LDP Author Guide

Jorge Godoy

Conectiva S.A.
Publishing Department

<godoy@metalab.unc.edu>

Emma Jane Hogbin

<emmajane@xtrinsic.com>

Mark F. Komarinski

<mkomarinski@wayga.org>

David C. Merrill

david -AT- lupercalia.net

2005-03-04

Revision History

Revision 4.84.74.64.54.44.34.24.1	2006-04-202005-03-042005-01-232004-07-142004-04-192004-04-042004-04-022004-01-27	Revised by: MGejhejhejhejhejhejh

Added notes about prefered submission formats, corrected links, packaged templates.Typo fixed in sample DocBook markup. Added new web-based authoring tool and information on LaTeX to DocBook conversions.Typos fixed in xmlto notes and book template. Copied information about DocBook-capable word processing tools into the "Converting Documents to DocBook XML" Appendix; added new XML editors; and information about tools to convert other formats to DocBook XML.Updated information regarding CVS accounts and connecting to the CVS server.Added editor credit requirements to the Using DocBook section. Updated the submission procedure. New documents can now only be added by one of the Review Coordinators after the successful completion of each of the required reviews.Removed the section Contributing to The LDP (replaced by Summary of The LDP Process).Added references for LyX to DocBook conversions in the bibliography.Updated the license requirements and added them to the table of contents (moved them out of the sub-section).

This guide describes the process of submitting and publishing a document with The Linux Documentation Project (TLDP). It includes information about the tools, toolchains and formats used by TLDP. The document's primary audience is new TLDP authors, but it also contains information for seasoned documentation authors.

Table of Contents

Chapter 1. About this Guide....1
- 1.1. About this Guide....1
- 1.2. About The LDP....1
- 1.3. Feedback....2
- 1.4. Copyrights and Trademarks....2
- 1.5. Acknowledgments and Thanks....2
  - 1.5.1. Version 1 - Version 3....2
  - 1.5.2. Version 4....2
- 1.6. Document Conventions....2

Chapter 2. Authoring TLDP Documents: An Introduction....4
- 2.1. Summary of The LDP Process....4
- 2.2. Mailing Lists....5

Chapter 3. Writing Your Proposal....6
- 3.1. Choosing a Subject....6
- 3.2. Scope of Your Document....6
  - 3.2.1. Documentation Templates....7
- 3.3. Unmaintained and Out-of-date Documents....7
- 3.4. Developing an Outline....8
- 3.5. Research....9

Chapter 4. Write....10
- 4.1. Writing the Text....10
  - 4.1.1. Writing Style and Style Guides....10
  - 4.1.2. On-line Writing Resources....11
- 4.2. Edit and Proofread the Text....11
- 4.3. Tools for Writing, Editing and Maintaining your Document....12
  - 4.3.1. Editing Tools....12
  - 4.3.2. Concurrent Versions System (CVS)....12
  - 4.3.3. Spell Check....13

Chapter 5. Markup....14
- 5.1. Markup: A General Overview....14
- 5.2. DocBook: What it is and why we use it....14
- 5.3. XML and SGML: Why we use XML....15
- 5.4. Markup Languages Accepted by TLDP....16

Chapter 6. Distributing Your Documentation....17
- 6.1. Before Distributing Your Documentation....17
- 6.2. Licensing and Copyright....17
  - 6.2.1. Copyright....18
  - 6.2.2. Disclaimer....18
  - 6.2.3. Licensing source code....18
- 6.3. Acknowledgments....18
- 6.4. TLDP Review Process....19
- 6.5. Submission to LDP for publication....20

Table of Contents

Chapter 7. Maintenance....21
- 7.1. Maintaining Your Document....21
- 7.2. Fixing Errors....21
  - 7.2.1. Fixing Your Own Documents....21
  - 7.2.2. Fixing Other Documents in the Collection....21

References....23
- Markup and general information....23
  - DocBook References....23
  - LinuxDoc....24
  - Converting Other Formats to DocBook....24
  - LDP templates, tools & links....24
  - DocBook Transformations....25
  - General Writing Links and Style Guides....25
  - Related TLDP Documents....26
  - Software: CVS....27
  - Software: Emacs....27
  - XML Authoring Tools....27
  - Documentation Licenses....27

Appendix A. Templates....29
- A.1. Document Templates....29
- A.2. Style Sheets....29
- A.3. GNU Free Documentation License....29

Appendix B. System Setup: Editors, Validation and Transformations....30
- B.1. Tools for your operating system....30
- B.2. Editing tools....30
  - B.2.1. Word Processors....31
  - B.2.2. Text Editors....34
- B.3. Validation....42
  - B.3.1. Why Validate Your Document....42
  - B.3.2. Validation for the Faint of Heart....42
  - B.3.3. Validation for the Not So Faint Of Heart....42
  - B.3.4. Creating and modifying catalogs....43
  - B.3.5. Validating XML....45
- B.4. Transformations....47
  - B.4.1. DSSSL....48
  - B.4.2. The docbook-utils Package....50
  - B.4.3. XSL....51
- B.5. DocBook DTD....52
- B.6. Formatting Documents....52
  - B.6.1. Inserting a summary on the initial articles page....52
  - B.6.2. Inserting indexes automatically....53

Appendix C. Concurrent Version System (CVS)....55
- C.1. Getting a CVS account....55
- C.2. Using CVS....55

Table of Contents

Appendix C. Concurrent Version System (CVS)
C.2.1. Setting Up Your CVS Account....55
C.2.2. Getting the Documents....56
C.2.3. CVS Commands....56
C.3. CVS Resources....58

Appendix D. DocBook: Sample Markup....59
D.1. General Tips and Tricks....59
D.1.1. Useful markup....59
D.2. <section> and <sectN>: what's the difference?....62
D.3. Command Prompts....62
D.4. Encoding Indexes....63
D.5. Inserting Pictures....64
D.5.1. Graphics formats....65
D.5.2. Alternative Methods....65
D.6. Markup for Metadata....66
D.6.1. Crediting Translators, Converters and Co-authors....66
D.6.2. <revremark>s....66
D.6.3. Revision History....67
D.6.4. Date formats....67
D.6.5. Sample Article (or Book) Information Element....67
D.7. Bibliographies....68
D.8. Entities (shortcuts, text macros and re-usable text)....69
D.9. Customizing your HTML files....70
D.9.1. HTML file names....70
D.9.2. Headers and Footers....71

Appendix E. Converting Documents to DocBook XML....72
E.1. Text to DocBook....72
E.2. OpenOffice.org to DocBook....72
E.2.1. Open Office 1.0.x....72
E.2.2. Open Office 1.1....73
E.3. Microsoft Word to DocBook....74
E.4. LaTeX to DocBook....74
E.5. LyX to DocBook....74
E.6. DocBook to DocBook Transformations....75
E.6.1. XML and SGML DocBook....75
E.6.2. Changing DTDs....76

Glossary....77

Appendix F. GNU Free Documentation License....81
F.1. 0. PREAMBLE....81
F.2. 1. APPLICABILITY AND DEFINITIONS....81
F.3. 2. VERBATIM COPYING....82
F.4. 3. COPYING IN QUANTITY....82
F.5. 4. MODIFICATIONS....83
F.6. 5. COMBINING DOCUMENTS....84

Table of Contents

Appendix F. GNU Free Documentation License
F.7. 6. COLLECTIONS OF DOCUMENTS....84
F.8. 7. AGGREGATION WITH INDEPENDENT WORKS....84
F.9. 8. TRANSLATION....85
F.10. 9. TERMINATION....85
F.11. 10. FUTURE REVISIONS OF THIS LICENSE....85
F.12. Addendum....85
Notes....86

Chapter 1. About this Guide

1.1. About this Guide

This document was started on Aug 26, 1999 by Mark F. Komarinski after two day's worth of frustration getting tools to work. If even one LDP author is helped by this, then I did my job.

Version 4 of this document was released in early 2004 by Emma Jane Hogbin. A complete overhaul of the document was done to separate the authoring HOWTOs from the technical HOWTOs. The review took approximately eight months.

The newest version of this document can be found at the LDP homepage http://www.tldp.org. The original DocBook, HTML, and other formats can be found there.

There are many ways to contribute to the Linux movement that do not require the ability to produce software. One such way is to write documentation. The availability of easy-to-understand, technically accurate documentation can make a world of difference to someone who is struggling with Linux software. This Guide is designed to help you research and write a HOWTO which can be submitted to the LDP. The appendices also include sample templates, markup tips and information on how to transform your document from DocBook to another format (such as HTML) for easier proofreading.

1.2. About The LDP

The Linux Documentation Project (LDP) was started to provide new users a way of quickly getting information about a particular subject. It not only contains a series of books on administration, networking, and programming, but also has a large number of smaller works on individual subjects, written by those who have used it. If you want to find out about printing, you get the Printing HOWTO. If you want to do find out if your Ethernet card works with Linux, grab the Ethernet HOWTO, and so on.

The LDP provides documents to the world in a variety of convenient formats and also accepts submissions in a number of formats. The current standard for storing the source documentation is a format known as DocBook, see Section 5.2.

> The Linux Documentation Project (LDP) is working on developing free, high-quality documentation for the GNU/Linux operating system. The overall goal of the LDP is to collaborate in all of the issues of Linux documentation. This includes the creation of "HOWTOs" and "Guides". We hope to establish a system of documentation for Linux that will be easy to use and search. This includes the integration of the manual pages, info docs, HOWTOs, and other documents.
>
> --LDP Manifesto located at http://www.tldp.org/manifesto.html

The human readable version goes more like this: The LDP consists of a large group of volunteers who are working on documentation about Linux and the programs which run on the Linux kernel. These documents exist primarily as shorter HOWTOs and longer Guides. Both are available from http://www.tldp.org/. This Guide focuses primarily on how to write your own HOWTOs for submission to the LDP.

1.3. Feedback

Send feedback to `<discuss@en.tldp.org>`. Please reference the title of this document in your email. Please note: you must be subscribed in order to send email to the list.

1.4. Copyrights and Trademarks

Copyright 1999-2002 Mark F. Komarinski, David C. Merrill, Jorge Godoy

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the appendix entitled "GNU Free Documentation License."

1.5. Acknowledgments and Thanks

1.5.1. Version 1 - Version 3

Thanks to everyone that gave comments as I was writing this. This includes David Lawyer, Deb Richardson, Daniel Barlow, Greg Ferguson, Mark Craig and other members of the `<discuss@en.tldp.org>` list. Some sections I got from the HOWTO Index and the sgmltools documentation. The sections on network access to CVS was partially written by Sergiusz Pawlowicz (`<ser@metalab.unc.edu>`). Sections on DocBook were written by Jorge Godoy (`<godoy@conectiva.com>`). A great deal of thanks to both of them for their help.

1.5.2. Version 4

Thanks to Tabatha Marshall and Machtelt Garrels (Tille) for making sure I actually finished the document. Thanks to my reviewers: Charles Curley, Martin Brown and Tille; and to Saqib Ali for his on-line transformation and validation tools. I have also incorporated a number of useful emails from the LDP mailing lists. The original authors are credited within the document. Special personal thank yous are extended to Steve Champeon for getting me interested in markup languages and for being a wonderful mentor; and to my partner, Graig Kent, for being outrageously supportive. [EJH]

1.6. Document Conventions

This document uses the following conventions[1]:

Descriptions	**Appearance**
Information requiring special attention	This is a warning.
Caution	This cautions the reader.
Hint	

	This is a hint.
Notes	This is a note.
File Names	`file.extension`
Directory Names	`directory`
Commands to be typed	**command**
Applications Names	application
Prompt of users command under bash shell	bash$
Prompt of root users command under bash shell	bash#
Prompt of user command under tcsh shell	tcsh$
Environment Variables	`VARIABLE`
Emphasized word	*word*
Quoted text	"quote"
Code Example	`<para>Beginning and end of paragraph</para>`

Chapter 2. Authoring TLDP Documents: An Introduction

2.1. Summary of The LDP Process

The following section outlines the process of creating and/or maintaining a document for the Linux Documentation Project. This section includes all steps--some of which may not be relevant to your specific document.

1. Join the discuss mailing list. Authors who are interested in taking over the maintenance of someone else's document should also join this list. This is to make sure the LDP knows who is working on what documentation.

   If you have not yet written your documentation, please review our documents (current, unmaintained and in progress) and submit a proposal to the list. Your proposal should include reasons why your document will be different than those already in the collection; or identify a subject that is currently missing from our documentation. For more information about writing proposals, please read Chapter 3.

   For more information about the mailing lists, please read Section 2.2 or visit lists.tldp.org to subscribe. If your document has already been written, please submit a copy to the discuss list (or include the URL of where it can be found).
2. Write your document. If your document has not yet been written, please be sure to email the discuss list before you start writing. *You may choose whatever format you feel most comfortable in to write your document.* If it is not one of the formats accepted by the LDP a volunteer will convert it for you. For more information on writing technical documentation, please read Chapter 4.
3. If you are adding your own markup, you may also want to join the docbook mailing list. For more information about the LDP DocBook list please read Section 2.2. If you would like to start with a template, you may find the templates in Appendix A useful. There is also a general introduction to markup in Chapter 5 and a section full of examples at Appendix D.
4. Submit your document for technical, language and metadata reviews. Do this by emailing your document to `<submit@en.tldp.org>`. In the subject line be sure give the title of the document. In the body of the email say that you are ready for the review process. Outline any additional reviews your document may have already received. You should be assigned a reviewer within the week. The reviews may take an additional week each. For more information about this process, please read Chapter 6.

   If your document is not already in DocBook or LinuxDoc format, a reviewer will convert it for you.
5. Once your document has been through each of the reviews a Review Coordinator will add it to the CVS, update the version number to 1.0 and have the document published on the public Web site. For more information about your final submission to the LDP, please read Section 6.5.

If you don't submit your document in DocBook format

The volunteer adding markup to your document may choose any accepted markup language. The Author Guide, however, will refer only to DocBook. If you are submitting plain text or some other format, please let the LDP know if you prefer to maintain your document in either LinuxDoc or DocBook, which are the accepted formats for end-results.

2.2. Mailing Lists

You can subscribe to the following mailing lists:

- First is `<discuss@en.tldp.org>`, which is the main discussion group of the LDP.
- Another is the `<docbook@en.tldp.org>` list, which is for questions about DocBook use including markup and transformations. If you run into trouble with a particular markup tag, you can send your question here for answers.

You can subscribe to either list by sending a request message to either `<discuss-subscribe@en.tldp.org>` or `<docbook-subscribe@en.tldp.org>`. The subject of your message should read "subscribe" (no quotes). To remove yourself from the list, send an E-mail with the subject of "unsubscribe" to `<discuss-unsubscribe@en.tldp.org>` or `<docbook-unsubscribe@en.tldp.org>`.

If you are interested in DocBook beyond the simple markup of your LDP document, you may want to consider joining one of the OASIS DocBook mailing lists. Please see http://docbook.org/mailinglist/index.html for more information.

Chapter 3. Writing Your Proposal

3.1. Choosing a Subject

It is likely that if you are reading the Author Guide, you already have a subject in mind. The idea may have come from your own frustrations in trying to get something to work; or maybe you are already writing or have written documentation for other purposes and you want to submit it to the LDP. For example, if you posted a question to a mailing list and it took many posts to resolve the problem -- that might be an incentive to write documentation.

Regardless of how you picked your subject, it is important that the LDP community understand why your documentation should be included in its collection. If someone has requested documentation, or you worked with a mailing list to resolve a problem you should include the details in your proposal to the LDP discuss mailing list. It may help others to understand the need for your specific document.

3.2. Scope of Your Document

Now that you've got a subject, you will need to decide the *scope* of the document. The scope or subject area should be:

1. **Clearly defined.** Define the boundaries of your subject area before you begin. Do not repeat information that is in another HOWTO and do not leave gaps of information between your HOWTO and someone else's HOWTO.
2. **Not too broad, and not too narrow.** If you try to cover too much information you may sacrifice depth. It is better to cover a small subject area in detail than to cover a large subject area poorly. Linux tools are known for doing exactly one thing and doing that one thing *well*. Similarly, your HOWTO should cover one subject and cover it *well*.

   If the scope of your proposed document is very narrow, it might be better to include your information as part of another HOWTO. This makes it easier for readers to find the HOWTO they need. Search the LDP repository for topics which relate to your document. If you find a document which is a good match, email the author and ask them if they would like to include your contribution.
3. **Undocumented.** Before documenting a particular subject, always do a web search (and specifically a search within the LDP documents) to see if your topic is already covered in another document. If it is, refer to the other document instead of duplicating the information within your own document. You may wish to include a brief summary of information that can be found in other documents.

   If the HOWTO already in place is insufficient or needs updating, contact the author and offer to help. See also Section 3.3 for taking over old or unmaintained documents.

   Most authors appreciate any help offered. Additionally, sending comments and remarks to the author is usually regarded both as a reassurance and a reward: to the author, feedback is the ultimate proof that writing the documentation was not a pointless effort.
4. **Pre-approved by the LDP.** Before you proceed with your HOWTO, post to the discuss list and get some feedback from other LDP volunteers. Checking with the list *before* you begin can save you headaches *later*.

Stay in touch!

Joining the discuss list and following it regularly, even if you never post, is a good way to stay current on the activities, needs and policies of the LDP.

3.2.1. Documentation Templates

After you've decided the scope of your document you should start to consider the type of document that you will write. There are a number of different LDP documentation templates: Guides, HOWTOs, man pages and FAQs. Rahul Sundaram describes their scope in the Linux Documentation Project (LDP) FAQ. Here is a brief overview of what they are with pointers on how you can get started writing your own:

- **Guides.** A guide covers a broad subject and is quite long. The Author Guide (this document) is a guide. Other guides include: Introduction to Linux: A hands on guide, The Linux Kernel Module Programming Guide, etc. A full list of guides is available from: Linux Project Documentation Guides. Guides use the "book" templates located in Appendix A.
- **HOWTOs.** A HOWTO is typically a set of instructions that outlines, step-by-step, how a specific task is accomplished. Examples of HOWTOs are: CDROM-HOWTO Module-HOWTO. A full list of HOWTOs is available from: Single List of HOWTOs (warning: it's a BIG page). HOWTOs typically use the "article" template and are output to multiple HTML pages by default. Templates are available in Appendix A.
- **man pages.** man (Manual) pages are the standard form of help available for many linux applications and utilities. They can be viewed by typing **man applicationname** at a prompt. A full list of man pages is available from: Linux Man Pages. Since man pages are bundled with software there is no LDP template at this time.
- **FAQs.** FAQs (Frequently Asked Questions) are a list of questions and answers to help prevent new users from asking the same questions over and over again. A full list of FAQs is available from: Linux Documentation Project FAQs. FAQs typically use the "article" template with some specific DocBook elements to form the Question/Answer structure. You can find a template for writing a FAQ in Appendix A.

mini-HOWTOs and HOWTOs

The LDP no longer distinguishes between HOWTOs and mini-HOWTOs. All previously written mini-HOWTOs have been included in longer HOWTOs. All new documents must be at least HOWTO in length. This means the documents should cover a bigger subject area rather than a small one. If your document is very small you may wish to submit it for inclusion in another, larger HOWTO that already exists. If you're not sure what size your document is, email the discuss list and ask for feedback.

3.3. Unmaintained and Out-of-date Documents

If you wish to become the "owner" for an unmaintained document there are several steps you must go through.

