

LDP Author Guide

Mark F. Komarinski

VA Linux Systems

markk@linuxdoc.org

Jorge Godoy

[Conectiva S.A.](#)

Publishing Department

godoy@conectiva.com

godoy@metalab.unc.edu

David C. Merrill

dcmerrill@mindspring.com

List the tools, procedures, and hints to get LDP authors up to speed and writing.

Table of Contents

<u>Chapter 1. About this Guide.....</u>	<u>1</u>
<u>1.1. Purpose / Scope of this Guide.....</u>	<u>1</u>
<u>1.2. About the LDP.....</u>	<u>1</u>
<u>1.3. Feedback.....</u>	<u>1</u>
<u>1.4. Copyrights and Trademarks.....</u>	<u>1</u>
<u>1.5. Acknowledgments and Thanks.....</u>	<u>2</u>
<u>1.6. Documents.....</u>	<u>2</u>
<u>Chapter 2. Introduction to the LDP and SGML.....</u>	<u>4</u>
<u>2.1. The LDP.....</u>	<u>4</u>
<u>2.2. SGML.....</u>	<u>4</u>
<u>2.3. Why SGML instead of HTML or other formats?.....</u>	<u>4</u>
<u>2.4. SGML is not XML.....</u>	<u>5</u>
<u>2.5. For New Authors.....</u>	<u>5</u>
<u>2.6. Mailing Lists.....</u>	<u>5</u>
<u>Chapter 3. The tools.....</u>	<u>7</u>
<u>3.1. DSSSL.....</u>	<u>7</u>
<u>3.1.1. Norman Walsh DSSSL.....</u>	<u>7</u>
<u>3.1.2. LDP DSSSL.....</u>	<u>7</u>
<u>3.2. DocBook DTD (version 4.1 or 3.1).....</u>	<u>7</u>
<u>3.3. Jade.....</u>	<u>7</u>
<u>3.3.1. Jade.....</u>	<u>8</u>
<u>3.3.1.1. Using Jade.....</u>	<u>8</u>
<u>3.3.1.2. Jade in XML mode.....</u>	<u>8</u>
<u>3.3.2. OpenJade.....</u>	<u>9</u>
<u>3.4. Jade wrappers.....</u>	<u>9</u>
<u>3.4.1. sgmltools-lite.....</u>	<u>9</u>
<u>3.4.2. Cygnus DocBook Tools.....</u>	<u>9</u>
<u>3.4.2.1. Using the Cygnus Tools.....</u>	<u>10</u>
<u>3.5. Editing tools.....</u>	<u>10</u>
<u>3.5.1. Emacs (PSGML).....</u>	<u>10</u>
<u>3.5.1.1. Writing SGML using PSGML.....</u>	<u>11</u>
<u>3.5.2. VIM.....</u>	<u>13</u>
<u>3.5.2.1. Getting Started.....</u>	<u>13</u>
<u>3.5.2.2. Loading or starting new documents.....</u>	<u>14</u>
<u>3.5.3. WordPerfect 9 (Corel Office 2000).....</u>	<u>14</u>
<u>3.5.4. sedit.....</u>	<u>14</u>
<u>3.5.5. nedit.....</u>	<u>15</u>
<u>3.5.5.1. Using nedit.....</u>	<u>16</u>
<u>3.5.5.2. Tips and tricks with nedit.....</u>	<u>16</u>
<u>3.6. CVS.....</u>	<u>18</u>
<u>3.6.1. Getting a CVS account.....</u>	<u>18</u>
<u>3.6.2. Other CVS repository notes.....</u>	<u>19</u>
<u>3.6.2.1. Anonymous CVS access.....</u>	<u>19</u>
<u>3.6.2.2. CVS Files via web.....</u>	<u>19</u>
<u>3.6.2.3. Graphical access to CVS.....</u>	<u>19</u>

Table of Contents

3.6.3. Updating files and CVS	19
3.7. Other/Reference	20
3.7.1. DocBook: The Definitive Guide	20
3.7.2. SGML templates	20
3.7.3. Aspell	20
3.7.4. ispell	20
Chapter 4. Using DocBook Tags	21
4.1. Introduction	21
4.2. Configuration needed	21
4.3. Creating and modifying catalogues	21
4.3.1. Explaining the terminology system	22
4.3.2. Useful command for catalogues	23
4.4. Writing with DocBook elements	24
4.4.1. Useful commands	24
4.5. Encoding Indexes	27
4.6. Inserting Pictures	28
4.6.1. Alternative Methods	29
4.7. Tables	30
4.8. Listings and program codes	31
4.9. Crediting Translators and Converters	32
4.9.1. The <othercredit> Tag	33
4.9.2. The "Acknowledgements" section	33
4.9.3. The <revremark> tag	33
4.10. Tools & Hints	33
4.10.1. Compiling the sources	33
4.10.2. Inserting a summary on the initial articles page	35
4.10.3. Inserting indexes automatically	36
4.10.4. Making notes on the text while it's being written	37
4.10.5. Re-using parts of documents	37
4.11. Document samples	38
4.11.1. Article example	38
4.11.2. Book Example	39
Chapter 5. LDP Style Guide	40
5.1. Deciding on a Subject	40
5.2. Developing an Outline	40
5.3. Writing the Text	41
5.4. Editing and Proofing the Text	42
5.5. Maintaining Your HOWTO	43
5.6. References	43
Chapter 6. Additional Style-related Items	44
6.1. Date formats	44
6.2. Graphics formats	44
6.3. DocBook Versions	44
6.4. Depreciated Tags	44

