBook

- The DocBook 技術委員會，DocBook DTD的維護者
- DocBook：The Definitive Guide，DocBook DTD 的線上說明文件。
- The DocBook Open Repository contains DSSSL stylesheets and other resources for people using DocBook

附錄 A. 範例

These examples are not exhaustive—they do not contain all the elements that might be desirable to use, particularly in a document's front matter. For more examples of DocBook markup, examine the XML source for this and other documents available in the Subversion `doc` repository, or available online starting at `http://svnweb.FreeBSD.org/doc/` .

A.1. DocBook book

範例 A.1. DocBook book

```
<!DOCTYPE book PUBLIC "-//FreeBSD//DTD DocBook XML V5.0-Based Extension//EN"
 "http://www.FreeBSD.org/XML/share/xml/freebsd50.dtd">

<book xmlns="http://docbook.org/ns/docbook"
  xmlns:xlink="http://www.w3.org/1999/xlink" version="5.0"
  xml:lang="en">

  <info>
    <title>An Example Book</title>

    <author>
      <personname>
        <firstname>Your first name</firstname>
        <surname>Your surname</surname>
      </personname>

      <affiliation>
 <address>
   <email>foo@example.com</email>
 </address>
      </affiliation>
    </author>

    <copyright>
      <year>2000</year>
      <holder>Copyright string here</holder>
    </copyright>

    <abstract>
      <para>If your book has an abstract then it should go here.</para>
    </abstract>
  </info>

  <preface>
    <title>Preface</title>

    <para>Your book may have a preface, in which case it should be placed
      here.</para>
  </preface>

  <chapter>
    <title>My First Chapter</title>

    <para>This is the first chapter in my book.</para>

    <sect1>
      <title>My First Section</title>

      <para>This is the first section in my book.</para>
```

```
    </sect1>
  </chapter>
</book>
```

A.2. DocBook `article`

範例 A.2. DocBook `article`

```
<!DOCTYPE article PUBLIC "-//FreeBSD//DTD DocBook XML V5.0-Based Extension//EN"
 "http://www.FreeBSD.org/XML/share/xml/freebsd50.dtd">

<article xmlns="http://docbook.org/ns/docbook"
  xmlns:xlink="http://www.w3.org/1999/xlink" version="5.0"
  xml:lang="en">

  <info>
    <title>An Example Article</title>

    <author>
      <personname>
        <firstname>Your first name</firstname>
        <surname>Your surname</surname>
      </personname>

      <affiliation>
 <address>
   <email>foo@example.com</email>
 </address>
      </affiliation>
    </author>

    <copyright>
      <year>2000</year>
      <holder>Copyright string here</holder>
    </copyright>

    <abstract>
      <para>If your article has an abstract then it should go here.</para>
    </abstract>
  </info>

  <sect1>
    <title>My First Section</title>

    <para>This is the first section in my article.</para>

    <sect2>
      <title>My First Sub-Section</title>

      <para>This is the first sub-section in my article.</para>
    </sect2>
  </sect1>
</article>
```

索引