- Contact the author. Make sure the author no longer wishes to maintain the document in question. Note that the E-mail address shown may no longer be valid. In that case, try a search for the author. If the original author of a document cannot be found after a "good-faith" effort, let us know (`<discuss@en.tldp.org>`--subscription required).
- Determine if a more up-to-date copy of the document exists, outside of what is available on the LDP. If so, try to secure a copy for yourself to work on.
- Inform the LDP which document you would like to maintain, so that we can track who is working on what and prevent duplication of effort. We also suggest that you join the LDP general discussion list

(`<discuss@en.tldp.org>`). This step is also required for new documents.
- Submit the document to the LDP with any intended modifications. Make sure to continue to reference the original author somewhere within the document (Credits, Revision History, etc.). Once the document is re-submitted, we will remove the entry from the list of unmaintained documents.

Feedback wanted

Some of unmaintained documents may be outdated, or the content may be covered in another (actively maintained) HOWTO. If that is the situation, contact us (preferably on the discuss mailing list) and let us know.

3.4. Developing an Outline

Before you actually begin writing, prepare an outline. An outline will help you to get a clear picture of the subject matter and allow you to concentrate on one small part of the HOWTO at a time.

Unless your HOWTO is exceptionally small, your outline will probably be multilevel. When developing a multilevel outline, the top level should contain general subject areas, and sub-headings should be more detailed and specific. Look at each level of your outline independently, and make sure it covers all key areas of the subject matter. Sub-headings should cover all key areas of the heading under which they fall.

Each item in your outline should follow the item before it, and lead into the item that comes next. For example, a HOWTO about a particular program shouldn't have a section on *configuration* before one on *installation*.

You may choose to use the following outline for a HOWTO about using a specific program:

- development history
- where to download the software from
- how to install the software
- how to configure the software
- how to use the software

You may find it helpful to try a little "card sorting" at this stage of the writing process. Write all of your mini subject areas onto pieces of paper. Now sort these pieces of paper into main headings and their sub-sections. It may help to shuffle the slips of paper before you start. This will help to give you a fresh set of eyes while working.

When you are comfortable with your outline, look it over once more, with a critical eye. Have you covered every relevant topic in sufficient detail? Have you not wandered beyond the scope of the document? It is a good idea to show your outline to others (including The LDP discuss list) to get some feedback. Remember: it is much easier to reorganize your document at the outline stage than doing this after writing it.

Writing a HOWTO made easy

For help writing your HOWTO outline, and getting a head start on the markup of your document, check out The LDP HOWTO Generator. Note that this is for generating HOWTOs. Templates for FAQs and Guides are available in Appendix A.

You're not alone

You might have noticed a theme developing here. Just like Free software, Free documentation is best when you "release early, release often." The discuss list includes many experienced LDP authors, and you would be wise to seek their advice when making decisions about your contribution.

3.5. Research

While you are working on your outline you will most likely research your topic--especially to confirm the document you are about to write does not yet exist! Here are a few pointers that will keep you from pulling out your hair later:

- **Compile your resources as you research.** It is almost guaranteed you will not remember where to find a critical piece of information when you need it most. It will help to bookmark important (and even not so important) pages as you go. Make sure the bookmark's title reflects why the page is important to you. If there are multiple key ideas in one page, you may want to bookmark the same page with different titles.
- **Assume your most important resource will disappear.** The dreaded "Error 404: Page not found". Even if you have bookmarked a page it may not be there when you return to it. If a page contains a really critical piece of information: make a copy. You may do this by creating a text file with the title of the document, the author's name, the page's URL and the text of the page into a text file on your computer. You might also choose to "print" the file to a PDF (save as or convert to PDF format will capture the original URL on the page if you're using a smart browser).
- **Start your "Resources" page now.** As you find pages of interest add them to a Resources document. You may do this by exporting your bookmarks or by keeping a separate text file with the Resources sorted by sub-category. A little effort now will save you a lot of time later.

  There is more information about the DocBook markup of bibliographies in Section D.7.
- **Write down subject areas as you go.** If you are card sorting you may find it particularly useful to write topic cards as you find pages that cover that specific topic. At the top of the card write the subject area. In the main area of the card write a few notes about what you might cover under this topic--include the titles of pages that contain important information. If a sub-topic gets too big you may want to divide it into multiple cards.
- **Separate generic information from version-specific information.** A new version of the software that you describe might be released the day after you release your document. Other things, like where to download the software, won't change. Alternatively, you may choose to document old problems with specific software as a way of encouraging readers to upgrade to the latest version available: "Version X of the software is known for a specific bug. The bug was fixed as of Version Y."
- **Save all related emails.** People will often have interesting insight into the problem that you are writing about. Any questions that are asked about your topic should be addressed in the final document. If you are writing about software make sure to ask people what system they are using. Add information in your document about which system configurations your instructions have been tested on. (Having lots of friends with moderately different configurations can be very beneficial!) All of these personal experiences can add greatly to your final documentation.

Chapter 4. Write

4.1. Writing the Text

By now you should have organized your document; you collected bits of raw information and inserted them into the outline. Your next challenge is to massage all of the raw data you've collected into a readable, entertaining and understandable whole. If you are working from an existing document make sure any new pieces of information are in the best possible places.

It has taken quite a bit of work to get to the point where you can actually start writing, hasn't it? Well, the hard work begins to pay off for you now. At this stage, you can begin to really use your imagination and creativity to communicate this heap of information. Don't be too clever though! Your readers are already struggling with new concepts--do not make them struggle to understand your language as well.

There are a number of classic guides to writing--many of which are available on-line. Their language will seem old, but the messages are still valuable to authors today. These are listed in . Also listed in the resources section are a variety of sites that have information specific to technical writing.

The Author Guide wouldn't be complete without mentioning the Plain Language movement. Although directed at simplifying government documents, Writing user-friendly documents is quite useful. It includes before and after writing samples. There's also a PlainTrain writing tutorial.

And any document that discusses writing for the web wouldn't be complete without a nod toward useit.com. The following articles may be of specific interest:

- Writing for the Web
- Information pollution
- Be Succinct! (Writing for the Web)

There are many, many resources for writing web documents--a quick web search for "web writing" will find lots of resources. Don't get too distracted, though: the ultimate goal is to write, not to read about writing!

4.1.1. Writing Style and Style Guides

There are a number of industry style guides which define how language should be used in documents. A common example for American English is the Chicago Manual of Style. It defines things like: whether a period (.) should be inside or outside of "quotes"; and the format for citing another document. A style guide helps to keep documents consistent--most corporations will follow a style guide to format media releases and other promotional material. Style guides mays also define how words should be spelled (is it color or colour?).

The LDP does not require a specific style guide; however, you should use a consistent style throughout your document. Your document should be spell checked for a single language (e.g. American English vs. British English). The Reviewer's HOWTO currently lists a number of conventions for LDP documents and it is as close as it gets to an official LDP Style Guide.

A personal glossary

It helps to make a list of terms that you were new to you when you first started researching and writing your document. You can refer to this list while writing the text. You may also want to include it as a glossary in your final document.

You can save yourself a lot of time in the editing phase if you decide how you will write your document ahead of time. If you are taking over someone else's document you may need to either: modify your own style to fit the current document; or edit the document so that it melds with the style you would like to use from now on.

From a writing style perspective, use of the first-person in a HOWTO adds to its charm--an attribute most other documentation sorely lacks. Don't be afraid to speak for yourself--use the word "I" to describe your personal experiences and opinions.

4.1.2. On-line Writing Resources

In the section, you will find a list of resources that cover the subject better than this guide could hope to. Consult them, and follow their advice.

4.2. Edit and Proofread the Text

Once you've written the text of your HOWTO, it is time to polish and refine it. Good editing can make the difference between a good HOWTO and a great one.

One of the goals of editing is to remove *[extraneous]* material that has crept its way into your document. You should go through your HOWTO carefully, and ruthlessly delete anything that does not contribute to the reader's understanding of the subject matter. It is natural for writers to go off on tangents while in the process of writing. Now is the time to correct that. It often helps to leave a bit of time between when you write a document and when you edit it.

Make sure that the content of every section matches the title precisely. It's possible that your original subject heading is no longer relevant. If you've strayed from your original heading it could mean one of the following: the original title was poorly worded, the new material should actually go in a different section, or the new material is not really necessary for your document. If you need to change a title, check to see if the original subject heading is still important. If it is, make sure that topic is covered somewhere else in the document.

When editing and proofing your work, check for obvious mistakes, such as spelling errors and typos. You should also check for deeper, but less obvious errors as well, such as "holes" in the information. If you are creating a set of instructions it may help to test them on a new machine. Sometimes there are packages that need to be installed which you've forgotten to mention in your documentation, for instance.

When you are completely satisfied with the quality and accuracy of your work, forward it to someone else for third-party proofing. You will be too close to the work to see fundamental flaws. Have others test the instructions as well. Make sure they follow exactly what you have written. Ask anyone who tests your documentation to make specific notes in any places where they didn't follow your instructions (and the reason why they didn't follow them). For example: "I skipped step 2 because I already have the required packages installed."

In a sense, editing is like code review in software development. Having a programmer review their own code doesn't make much sense, and neither does having a writer edit their own document. Recruit a friend, or write the discuss list to find a volunteer to proofread before submitting your document. You may also want to submit your document to a mailing list that is relevant to your document's topic. List members should be able to help check the facts, clarity of your instructions and language of the document.

Native speaker?

If you are writing in a language in which you are not fluent, find an editor who is. Technical documentation, more than any other type of writing, must use extremely precise grammar and vocabulary. Misuse of language makes your document less valuable.

4.3. Tools for Writing, Editing and Maintaining your Document

Reminder

You do *not* need to submit your initial document to the LDP in anything more than plain text! Other open submission formats are accepted as well, for instance OpenOffice documents, RTF files or HTML. The LDP volunteers will convert your document to DocBook for you. Once it has been converted you will need to maintain your document in DocBook format, but that should be obvious.

4.3.1. Editing Tools

You may use any word processing or text editing tool to write your initial document. When you get to the markup stage you may want to use a text editor which recognizes DocBook files. At a minimum a program which adds syntax highlighting to your markup will make life a lot easier. For a description of editors which can handle DocBook files please skip to Section B.2.

4.3.2. Concurrent Versions System (CVS)

The LDP provides optional CVS access to its authors. This enables collaborative writing and has the following positive effects:

1. CVS will keep an off-site backup of your documents. In the event that you hand over a document to another author, they can just retrieve the document from CVS and continue on. In the event you need to go back to a previous version of a document, you can retrieve it as well.
2. However difficult from an organizational point of view, it's great to have multiple people working on the same document. CVS enables you to do this. You can have CVS tell you what changes were made by another author while you were editing your copy, and integrate those changes.
3. CVS keeps a log of what changes were made. These logs (and a date stamp) can be placed automatically inside your documents when they are published.
4. CVS can be combined with scripts to automatically update the LDP web site with new documentation as it's written and submitted. This is not in place yet, but it is a goal. Currently, CVS updates signal the HOWTO coordinator to update the LDP web page, meaning that if you use CVS, you're not required to e-mail your XML code. (Although you do still need to send the submit list an email when you are ready for your document to be published, because the whole publishing process has not been fully automated yet.)

Access to our CVS repository

Only authors with at least three submissions get access to our CVS, see Appendix C.

For more information on how to use CVS to maintain your LDP documents, please read Appendix C.

4.3.3. Spell Check

Some writing tools will come with their own built-in spell check tools. This list is only if your application does not have a spell check option.

Spell Check Software

aspell http://aspell.sourceforge.net
: This spell check application can work around XML tags. By distinguishing between content and markup aspell is able to check your content and ignore the bits it shouldn't be looking at. If you are getting spelling errors in your markup tags you may be using an old version and should upgrade.

  The **aspell** command comes with the aspell package, included on most Linux distributions. Use the command as follows:

  aspell `-c` `file`

  An interactive user interface allows for fast and easy correction of errors. Use the `--help` to read more about **aspell** features.

ispell http://www.cs.hmc.edu/~geoff/ispell.html
: Similar to aspell, but tries to spell check your markup tags. If you have a choice, use aspell, if not, ispell is a very acceptable substitute.

Chapter 5. Markup

5.1. Markup: A General Overview

A markup language is a system for marking or tagging a document to define the structure of the document. You may add tags to your document to define which parts of your document are paragraphs, titles, sections, glossary items (the list goes on!). There are many markup languages in use today. XHTML and HTML will be familiar to those who author web documents. The LDP uses a markup language known as DocBook. Each of these markup languages uses its own "controlled vocabulary" to describe documents. For example: in XHTML a paragraph would be marked up with the tagset <p></p> while in DocBook a paragraph would be marked up with `<para></para>`. The tagsets are defined in a quasi dictionary known as a Document Type Definition (DTD).

Markup languages also follow a set of rules on how a document can be assembled. The rules are either SGML (Standard Generalized Markup Language) or XML (eXtensible Markup Language). These rules are essentially the "grammar" of a document's markup. SGML and XML are very similiar. XML is a sub-set of SGML, but XML requires more precise use of the tags when marking up a document. The LDP accepts both SGML and XML documents, but prefers XML.

There are three components to an XML/SGML document which is read by a person.

- **Content.** As a TLDP author it is good to remember that this is the most important piece. Many authors will write the content first and add their markup later. Content may include both plain text and graphics. This is the only part that is required of LDP authors!
- **Markup.** To describe the structure of a document a controlled vocabulary is added on top of the content. It is used to distinguish different kinds of content: paragraphs, lists, tables, warnings (and so on). The markup must also conform to either SGML or XML rules. If you are not comfortable adding markup to your document, a TLDP volunteer will do it for you.
- **Transformation.** Finally the document is transformed from DocBook to PDF, HTML, PostScript for display in digital or paper form. This transformation is controlled through the Document Style Semantics and Specification Language (DSSSL). The DSSSL tells the program doing the transformation how to convert the raw markup into something that a human can read. The LDP uses a series of scripts to automate these transformations. You are not required to transform your own documents.

Content, markup and transformations

Steve Champeon does a great job of explaining how content, markup languages, and transformations all fit together in his article The Secret Life of Markup. Although he is writing from an HTML perspective, the ideas are relevant and there is an example of DocBook markup.

5.2. DocBook: What it is and why we use it

According to the official DocBook web site,

> DocBook is a general purpose XML and SGML document type particularly well suited to books and papers about computer hardware and software (though it is by no means limited to these applications).
>
> --DocBook.org

For the impatient

In the next sections we will be explaining about the theoretical side of DocBook, its origins, development, advantages and disadvantages. If you just want the practical side, check out these sections for an overview of HOWTO DocBook: , Appendix D, and Appendix E from this guide.

For the beginner

We wish to stress again the fact that any open document format will be accepted. If you feel more comfortable with plain text, OpenOffice or HTML, that is fine with us. If you do not look forward to learning DocBook, LDP volunteerd will convert your document to DocBook XML. To us, the most important task for our authors is the actual writing, not the formatting, keep that in mind!

From the point of submission onwards, however, you will have to maintain your document in this XML format, but that's a piece of cake. Promised.

Although there are other DTDs used to write documentation, there are a few reasons not to use them.

- DocBook is the most popular DTD, being used by more than a dozen major open source projects from GNOME to Python to FreeBSD.
- The tools for DocBook are more developed than others. DocBook support is included in most Linux distributions, allowing you to send raw files to be processed at the receiver's end.
- And finally, DocBook has an extensive set of tags (over 300 in all) which is very useful when you are trying to describe the content of a document. Fortunately for new authors the majority of them do not need to be used for simple documentation.

Still not convinced? Eric Raymond has written a DocBook Demystification HOWTO which may help.

Convinced, but still not comfortable with the thought of working with DocBook? Give David Lawyer's Howtos-with-LinuxDoc-mini-HOWTO a try.

5.3. XML and SGML: Why we use XML

DocBook comes in a couple of different flavors--including both XML and SGML formats. This means that you may use either the SGML grammar/rules when adding markup, or you may use the XML grammar/rules. Either way you may only use one set of rules throughout your document, and you must define which one you are using at the top of your document.

The LDP prefers the XML flavor of DocBook. There are a number of reasons for this including the following:

1. Libraries for handling XML files are developing at a rapid pace. These utilities may make it easier for new authoring tools to become available.
2. Many popular word processing programs are now creating XML output. While it may not be DocBook XML, this does make it easier for application writers to either add DocBook XML support, or provide some method of translating between their format and DocBook XML.
3. Everyone else is doing it. While this might not be a real reason, it allows the LDP to keep up-to-date with similar projects. Tools, procedures, and issues can be worked out in a common framework.

Still not convinced? Fortunately the LDP does accept a number of other file formats for input. The list of accepted markup languages can be found in Section 5.4

5.4. Markup Languages Accepted by TLDP

The LDP stores its documents in the following markup languages:

1. DocBook XML version 4.2 (preferred), 4.1.2
2. DocBook SGML version 4.2, 4.1, or 3.x
3. LinuxDoc SGML

New Documents

A new document may be submitted to the LDP in any format. Documents which are not in DocBook or LinuxDoc will be converted by a volunteer. The author is responsible for adding markup to any revisions which are made.

Chapter 6. Distributing Your Documentation

6.1. Before Distributing Your Documentation

Before you distribute your documentation, there are a few more things that you will need to check and possibly add to your document.

Spelling and Grammar Check
: You can read more about helper applications in Section 4.3.3. You should also check your document for its overall flow and clarity.

Abstract and Other Meta Data
: Add a short paragraph which clearly defines the scope of your document. For more information on how to add this information using DocBook please read Section D.6

Acknowledgments
: Give credit where credit is due. For more information about when to give credit, read Section 6.3.

License and Copyright
: The LDP distributes documents, however, the author maintains the copyright on the document. All documents accepted by the LDP must contain a license and copyright notice. You can read more about this in Section 6.2.1. You may also want to add a Disclaimer, but this is optional. More about this in Section 6.2.2.

Validate the Markup
: If you are submitting a DocBook or LinuxDoc document, make sure the markup is valid. Read why in Section B.3.1.

Obtain Peer Reviews
: You may want to have others review your document before submitting it to the LDP. Ask people for a Peer Review and/or a Technical Accuracy Review. Since not all mailing lists will respond favorably to attachments, you may wish to set up a temporary web site which houses your document. Note: this is absolutely *not* required.

6.2. Licensing and Copyright

In order for a document to be accepted by the LDP, it must be licensed and conform to the "LICENSE REQUIREMENTS" section of the LDP Manifesto located at http://www.tldp.org/manifesto.html.

We recommend using the GNU Free Documentation License (GFDL), one of the Creative Commons Licenses (Share-Alike, or Attribution-Share-Alike), or the LDP license (currently under review). The full text of the license must be included in your document, including the title and version of the license you are using. The LDP will not accept new documents that do not meet licensing requirements.

Debian-compatible licenses

The Debian package maintainer for LDP documents has divided the LDP documents into those with a "free" license and those with a "non-free" license. For a summary of this list, please read Debian License Summaries. Currently the Artistic License, BSD License and the GNU General Public License are listed as "free". These licenses will also be accepted by the LDP. The definition of "non-free" has not been made transparent. By choosing another license that has any kind of restriction on redistribution or whether or not the document may be modified, your document *may* be put into the "non-free" package instead of the "free" package. We are working with Debian to clarify how these decisions are made.

You can get DocBook markups of both the GNU GPL and the GNU FDL from the GNOME Documentation Project. You can then merely include the license in its entirety in your document. A DocBook-formatted copy of the license is available in Appendix A.

For more information about open source documentation and licensing, please check .

6.2.1. Copyright

As an author, you may retain the copyright and add other restrictions (for example: require approval for any translations or derivative works). If you are a new maintainer of an already-existing HOWTO, you must include the previous copyright statements of the previous author(s) and the dates they maintained that document.