Table of Contents

6.5. Tag Minimization	44
6.6. Conventions	45
Chapter 7. Tips and Tricks with DocBook	46
7.1. Including Images	46
7.2. Naming separate HTML files	46
7.3. Using ldp.dsl	47
Chapter 8. Distributing your documentation	48
8.1. Before you distribute	48
8.1.1. Validate your SGML code	48
8.2. Copyright and Licensing issues	48
8.3. Submission to LDP	49
8.4. Maintaining Your HOWTO	49
Chapter 9. FAQs about the LDP	50
Glossary	51

Chapter 1. About this Guide

1.1. Purpose / Scope of 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.

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

There are many ways to contribute to the Linux movement without actually writing code. One of the most important is writing documentation, allowing each person to share their knowledge with thousands of others around the world. This Guide is designed to help you get familiar with how the LDP works, and what tools you'll need to write your own HOWTO.

1.2. About the LDP

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.linuxdoc.org/manifesto.html>

The human readable version goes more like this: The LDP consists of a large group of volunteers that are working on documentation for the Linux OS. The most visible documentation are the HOWTOs located at <http://www.linuxdoc.org/>. This Guide focuses primarily on how to write your own HOWTOs for submission to the LDP.

1.3. Feedback

Comments on this Guide may be directed to the author (<markk@linuxdoc.org>).

1.4. Copyrights and Trademarks

Copyright 1999–2000 Mark F. Komarinski, David C. Merrill, Jorge Godoy

This manual may be reproduced in whole or in part, without fee, subject to the following restrictions:

- The copyright notice above and this permission notice must be preserved complete on all complete or partial copies

- Any translation or derived work must be approved by the author in writing before distribution.
 - If you distribute this work in part, instructions for obtaining the complete version of this manual must be included, and a means for obtaining a complete version provided.
 - Small portions may be reproduced as illustrations for reviews or quotes in other works without this permission notice if proper citation is given. Exceptions to these rules may be granted for academic purposes: Write to the author and ask. These restrictions are here to protect us as authors, not to restrict you as learners and educators. Any source code (aside from the SGML this document was written in) in this document is placed under the GNU General Public License, available via anonymous FTP from the GNU archive.
-

1.5. Acknowledgments and Thanks

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 ldp-discuss@lists.linuxdoc.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 Serek (ser@serek.arch.pwr.wroc.pl). Sections on DocBook were written by Jorge Godoy (godoy@conectiva.com). A great deal of thanks to both of them for their help.

1.6. Documents

This document uses the following conventions^[1]:

Descriptions Appearance

Warnings

Caution
Warnings.

Hint

Tip:
Hint.

Notes

Note: Note.

Information requiring special attention

Warning
Warning.

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

shellbash#*Prompt* of user command under tcsh shelltcsh\$Environment VariablesVARIABLEEmphasized
wordwordCode Example

<para>Beginning and end of paragraph</para>

Chapter 2. Introduction to the LDP and SGML

2.1. The LDP

The Linux Documentation Project (LDP) was started to provide new users a way of getting information quickly about a particular subject. It not only contains a series of books on administration, networking, and programming, but 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. At first, many of these works were in text or HTML. As time went on, there had to be a better way of managing these documents. One that would let you read it from a web page, a text file on a CD-ROM, or even your hand-held PDA. The answer, as it turns out, is SGML.

2.2. SGML

The Standard Generalized Markup Language (SGML) is a language that is based on embedding codes within a document. In this way, it is similar to HTML, but there is where any similarities end. The power of SGML is that unlike WYSIWYG (What You See Is What You Get), you don't define things like colors, or font sizes, or even some kinds of formatting. Instead, you define elements (paragraph, section, numbered list) and let the SGML processor and the end program worry about placement, colors, fonts, and so on. HTML does the same thing, and is actually a subset of SGML. SGML has really three parts that make it up. First is the Structure, which is what is commonly called the DTD, or Document Type Definition. The DTD defines the relationship between each of the elements (or tags). The DocBook DTD, used to create this document, is an example of this. The DTD lists the rules that the content must follow. Second is the DSSSL or Document Style Semantics and Specification Language. The DSSSL tells the program doing the rendering how to convert the SGML into something that a human can read. It tells the renderer to convert a <title> tag into 14 point bold if it is going to RTF format, or to turn it into a <h1> tag if you're going to HTML. Finally there is the Content, which is what gets rendered by the SGML processor and is eventually seen by the user. This paragraph is content, but so would a graphic image, table, numbered list, and so on. Content is surrounded by tags to separate out each element.