6.2.2. Disclaimer

If you would like to include a disclaimer, you may choose to use the following:

> No liability for the contents of this document can be accepted. Use the concepts, examples and information at your own risk. There may be errors and inaccuracies, that could be damaging to your system. Proceed with caution, and although it is highly unlikely that accidents will happen because of following advice or procedures described in this document, the author(s) do not take any responsibility for any damage claimed to be caused by doing so.
>
> All copyrights are held by their by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Naming of particular products or brands should not be seen as endorsements.

6.2.3. Licensing source code

If your HOWTO includes bits of source code that you want others to use, you may choose to release the source code under GPL.

6.3. Acknowledgments

Your document should have an "Acknowledgments" section, in which you mention everyone who has contributed to your document in any meaningful way. You should include translators and converters, as well as people who have sent you lots of good feedback, perhaps the person who taught you the knowledge you are now passing on, and anybody else who was instrumental in making the document what it is. Most authors put this section at the end of their document.

When someone else assists in the production of an LDP document, you should give them proper attribution, and there are DocBook tags designed to do this. This section will show you the tags you should use, as well as other ways of giving credit where credit is due. Crediting editors is easy - there is already an `<editor>`tag in DocBook. But there are two special cases where you need to credit someone, but DocBook doesn't provide a special tag. These are *translators* and *converters*.

A *converter* is someone who performs a source code conversion, for instance from HTML to DocBook XML. Source code conversions help the LDP achieve long term goals for meta-data, and allow us to distribute documentation in many different formats.

Translators take your original document and translate it into other human-readable languages, from English to Japanese for example, or from German to English. Each translation allows many more people all over the world to make use of your document and of the Linux operating system!

We recommend that you acknowledge converters in the comment for the initial version released in the new format, and we recommend that you credit translators in each version which they have translated.

Acknowledgments translated in DocBook

For more information on how to add these credits using DocBook please read Section D.6

6.4. TLDP Review Process

Before your document is accepted to the LDP collection it will undergo at least three formal reviews. These reviews include a technical accuracy review, a language review and a metadata review. All new documents must pass these reviews before being accepted into the collection.

When you feel your document is finished, email a copy to the submit mailing list (`<submit@en.tldp.org>`). Please include the title of your document and "Final Review Required" in the subject line of your email. A team of volunteers will be assigned to your document for each of the reviews. It may take up to a week to gather a team who is qualified to review your document. Typically the technical review happens first, followed by the language review and finally the metadata review. Your reviewers will read your document give you feedback on whether or not they think your document is ready for publication in the LDP collection.

Your reviewers may have specific points that must be changed. Once you have made the changes submit your document back to your review team. They will review the document again and advise you on whether or not your document is ready for inclusion in the LDP collection. You may need to undergo several edits before your document is ready. Or it may not require any additional work. Be prepared to make at least one round of changes for both the technical and language reviews. Ideally this exchange will happen in the LDP's CVS to better track each of the changes that are made, and keep track of the most current version of your document.

Once your document has passed both the technical and language reviews, you may submit it by following the instructions in Section 6.5.

Comparing Two Source Files

Your reviewer may make changes directly to your source file, or they may put their suggestions in a separate email. If they are working with the source file directly, and your document is using DocBook XML, you may find XMLdiff useful to see the changes that your reviewer has made to your source file. It is a python tool that figures out the differences between two similar XML files, in the same way the diff utility compares text files.

XMLdiff is available from http://www.logilab.org/projects/xmldiff.

For more information on what the reviewers will be looking for, please read the Linux Documentation Project Reviewer HOWTO.

6.5. Submission to LDP for publication

The final step

This section contains information on what to do after your document has received both a technical and language review by the LDP volunteers.

As part of the review process a Review Coordinator will add your document to the CVS (including any associated image files) and notify the submit mailing list that your document is ready for publication.

If you do not already have a CVS account, please apply for one when your document is submitted for publication. You can apply for an account contacting LDP CVS master Sergiusz mailto:ser@gnu.org

Chapter 7. Maintenance

7.1. Maintaining Your Document

Just because your document has now been published does not mean your job is done. Linux documentation needs regular maintenance to make sure it is up to date, and to improve it in response to readers' ideas and suggestions. TLDP is a living, growing body of knowledge, not just a publish-and-forget-it static entity.

Add relevant mailing lists to your document where people can get support. If you have the time, follow these mailing lists yourself to stay up-to-date on the latest information.

Put your email address in the document, and politely request feedback from your readers. Once you are officially published, you will begin to receive notes with suggestions. Some of these emails will be very valuable. Create a folder in your mail program for incoming suggestions--when the time is right review the folder and make updates to your document. If you are following a related mailing list you may also choose to save a copy of important emails from the list to this folder.

We are not a free support service, but...

Some people who email you will request personal assistance. You should feel free to decline personal assistance if you cannot spare the time. Writing a contribution to the LDP does not commit you to a lifetime of free support for anyone on the net; however, do try to reply to all requests and suggest a mailing list that will (hopefully) be able to provide support to your reader.

7.2. Fixing Errors

7.2.1. Fixing Your Own Documents

If you find an error in your own document, please fix it and re-submit the document. You can re-submit your files by emailing them to `<submit@en.tldp.org>`.

If you have been using the CVS, you can submit your changes to the CVS tree and then send a note to the submit mailing list (`<submit@en.tldp.org>`). In your email please give the exact path of your document in the CVS tree.

Remember to update the revision history at the top of the document.

7.2.2. Fixing Other Documents in the Collection

If you find an error in someone else's document please contact the author of the document with the correction. If you do not hear back from the author within a "reasonable" amount of time, please email the LDP coordinator at `<discuss@en.tldp.org>` (subscription required) and describe the problem and how you think it needs to be fixed. If the license permits, you may be asked to make the change directly to the document. If the problems are serious, the document may be removed from the collection, or moved to the "Unmaintained" section.

Taking over unmaintained documentation

For more information on how to deal with unmaintained documents, please read: Unmaintained (includes a list of steps to take to take over "ownership" of unmaintained documents, and a list of unmaintained documents).

References

Markup and general information

Secret Life of Markup, The, http://hotwired.lycos.com/webmonkey/02/42/index4a.html, Steve Champeon.

Progressive Enhancement and the Future of Web Design: Where We Are Now , http://hotwired.lycos.com/webmonkey/03/21/index3a_page2.html, Steve Champeon.

SGML, http://www.w3.org/MarkUp/SGML/.

DocBook References

DocBook XML 4.1.2 Quick Start Guide, http://www.jimweller.net/jim/dbxmlqs/index.html, Jim Weller.

> Describes how to install, configure and use the tools and resources for DocBook XML 4.1.2. The purpose of this quick start guide is to get new docbook authors, editors, and contributors up and running fast with the DoocBook tools. These are powerful tool in the hands of an author. It assumes a fair knowledge of building and installing source packages. There are probably a million and one ways to accomplish my ultimate goal of installing and using these tools. This one works well for me.
>
> --Jim Weller

Installing And Using An XML/SGML DocBook Editing Suite Setting Up A Free XML/SGML DocBook Editing Suite For Windows And Unix/Linux/BSD, http://supportweb.cs.bham.ac.uk/documentation/tutorials/docsystem/build/tutorials/docbooksys/docbooksys.html , Ashley J.S. Mills, 2002.

Getting Upto Speed With DocBook, http://supportweb.cs.bham.ac.uk/documentation/tutorials/docsystem/build/tutorials/UniDocBook/UniDocBook.html, Ashley J.S. Mills, 2002.

DocBook: The Definitive Guide, http://www.docbook.org/, Norman Walsh, Leonard Muellner, 1999, 1-56592-580-7, O'Reilly & Associates, Inc..

This book was released by O'Reilly in October 1999, and is a great reference to DocBook. I have not found it to be a great practical book. You can pick it up at the book vendor of choice, and the entire book is also available online (in HTML and SGML formats) at the above URL.

Simplified DocBook: The Definitive Guide, http://www.docbook.org/tdg/simple/en/html/sdocbook.html, Norman Walsh, Leonard Muellner, 1999.

Simplified DocBook, http://www.oasis-open.org/docbook/xml/simple/.

XML Matters: Getting started with the DocBook XML dialect, http://www-106.ibm.com/developerworks/xml/library/xml-matters3.html.

FAQ for DocBook markup, http://www.dpawson.co.uk/docbook/markup.html.

Single-Source Publishing with DocBook XML, , http://www.lodestar2.com/people/dyork/talks/2002/ols/docbook-tutorial/frames/frames.html, Dan York, 2002.

DocBook Install mini-HOWTO, http://tldp.org/HOWTO/mini/DocBook-Install/, Robert Easter.

Exploring SGML DocBook, http://lwn.net/2000/features/DocBook/, Giorgio Zoppi.

LinuxDoc

Howtos-with-LinuxDoc-mini-HOWTO, http://www.tldp.org/HOWTO/Howtos-with-LinuxDoc.html, David Lawyer.

LinuxDoc-SGML User's Guide, http://www.nmt.edu/tcc/doc/guide.html.

Converting Other Formats to DocBook

Converting HTML to Docbook SGML/XML Using html2db, http://www.cise.ufl.edu/~ppadala/tidy/.

5-Minute Review: Using LyX for Creating DocBook, http://www.teledyn.com/help/XML/lyx2db/t1.html.

Document processing with LyX and SGML, http://www.karakas-online.de/mySGML/.

LDP templates, tools & links

LDP Templates, , http://www.tldp.org/authors/index.html#resources .

Contains links to SGML templates and their resulting HTML output to help you see what your document will look like. Many of the tags just need to be replaced with information unique to your HOWTO. Also contains links to tools and other useful information.

Linux Documentation Project HOWTO Generator, The, http://www.nyx.net/~sgjoen/The_LDP_HOWTO_Generator.html.

This is a standalone web page with a number of fields to fill in and a few buttons. When ready the compile button starts the compilation of all the input fields and wraps it all in proper LinuxDoc SGML, ready to process with the LinuxDoc SGML tools.

The compiled output is outputted to a read-only text area near the bottom of the webpage, so the text has to be copied and pasted into a file for compilation.

DocBook is not currently supported.

DocBook Transformations

DocBook XML/SGML Processing Using OpenJade, , http://tldp.org/HOWTO/DocBook-OpenJade-SGML-XML-HOWTO/, Saqib Ali.

General Writing Links and Style Guides

Guide to Grammar and Style, http://newark.rutgers.edu/~jlynch/Writing/, Jack Lynch.

Purdue's Online Writing Lab, http://owl.english.purdue.edu/.

Chicago Manual of Style, http://www.chicagomanualofstyle.org/.

Plain Language Resources, http://www.plainlanguagenetwork.org/Resources/.

Writing User-Friendly Documents, http://www.blm.gov/nhp/NPR/pe_toc.html.

This is quite useful. It includes before and after writing samples.

PlainTrain Writing Tutorial, http://www.web.net/~plain/PlainTrain/IntroducingPlainLanguage.html.

Writing for the Web, http://www.useit.com/papers/webwriting/.

Information Pollution, http://useit.com/alertbox/20030811.html.

Be Succinct! (Writing for the Web), http://useit.com/alertbox/9703b.html.

Politics and the English Language, http://www.k-1.com/Orwell/index.cgi/work/essays/language.html, George Orwell.

A classic text on writing.

Elements of Style, The, http://www.bartleby.com/141/, Strunk and White.

A classic text on writing.

A Short Handbook and Style Sheet, http://newark.rutgers.edu/~jlynch/Writing/m.html#mechanics, Thomas Pinney.

Technical Writing Links, http://www.techcomplus.com/tips.htm.

Technical Writing Tutorial, http://psdam.mit.edu/rise/tutorials/writing/technical-writing.html.

Strategies to succeed in technical writing, http://www.school-for-champions.com/techwriting.htm.

User Guides Online Tutorial, http://www.klariti.com/technical-writing/User-Guides-Tutorial.shtml.

DMOZ Technical Writing Links, http://dmoz.org/Arts/Writers_Resources/Non-Fiction/Technical_Writing/.

techwr-L, http://www.raycomm.com/techwhirl/magazine/.

Technical Writing Links, http://academic.middlesex.cc.ma.us/PeterHarbeson/links.html.

An omnibus of links--scrounge for goodies.

Related TLDP Documents

Linux Documentation Project (LDP) FAQ, http://tldp.org/FAQ/LDP-FAQ/index.html, Rahul Sundaram.

LDP HOWTO-INDEX, http://tldp.org/HOWTO/HOWTO-INDEX/, Guylhem Aznar, Joshua Drake, Greg Ferguson.

Software: CVS

CVS: Project Management, http://doc.cs.byu.edu/programming/cvs/, Byron Clark.

CVS, http://supportweb.cs.bham.ac.uk/documentation/tutorials/docsystem/build/tutorials/cvstute/cvstute.html, Ashley J.S. Mills, Alan P. Sexton.

CVS--Concurrent Versions System, http://www.loria.fr/~molli/cvs/doc/cvs_toc.html, Pascal Molli.

Learning About CVS , http://cvshome.org/docs/.

Software: Emacs

Information about PSGML , http://www.lysator.liu.se/~lenst/about_psgml/.

Emacs: The Free Software IDE, http://www.linuxjournal.com/article.php?sid=576.

Emacs/PSGML Quick Reference, http://www.snee.com/bob/sgmlfree/psgmqref.html, Bob Ducharme.

NT Emacs Installation, http://www.charlescurley.com/emacs.html, Charles Curley.

Cygwin Packages for DocBook Authoring, http://www.docbook.org/wiki/moin.cgi/CygwinPackages/.

SGML for Windows NT: Setting up a free SGML/XML editing and publishing system on Windows/Cygwin , http://ourworld.compuserve.com/homepages/hoenicka_markus/cygbook1.html, Markus Hoenicka, 2000.

Vim, http://www.newriders.com/books/opl/ebooks/0735710015.html, Steve Oualline.

XML Authoring Tools

Saqib's list of XML Authoring Tools, http://www.xml-dev.com/xml/editors.html.

Documentation Licenses

Licensing HOWTO, http://www.catb.org/~esr/Licensing-HOWTO.html, Eric Raymond, Catherine Raymond.

> This document explains how U.S. copyright and licensing law applies to open-source software projects. It compares the strengths and weaknesses of the existing open-source licenses, and gives guidance on how to choose a license for your project. It also explains the legalities of changing a project's license. It suggests new practice for coping with today's high-threat legal environment--this part is a must-read for all project leaders.
>
> --Eric Raymond and Catherine Raymond

OPL, http://www.opencontent.org/openpub/.

OpenContent officially closed June 30, 2003.

Creative Commons, http://creativecommons.org/licenses/by-sa/1.0/.

GNU Free Documentation License, http://www.gnu.org/copyleft/fdl.html.

GNU General Public License, http://www.gnu.org/licenses/licenses.html#GPL.

If you would like your documents to be included in the main Debian distribution, you should use this license. It is not, however, the LDP's license of choice.

Appendix A. Templates

The LDP stores its documents in the following markup languages:

1. DocBook XML version 4.2 (preferred), 4.1.2
2. DocBook SGML version 4.2, 4.1, or 3.x
3. LinuxDoc SGML

New Documents

A new document may be submitted to the LDP in any format. Documents which are not in DocBook or LinuxDoc will be converted by a volunteer. The author is responsible for adding markup to any revisions which are made.

A.1. Document Templates

1. **HOWTO (Article) templates/ldp-howto.zip.** Most HOWTO documents will use this template.
2. **Guide (Book) templates/ldp-guide.zip.** Use this template to create a full book (like this Author Guide, for instance). Templates provided by Tille Garrels.
3. **FAQ templates/ldp-faq.zip.** A standard article for writing a FAQ (Frequently Asked Questions) list.
4. **LinuxDoctemplates/ldp-linuxdoc.zip.** A standard template both in HOWTO length and Guide length.
5. **Disclaimer disclaimer.xml.** A standard disclaimer which warns readers that (1) your document may not be perfect and (2) you are not responsible if things end up broken because of it.

A.2. Style Sheets

The following style sheets can be used to make your document nicer to look at. Remember that the LDP will use its own style sheets to distribute your documentation.

1. **DSL Style Sheet style.dsl.** This DSL style sheet was provided by Tille and is to be used with DSSSL transformations.
2. **Cascading Style Sheet style-ob.css.** This CSS file was provided by Saqib Ali and Emma Jane Hogbin. The "ob" is for "orange and blue". Use this CSS file with an HTML file. Instructions are included in the CSS file.

A.3. GNU Free Documentation License

The GFDL (GNU Free Documentation License) is available in XML format at http://www.gnu.org/licenses/fdl.xml. For a version in appendix format suitable for including in your document, you can see get the XML for this document from CVS at http://cvsview.tldp.org/index.cgi/LDP/guide/docbook/LDP-Author-Guide/fdl-appendix.xml.

TLDP template files for DocBook (XML and SGML) and Linuxdoc SGML are available from the TLDP website at http://www.tldp.org/authors/index.html#resources.

Appendix B. System Setup: Editors, Validation and Transformations

In this section, we will cover some of the tools that you may want to use to create your own LDP documentation. If you use some other tool to assist you in writing documents for the LDP, please drop us a line and we'll add a blurb for it here. Section 1.3 has contact information.

B.1. Tools for your operating system

A few notes on setting up your system for DocBook publishing. These tools focus more on the transformation of documents from DocBook to XHTML (for example).

Tools For Your Operating System

Debian

http://www.docbook.org/wiki/moin.cgi/DebianGnuLinuxPackages

Morgon Kanter suggests **apt-get install docbook-xml docbook-xsl xsltproc** as the minimum requirements. http://lists.tldp.org/index.cgi?1:mss:4851

Fedora (aka the new RedHat)

Notes contributed by Charles Curley.

Tools for Docbook SGML and XML are included in the distrubution. So are Emacs and PSGML mode, although you will have to customize your `.emacs`. If you are missing a package after installing Fedora, get familiar with yum or apt.

Installation instructions: none; use Red Hat 9 until they are written: http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/.

Mandrake

Notes contributed by Artemio.

In Mandrake (as of my current 9.2), all the stuff including openjade, xmlto, docbook-utils etc. comes as standard.

So I just needed to get the TLDP XSL sheet and that's all. Didn't ever have any dependency or other problems, everything works fine (knock on wood :-)).

RedHat

According to Hal Burgiss, your system is likely already ready to edit and process DocBook documents without installing any additional packages.

B.2. Editing tools

Editing tools have come a long way in their support for XML (and specifically DocBook). There are two types of editors outlined in this section: text editors (emacs, vim, etc); and word processors (OpenOffice, AbiWord, etc). New authors who are not comfortable working with markup languages should probably choose a word processor that can output DocBook files. For this reason the word processors are listed first.

Although many editors can also validate your DocBook files, this information has been separated into Section B.3.

More info

Check the resources section for more .

B.2.1. Word Processors

Even if you are not comfortable working DocBook's tagset in a text editor you can still produce valid DocBook documents using a word processor. Support at this point is very limited, but it does exist in the following programs. The up side, of course, is that things like spell check are built in to the program. In addition to this, support for DocBook (and XML) is constantly improving.

Converting Microsoft Word documents

Even if you want to use MS Word to write your documents, you may find w2XML useful. Note that this is not free software--the cost is around $130USD. There is, however, a trial version of the software.

Work on the content!

Remember that all formatting changes you make to your document will be ignored when your document is released by the LDP. Instead of focusing on how your document *looks*, focus on the content.

B.2.1.1. AbiWord

Through word of mouth I've heard that AbiWord can work (natively) with DocBook documents. This will need to be tested by someone (possibly me) and should definitely be included if it is the case.

B.2.1.2. OpenOffice.org

http://openoffice.org

As of OpenOffice.org (OOo) 1.1RC there has been support for exporting files to DocBook format.

Although OOo uses the full DocBook document type declaration, it does not actually export the full list of DocBook elements. It uses a "simplified" DocBook tagset which is geared to on-the-fly rendering. (Although it is not the official Simplified DocBook which is described in Section B.5.) The OpenOffice simplified (or "special" docbook) is available from http://xml.openoffice.org/xmerge/docbook/supported_tag_table.html.

B.2.1.2.1. Open Office 1.0.x

OOo has been tested by LDP volunteers with mostly positive results. Thanks to Charles Curley (charlescurley.com) for the following notes on using OOo version 1.0.x:

Check the version of your OpenOffice

These notes may not apply to the version of OOo you are using.

- To be able to export to DocBook, you must have a Java runtime environment (JRE) installed and registered with OOo--a minimum of version 4.2.x is recommended. The configuration instructions will depend on how you installed your JRE. Visit the OOo web site for help with your setup.

Contrary to the OOo documentation, the Linux OOo did not come with a JRE. I got one from Sun.

- The exported file has lots of empty lines. My 54 line exported file had 5 lines of actual XML code.
- There was no effort at pretty printing.
- The header is: `<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN" "http://www.oasis-open.org/docbook/xml/4.1.2/docbookx.dtd">`
- The pull-down menu in the File->Save As dialog box for file format indicates that the export format is "DocBook (simplified)." There is no explanation of what that "simplified" indicates. Does OOo export a subset of DocBook? If so, which elements are ignored? Is there any way to enter any of them manually?
- There is NO documentation on the DocBook export filter or whether OOo will import it again.

Conclusions: OOo 1.1RC is worth looking at if you want a word processor for preparing DocBook documents.

However, I hope they cure the lack of documentation. For one thing, it would be nice to know which native OOo styles map to which DocBook elements. It would also be nice to know how to map one's own OOo styles to DocBook elements.

B.2.1.2.2. Open Office 1.1

Tabatha Marshall offers the following additional information for OOo 1.1.

> The first problem was when I tried to do everything on version 1.0.1. That was obviously a problem. I have RH8, and it was installed via rpm packages, so I ripped it out and did a full, new install of OpenOffice 1.1. It took a while to find out 1.1 was a requirement for XML to work.
>
> During the install process I believe I was offered the choice to install the XML features. I have a tendency to do full installs of my office programs, so I selected everything.
>
> I can't offer any advice to those trying to update their current OO 1.1. Their "3 ways" aren't documented very well at the site (xml.openoffice.org) and as of this writing, I can't even find THAT on their site anymore. I think more current documentation is needed there to walk people through the process. Most of this was unclear and I had to pretty much experiment to get things working.
>
> Well, after I installed everything I had some configuration to do. I opened the application, and got started by opening a new file, choosing templates, then selecting the DocBook template. A nice menu of Paragraph Styles popped up for me, which are the names for all those tags, I noticed (you can see I don't use WYSIWYG often).
>
> With a blank doc before me (couldn't get to the XML Filter Settings menu unless some type of doc was opened), I went into Tools->XML Filter Settings, and edited the entry for DocBook file. I configured mine as follows:
>
> - Doctype **`-//OASIS//DTD DocBook XML V4.2//EN`**

- DTD
  http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd
- XSLT for export
  /usr/local/OpenOffice.org1.1.0/share/xslt/docbook/ldp-html.xsl
- XSLT for import
  /usr/local/OpenOffice.org1.1.0/share/xslt/docbook/docbooktosoffheadings.xsl
  (this is the default)
- Template for import
  /home/tabatha/OpenOffice/user/template/DocBook File/DocBookTemplate.stw

At first, if I opened an XML file that had even one parsing error, it would just open the file anyway and display the markup in OO. I have many XML files that use © and other types of entities which show up as parse errors (depending on the encoding) even though they can be processed through. But today I was unable to open any of those files. I got input/output errors instead. Still investigating that one.

However when you do successfully open a document (one parsing with no errors), it puts it automatically into WYSIWYG based on the markup, and you can then work from the paragraph styles menu like any other such editor.

To validate the document, I used Tools->XML Filter Settings, then clicked the Test XSLTs button. On my screen, I set up the XSLT for export to be `ldp-html.xsl`. If you test and there are errors, a new window pops up with error messages at the bottom, and the lines that need to be changed up at the top. You can change them there and progress through the errors until they're all gone, and keep testing until they're gone.

If you want to open a file to see the source instead of the processed results, go to Tools->XML Filter Settings->Test XSLTs, and then under the Import section, check the Display Source box. My import XSLT is currently `docbooktosoffheadings.xsl` (the default) and the template for import is `DocBookTemplate.stw` (also default).

I think this might work for some people, but unfortunately not for me. I've never used WYSIWYG to edit markup. Emacs with PSGML can tell me what my next tag is no matter where I am, validate by moving through the trouble spots, and I can parse and process from command line.

With OpenOffice, you have to visit http://xml.openoffice.org/filters.html to find conversion tools.

B.2.1.3. WordPerfect 9 (Corel Office 2000)

http://www.corel.com/

WordPerfect 9 for the MS Windows platform has support for SGML and DocBook 3.0. WordPerfect 9 for Linux has no SGML capabilities.

If you are using WordPerfect on the Linux operating system, please read: WordPerfect on Linux FAQ

B.2.1.4. XMLmind's XML editor

http://www.xmlmind.com/xmleditor/

Although strictly speaking, it is not a word processor, XMLmind's XML editor allows authors to concentrate on the content, and not the markup. It has built in spelling and conversion utilities which allow you to transform your documents without having to install and configure an additional processing tool such as jade. There is a free "standard edition", which is a simplified version of their "professional edition."

B.2.1.5. Conglomerate

http://www.conglomerate.org

According to their web site, "Conglomerate aims to be an XML editor that everyone can use. In particular, our primary goal is to create the ultimate editor for DocBook and similar formats. It aims to hide the jargon and complexity of XML and present the information in your documents in a way that makes sense."

B.2.1.6. Vex: a visual editor for XML

http://vex.sourceforge.net/

According to their web site, "The visual part comes from the fact that Vex hides the raw XML tags from the user, providing instead a wordprocessor-like interface. Because of this, Vex is best suited for document-style XML documents such as XHTML and DocBook rather than data-style XML documents."

B.2.2. Text Editors

For advanced writers

> The tools outlined in this section allow you to work with the DocBook tags directly. If you are not comfortable working with markup languages you may want to use a word processor instead. Word processors that support DocBook are described in Section B.2.1.

If you are comfortable working with markup languages and text editors, you'll probably want to customize your current editor of choice to handle DocBook files. Below are some of the more common text editors that can, with some tweaking, handle DocBook files.

B.2.2.1. Emacs (PSGML)

http://www.lysator.liu.se/~lenst/about_psgml/

Emacs has an SGML writing mode called psgml that is a major mode designed for editing SGML and XML documents. It provides:

- "syntax highlighting" or "pretty printing" features that make the tags stand out
- a way to insert tags other than typing them by hand
- and the ability to validate your document while writing

For users of Emacs, it's a great way to go. PSGML works with DocBook, LinuxDoc and other DTDs equally well.

B.2.2.1.1. Verifying PSGML is Installed

If you have installed a recent distribution, you may already have PSGML installed for use with Emacs. To check, start Emacs and look for the PSGML documentation (**C-himpsgml**).

Dependencies

If you don't have PSGML installed now might be a good time to upgrade Emacs. The rest of these instructions will assume you have PSGML installed.

B.2.2.1.2. Configuring Emacs for Use With PSGML

If you want GNU Emacs to enter PSGML mode when you open an `.xml` file, it will need to be able to find the DocBook DTD files. If your distribution already had PSGML set up for use with GNU Emacs, you probably won't need to do anything.

Tuning Emacs

For more information on how to configure Emacs, check out .

Once you've configured your system to use PSGML you will need to override Emacs' default *sgml-mode* with the *psgml-mode*. This can be done by configuring your `.emacs` file. After you've edited the configuration file you will need to restart Emacs.

B.2.2.1.3. Creating New DocBook XML Files

There are a number of steps to creating a new DocBook XML file in Emacs.

- Create a new file with an `xml` extension.
- On the first line of the file enter the doctype for the version of DocBook that you would like to use. If you're not sure what a doctype is all about, check Section B.5
- Enter **C-c C-p**. If Emacs manages to parse your DTD, you will see `Parsing prolog...done` in the minibuffer.
- Enter **C-c C-e Enter** to auto-magically insert the parent element for your document. (New authors are typically writing `article`s.)
- If things are working correctly, you should see new tags for the parent element for your document right after the document type declaration. In other words you should now see two extra tags: `<article>` and `</article>` in your document.

B.2.2.1.4. Spell Checking in Emacs

Emacs can be configured to use aspell by adding the following to your `~/.emacs` file. Thanks to Rob Weir for this configuration information.

```
;; Use aspell
(setq-default ispell-program-name "aspell")
;;Setup some dictionary languages
(setq ispell-dictionary "british")
(setq flyspell-default-dictionary "british")
```

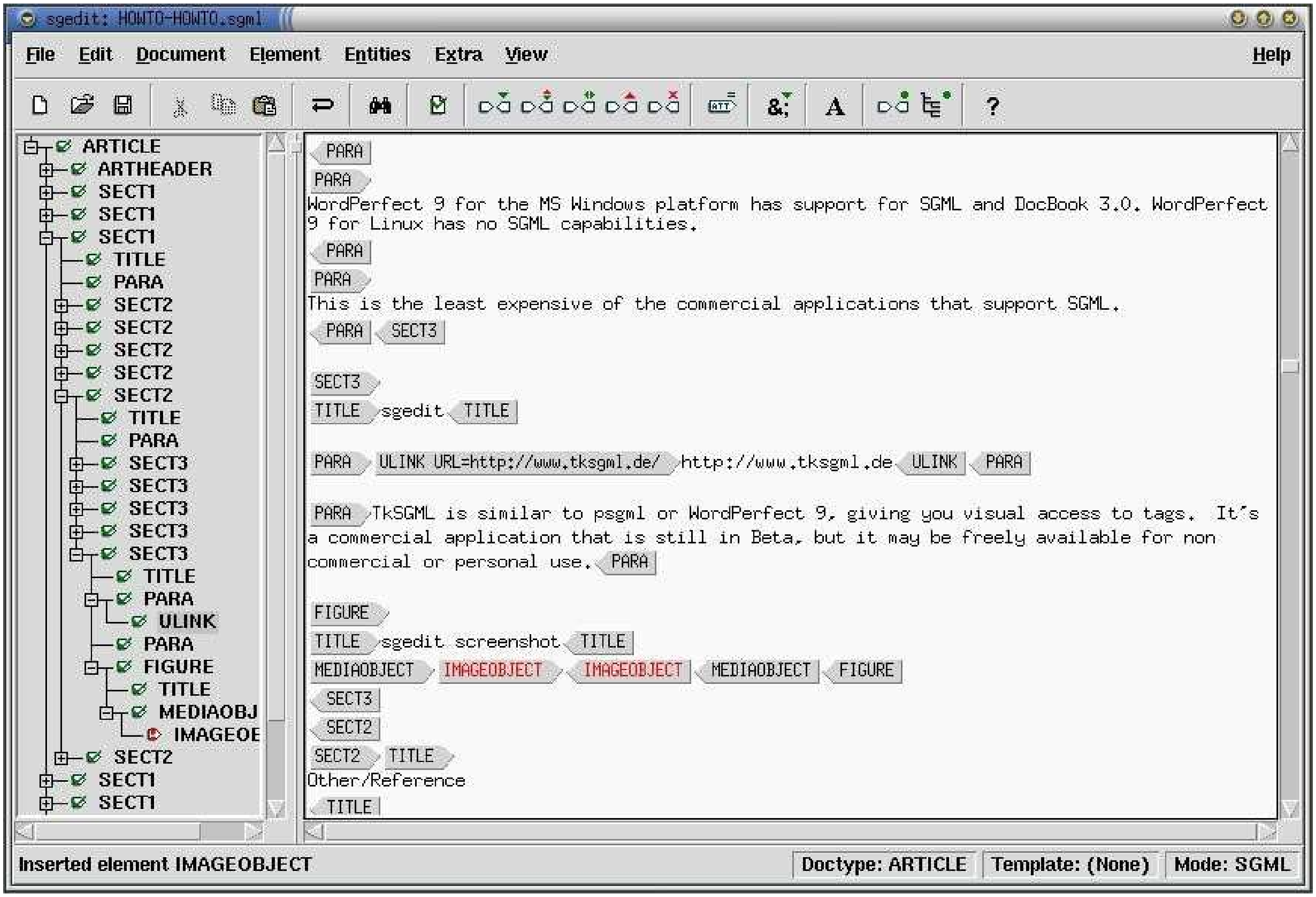
B.2.2.2. epcEdit

http://www.tksgml.de

The epcEdit program allows you to edit XML files. It has the advantages of not needing to know Emacs or vi before starting, and is cross-platform, working in both Windows and Linux. This is a commercial application, and pricing can be found at http://www.tksgml.de/pricing.html

Along with visual editing, epcEdit will also validate documents on loading, and on demand by using the Document->Validate command.

Figure B-1. epcEdit screen shot



B.2.2.3. Morphon XML editor

http://www.morphon.com/xmleditor/index.shtml

This is a commercial application which is currently available for free (with an optional user registration). It is written in Java, allowing it to run on any platform that has a Java Virtual Machine (that is, works in both Windows and Linux).

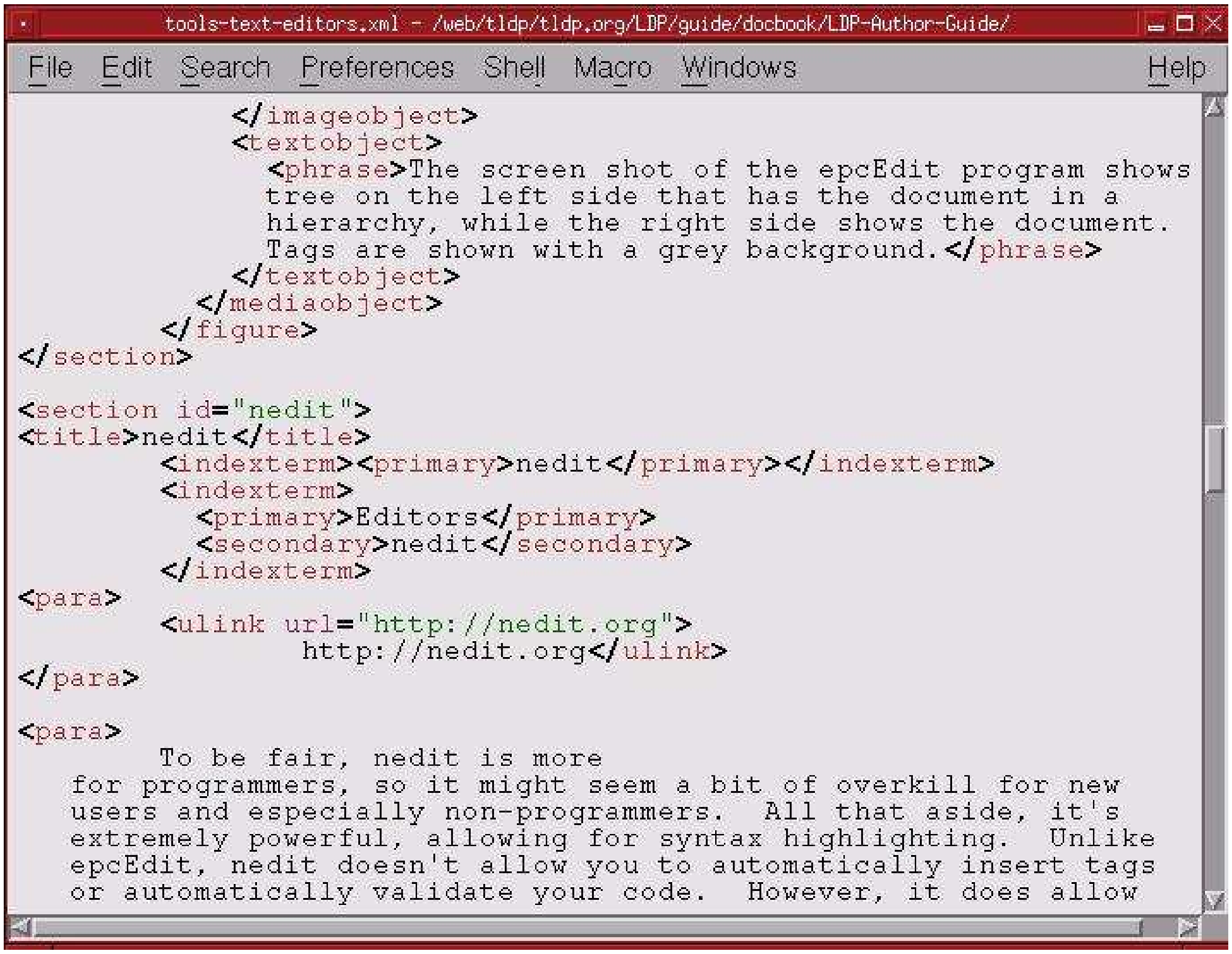
On the plus sides of XMLEditor is the left side of the screen shows the hierarchy of the document (starting with Book and so on). Selecting an item in the list brings you to that part of the document so you can edit it. The right part of the screen shows the text without any markup or tags being shown. If you have external files as ELEMENTS (as the LDP Author Guide does), XMLEditor will follow the links and load the files, so you always work on the entire work. On the minus side of this, you will get errors if a file is missing.

B.2.2.4. nedit

http://nedit.org

To be fair, nedit is more for programmers, so it might seem a bit of overkill for new users and especially non-programmers. All that aside, it's extremely powerful, allowing for syntax highlighting. Unlike epcEdit, nedit doesn't allow you to automatically insert tags or automatically validate your code. However, it does allow for shell commands to be run against the contents of the window (as opposed to saving the file, then checking).

Figure B-2. nedit screen shot



B.2.2.4.1. Using nedit

When you open your DocBook file, nedit should already have syntax highlighting enabled. If it does not you can turn it on explicitly using: Preferences->Language Mode->SGML HTML

If you have line numbers turned on (using Preferences->Show Line Numbers) then finding validation errors is much simpler. nsgmls, the validation tool we'll use, lists errors by their line number.

B.2.2.4.2. Configuring nedit

Since you can feed the contents of your window to outside programs, you can easily extend nedit to perform repetitive functions. The example you'll see here is to validate your document using nsgmls. For more information about nsgmls and validating documents please read Section B.3.