2.3. Why SGML instead of HTML or other formats?

SGML provides for more than just formatting. You can automatically build indexes, table of contents, and links within the document or to outside. The Jade and OpenJade packages also let you export (I'll call it render from here on) SGML to LaTeX, info, text, HTML, and RTF. From these basic formats, you can then create other formats such as MS Word, PostScript, PDF and so on. Programs like LyX allow you to write in TeX format, then export it as SGML and render from SGML to whatever you chose. In the end, SGML is more concerned about the way elements work instead of the way they look. A big distinction, and one that will let you write faster, since you don't have to worry about placement of paragraphs, font sizes, font types, and so on.

2.4. SGML is not XML

There has been a lot of press recently about XML, and DocBook support for XML. DocBook is a collection of markup tags that follow a specified hierarchy. XML DocBook v4.1.2 is now officially supported by the LDP. If you're familiar with SGML and want to convert to XML, please keep the following in mind (list borrowed from DocBook: The Definitive Guide):

- All XML markup is case sensitive. All element, attribute, and entity names have to be in lowercase. SGML is not case sensitive.
 - All attributes have to be quotes using either straight (') or double (") quotes.
 - Empty elements (like xref) have to end with a /: <xref/>.
 - Tag minimizations (</>) are not supported. The LDP discourages their use in SGML as well.
-

2.5. For New Authors

If you are a new to the LDP and want to pick up an unmaintained HOWTO or write a new HOWTO document, contact the HOWTO coordinator at <ldp-discuss@lists.linuxdoc.org>. This is to make sure the HOWTO coordinator can know who is working on what documentation.

Once that part is complete, you may write your documentation in the format of your choice and submit a draft to <ldp-submit@lists.linuxdoc.org> and the draft will be reviewed by an LDP volunteer. In a few short days you will get the draft and comments from the volunteer. After applying the comments, you may send this version to the ldp-submit list again for final submission into the LDP.

At this point, another LDP volunteer will translate your document into DocBook and send you the finished DocBook document. From here on, all submissions to the LDP has to be in DocBook format. If you have markup questions, you may ask the volunteer who assisted you, or ask the LDP DocBook list.

If you choose to start your document off in DocBook, there are plenty of templates to get you started:

- <http://www.linuxdoc.org/authors/template-ld/big-howto-template-ld.sgml> – This template is written by Stein Gojen and is based off the LinuxDoc template.
 - <http://www.linuxdoc.org/authors/template/big-howto-template.sgml> – This template is based on Stein's work, but is much larger and complicated than the above. It uses more features of DocBook.
-

2.6. Mailing Lists

There are a few mailing lists to subscribe to so you can take part in how the LDP works. First is <ldp-discuss@lists.linuxdoc.org>, which is the main discussion group of the LDP. To subscribe, send a message with the subject reading "subscribe" to <ldp-discuss-request@lists.linuxdoc.org>. To unsubscribe, send an e-mail with the subject of "unsubscribe" to <ldp-discuss-request@lists.linuxdoc.org>.

Another list is the <ldp-docbook@lists.linuxdoc.org> list, which is for markup or other questions about DocBook itself. If you run into trouble with a particular markup tag, you can send your question here for answers. You can subscribe to the DocBook list by sending a "subscribe" message to <ldp-docbook-request@lists.linuxdoc.org>.

Chapter 3. The tools

In this section, we will cover some of the tools that you'll need or want to use to create your own LDP documentation. I'll describe them here, and better define them later on, along with how to install them. If you use some other tool to assist in writing LDP, please let me know and I'll add a blurb here for it.

3.1. DSSSL

The Normal Walsh version is required, the LDP is optional.

3.1.1. Norman Walsh DSSSL

<http://nwalsh.com/docbook/dsssl/>

The Document Style Semantics and Specification Language tells jade how to render a SGML document into print or online form. The DSSSL is what converts a title tag into an <H1> tag in HTML, or bold, 14 point 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).

3.1.2. LDP DSSSL

<http://www.linuxdoc.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.

3.2. DocBook DTD (version 4.1 or 3.1)

Required – <http://www.oasis-open.org/docbook/sgml/4.1/docbk41.zip> or <http://www.oasis-open.org/docbook/sgml/3.1/docbk31.zip>

The XML DTD is available from <http://www.oasis-open.org/xml/4.1.2/>.

The DocBook DTD defines the tags and structure of a DocBook SGML document. Modifying the DTD, such as adding a new tag, doesn't make it DocBook anymore.

3.3. Jade

Jade and OpenJade are two of the programs that do most of the rendering and validation of code based off the DTD and DSSSL. One of the following is required and should be installed after the DTD and DSSSL have been installed.

3.3.1. Jade

<ftp://ftp.jclark.com/pub/jade/jade-1.2.1.tar.gz>

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

3.3.1.1. Using Jade

This is the quick and dirty way that should work for all distributions, no matter what distribution you're using.

1. Create a base directory to store everything such as `/usr/local/sgml/`. We'll call this `$_toolroot` from here on.
2. Install Jade, DocBook DTD, and DSSSL such that the base of each is under `$_toolroot`, creating:

- ◆ `$_toolroot/jade-1.2.1`
- ◆ `$_toolroot/dtd`
- ◆ `$_toolroot/dsssl`

3. You'll need to set the `SGML_CATALOG_FILES` environment variable to the catalogs that you have under `$_toolroot`. You can do this with the command:

```
bash$ export SGML_CATALOG_FILES=$_toolroot/dtd/docbook.cat:\
$_toolroot/dsssl/docbook/catalog:$_toolroot/jade-1.2.1/dsssl/catalog
```

4. Now you can start using Jade. To create individual HTML files:

```
$_toolroot/jade-1.2.1/jade/jade -t sgml -i html \
-d $_toolroot/dsssl/docbook/html/docbook.dsl howto.sgml
```

5. To create one large HTML file, add `-V nochunks` to the jade command.
-

3.3.1.2. Jade in XML mode

Once configured for XML, jade and openjade will work the same way as for SGML DocBook.

After extracting the XML DTD, you may want to make a symlink from the `docbook.cat` file to "catalog", the default filename for jade/openjade catalogs. Replace `$_xml_root` with the location of your XML DTD.

```
bash$ cd $_xml_root
bash$ ln -s docbook.cat catalog
bash$ export SGML_CATALOG_FILES=$_xml_root/catalog:$_toolroot/dsssl/catalog:\
$_toolroot/dtd/docbook/catalog
bash$ jade -t sgml -i html -d style $_jade_path/pubtext/xml.dcl foo.xml
```

You'll need the catalogs for XML, the DSSSL, and DocBook, respectively. `$_toolroot` was defined above.

Replace style with the style you wish to use. The pointer to `xml.dcl` is required for jade to work, and it

has to be listed immediately before the pointer to your XML document.

You may get the following warnings when processing XML documents. They don't impact the output, and the cause is being looked into.

```
<xml_dtd_pth>/ent/iso-lat2.ent:119:18:E: "X0176" is not a function name
<xml_dtd_pth>/ent/iso-lat2.ent:120:17:E: "X0178" is not a function name
```

If you want to convert your existing SGML DocBook into XML docbook, use this as your declaration (the lines at the very start of your document).

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN">
```

If you have followed LDP guidelines, there should be no other changes required to your document.

3.3.2. 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.

3.4. Jade wrappers

These tools are optional and may be installed after Jade, the DSSSL, and DTD have been installed.

3.4.1. sgmltools-lite

<http://sgmltools-lite.sourceforge.net/>

This is the successor to the sgmltools project, which has officially been disbanded for over a year. Since then, Cees de Groot has created a slightly different project, which acts as a wrapper to the jade SGML processor. It hides much of the ugliness of syntax. This author was able to install the old sgmltools package followed by the sgmltools-lite and could format this document quite easily. There's even a man page for sgmltools showing syntax.

3.4.2. Cygnus DocBook Tools

May be Red Hat specific – <http://www.redhat.com/>

Red Hat distributes three packages, starting with the 6.2 release, that include DocBook support and some tools. The tools are easily installed, allowing you to focus more on writing than wrestling with the tools.

TeX, Jade, and JadeTeX must be installed first. All three of these packages are available on the installation CD.

3.4.2.1. Using the Cygnus Tools

These tools are provided with Red Hat 6.2. Make sure the following packages are installed:

- sgml-common-0.1-8.noarch
- docbook-3.1-4.noarch
- stylesheets-1.54.13rh-1.noarch

Red Hat has the latest version on their web site:

<http://www.redhat.com/support/errata/RHBA-2000022-01.html>.

Download/get/sneaker-net the RPMs to your machine and install in the usual manner (become root, then **rpm -Uvh filename**). Once the RPMs are installed, you can use the following commands to render DocBook:

```
bash$ db2html filename
```

Renders DocBook into HTML. A subdirectory with the filename (minus the .sgml extension) is created and the HTML files are placed there.

```
bash$ db2pdf filename
```

Renders DocBook into a PDF file. Note that there is currently a problem with db2pdf, and pd2ps caused by JadeTeX. This has been [registered as a bug with RedHat](#).

3.5. Editing tools

The following tools may be used to create, edit, or validate your HOWTO.

3.5.1. Emacs (PSGML)

Optional – 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 SGML 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, and many believe it to allow more versatility than any other SGML documentation tool. It works with DocBook, LinuxDoc and other DTDs equally well.

3.5.1.1. Writing SGML using PSGML

3.5.1.1.1. Introduction

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**).

From here on, we assume you have PSGML installed for use with a recent version of GNU Emacs. If that all went by too fast for you, see the free chapter from Bob Ducharme's SGML CD book:

<http://www.snee.com/bob/sgmlfree/>.

3.5.1.1.2. Updating your .emacs to use PSGML

If you want GNU Emacs to enter PSGML mode when you open a *.sgml* file and be ready for SGML editing, make sure PSGML can find the DocBook DTD. If your distribution already had PSGML set up for use with GNU Emacs, you probably do not have to do anything to get this to work. Otherwise, you may need to set an environment variable that tells PSGML where to look for the SGML catalog (the list of DTDs).

For example:

```
bash$ export SGML_CATALOG_FILES=/usr/lib/sgml/catalog
```

Then add something like the following to your *.emacs* file:

```
;; *****
;; set up psgml mode...
;; use psgml-mode instead of emacs native sgml-mode
;;