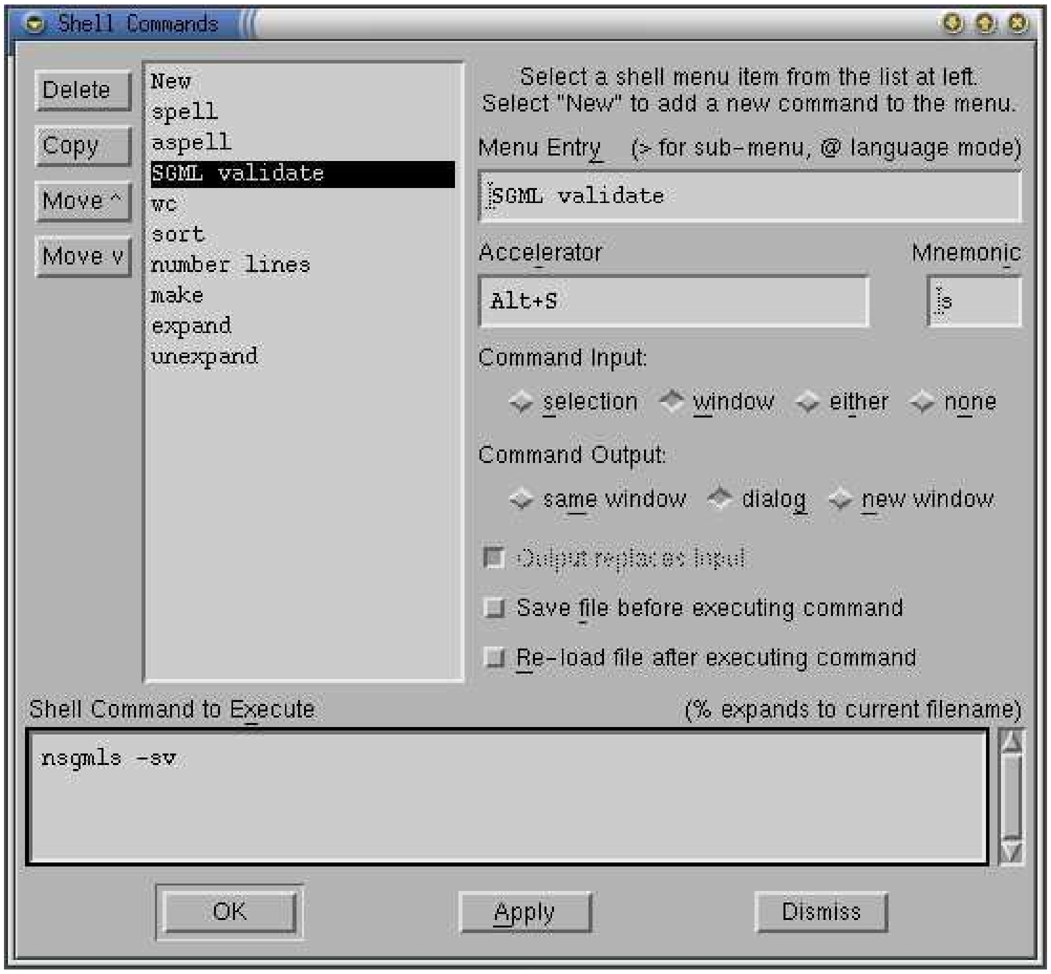
- Select Preferences->Default Settings->Customize Menus->Shell Menu.... This will bring up the Shell Command dialog box, with all the shell commands nedit has listed under the Shell menu.
- Under Menu Entry, enter "Validate DocBook." This will be the entry you'll see on the screen.
- Under Accelerator, press **Alt**-**S**. Once this menu item is set up, you can press **Alt**-**S** to have the Validate DocBook automatically run.
- Under Command Input, select window, and under Command Output, select dialog.

- Under Command to Execute, enter **nsgmls *-sv***. Using *-v* outputs the version number is output to the screen so that you know the command has run.

Check the PATH

Note that nsgmls has to be in your `PATH` for this to work properly.

Figure B-3. Adding shell commands to nedit



- Click OK and you'll now be back at the main nedit screen. Load up an XML file, and select Shell->Validate DocBook or press **Alt**-**S**.
- The **nedit** program will fire up and check the contents of the window.
- If all you see is a version number for nsgml then your document is valid. Any errors are reported by line number in your document.

Figure B-4. nsgmls output on success

B.2.2.5. VIM

http://www.vim.org

No mention of text editors is complete without talking about vi. The VIM (Vi IMproved) editor has the functionality of regular vi and includes "syntax highlighting" of tags.

B.2.2.5.1. Getting Started

There are many versions of vi. New authors will likely want one of the more feature-packed versions for syntax highlighting and a graphical interface including mouse control.

Red Hat users will want the following packages: vim-common, vim-minimal and vim-enhanced. Debian users will want the following package: vim. For an X interface (including GUI menus and mouse control) users will want gvim. The "g" in gvim is for "Graphical".

VIM compiles very easy should you need to build your own. Both **vim** and **gvim** are built by default. Syntax highlighting is included but not enabled by default if you have to start from scratch; use the **:syntax enable** command in VIM to turn this feature on.

B.2.2.5.2. Creating New DocBook XML Files

In both vim and gvim, `.xml` files will be recognized and enter into "SGML mode". A series of known DocBook tags and attributes have been entered into vim and will be highlighted one color if the name is known and another if it is not (for this author the colors are yellow and blue).

Having the correct document type declaration at the top of your document should add the syntax highlighting. If you do not see this highlighting you will need to force VIM into SGML mode (even for XML files) using the command **:set ft=sgml**. If you are working with multiple files for a single XML document you can add your document type in <-- comments --> to the top of the file to get the correct syntax highlighting (you will need to restart the program to see the change in highlighting). The top line of this file (`tools-text-editors.xml`) looks like this:

```
<!-- <!DOCTYPE book PUBLIC '-//OASIS//DTD DocBook XML V4.2//EN'> -->
```

B.2.2.5.3. Spell Check

As in Emacs, Vim, will work quite happily with aspell. It can be run from within Vim with the following: **`:! aspell -c %`**.

For more sophisticated spell check alternatives, give Cream or vimspell a try.

B.2.2.5.4. Tag Completion

The following information is provided by Kwan Lowe.

Vim has a DocBook helper script which can be easily copied into your `.vimscripts` directory and used to "auto complete" tags while writing DocBook documents. The script can be downloaded from: http://www.vim.org/scripts/script.php?script_id=38.

> Grab the file, then untar it. Copy the `dbhelper.vim` to your `.vimscripts` directory if you have one.
>
> ```
> $ mkdir .vimscripts
> $ cp dbhelper.vim .vimscripts
> ```
>
> You'll also have to convert the `dbhelper.vim` file to unix formatting:
>
> ```
> $ dos2unix dbhelper.vim
> ```
>
> Next, edit your `.vimrc` file and add the line: **`source /home/yourname/.vimscripts/dbhelper.vim`**
>
> To use the scripts, enter vi and go into insert mode. Press **,** (comma) followed by the shortcut. For example: **`,dtbk`**

B.2.2.6. XMLForm

http://www.datamech.com/XMLForm/

This web-based application allows you to put in the URL for XML source, or copy and paste the XML directly into the web form. The application then breaks down your document into a series of form fields that hide the DocBook tags so that you may edit the content directly. Version 5 is available from http://www.datamech.com/XMLForm/formGenerator5.html. This application is best on shorter documents (less than 20 pages printed).

As this is an on-line tool, it will be good for small updates only.

B.2.2.7. XMLmind XML Editor (XXE)

http://www.xmlmind.com/xmleditor

David Horton offers the following information:

> I am a big fan of XMLMind's XXE editor and XFC FO converter. It is "free as in beer," but not necessarily "free as in speech." Very liberal license for personal use however. It's Java-based so it works on all sorts of OS's.

B.3. Validation

B.3.1. Why Validate Your Document

The LDP uses a number of scripts to distribute your document. These scripts submit your document to the LDP's CVS (a free document version management system), and then they transform your document to other formats that users then read. Your document will also be mirrored on a number of sites worldwide (yet another set of scripts).

In order for these scripts to work correctly, your document must be both "well formed" and use "valid markup". *Well formed* means your document follows the rules that XML is expecting: it complies with XML grammar rules. *Valid markup* means you only use elements or tags which are "valid" for your document: XML vocabulary rules are applied.

If your document is not well formed or uses invalid markup, the scripts will not be able to process it. As a result, your revised document will not be distributed.

The Docbook Section

There is more information about how to validate your document in the DocBook section. Check out Section B.3 for more help with validating your document.

B.3.2. Validation for the Faint of Heart

Your life is already hard enough without having to install a full set of tools just to see if you validate as well. You can upload your raw XML files to a web site, then go to http://validate.sf.net, enter the URL to your document, then validate it.

External entities

When this information was added to the Author Guide external entities were not supported. Follow the instructions provided on the Validate site if you have trouble.

B.3.3. Validation for the Not So Faint Of Heart

B.3.3.1. Catalogs

XML and SGML files contain most of the information you need; however, there are sometimes entities which are specific to SGML in general. To match these entities to their actual values you need to use a *catalog*. The role of a catalog is to tell your system where to find the files it is looking for. You may want to think of a catalog as a guide book (or a map) for your tools.

Most distributions (Red Hat/Fedora and Debian at least) have a common location for the main SGML catalog file, called `/etc/sgml/catalog`. In times past, it could also be found in `/usr/lib/sgml/catalog`.

The structure of XML catalog files is not the same as SGML catalog files. The section on tailoring a catalog (see Section B.3.4) will give more details about what these files actually contain.

If your system cannot find the catalog file, or you are using custom catalog files, you may need to set the

`SGML_CATALOG_FILES` and `XML_CATALOG_FILES` environment variables. Using **echo `$SGML_CATALOG_FILES`**, check to see if it is currently set. If a blank line is returned, the variable has not been set. Use the same command to see if `XML_CATALOG_FILES` is set as well. If the variables are not set, use the following example to set them now.

Example B-1. Setting the SGML_CATALOG_FILES and XML_CATALOG_FILES Environmental Variables

```
bash$ export SGML_CATALOG_FILES="/etc/sgml/catalog"
```

To make this change permanent, you can add the following lines to your `~/.bashrc` file.

```
SGML_CATALOG_FILES="/etc/sgml/catalog"
export SGML_CATALOG_FILES
```

If you installed XML tools via a RedHat or Debian package, you probably don't need to do this step. If you are using a custom XML catalog you will definitely need to do this. There is more on custom catalogs in the next section. To ensure my backup scripts grab this custom file, I have added mine in a sub-directory of my home directory named "docbook".

```
bash$ export XML_CATALOG_FILES="/home/user/docbook/db-catalog.xml"
```

You can also change your `.bashrc` if you want to save these changes.

```
XML_CATALOG_FILES="/home/user/docbook/db-catalog.xml"
export XML_CATALOG_FILES
```

If you are adding the changes to your `.bashrc` you will not see the changes until you open a new terminal window. To make the changes immediate in the current terminal, "source" the configuration file.

B.3.4. Creating and modifying catalogs

In the previous section I mentioned a catalog is like a guide book for your tools. Specifically, a catalog maps the rules from the public identifier to your system's files.

At the top of every DocBook (or indeed every XML) file there is a DOCTYPE which tells the processing tool what kind of document it is about to be processed. At a minimum this declaration will include a public identifier, such as `-//OASIS//DTD DocBook V4.2//EN`. This public identifier has a number of sections all separated by `//`. It contains the following information: ISO standard if any (`-` -- in this case there is no ISO standard), author (OASIS), type of document (DTD DocBook V4.2), language (English). Your DOCTYPE may also include a URL.

A public identifier is useless to a processing tool, as it needs to be able to access the actual DTD. A URL is useless if the processing tool is off-line. To help your processor deal with these problems you can download all of the necessary files and then "map" them for your processing tools by using a catalog.

If you are using SGML processing tools (for instance Jade), you will need an SGML catalog. If you are using XML processing tools (like XSLT), you will need an XML catalog. Information on both is included.

B.3.4.1. SGML Catalogs

Example B-2. Example of an SGML catalog

```
❶
-- Catalog for the Conectiva Styles --

OVERRIDE YES
                                                          ❷
PUBLIC "-//Conectiva SA//DTD DocBook Conectiva variant V1.0//EN"
                        "/home/ldp/styles/books.dtd"

DELEGATE "-//OASIS"
                "/home/ldp/SGML/dtds/catalog.dtd"
                                                          ❸
DOCTYPE BOOK /home/ldp/SGML/dtds/docbook/db31/docbook.dtd

-- EOF --
```

❶
Comment. Comments start with "--" and follow to the end of the line.

❷
The public type association *"-//Conectiva SA//DTD books V1.0//EN"* with the file `books.dtd` on the directory `/home/ldp/styles`.

❸
Comment signifying the end of the file.

As in the example above, to associate an identifier to a file just follow the sequence shown:

1. Copy the identifier *PUBLIC*
2. Type the identifying text
3. Indicate the path to the associated file

B.3.4.1.1. Useful commands for catalogs

The most common mappings to be used in catalogs are:

`PUBLIC`
: The keyword `PUBLIC` maps public identifiers for identifiers on the system.

`SYSTEM`
: The `SYSTEM` keyword maps system identifiers for files on the system.

SYSTEM "http://nexus.conectiva/utilidades/publicacoes/livros.dtd" "publicacoes/livros.dtd"

`SGMLDECL`
: The keyword `SGMLDECL` designates the system identifier of the SGML statement that should be used.

SGMLDECL "publishings/books.dcl"

`DTDDECL`

Similar to the SGMLDECL the keyword DTDDECL identifies the SGML statement that should be used. DTDDECL makes the association of the statement with a public identifier to a DTD. Unfortunately, this association isn't supported by the open source tools available. The benefits of this statement can be achieved somehow with multiple catalog files.

DTDDECL "-//Conectiva SA//DTD livros V1.0//EN" "publicacoes/livros.dcl"

CATALOG
: The keyword CATALOG allows a catalog to be included inside another. This is a way to make use of several different catalogs without the need to alter them.

OVERRIDE
: The keyword OVERRIDE informs whether an identifier has priority over a system identifier. The standard on most systems is that the system identifier has priority over the public one.

DELEGATE
: The keyword DELEGATE allows the association of a catalog to a specific type of public identifier. The clause DELEGATE is very similar to the CATALOG, except for the fact that it doesn't do anything until a specific pattern is specified.

DOCTYPE
: If a document starts with a type of document, but has no public identifier and no system identifier the clause DOCTYPE associates this document with a specific DTD.

B.3.4.2. XML Catalogs

The following sample catalog was provided by Martin A. Brown.

Example B-3. Sample XML Catalog file

```
<?xml version="1.0"?>
<!DOCTYPE catalog PUBLIC "-//OASIS/DTD Entity Resolution XML Catalog V1.0//EN"
          "http://www.oasis-open.org/committees/entity/release/1.0/catalog.dtd">

<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">

<public publicId="-//OASIS//DTD DocBook XML V4.2//EN"
       uri="/home/mabrown/docbook/dtds/4.2/docbookx.dtd"/>
   <uri name="http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd"
       uri="/home/mabrown/docbook/dtds/4.2/docbookx.dtd"/>
   <uri name="http://docbook.sourceforge.net/release/xsl/current/xhtml/docbook.xsl"
       uri="/home/mabrown/docbook/xsl/xhtml/docbook.xsl"/>
   <uri name="http://docbook.sourceforge.net/release/xsl/current/xhtml/chunk.xsl"
       uri="/home/mabrown/docbook/xsl/xhtml/chunk.xsl"/>
   <uri name="http://docbook.sourceforge.net/release/xsl/current/xhtml/profile-chunk.xsl"
       uri="/home/mabrown/docbook/xsl/xhtml/profile-chunk.xsl"/>
</catalog>
```

B.3.5. Validating XML

B.3.5.1. nsgmls

You can use nsgmls, which is part of the jade suite (on Debian apt-get the docbook-utils package, see Section B.4.2), to validate SGML or XML documents.

```
bash$ nsgmls -s HOWTO.xml
```

If there are no issues, you'll just get your command prompt back. The *-s* tells nsgmls to show only the errors.

Function not found

If you get errors about a function not being found, or something about an ISO character not having an authoritative source, you may need to point **nsgmls** to your `xml.dcl` file. For Red Hat 9, it will look like this: **nsgmls *-s* /usr/share/sgml/xml.dcl *HOWTO.xml***

For more information on processing files with Jade/OpenJade please read DocBook XML/SGML Processing Using OpenJade.

B.3.5.2. onsgmls

This is an alternative to nsgmls. It ships with the OpenJade package. This program gives more options than nsgmls and allows you to quietly ignore a number of problems that arise while trying to validate an XML file (as opposed to an SGML file). This also means you don't have to type out the location of your `xml.dcl` file each time.

I was able to simply use the following to validate a file with only error messages that were related to my markup errors.

```
bash$ onsgmls -s HOWTO.xml
```

According to Bob Stayton you can also turn off specific error messages. The following example turns off XML-specific error messages.

```
bash$ onsgmls -s -wxml -wno-explicit-sgml-decl HOWTO.xml
```

B.3.5.3. xmllint

You can also use the xmllint command-line tool from the libxml2 package to validate your documents. This tool does a simple check on completeness of tags and whether all tags that are opened, are also closed again. By default xmllint will output a results tree. So if your document comes out until the last line, you know there are no heavy errors having to do with tag mismatches, opening and closing errors and the like.

To prevent printing the entire document to your screen, add the *--noout* parameter.

```
bash$ xmllint --noout HOWTO.xml
```

If nothing is returned, your document contains no syntax errors. Else, start with the first error that was reported. Fix that one error, and run the tool again on your document. If it still returns output, again fix the first error that you see, don't botter with the rest since further errors are usually generated because of the first one.

If you would like to check your document for any errors which are specific to your Document Type Definition, add *--valid*.

```
bash$ xmllint --noout --valid HOWTO.xml
```

The **xmllint** tool may also be used for checking errors in the XML catalogs, see the man pages for more info on how to set this behavior.

If you are a Mac OSX or Windows user, you may also want to check out tkxmllint, a GUI version of xmllint. More information is available from: http://tclxml.sourceforge.net/tkxmllint.html.

Example B-4. Debugging example using xmllint

The example below shows how you can use xmllint to check your documents. I've created some errors that I made a lot, as a beginning XML writer. At first, the document doesn't come through, and errors are shown:

```
bash$ xmllint ldp-history.xml
ldp-history.xml:22: error: Opening and ending tag mismatch: articlinfo line 6 and articleinfo
</articleinfo>
              ^
ldp-history.xml:37: error: Opening and ending tag mismatch: listitem line 36 and orderedlist
</orderedlist>
              ^
ldp-history.xml:39: error: Opening and ending tag mismatch: orderedlist line 34 and sect2
</sect2>
        ^
ldp-history.xml:46: error: Opening and ending tag mismatch: sect1 line 41 and para
for many authors to contribute their part in their area of specialization.</para
                                                                               ^
ldp-history.xml:57: error: Opening and ending tag mismatch: para line 55 and sect1
</sect1>
        ^
ldp-history.xml:59: error: Opening and ending tag mismatch: sect2 line 31 and article
</article>
          ^
ldp-history.xml:61: error: Premature end of data in tag sect1 line 24

^
ldp-history.xml:61: error: Premature end of data in tag article line 5

^
```

Now, as we already mentioned, don't worry about anything except the first error. The first error says there is an inconsistency between the tags on line 6 and line 22 in the file. Indeed, on line 6 we left out the "e" in "articleinfo". Fix the error, and run **xmllint** again. The first complaint now is about the offending line 37, where the closing tag for list items has been forgotten. Fix the error and run the validation tool again, until all errors are gone. Most common errors include forgetting to open or close the paragraph tag, spelling errors in tags and messed up sections.

B.4. Transformations

TLDP will convert your document

This section is about how to transform documents from DocBook to other formats. If you do not need to transform documents for your own web site, or to proof read the content, please *skip this section*.

If you would like to transform your documents for proofreading purposes, please use the XML to HTML on-line converter. You will need to upload your XML file(s) to a web site. Then simply drop the URL into the form and click the submit button. Your document will be magically transformed into a beautiful (and legible) HTML document. External files are supported. You may use either absolute or relative URIs.