(autoload 'sgml-mode "psgml" "Major mode to edit SGML files." t)
(setq auto-mode-alist
  (append
    (list
      ('("\\.sgm$" . sgml-mode)
      ('("\\.sgml$" . sgml-mode)
    )
    auto-mode-alist))

;; set some psgml variables

(setq sgml-auto-activate-dtd t)
(setq sgml-omittag-transparent t)
(setq sgml-balanced-tag-edit t)
(setq sgml-auto-insert-required-elements t)
(setq sgml-live-element-indicator t)
(setq sgml-indent-step nil)

;; create faces to assign to markup categories

(make-face 'sgml-comment-face)
(make-face 'sgml-start-tag-face)
(make-face 'sgml-end-tag-face)
(make-face 'sgml-entity-face)
(make-face 'sgml-doctype-face) ; DOCTYPE data
(make-face 'sgml-ignored-face) ; data ignored by PSGML
```

```

(make-face 'sgml-ms-start-face) ; marked sections start
(make-face 'sgml-ms-end-face) ; end of marked section
(make-face 'sgml-pi-face) ; processing instructions
(make-face 'sgml-sgml-face) ; the SGML declaration
(make-face 'sgml-shortref-face) ; short references

;; view a list of available colors with the emacs-lisp command:
;;
;; list-colors-display
;;
;; please assign your own groovy colors, because these are pretty bad
(set-face-foreground 'sgml-comment-face "coral")
;(set-face-background 'sgml-comment-face "cornflowerblue")
(set-face-foreground 'sgml-start-tag-face "slateblue")
;(set-face-background 'sgml-start-tag-face "cornflowerblue")
(set-face-foreground 'sgml-end-tag-face "slateblue")
;(set-face-background 'sgml-end-tag-face "cornflowerblue")
(set-face-foreground 'sgml-entity-face "lavender")
;(set-face-background 'sgml-entity-face "cornflowerblue")
(set-face-foreground 'sgml-doctype-face "lavender")
;(set-face-background 'sgml-doctype-face "cornflowerblue")
(set-face-foreground 'sgml-ignored-face "cornflowerblue")
;(set-face-background 'sgml-ignored-face "cornflowerblue")
(set-face-foreground 'sgml-ms-start-face "coral")
;(set-face-background 'sgml-ms-start-face "cornflowerblue")
(set-face-foreground 'sgml-ms-end-face "coral")
;(set-face-background 'sgml-ms-end-face "cornflowerblue")
(set-face-foreground 'sgml-pi-face "coral")
;(set-face-background 'sgml-pi-face "cornflowerblue")
(set-face-foreground 'sgml-sgml-face "coral")
;(set-face-background 'sgml-sgml-face "cornflowerblue")
(set-face-foreground 'sgml-shortref-face "coral")
;(set-face-background 'sgml-shortref-face "cornflowerblue")

;; assign faces to markup categories

(setq sgml-markup-faces '
  (
    (comment . sgml-comment-face)
    (start-tag . sgml-start-tag-face)
    (end-tag . sgml-end-tag-face)
    (entity . sgml-entity-face)
    (doctype . sgml-doctype-face)
    (ignored . sgml-ignored-face)
    (ms-start . sgml-ms-start-face)
    (ms-end . sgml-ms-end-face)
    (pi . sgml-pi-face)
    (sgml . sgml-sgml-face)
    (shortref . sgml-shortref-face)
  ))

;; tell PSGML to pay attention to face settings
(setq sgml-set-face t)
;; ...done setting up psgml-mode.
;; *****

```

Then restart Emacs

3.5.1.1.3. SGML Smoke Test

Try the following smoke test. Start a new file, `/tmp/test.sgml` for example, and enter the following:

```
<!DOCTYPE test [
<!ELEMENT test - - (#PCDATA)>
]>
```

Enter **C-cC-p**. If Emacs manages to parse your DTD, you will see *Parsing prolog...done* in the minibuffer. Try **C-c C-e RETURN** to insert a `<test>` element. If things are working correctly, you should see the following in Emacs:

```
<!DOCTYPE test [
<!ELEMENT test - - (#PCDATA)>
]>
<test></test>
```

3.5.1.1.4. Writing a New HOWTO in DocBook

Start a new file for your HOWTO and enter the following:

```
<!DOCTYPE ARTICLE PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
```

Enter **C-cC-p** and hold your breath. If everything goes as planned, you will see Emacs chewing for a few seconds and then *Parsing prolog...done* in the minibuffer.

At this point, enter **C-cC-eRETURN** to insert an `<article>` element and proceed to write your HOWTO.

3.5.1.1.5. Quick Reference for Emacs with PSGML

See Nik Clayton's primer for FreeBSD documentation:

<http://www.freebsd.org/tutorials/docproj-primer/psgml-mode.html>

3.5.2. VIM

<http://www.vim.org>

No mention of Emacs is complete without talking about vi. The VIM (Vi IMproved) editor has the functionality of regular vi, but also has an SGML mode that will color-coordinate your screen to show where tags are.

3.5.2.1. Getting Started

The vim program comes really in multiple parts. There is first the plain vim program, which has compatibility with the vi program and its commands. Red Hat users will want the vim-common and vim-minimal packages. Next is the enhanced **vim**, which includes the highlighting and other features of vim over regular vi. Red Hat users will want vim-enhanced. Last, but certainly not least, is the X interface, which

gives a graphical interface, menus, and mouse control. To separate this from vim or vi, the command for graphical access is called **gvim**.

3.5.2.2. Loading or starting new documents

In both **vim** and **gvim** modes, `.sgml` files will be automatically recognized and enter into "sgml mode". A series of known DocBook tags have been entered into **vim** and will be highlighted in brown if a tag is known. If it isn't, it will appear in light blue. attributes to known tags are in light blue, and values for the attributes are in pink. From here on, you can use regular **vi** commands to navigate and edit.

3.5.3. 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.

This is the least expensive of the commercial applications that support SGML.

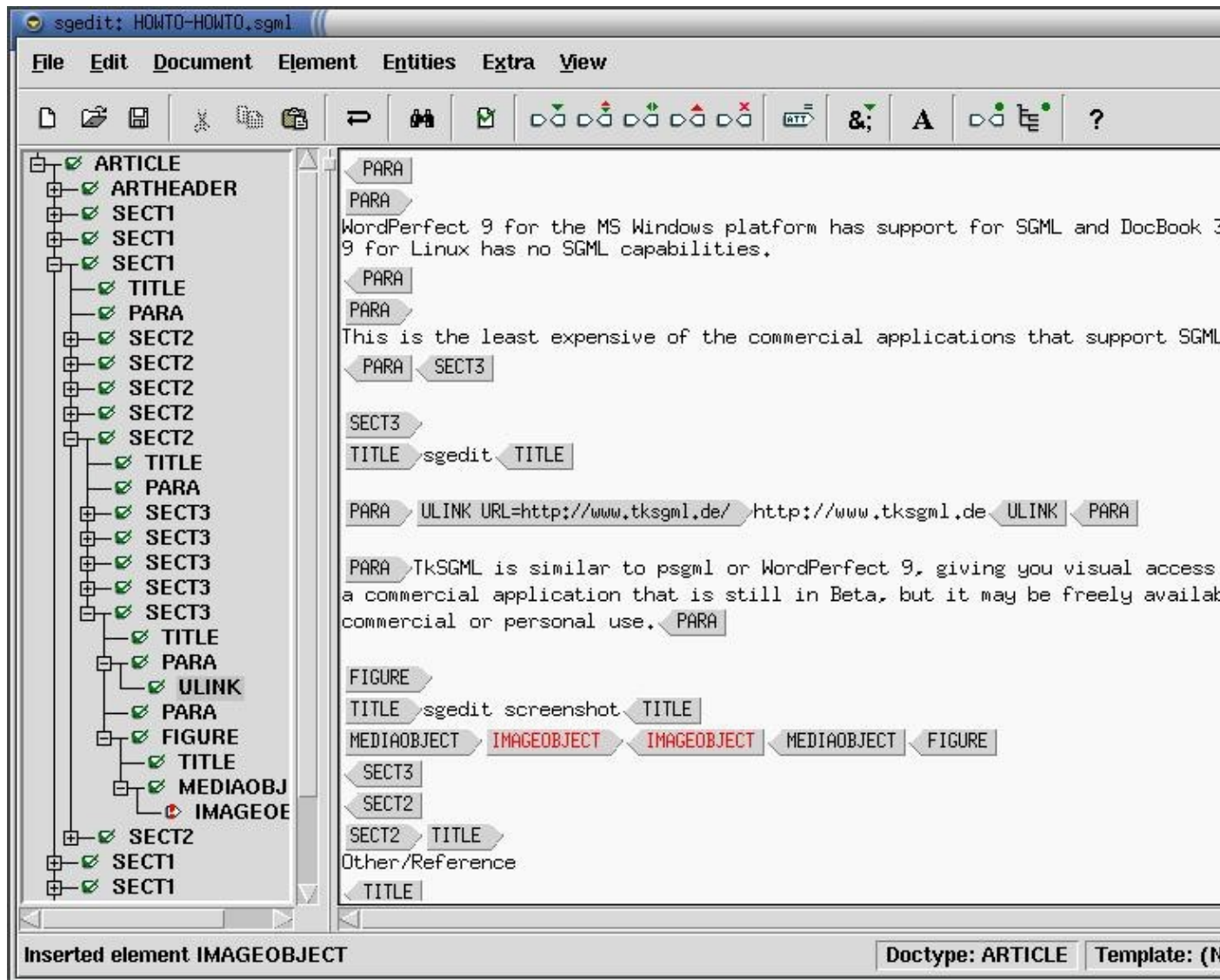
3.5.4. sgedit

<http://www.tksgml.de/>

The sgedit program allows you to visually edit SGML 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. It's a commercial application, but pricing has not been set. There will be free licenses for private and academic use.

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