Another easy-to-use package is xmlto. It is a front-end for xsltproc. It is available as a RedHat, Debian (etc) package or can be downloaded from http://cyberelk.net/tim/xmlto/. You can use it to convert documents with:

```
bash$ xmlto html mydoc.xml
bash$ xmlto txt mydoc.xml
```

You do not ever need to transform documents before submitting to the LDP. The LDP volunteers have a system which transforms your DocBook file into HTML, PDF and plain text formats. There, you've been warned.

Still here? Great! Transformations are a pretty basic requirement to get what you've written from a messy tag-soup into something that can be read. This section will help you get your system set up and ready to transform your latest document into other formats. This is very useful is you want to *see* your document before you release it to the world.

There are currently two ways to transform your document: Document Style Semantics and Specification Language (DSSSL); and XML Style sheets (XSLT). Although the LDP web site uses DSSSL to convert documents you may use XSLT if you want. You only need to choose *one* of these methods. For more information about DSSSL read: Section B.4.1, for more information about XSLT read: Section B.4.3.

B.4.1. DSSSL

There are three basic requirements to transform a document using DSSSL:

- The Document Style and Semantics Specification Language files (these are plain text files). Section B.4.1.1
- The Document Type Definition file which matches the DOCTYPE of your document (this is a plain text file). Section B.5
- A processor to do the actual work. Section B.4.1.2

B.4.1.1. The Style Sheets

There are two versions of the Document Style Semantics and Specification Language used by the LDP to transform documents from your raw DocBook files into other formats (which are then published on the Web). The LDP version of the style sheets requires the Norman Walsh version--which basically means if you're using DSSSL the Norman Walsh version can be considered a requirement for system setup.

Norman Walsh DSSSL http://docbook.sourceforge.net/projects/dsssl/. The Document Style Semantics and Specification Language tells Jade (see Section B.4.1.2) how to render a DocBook document into print or on-line form. The DSSSL is what converts a `title` tag into an <h1> HTML tag, or to 14 point bold Times Roman for RTF, for example. Documentation for DSSSL is located at the same site. Note that modifying the DSSSL doesn't modify DocBook itself. It merely changes the way the rendered text looks. The LDP uses a modified DSSSL (see below).

LDP DSSSL http://www.tldp.org/authors/tools/ldp.dsl. The LDP DSSSL requires the Norman Walsh version (see above) but is a slightly modified DSSSL to provide things like a table of contents.

Example B-5. "Installing" DSSSL style sheets

Create a base directory to store everything such as `/usr/share/sgml/`. Copy the DSSSL style sheets into a sub-directory named `dsssl`.

B.4.1.2. DSSSL Processors

There are two versions of the Jade processor: the original version by James Clark; and an open-source version of approximately the same program, OpenJade. You only need *one* of these programs. It should be installed *after* the DTD and DSSSL have been "installed."

DSSSL Transformation Tools

Jade
: ftp://ftp.jclark.com/pub/jade/

  Currently, the latest version of the package is `jade-1.2.1.tar.gz`.

  Jade is the front-end processor for SGML and XML. It uses the DSSSL and DocBook DTD to perform the verification and rendering from SGML and XML into the target format.

OpenJade
: http://openjade.sourceforge.net/

  An extension of Jade written by the DSSSL community. Some applications require jade, but are being updated to support either software package.

B.4.1.3. System Setup for DSSSL Transformations

1. Tell your system where to find the SGML_CATALOG_FILES (yes, even if you are using XML). You can find an example of how to do this in Example B-1.
2. Download the DSSSL and DTD files and copy them into your working directory. You can find an example of how to do this in Example B-5 and Example B-7.

B.4.1.4. Transformations with DSSSL

Once your system is configured (see the previous section), you should be able to start using jade to transform your files from XML to XHTML.

To create individual HTML files, point jade at the correct DSL (style sheet). The following example uses the LDP style sheet.

```
bash$ jade -t xml -i html \
        -d /usr/local/sgml/dsssl/docbook/html/ldp.dsl#html \
HOWTO.xml
```

If you would like to produce a single-file HTML page, add the *-V nochunks* parameter. You can specify the name of the final HTML file by appending the command with *> output.html*.

```
bash$ jade -t xml -i html -V nochunks \
        -d /usr/local/sgml/dsssl/stylesheets/ldp.dsl#html \
HOWTO.sgml > output.html
```

***Not a function name* errors**

If you get an error about "is not a function name", you will need to add a pointer to `xml.dcl`. It has to be listed immediately before the pointer to your XML document. Try one of the following locations: `/usr/lib/sgml/declaration/xml.dcl`, or `/usr/share/sgml/declaration/xml.dcl`. Use locate to find the file if it is not in either of those two places. The modified command would be as follows:

```
bash$ jade -t xml -i html \
       -d /usr/local/sgml/dsssl/docbook/html/ldp.dsl#html \
       /usr/lib/sgml/declaration/xml.dcl HOWTO.xml
```

If you would like to create print-friendly files instead of HTML files, simply change the style sheet that you are using. In the file name above, note "html/ldp.dsl" at the end. Change this to "print/docbook.dsl", or if you want XHTML output, instead of HTML, change the file name to "xhtml/docbook.dsl".

B.4.1.4.1. Changing CSS Files

If you want your HTML files to use a specific CSS stylesheet, you will need to edit `ldp.dsl`. Immediately after `;; End of $verbatim-display$ redefinition` add the following lines:

```
(define %stylesheet-type%
  ;; The type of the stylesheet to use
  "text/css")

(define %stylesheet%
  ;; Name of the css stylesheet to use, use value #f if you don't want to
  ;; use css stylesheets
  "base.css")
```

Replace `base.css` with the name of the CSS file you would like to use.

B.4.2. The docbook-utils Package

The docbook-utils provide commands like **db2html**, **db2pdf** and **db2ps**, based on the **jw** scripts, that is a front-end to Jade. These tools ease the everyday management of documentation and add comfortable features.

The package, originally created by RedHat and available from http://sources.redhat.com/docbook-tools/ can be installed on most systems.

Example B-6. Example creating HTML output

After validating your document, simply issue the command **db2html** **`mydoc.xml`** to create (a) HTML file(s). You can also use the docbook-utils as validation tools. Remember: when errors occur, always start by solving only the first problem. The rest of the problems may be fixed when you fix the first error.

If you get errors about a function name, please read .

B.4.2.1. Using CSS and DSL for pretty output

You can define your own additional DSL instructions, which can include a pointer to a personalized CSS file. Sample DSL and CSS files are provided in Appendix A.

The sample DSL file will create a table of contents, and have all HTML files start with the prefix `intro2linux-` and end with a suffix of `.html`. The `%stylesheet%` variable points to the CSS file which should be called by your HTML file.

To use a specific DSL style sheet the following command should be used:

db2html -d mystyle.dsl mydoc.xml

You can compare the result here: http://tille.xalasys.com/training/unix/ is a book formatted with the standard tools; http://tille.xalasys.com/training/tldp/ is one using personalized DSL and CSS files. Soft tones and special effects, for instance in buttons, were used to achieve maximum effect.

B.4.3. XSL

There are alternatives to DSSSL and Jade/OpenJade.

When working with DocBook XML, the LDP offers a series of XSL[2] style sheets to process documents into HTML. These style sheets create output files using the XML tool set that are similar to those produced by the SGML tools using ldp.dsl.

The major difference between using `ldp.dsl` and the XSL style sheets is the way that the generation of multiple files is handled, such as the creation of a separate file for each chapter, section and appendix. With the SGML tools, such as jade or openjade, the tool itself was responsible for generating the separate files. Because of this, only a single file, `ldp.dsl` was necessary as a customization layer for the standard DocBook DSSSL style sheets.

With the DocBook XSL style sheets, generation of multiple files is controlled *by the style sheet*. If you want to generate a single file, you call one style sheet. If you want to generate multiple files, you call a different style sheet. For that reason the LDP XSL style sheet distribution is comprised of four files:

1. `tldp-html.xsl` - style sheet called to generate a *single* file.
2. `tldp-html-chunk.xsl`[3] - style sheet called to generate multiple files based on chapter, section and appendix elements.
3. `tldp-html-common.xsl` - style sheet containing the actual XSLT transformations. It is called by the other two HTML style sheets and is *never* directly called.
4. `tldp-print.xsl` - style sheet for generation of XSL Formatting Objects for print output.

You can find the latest copy of the files at http://my.core.com/~dhorton/docbook/tldp-xsl/. The package includes installation instructions which are duplicated at http://my.core.com/~dhorton/docbook/tldp-xsl/doc/tldp-xsl-howto.html. The short version of the install instructions is as follows: Download and unzip the latest package from the web site. Take the files from the `html` directory of TLDP-XSL and put them in the `html` directory of Norman Walsh's stylesheets. Take the file from the TLDP-XSL `fo` directory and put it in the Norman Walsh `fo` directory.

Once you have installed these files you can use xsltproc to generate HTML files from your XML documents. To transform your XML file(s) into a single-page HTML document use the following command:

```
bash$  xsltproc -o HOWTO.html /usr/local/sgml/stylesheets/tldp-html.xsl HOWTO.xml
```

To generate a set of linked HTML pages, with a separate page for each `chapter`, `sect1` or `appendix`, use the following command:

```
bash$ xsltproc /usr/share/sgml/stylesheets/tldp-html-chunk.xsl HOWTO.xml
```

Note that you never directly call the style sheet `tldp-html-common.xsl`. It is called by both of the other two style sheets.

B.4.3.1. Changing CSS Files

If you want your HTML files to use a specific CSS stylesheet, you will need to edit `tldp-html-common.xsl`. Look for a line that ressembles `<xsl:param name="html.stylesheet" select="'style.css'"/>`.

Replace `style.css` with the name of the CSS file you would like to use.

B.5. DocBook DTD

The DocBook DTD defines the structure of a DocBook document. It contains rules about how the elements may be used; and what the elements ought to be describing. For example: it is the DocBook DTD which states all `warning`s are to *warn* the reader (this is the definition of the element); and may not contain plain text (this is the content model--and the bit which forces you to wrap your warning text in a `para` or perhaps a list).

Versions

It is important that you download the version(s) that match your document. You may want to configure your system now to deal with "all" DocBook DTDs if you are going to be editing older LDP documents. If you are a new author, you only need the first one listed: XML DTD for DocBook version 4.2.

The XML DTD is available from http://www.oasis-open.org/xml/4.2/. The LDP prefers this version of the DocBook DTD.

If you are going to be working with SGML versions of DocBook you will need one (or both) of: http://www.oasis-open.org/docbook/sgml/4.1/docbk41.zip or http://www.oasis-open.org/docbook/sgml/3.1/docbk31.zip

Example B-7. "Installing" DocBook Document Type Definitions

Create a base directory to store everything such as `/opt/local/sgml/`. Copy the DTDs into a sub-directory named `dtd`.

Do not edit DTD files

The DocBook standard is described in these files. If you change these files, you are no longer working with DocBook.

B.6. Formatting Documents

B.6.1. Inserting a summary on the initial articles page

A feature that might be valuable in some cases is the insertion of the summary on the initial page of an article. DocBook articles do not include it as a standard feature.

To enable this, it is necessary to modify the style sheet file.

Example B-8. Style sheet to insert summaries in articles

```
<!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN" [
<!entity html-docbook PUBLIC "-//Norman Walsh//DOCUMENT DocBook HTML Stylesheet//EN" CDATA DSSSL>
<!entity print-docbook PUBLIC "-//Norman Walsh//DOCUMENT DocBook Print Stylesheet//EN" CDATA DSSSL>
]>

<style-sheet>
<style-specification use="html">
<style-specification-body>

; Includes a summary at the beginning of an item.
(define %generate-article-toc% #t)

</style-specification-body>
</style-specification>
<style-specification use="print">
<style-specification-body>

; Includes a summary at the beginning of an item.
(define %generate-article-toc% #t)

</style-specification-body>
</style-specification>
<external-specification id="html" document="html-docbook">
<external-specification id="print" document="print-docbook">
</style-sheet>
```

B.6.2. Inserting indexes automatically

Although DocBook has markups for the composition of them, indexes are not generated automatically. The **collateindex.pl** command allows indexes to be generated from the source DocBook for inclusion into the finished/transformed document.

1. Process the document with jade using the style to *HTML* with the option `-V html-index`.

   ```
   bash$ jade -t sgml \
           -d html/docbook.dsl -V html-index document.sgml
   ```
2. Generate the `index.sgml` file with **collateindex.pl**.

   ```
   bash$  perl collateindex.pl \
           -o index.sgml HTML.index
   ```

For the above example to work, it is necessary to define an *external entity* by calling the file `index.sgml`.

Example B-9. Configuring an external entity to include the index

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD
DocBook V3.1//EN" [

<!-- Insertion of the index --> <!entity index SYSTEM
```

```
"index.sgml"> ]>
```

See also Section D.4 for information on how to insert necessary information on the text.

Odd behavior generating indexes for print versions

Remember that if you're trying to get Tables of Contents or Indexes on PS or PDF output you'll need to run jadetex or pdfjadetex at least three times. This is required by TeX but not by DocBook or related applications.

Appendix C. Concurrent Version System (CVS)

The LDP provides optional CVS access to its authors. This enables collaborative writing and has the following positive effects:

1. CVS will keep an off-site backup of your documents. In the event that you hand over a document to another author, they can just retrieve the document from CVS and continue on. In the event you need to go back to a previous version of a document, you can retrieve it as well.
2. However difficult from an organizational point of view, it's great to have multiple people working on the same document. CVS enables you to do this. You can have CVS tell you what changes were made by another author while you were editing your copy, and integrate those changes.
3. CVS keeps a log of what changes were made. These logs (and a date stamp) can be placed automatically inside your documents when they are published.
4. CVS can be combined with scripts to automatically update the LDP web site with new documentation as it's written and submitted. This is not in place yet, but it is a goal. Currently, CVS updates signal the HOWTO coordinator to update the LDP web page, meaning that if you use CVS, you're not required to e-mail your XML code. (Although you do still need to send the submit list an email when you are ready for your document to be published, because the whole publishing process has not been fully automated yet.)

Access to our CVS repository

Only authors with at least three submissions get access to our CVS, see Appendix C.

You can browse the LDP CVS repository via the web at http://cvs.tldp.org/.

C.1. Getting a CVS account

CVS accounts will not be granted to all applicants

To be granted a CVS account you must qualify under one of the following categories:

- authors with documents already in the collection who have made a minimum of three submits to the LDP through `<submit@en.tldp.org>`
- technical and language reviewers approved by one of the Review Coordinators
- new authors in the review process (also requires approval from one of the Review Coordinators)

Please do not apply for a CVS account if you do not qualify.

If you qualify for a CVS account you may apply for one contacting CVS master Sergiusz mailto:ser@gnu.org Include information about which documents you maintain.

C.2. Using CVS

C.2.1. Setting Up Your CVS Account

First you'll need to get an account at the LDP's CVS Repository. Please see the notes above on obtaining an account. This repository houses various documents including HOWTOs and Guides. Documents are sorted by the type of document (for example a HOWTO or a Guide), and by the markup language the document uses (for example DocBook or LinuxDoc).

When your account is ready you can log in using one of the following commands. In all instances `your_userid` should be replaced by the user name you were issued in the response email. You will be prompted for a password after this first step.

Initializing Your CVS Account

Linux system
: **cvs *-d :pserver:your_userid@cvs.tldp.org:/cvsroot* login**

Windows system
: **set CVSROOT=":pserver:*your_userid*@cvs.tldp.org:/cvsroot"**

  cvs *-d %CVSROOT%* login

Wait patiently while the system tries to log you in. It can often take more that 10-20 seconds for the system to either accept (or reject) your password. Once you've used **cvs login** for the first time and have been given access to the system, your password is stored in `.cvspass` and you will not have to use **cvs login** again. Just set the CVSROOT with the export command listed above and continue on. If TLDP's CVS server is the only one you work with, you might also add an **export CVSROOT** line to your `~/.bashrc` shell configuration file.

C.2.2. Getting the Documents

You can get the entire repository (about 150 MB) with: **cvs checkout LDP**

Or you can get the source for your own document with: **cvs checkout LDP/howto/docbook/YOUR-HOWTO.xml** OR **cvs checkout LDP/guide/docbook/YOURGUIDE**

Windows users will need to use a modified version of this command. Instead they should use: **cvs -d %CVSROOT% checkout LDP/howto/docbook/YOUR-HOWTO.xml**

Keep an overview

checkout will add the full directory structure from tldp.org on down. Although it doesn't really matter where you put these files on your local file system you may not want to bury the directories too deeply.

C.2.3. CVS Commands

CVS Commands: a brief reminder

commit
: This CVS command will upload your changes to the CVS server.

  Please be sure to include a useful description of the changes that have been made to your document.

  If you want to bypass the editor screen you can use

  cvs commit -m "A description of the work done on this version of the document."

  Ready for publication warning

You must still email `<submit@en.tldp.org>` when you are ready to have your changes appear on the live site. Your email should include the relative path to the file(s) in the LDP CVS tree that you wish to update.

add

You can add new files to your CVS repository. These may be image files or additional XML files. First check that your HOWTO is in its own directory. You may want to coordinate with the people at `<submit@en.tldp.org>` to ensure you can add graphics or other files to your HOWTO.

Copy the files you want to add into your local CVS repository (where all of your downloaded/working files are). Then:

cvs add *filename*

After you've added the files, you still need to **commit** them to the repository (see above).

remove

$Id: cvs.xml,v 1.32 2011/01/14 16:24:52 serek Exp $

While this is not a CVS "command" it can be used to automatically insert information about the file including the time and date it was last modified, the version number it was assigned by the CVS and the filename of this particular file. The output will look like this: `$Id: cvs.xml,v 1.9 2002/04/21 09:44:26 serek Exp $`

If you need to change a file name, you still need to use the **add** command. First remove the copy of the file from your local disk. Then remove it from the CVS tree with: **cvs remove *filename***. As with the **add** command, you need to **>commit** your removed file. Finally, now that the old file has been removed, add your new file using the instructions above (first **add** and then **commit** the additional file).

C.2.3.1. Recovering old versions

There you are, typing away, when you screw up. Real bad. Doesn't matter what it is, but suffice it to say that you've toasted not only the version on your local drive, but created a new version on the CVS server. What you need to do is go back in time and resurrect an older version of your file.

To do this, you'll need to know the version number of the file you want to retrieve. **cvs diff** will give a list of revisions if there are differences. You can pick the revision number, subtract one, and that is probably the revision you want to look at.

The command

cvs -Q update -p -r *revision filename*

will output to stdout the contents of the *revision* version of *filename*. You can pipe it to **more** or redirect the output to a file. Conveniently, you can redirect stdout to a file called *filename*. Your local file is now the revision you want, and

cvs update

will update the CVS server with the new (old) version of *filename*.

C.3. CVS Resources

If you're completely new to CVS, there are a few web pages you may want to look at which can help you out:

- http://cvshome.org/docs/blandy.html
- http://www.loria.fr/~molli/cvs/doc/cvs_toc.html

Appendix D. DocBook: Sample Markup

D.1. General Tips and Tricks

For a general overview of what markup is all about, please read Chapter 5

- An editor capable of inserting elements according to the DTD will help a lot since it will enforce the DTD. This way you can be sure that no invalid elements were added anywhere in your document.
- In order to ensure that future changes are as easy as possible, authors should try to keep compatibility with the XML version of the DocBook DTD. This means keeping element names in lower case, using double quotes in all attributes, and not omitting end tags. Most tools that automatically insert elements (like psgml+emacs) follow these rules automatically or with some fine tuning.
- Each type of document created has a specific structure. This document is in "book" format. Most authors, however, will want to use the shorter "article" format instead. Templates are available from Appendix A.

D.1.1. Useful markup

Table D-1 shows some markup that is useful for generating generic documents. Remember that some elements are valid only on some contexts.

Check several formats

Sometimes the appearance of a particular tag changes from one conversion format to another. As a beginner in DocBook writing, you may wish to see how your document looks in several formats before you publish them. You are advised to look at how your document is presented in HTML, PDF and PostScript, since these formats will be made available by TLDP once you publish your document.

Better too much than not enough

Since the formatting depends on the output style chosen, it's recommended to use as much markup as possible. Even if the appearance of the output doesn't seem to change with the standard output style, there may be specific output formats that will make these tags stand out.

Table D-1. Useful markup

Description	Sample markup	Result
E-mail address	`<email>`address@domain`</email>`	`<address@domain>`
About the author	`<author>`...`</author>`	(see example below)

Author's name	`<firstname>Mary</firstname>`
	`<othername>Margaret</othername>`
	`<surname>O'Hara</surname>`
		Mary Margaret O'Hara

Keys' name (printings on the keyboard)	`<keycap>`F1`</keycap>`	**F1**
Symbol represented by	`<keysym>`KEY_F1`</keysym>`	KEY_F1

the keys		
Key's code	<keycode>0x3B</keycode>	0x3B
Combinations or sequences of keys	<keycombo> <keycap>Ctrl</keycap> <keycap>S</keycap> </keycombo>	**Ctrl**-**S**
Program Menus	<guimenu>File</guimenu>	File
Menu Items	<guimenuitem>Save</guimenuitem>	Save
Menu Sequences	<menuchoice> <shortcut> <keycombo> <keycap>Ctrl</keycap> <keycap>S</keycap> </keycombo> </shortcut> <guimenu>File</guimenu> <guimenuitem>Save</guimenuitem> </menuchoice>	File->Save (**Ctrl**-**S**)
Mouse Button	<mousebutton>left</mousebutton>	left
Application Names	<application>application</application>	application
Text Bibliographical Reference	<citation>reference</citation>	[reference]
Quote	<blockquote> <attribution>Text Author</attribution> <para>Quote Text.</para> </blockquote>	Quote Text. --Text Author
Index	(NA)	See Section D.4.
File Names	<filename>file</filename>	`file`
Directories	<filename id="directory">directory</filename>	`directory/`
Emphasize Text[a]	<emphasis>text</emphasis>	*text*
Footnotes	<footnote> <para>Footnote text</para> </footnote>	(See note at the end of this table for an example)
URLs	<ulink url="http://www.conectiva.com">Conectiva S.A.</>	Conectiva S.A.
Itemized (unnumbered) List	<itemizedlist> <listitem> <para>item</para> </listitem> <listitem> <para>item</para> </listitem> </itemizedlist>	• item • item
Ordered (numbered) List	<orderedlist> <listitem> <para>item</para> </listitem>	1. item 2. item

	<listitem> <para>item</para> </listitem> </orderedlist>	
Segmented List	<segmentedlist> <title>Binary to decimal conversion</title> <segtitle>Binary</segtitle> <segtitle>Decimal</segtitle> </seglistitem><seg>00</seg><seg>0</seg> </seglistitem> <seglistitem><seg>01</seg><seg>1</seg> </seglistitem> <seglistitem><seg>10</seg><seg>2</seg> </seglistitem> </segmentedlist>	**Binary to Decimal Conversion** **Binary:** 00 **Decimal:** 0 **Binary:** 01 **Decimal:** 1 **Binary:** 10 **Decimal:** 2
Variable List	<variablelist> <varlistentry> <term>Entry 1</term> <listitem> <para>Description</para> </listitem> </varlistentry> <varlistentry> <term>Entry 2</term> <listitem> <para>Description</para> </listitem> </varlistentry> </variablelist>	Entry 1 Description Entry 2 Description
Simple Lists	<simplelist type="horiz" columns="3"> <member>1</member> <member>2</member> <member>3</member> <member>4</member> <member>5</member> <member>6</member> </simplelist> <simplelist type="inline"> <member>A</member> <member>B</member> <member>C</member> <member>D</member> <member>E</member> <member>F</member> </simplelist>	1 2 3 4 5 6 A, B, C, D, E, F
Pictures	(NA)	See Section D.5
Glossary	<glossentry> <glossterm>Term</glossterm> <glossdef> <para>Definition</para> </glossdef> </glossentry>	(See the glossary at the end of this document)
		(NA)

Crossed References	<section id="secao"> ... </section> <section id="reference the other section"> ... <para>Please, see<xref linkend="secao" /> for more information.	
Notes: a. Text can be emphasized in a few ways. The most common ways are italics and bold. DocBook, however, supports only italics. The use of bold requires additional settings on the style sheet used.		

D.2. `<section>` and `<sectN>`: what's the difference?

Acknowledgment

These notes were provided by: John Daily and Saqib Ali (<saqib@seagate.com>).

<section> versus <sectN> is largely a question of flexibility. The stylesheets can make a <section> in a <section> look just like a <sect2> within a <sect1>, so there's no output advantage.

But, a <section> within a <section> can be extracted into its own top-level section, nested even more deeply, or moved to an entirely different part of the document, without it and its own <section> children being renamed. That is not true of the numbered section tags, which are very sensitive to rearrangements. This can be easier for authors who are new to DocBook than using <sectN>.

The main idea behind creating structured data is that it should be easy to access and query. One should be able to retrieve a subsection of any structured data, by using querying languages for XML (XPath and XQuery). <sectN> are useful when traversing a document using XPath/XQuery. <sectN> gives more flexibility, and control while writing an XPath expression.

A well-defined, valid and well-structured document makes it easier for one to write XPath/Query to retreive "specific" data from a document. For example you can use XPath to retrieve information in the <sect3> block with certain attributes. However if you just used <section> for all subsections, it becomes harder to retrieve the block of information that you need.

So which one should you use? The one you feel most comfortable with is a good place to start. This document is written with <section>s. You may, or may not, think this is a good idea. :)

D.3. Command Prompts

There are likely as many ways of doing this as there are DocBook authors; however, here are two ways that you might find useful. Thanks to Y Giridhar Appaji Nag and Martin Brown for the markup used here.

Example D-1. Command Prompt with `programlisting`

```
<programlisting>
<prompt># </prompt><userinput><command>cd</command> /some/dir</userinput>
<prompt># </prompt><userinput><command>ls</command> -l</userinput>
</programlisting>
```

Displays as:

```
# cd /some/dir
# ls -l
```

Example D-2. Command Prompt with screen

First create a general entity in the internal subset at the very beginning of your document. This entity will define a name for the shortcut which you can use to display the full prompt at any point in your document. `<!ENTITY prompt "<prompt>[user@machine ~/dir]$</prompt>">`

For more information about entities, read Section D.8.

```
<screen>
&prompt; <userinput>cd /some/dir</userinput>
&prompt; <userinput>ls -l</userinput> </screen>
```

Displays as:

```
[user@machine ~/dir]$ cd /some/dir
[user@machine ~/dir]$ ls -l
```

If you would like to add the output of your commands you can add `<computeroutput>` text`</computeroutput>` within the `<screen>` or `<programlisting>` as appropriate.

D.4. Encoding Indexes

The generation of indexes depends on the markups inserted in the text.

Such markups will be processed afterwards by an external tool and will generate the index. An example of such a tool is the `collateindex.pl` script (see Section B.6.2). Details about the process used to generate these indexes are shown in Section B.6.2.

The indexes have nesting levels. The markup of an index is done by the code Example D-3.

Example D-3. Code for the generation of an index

```
<indexterm>
   <primary>Main level</primary>
   <secondary>Second level</secondary>
<tertiary>Third level</tertiary>
</indexterm>
```

It is possible to refer to chapters, sections, and other parts of the document using the *attribute* `zone`.

Example D-4. Use of the attribute zone

```
<section id="encoding-index">
<title>Encoding Indexes</title>
<indexterm zone="encoding-index">
<primary>edition</primary> <secondary>index</secondary>
</indexterm>
```

```
<para>
       The generation of indexes depend on the inserted markups on the text.
</para>
```

The Example D-4 has the code used to generate the entry of this edition on the index. In fact, since the attribute `zone` is used, the index statement could be located anywhere in the document or even in a separate file.

However, to facilitate maintenance the entries for the index were all placed after the text to which it refers.

Example D-5. Usage of values `startofrange` and `endofrange` on the attribute`class`

```
<para>Typing the text normally
    sometimes there's the need of
   <indexterm class="startofrange"
   id="example-band-index">
      <primary>examples</primary> <secondary>index</secondary>
   </indexterm> mark large amounts of
   text.</para>

   <para>Keep inserting the paragraphs
   normally.</para>

   <para>Until the end of the section
   intended to be indexed.  <indexterm
   startref="example-band-index" class="endofrange">.
   </para>
```

D.5. Inserting Pictures

It is necessary to insert pictures formats for all possible finished document types. For example: JPG files for web pages and EPS for PostScript and PDF files.

If you use the TeX format you'll need the images as a PostScript file. For on-line publishing you can use any kind of common image file, such as JPG, GIF or PNG.

The easiest way to insert pictures is to use the `fileref` attribute. Usually pictures are generated in JPG and in PostScript (PS or EPS).

Example D-6. Inserting a picture

```
<figure>
   <title>Picture's
   Title</title> <graphic fileref="images/file"></graphic>
</figure>
```

Replacing `<figure>` by `<informalfigure>` eliminates the need to insert a title for the picture.

There's still the `float` attribute on which the value `0` indicates that the picture should be placed exactly where the tag appears. The value `1` allows the picture to be moved to a more convenient location (this location can be described on the style sheet, or it can be controlled by the application).

D.5.1. Graphics formats

When submitting graphics to the LDP, please submit one set of graphics in `.eps`, and another in `.gif`, `.jpg` or `.png`. The preferred format is `.png` or `.jpg` due to patent problems with `.gif`.

You can use `.jpg` files for continuous-tone images such as photos or images with gradual changes in color. Use `.png` for simple images like diagrams, some screen shots, and images with a low number of colors.

D.5.2. Alternative Methods

The first alternative to Example D-6 is to eliminate the `<figure>` or `<informalfigure>` elements.

Another interesting alternative when you have decided to publish the text on media where pictures are not accepted, is the use of a wrapper, `<imageobject>`.

Example D-7. Using `<imageobject>`

```
<figure>
   <title>Title</title>
   <mediaobject>
      <imageobject>
 <imagedata fileref="images/file.eps" format="EPS">
      </imageobject>
   <imageobject>
 <imagedata fileref="images/file.jpg" format="JPG">
      </imageobject>
      <textobject>
 <phrase>Here there's an image of this example</phrase>
      </textobject>
   <caption>
   <para>Image Description. Optional. </para>
   </caption>
   </mediaobject>
</figure>
```

Files using the following formats are available BMP, CGM-BINARY, CGM-CHAR, CGM-CLEAR, DITROFF, DVI, EPS, EQN, FAX, GIF, GIF87A, GIF89A, IGES, JPEG, JPG, LINESPECIFIC, PCX, PIC, PS, SGML, TBL, TEX, TIFF, WMF, WPG.

This method presents an advantage: a better control of the application. The elements `<imageobject>` are consecutively tested until one of them is accepted. If the output format does not support images the `<textobject>` element will be used. However, the biggest advantage in usage of the format Example D-7 is that in DocBook `5.0`, the `<graphic>` element will cease to exist.

As a disadvantage, there is the need for more than one representation code of the same information. It is up to the author to decide how they will implement illustrations and pictures in the document, but for compatibility with future versions *I recommend* the use of this method for pictures and graphics.

Title page exception

`<mediaobject>`s will not display if they are used on a title page. For more information read: http://www.sagehill.net/docbookxsl/HtmlCustomEx.html#HTMLTitlePage

ASCII Images

You may also want to try converting your image to an ASCII representation of the file. JavE (Java ASCII Versatile Editor) can do this conversion for you. It can be downloaded from http://www.jave.de/. It has an easy to use GUI interface.

If you're more command-line oriented you may want to try: tgif (http://bourbon.usc.edu:8001/tgif/) and AA-Lib (http://aa-project.sourceforge.net/).

D.6. Markup for Metadata

D.6.1. Crediting Translators, Converters and Co-authors

There are several ways that these folks, as well as other contributors to your document, can be given some recognition for the help they've given you.

D.6.1.1. `<othercredit>`

All translators, converters and co-authors should be credited with an `<othercredit>` tag entry. To properly credit a translator or converter, use the `<othercredit>` tag with the `role` attribute set to "converter" or "translator", as indicated in the example below:

Example D-8. Other Credit

```
<othercredit role='converter'>
  <firstname>David</firstname>
  <surname>Merrill</surname>
  <contrib>Conversion from HTML to DocBook v3.1 (SGML).</contrib>
</othercredit>
```

D.6.1.2. Crediting Editors and Reviewers

To help track the review process all new documents must include a reference to the reviewers for the technical, language and metadata reviews.

Example D-9. Editor

```
<editor>
  <firstname>Tabatha</firstname>
  <surname>Marshall</surname>
  <contrib>Language review of version 0.8</contrib>
</editor>
```

D.6.2. `<revremark>`s

Within the `<revision>` tag hierarchy is a tag called `<revremark>`. Within this tag, you can make any brief notes you wish about each particular revision of your document.

D.6.3. Revision History

The <revhistory> tag should be used to denote the various revisions of the document. Specify the date, revision number and comments regarding what has changed.

Revisions should be listed with the most-recent version at the top (list in descending order).

D.6.4. Date formats

The <pubdate> tag in your header should list the publication date of this particular version of the document (coincide with the revision date). It should be in the following format:

```
<pubdate>2002-04-25</pubdate>
```

The date is in the format YYYY-MM-DD, which is one of the ISO 8601 standard formats for representing dates. For you Yanks out there (me too), think of it as going from the largest unit of time to the smallest.

D.6.5. Sample Article (or Book) Information Element

Here is a sample of a complete DocBook (SGML or XML) <articleinfo> element which contains some of the items and constructs previously described.

Example D-10. Sample Meta Data

```
<!--
   Above these lines in a typical DocBook article would be the article
   element (the immediate parent of the articleinfo element) and above
   that typically, the DOCTYPE declaration and internal subset.
  -->

<articleinfo>

  <!--
        Use "HOWTO", "mini HOWTO", "FAQ" in title, if appropriate
  -->

<title>Sample HOWTO</title>

<author>
<firstname>Your Firstname</firstname>
<surname>Your Surname</surname>
<affiliation>
<address><email>your email</email></address>
</affiliation>
</author>

<editor>
  <firstname>Tabatha</firstname>
  <surname>Marshall</surname>
  <contrib>Language review of version 0.8</contrib>
</editor>

<othercredit role='converter'>
  <firstname>David</firstname>
  <surname>Merrill</surname>
```

```
  <contrib>Conversion from HTML to DocBook v3.1 (SGML).</contrib>
</othercredit>

   <pubdate>YYYY-MM-DD</pubdate>

<revhistory>
<revision>
<revnumber>1.0</revnumber>
<date>YYYY-MM-DD</date>
<authorinitials>ABC</authorinitials>
</revremark>first official release</revremark>
</revision>

<revision>
<revnumber>0.9</revnumber>
<date>YYYY-MM-DD</date>
<authorinitials>ABC</authorinitials>
<revremark>First draft</revremark>
</revision>
</revhistory>

   <!--
                Provide a good abstract; a couple of sentences is sufficient
   -->

  <abstract>
<para>
       This is a sample DocBook (SGML or XML) HOWTO which has been
       constructed to serve as a template.
</para>
</abstract>

</articleinfo>
```

D.7. Bibliographies

Not everyone will choose to use the correct formatting for a bibliography. Most will use a list of some kind. And that's ok. But when you're ready to move to the next level, here's how to do it.

Below are two examples of bibliographic entries. The first is a very simple entry. It has only a title, URL and possibly a short description (abstract). The second is a little more complex and is for a full entry (for instance a book) with an ISBN, publisher and copyright date.

DocBook requirements for bibliographic references

At least one element within `<biblioentry>` is required, but it doesn't matter which one you have. So you might have a `<title>`, or a URL (`<bibliosource>`), or an `<author>`, or, ...

For a full list of all options, visit http://docbook.org/tdg/en/html/biblioentry.html. For more examples visit http://docbook.org/tdg/en/html/bibliography.html.

Displaying `<abstract>` content

By default `<abstract>`s do not display on web pages. You need to modify the `biblio.xsl` file. Do a search for the word "abstract" and then add this information inside the <xsl:template> tags. If that doesn't make sense, don't worry about it too much, but do be aware that it's required for the abstracts to show up.

```
<div xmlns="http://www.w3.org/1999/xhtml" class="{name(.)}">
    <xsl:apply-templates mode="bibliography.mode"/>
</div>
```

Example D-11. A Bibliography

```
<bibliography>
<title>Bibliography title</title>

<bibliodiv>
<title>Section title</title>
<biblioentry>
<title>Book or Web Site Title</title>
<bibliosource><ulink url=""/></bibliosource>
<abstract></abstract>
</biblioentry>

<biblioentry>
<title></title>
<bibliosource><ulink url=""/></bibliosource>
<author><firstname></firstname><surname></surname></author>
<copyright><year></year>
<holder></holder></copyright>
<editor><firstname></firstname><surname></surname></editor>
<isbn></isbn>
<publisher>
<publishername></publishername>
</publisher>
<abstract></abstract>
</biblioentry>
</bibliodiv>
</bibliography>
```

View *References* to see this in action.

D.8. Entities (shortcuts, text macros and re-usable text)

There may be some segments of text, or markup that you want to use over and over again. Instead of typing it up multiple times (and then having to edit it multiple times if you want to make a change) you can use *external entities*. Entities can give you a short cut to: re-using whole documents, text snippets, and special markup. Some common ways to use an entity would be for:

- **text macros for markup.** An example would be a company URL. By using a parameter entity you could refer simply to &my-company-url; instead of typing out the full <ulink url="http://foo.bar">Foo.bar</ulink> each time.
- **software license.** Such as the GNU Free Documentation License: it is long. And must be included in full in each document. By leaving the license in a separate file you can easily include it in many documents without having to redo the markup each time.
- **repeated text.** For instance an introduction to a topic which is included both as a summary in one section and as an introduction to the full information in another. Saves on editing time if you need to make changes to both sets of text.

Example D-12. Adding Entities

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
"http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd" [
<-- I can add comments here -->
<!ENTITY shortcut "Replace 'shortcut' with this text.">

<!ENTITY sc-to-a-file SYSTEM "anotherfile.xml">
<-- note: the square bracket on the third line is closed on the next
line--> ]>
```

To use these entities simply insert the name you gave the entity between a "&" (ampersand) and a ";" (semicolon). For example: "&shortcut;" would expand into "Replace 'shortcut' with this text"; "&sc-to-a-file;" would include the full contents of whatever is in *anotherfile.xml*.

An important feature while writing a text is the ability to check whether or not it will be presented in the final draft. It is common to have several parts of the text marked as draft, especially when updating an already existing document.

With the use of parameter entities, you can include or eliminate these drafts by altering only one line at the beginning of a document.

Example D-13. Use of parameter entities

```
<!ENTITY % review "INCLUDE"> ...
<![%review;[ <para>This paragraph will
be included on the draft when the entity "review" is defined with the
value "INCLUDE". </para> ]]>
```

The entity `review` might have several texts defined, as in Example D-13. When the changes to the text are considered final, you need only to remove parts of the text between the 3rd. and 6th. lines.

To keep the draft definitions and hide the text in the final draft, just alter the specification of the entity from `INCLUDE` to `IGNORE`.