Figure 3-1. sgedit screen shot

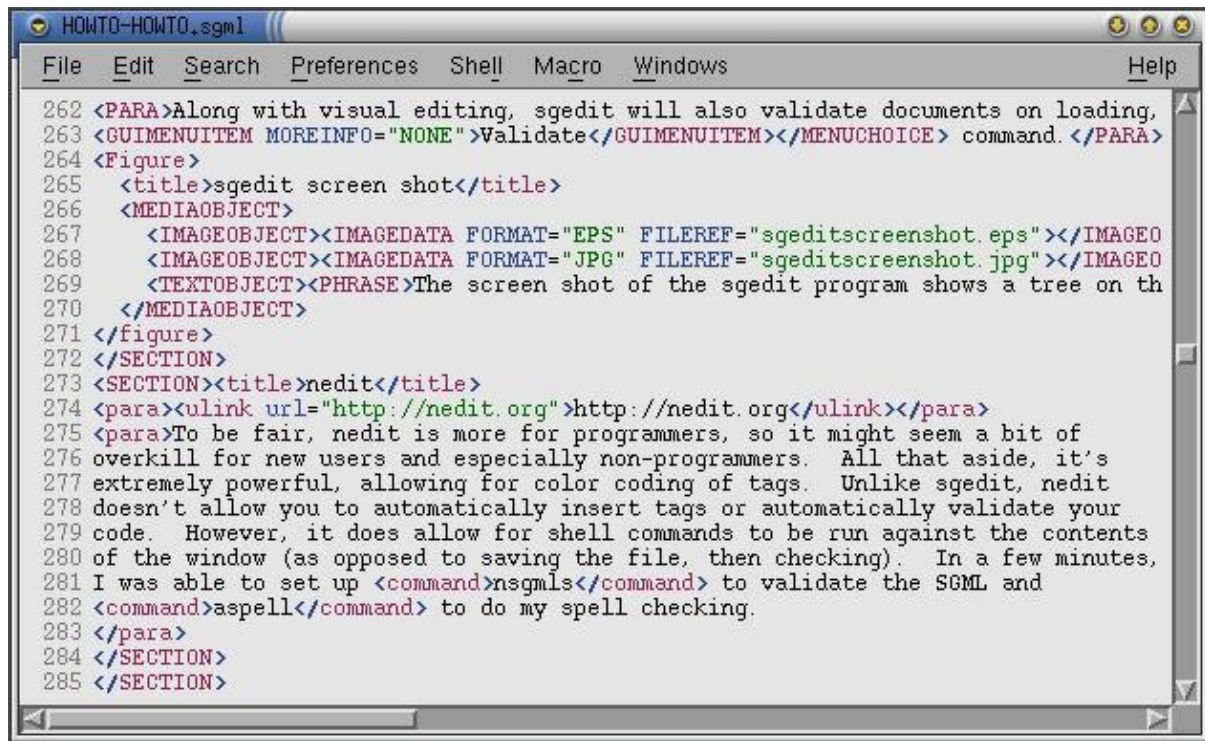


3.5.5. nedit

<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 color coding of tags. Unlike sedit, 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). In a few minutes, I was able to set up **nsgmls** to validate the SGML and **aspell** to do my spell checking.

Figure 3–2. nedit screen shot



3.5.5.1. Using nedit

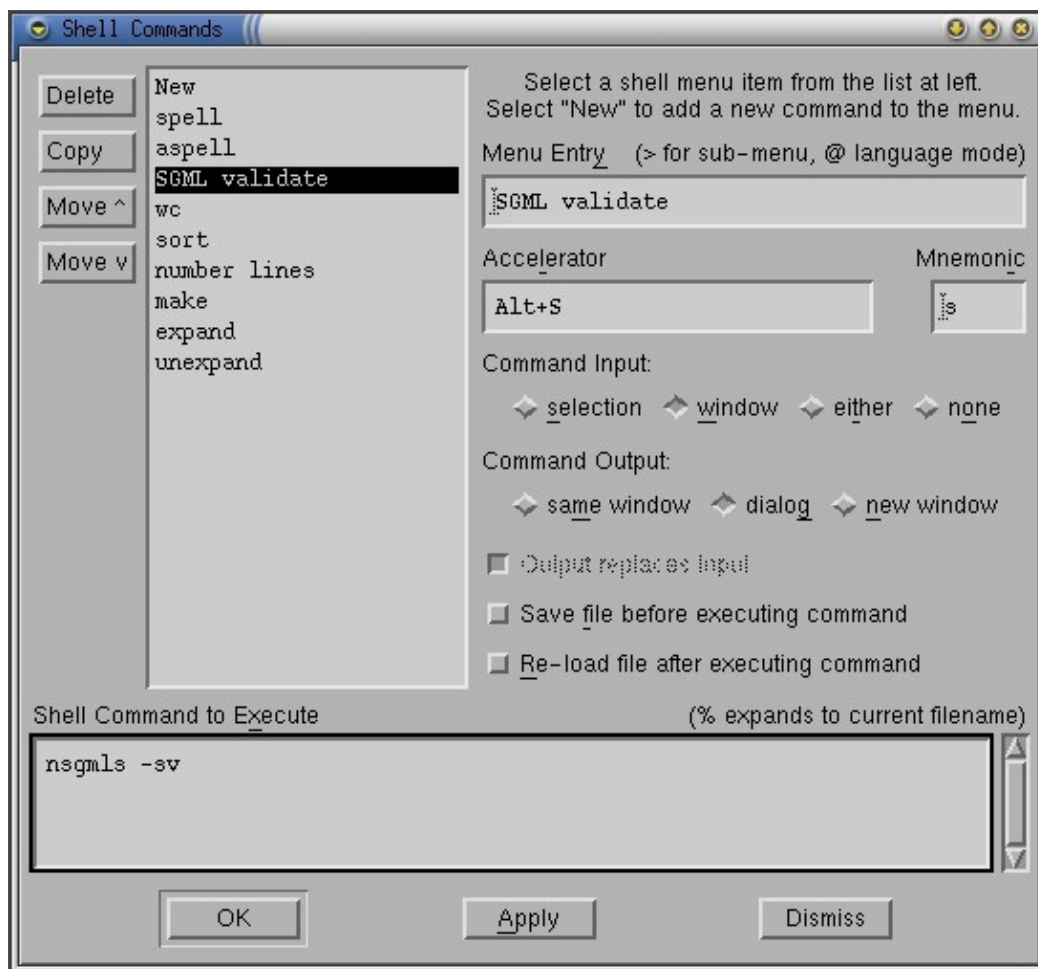
For writing new documentation, it is recommended that you download and use the LDP DocBook template. Once you have the file, you can start up nedit and start editing. If the file is saved with a .sgml extension, nedit will load the file up with SGML/HTML tags enabled. You can turn this on explicitly using the Preferences→Language Mode→SGML HTML command.

3.5.5.2. Tips and tricks with 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 the SGML code using **nsqmls**.

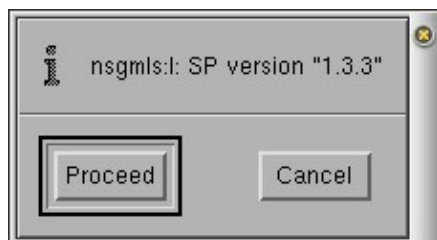
Select Preferences→Default Settings→Customize Menus→Shell Menu.... This will bring up the Shell Command dialog box, with all the shell commands nedit has listed under the Shell menu. Under Menu Entry, enter "SGML Validate". This will be the entry you'll see on the screen. Under Accelerator, press **Alt-S**.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

Figure 3–3. Adding shell commands to nedit



Click OK and you'll now be back at the main nedit screen. Load up an SGML file, and select Shell→SGML Validate or press **Alt-S**. The **nedit** program will fire up and check the contents of the window. The reason for using `-sv` is that the `-v` tells **nsgmls** to output the version of the program, so you'll always get output and know that **nsgmls** has run. If all you get is a version number, then there are no errors with the document. If there are errors, then they'll be listed in the separate window for you to see. If you have line numbers turned on (using Preferences→Show Line Numbers) then finding the errors is much simpler, as **nsgmls** will list errors by their line number.

Figure 3–4. nsgmls output on success



3.6. CVS

The LDP is in the process of providing CVS access to authors. There are a few good reasons for this:

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. It's great if you have many people working on the same document. You can have CVS tell you what changes were made while you were editing your copy by another author, and integrate those changes in.