D.9. Customizing your HTML files

D.9.1. HTML file names

By default, when separate HTML files are made, the SGML processor will assign arbitrary names to the resulting files. This can be confusing to readers who may bookmark a page only to have it change. Whatever your reasoning, here's how to make separate files named the way you want:

In your first `<article>` tag (which should be the only one) include an *id* parameter and call it "index". This will make your tag look like this:

```
<article id="index">
```

Do not modify the first `<chapter>` tag, as it's usually an introduction and you want that on the first page. For each other `<section>` tag, include the id parameter and name it. A name should include only alphanumeric characters, and it should be short enough to understand what it is.

```
<chapter id="tips">
```

Pick section IDs intelligently

We all know that Cool URIs Don't Change. This means your ids should not change either. Unless of course the content for the id has changed substantially and the id name is no longer relevant.

HTML file name generation using Jade

If you are using Jade to transform your DocBook into HTML you must use the following parameter: *`-V %use-id-as-filename%`*.

D.9.2. Headers and Footers

There is no "easy" way to add headers and footers to your document. If you are using DocBook XSL and doing your own document transformations you may customize the XSL template to suit your needs. For more information read http://www.sagehill.net/docbookxsl/HTMLHeaders.html.

Appendix E. Converting Documents to DocBook XML

A variety of free and commercial tools exist for doing "up conversion" of non-XML formats to DocBook. A few are listed here for your convenience. A more comprehensive list is available from Convert Other Formats to DocBook.

E.1. Text to DocBook

The txt2docbook tool allows one to convert a ascii (README-like) file to a valid docbook xml document. It simplifies the publishing of rather small papers significantly. The input format was inspired by the APT-Convert tool from http://www.xmlmind.com/aptconvert.html.

The script can be downloaded from http://txt2docbook.sourceforge.net/.

E.2. OpenOffice.org to DocBook

As of OpenOffice.org (OOo) 1.1RC there has been support for exporting files to DocBook format.

Although OOo uses the full DocBook document type declaration, it does not actually export the full list of DocBook elements. It uses a "simplified" DocBook tag set which is geared to on-the-fly rendering. (Although it is not the official Simplified DocBook which is described in Section B.5.) The OpenOffice simplified (or "special" docbook) is available from http://www.chez.com/ebellot/ooo2sdbk/.

E.2.1. Open Office 1.0.x

OOo has been tested by LDP volunteers with mostly positive results. Thanks to Charles Curley (charlescurley.com) for the following notes on using OOo version 1.0.x:

Check the version of your OpenOffice

These notes may not apply to the version of OOo you are using.

- To be able to export to DocBook, you must have a Java runtime environment (JRE) installed and registered with OOo--a minimum of version 4.2.x is recommended. The configuration instructions will depend on how you installed your JRE. Visit the OOo web site for help with your setup.

  Contrary to the OOo documentation, the Linux OOo did not come with a JRE. I got one from Sun.
- The exported file has lots of empty lines. My 54 line exported file had 5 lines of actual XML code.
- There was no effort at pretty printing.
- The header is: `<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN" "http://www.oasis-open.org/docbook/xml/4.1.2/docbookx.dtd">`
- The pull-down menu in the File->Save As dialog box for file format indicates that the export format is "DocBook (simplified)." There is no explanation of what that "simplified" indicates. Does OOo export a subset of DocBook? If so, which elements are ignored? Is there any way to enter any of them manually?
- There is NO documentation on the DocBook export filter or whether OOo will import it again.

Conclusions: OOo 1.1RC is worth looking at if you want a word processor for preparing DocBook documents.

However, I hope they cure the lack of documentation. For one thing, it would be nice to know which native OOo styles map to which DocBook elements. It would also be nice to know how to map one's own OOo styles to DocBook elements.

E.2.2. Open Office 1.1

Tabatha Marshall offers the following additional information for OOo 1.1.

> The first problem was when I tried to do everything on version 1.0.1. That was obviously a problem. I have RH8, and it was installed via rpm packages, so I ripped it out and did a full, new install of OpenOffice 1.1. It took a while to find out 1.1 was a requirement for XML to work.
>
> During the install process I believe I was offered the choice to install the XML features. I have a tendency to do full installs of my office programs, so I selected everything.
>
> I can't offer any advice to those trying to update their current OO 1.1. Their "3 ways" aren't documented very well at the site (xml.openoffice.org) and as of this writing, I can't even find THAT on their site anymore. I think more current documentation is needed there to walk people through the process. Most of this was unclear and I had to pretty much experiment to get things working.
>
> Well, after I installed everything I had some configuration to do. I opened the application, and got started by opening a new file, choosing templates, then selecting the DocBook template. A nice menu of Paragraph Styles popped up for me, which are the names for all those tags, I noticed (you can see I don't use WYSIWYG often).
>
> With a blank doc before me (couldn't get to the XML Filter Settings menu unless some type of doc was opened), I went into Tools->XML Filter Settings, and edited the entry for DocBook file. I configured mine as follows:
>
> - Doctype **`-//OASIS//DTD DocBook XML V4.2//EN`**
> - DTD **`http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd`**
> - XSLT for export **`/usr/local/OpenOffice.org1.1.0/share/xslt/docbook/ldp-html.xsl`**
> - XSLT for import **`/usr/local/OpenOffice.org1.1.0/share/xslt/docbook/docbooktosoffheadings.xsl`** (this is the default)
> - Template for import **`/home/tabatha/OpenOffice/user/template/DocBook File/DocBookTemplate.stw`**
>
> At first, if I opened an XML file that had even one parsing error, it would just open the file anyway and display the markup in OO. I have many XML files that use © and other types of entities which show up as parse errors (depending on the encoding) even though they can be processed through. But today I was unable to open any of those files. I got input/output

errors instead. Still investigating that one.

However when you do successfully open a document (one parsing with no errors), it puts it automatically into WYSIWYG based on the markup, and you can then work from the paragraph styles menu like any other such editor.

To validate the document, I used Tools->XML Filter Settings, then clicked the Test XSLTs button. On my screen, I set up the XSLT for export to be `ldp-html.xsl`. If you test and there are errors, a new window pops up with error messages at the bottom, and the lines that need to be changed up at the top. You can change them there and progress through the errors until they're all gone, and keep testing until they're gone.

If you want to open a file to see the source instead of the processed results, go to Tools->XML Filter Settings->Test XSLTs, and then under the Import section, check the Display Source box. My import XSLT is currently `docbooktosoffheadings.xsl` (the default) and the template for import is `DocBookTemplate.stw` (also default).

I think this might work for some people, but unfortunately not for me. I've never used WYSIWYG to edit markup. Emacs with PSGML can tell me what my next tag is no matter where I am, validate by moving through the trouble spots, and I can parse and process from command line.

With OpenOffice, you have to visit http://xml.openoffice.org/filters.html to find conversion tools.

E.3. Microsoft Word to DocBook

Even if you want to use MS Word to write your documents, you may find w2XML useful. Note that this is not free software--the cost is around $130USD. There is, however, a trial version of the software.

E.4. LaTeX to DocBook

Siep Kroonenberg reports that there is a package tex4ht http://www.cse.ohio-state.edu/~gurari/TeX4ht/ that converts TeX and LaTeX to various flavors of XML. Currently, its support for DocBook output is limited. If you want to use tex4ht in its current state for LDP you will probably have to hack your LaTeX source beforehand and the generated XML afterwords.

E.5. LyX to DocBook

This section is contributed by Chris Karakas.

You can use the LyX-to-X package to write your document conveniently in LyX (a visual editor originally developped as a graphical frontend to LaTeX), then export it to DocBook SGML and submit it to The LDP. This method is presented by Chris Karakas *Document processing with LyX and SGML*.

In the LyX-to-X project, LyX is used as a comfortable graphical SGML editor. Once the document is exported to SGML from LyX, it undergoes a series of transformations through sed and awk scripts that correct the SGML code, computes the Index, inserts the Bibliography and the Appendix and takes care of the correct invocation of openjade, pdftex, pdfjadetex and all the other necessary programs for the generation of HTML (chunked or not), PDF (with images, bookmarks, thumbnails and hyperlinks), PS, RTF and TXT versions.

All aspects of document processing are handled, including automatic Index generation, display of Mathematics in TeX quality both online and in print formats, as well as the use of bibliographic databases with RefDB. Special care is taken so that the document processing is as transparent to the user as possible - the aim being that the user writes in LyX, then presses a button, and the LyX-to-X script does the rest. Download the documentation and the LyX-to-X package from the Formats section.

E.6. DocBook to DocBook Transformations

E.6.1. XML and SGML DocBook

There are a few changes between DocBook XML and SGML. Handling these differences should be relatively easy for most small documents, and many authors will not need to make any changes to convert their documents other than the XML and DocBook declarations at the start of their document.

For others, here is a list of what you should keep in mind when converting your documents from SGML to XML.

Differences between XML and SGML elements

An XML element typically has three parts: the start tag, the content (your words) and the end tag. Qualifiers are added in the start tag and are known as attributes. They will always have a name and a quoted value.

```
<filename class="directory">/usr/local<filename>
```

The start tag contains one attribute (class) with a value of "directory". The end tag (also filename) must not contain any attributes.

- Element names (tags) and their attributes are case-dependent--typically lowercase. The following will not validate because the end tag <PARA> is uppercase:
  ```
  <para>This part will fail XML validation</PARA>
  ```
- All attributes in the start tag must be "quoted". This can be either single (') or double (") quotes, but not reverse (`) or "smart quotes". The quote used to start a name="value" pair must be the same quote used at the end of the value. In other words: "this" would validate, but 'that" would not.
- Tags that have a start tag, but no end tag are referred to as "empty" because they do not contain (wrap around) anything. These tags must still be closed with a trailing slash (/). For example: `xref` must be written as <xref linkend="software"/>. You may not have any spaces between the / and >. (Although you may have a space after the final attribute: <xref linkend="foo" />.)
- Processing instructions that get sent to the transformation engine (DSSSL or XSLT) and must have a question mark at the end of the tag. All processing instructions are removed from the output stream. The XML version of this tag would look like this:
  ```
  <?dbhtml filename="foo"?>
  ```
- If you're converting from SGML to XML, be sure file names refer to .xml files instead of .sgml. Some tools may get confused if a .sgml file contains XML.
- Tag minimizations were used in SGML instead of writing out the element name in the end tag. Example: `<para>`This is foo.`</>` Tag minimizations are not supported in XML and their use is discouraged in DocBook.

E.6.2. Changing DTDs

The significant changes between version changes in the DTD involve changes to the elements (tags). Elements may be: deprecated (which means they will be removed in future versions); removed; modified; or added. Almost all authors will run into a changed or deprecated tag when going from a lower version of DocBook to a higher version.

DocBook: The Definitive Guide does an excellent job of showing you how elements fit together. For each element it tells you what an element must contain (its content model) and what is may be contained in (who its parents are). For example: a `note` must contain a `para`. If you try to write <note>Content in a note</note> your document will not validate. Learning how elements are assembled will make it a lot easier to understand any validation errors that are thrown at you. If you get truly stuck you can also email the LDP's docbook mailing list for extra hints. Information on subscribing is available from Section 2.2

All tags that have been deprecated or changed for 4.x are listed in *DocBook: The definitive guide*, published by O'Reilly and Associates. This book is also available on-line from http://www.docbook.org.

E.6.2.1. Differences between version 3.x and 4.x

Here are a few elements that are of particular relevance to LDP authors:

- **artheader.** has been changed to `articleinfo`. Most other header elements have been renamed to info.
- **graphic.** has been deprecated and will be removed as of DocBook 5.x. To prepare for this, start using `mediaobject`. There is more information about `mediaobject` in Section D.5.
- **imagedata.** file formats must now be written in UPPERCASE letters. If you use lowercase or mixed-case spellings for your file formats, it will fail.

  Valid:

  ```
  <imagedata format="EPS" fileref="foo.eps">
  ```
  Invalid:

  ```
  <imagedata format="eps" fileref="foo.eps">
  ```

Glossary

Abiword
: Open Source word processor.

aspell
: Spell check program.

attribute
: An attribute makes available extra information regarding the element on which it appears. The attributes always appear as a name-value pair on the initialization pointers (i.e. the "start tag"). Example of an attribute is `id="identification"`, which gives the attribute `id` the value `identification`.

Cascading Style Sheet (CSS)
: Set of overlay rules that are read by your HTML browser, which uses these rules for doing the display, layout and formatting of the XML-generated HTML file(s). CSS allows for fast changes in look and feel without having to plunge in the HTML file(s).

Catalog
: Helper file for the display and transformation tools, which maps public identifiers and URLs to the local file system.

Concurrent Versions System (CVS)
: A common document management system used by the LDP.

DocBook
: An SGML (and XML) application, describing a document format that allows easy management of documentation.

docbook-utils
: Software package easing XML conversions.

Document Type Definition (DTD)
: A group of statements that define element names and their attributes specifying the rules for combinations and sequences. It's the DTD that defines which elements can or cannot be inserted in the given context.

DSSSL
: DSSSL stands for Document Style Semantics and Specification Language. It's an ISO standard (ISO/IEC 10179:1996). The DSSSL standard is internationally used as a language for documents style sheets pages for SGML.

element
: The elements describe the content's structure in a document. Most elements contain a start tag, content and a closing tag. For example a paragraph element includes all of the following `<para>`This is the paragraph.`</para>`. Some elements are "empty" and do not contain content and a closing tag. An example of this is a link to an external document where the URL is printed to the page. This element would include only the following `<ulink url="http://google.com/>`.

Emacs
: Popular text editor, especially on UNIX systems or alikes.

entity
: An entity is a name designated for some part of data so that it can be referenced by a name. The data could be anything from from simple characters to chapters to sets of statements in a DTD. Entity parameters can be generic, external, internal or SGML data. An entity is similar to a variable in a programming language, or a macro.

epcEdit
: Cross-platform XML editor.

external entity

An external entity points to an external document. External entities are used to include texts on certain locations of a SGML document. It could be used to include sample screens, legal notes, and chapters for example.

float
: Objects such as side bars, pictures, tables, and charts are called floats when they don't have a fixed placement on the text. For printed text, a chart can appear either at the top or at the bottom of the page. It can also be placed on the next page if it is too large.

Frequently Asked Questions (FAQ)
: LDP hosts a number of documents that are a list in the form of questions and answers. These documents are called FAQs. A FAQ is usually a single-page document.

generic entities
: An entity referenced by a name, which starts with "&" and ends with semicolon is a generic entity. Most of the time this type of entity is used in the document and not on the DTD. There are two types of entities: external and internal. They can refer to special characters or to text objects such as repeated sentences, names or chapters.

GNU Free Documentation License (GFDL)
: Like the GNU Public License for free software, but with specifics for written text and documentation with software.

GNU Public License (GPL)
: License type for software that guarantees that the software remains freely distributable, that the source code is available, that you can make changes to it and redistribute those changes if you want, on the condition that you keep on using the same license for your derived works.

Guide
: TLDP documents that are too long to be a HOWTO are usually stored as guides. These are more like entire books that treat a particular subject in-dept.

HOWTO
: Documents that discuss how to do something with a system or application. Most documents hosted at TLDP are HOWTOs, explaining how to install, configure or manage tens of applications on a variety of systems. HOWTOs are typically 10-25 pages.

internal entity
: An internal entity refers to part of the text and is often used as a shortcut for frequently repeated text.

ispell
: Spell check program.

Jade
: An application which applies the rules defined in a DSSSL style sheet to an SGML or XML document, transforming the document into the desired output.

Markup, markup language (ML)
: Code added to the content of a document, describing its structure.

Metadata
: Text in your document that is not important for understanding the subject, but that should be there anyway, such as version information, co-authors, credits to people etc.

nedit
: Text editor oriented to programmers.

nsgmls, onsgmls
: SGML document parser and validator program.

OpenOffice (OOo)
: Open Source office suite, compatible with Microsoft Office.

parameter entity
: An entity type often used in the DTD or a document's internal subset. The entity's name starts with a percent sign (%) and ends with a semicolon.

PSGML

Emacs *major mode* that customizes Emacs for editing SGML documents.

Organization for the Advancement of Structured Information Standards (OASIS)

OASIS is a non-profit, global consortium that drives the development, convergence and adoption of e-business standards.

Outline

Draft of your document that conceptualizes the subject and scope. Summary and To-Do list for the work to come.

Portable Document Format (PDF)

Standard document type supported on a wide range of operating systems.

processing instruction

A processing instruction is a command passed to the document formatting tool. It starts with "<?". This document uses processing instructions for naming files when it is rendered into HTML: `<?dbhtml filename="file.html">`

PostScript (PS)

Document format designed for printable documents. PS is the standard print format on UNIX(-alikes).

Reviewer, review process

TLDP doesn't accept just anything. Once you submit a document, it will be checked for consistency, grammar, spelling and style by a reviewer, a volunteer assigned by the review coordinator.

SGML

Standard Generalized Markup Language. It is an international standard (ISO8879) that specifies rules for the creation of electronic documents in markup systems, regardless of the platform used.

Subject and scope

Obviously, the subject is what your documentation is about. The scope defines which areas of the subject you are going to discuss, and how much detail will be involved.

tag

An SGML element bounded by the marks "<" and ">". Tags are used to mark the semantic or logical structure of a document. A sample is the tag *<title>* to mark the beginning of a title.

TeX

Popular UNIX text formatting and typesetting tool.

Transformation

The process of converting a document from its original DocBook XML form to another format, such as PDF, HTML or PostScript.

Validation

The process of checking your XML code to ensure it complies with the XML DTD you declared at the top of your document.

vi Improved (vIm)

Popular text editor on UNIX and alike systems.

WordPerfect (WP)

Popular word processor, runs on many systems.

XML

eXtensible Markup Language. A sub-product of SGML created specifically for Internet use.

xmllint

Command line XML parser and validator.

XMLmind XML Editor (XXE)

Free but not Open XML editor.

xmlto

Command line XML transformation program.

XSL

XML Style Language. XSL is to a XML document what a DSSSL style is for a SGML document. The XSL is written in XML.

Extensible Stylesheet Transformation (XSLT)

Framework for managing documents, consisting of the XSLT transformation language, the XPath expression language and XSL formatting objects.

Appendix F. GNU Free Documentation License

F.1. 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

F.2. 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

F.3. 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

F.4. 3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

F.5. 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- **A.** Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- **B.** List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- **C.** State on the Title Page the name of the publisher of the Modified Version, as the publisher.
- **D.** Preserve all the copyright notices of the Document.
- **E.** Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- **F.** Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- **G.** Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- **H.** Include an unaltered copy of this License.
- **I.** Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- **J.** Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- **K.** In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- **L.** Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- **M.** Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- **N.** Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version .

F.6. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

F.7. 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and dispbibute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

F.8. 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document , on account of their being thus compiled, if they are not themselves derivative works of the Document. If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

F.9. 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

F.10. 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

F.11. 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

F.12. Addendum

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME.
>
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Notes

[1] Please, take a look at the source to see how to get similar results on your documents. You should also remember that the way this appears to you depends on the format in which you are reading this document: online appearance is slightly different from the PostScript or PDF ones.

[2] In truth, "XSL" is actually comprised of three components: the *XSLT* transformation language, the *XPath* expression language (used by XSLT), and XSL Formatting Objects (FO) that are used for describing a page. The style sheets are actually written in XSLT and generate either HTML or (for print output) FO. The FO file is then run through a FO processor to create the actual print (PDF or PostScript) output. See the W3C web site for more information.

[3] In XSL terminology, the process of generating multiple files is referred to as "chunking".