3. Keeps a log of what changes were made. These logs (and a date stamp) can be placed automatically inside the document when you use some special tags that get processed before the SGML processor.
4. Can provide for a way for a program to automatically update the LDP web site with new documentation as it's written and submitted. This is not in place yet, but is a potential 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 SGML code.

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://www.sourceforge.com/ CVS/Docs/blandy>
- <https://wroclaw.art.pl/~ser/docs/cvs.html>

3.6.1. Getting a CVS account

First you'll need to get an account at the LDP's CVS Repository. This is pretty much the root directory that is used by CVS, with various projects (HOWTOs, mini HOWTOs, etc.) created as subdirectories of that.

You will need to create a hashed password and userid for your account. The hashed password allows you to send an encrypted password to the CVS group without them needing to know your password. You can do this with the following command, from bash (or sh):

```
bash$ echo your_password | perl -e "print crypt(<>,\njoin ' ', ('.', '/', 0..9, 'A'..'Z', 'a'..'z')[rand 64, rand 64]), \"\\n\\n\""
```

Take the output of this command, and send it with your proposed userid to cvsgadmin@cvsglist.linuxdoc.org. Your unique CVSROOT directory will be created and you'll get an e-mail with a response. When you get your response, log into your CVSROOT and make sure everything is set up properly:

```
bash$ export CVSROOT=:pserver:your_userid@cvsg.linuxdoc.org:/cvsgroot
bash$ cvs -d $CVSROOT login
```

(Replace the *your_userid* with what you were sent in the response e-mail).

You will be asked for your password, and then given access to the CVS Repository in read-write mode. Once you've used **cvs login** once 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 and continue on. You can get the entire LinuxDoc repository with this command:


```
bash$ cvs get LDP
```

Or you can get the SGML source for your own document with these commands:

```
bash$ cvs get LDP/howto/docbook/YOUR-HOWTO.sgml
bash$ cvs get
guide/docbook/YOURGUIDE
```

3.6.2. Other CVS repository notes

3.6.2.1. Anonymous CVS access

Anonymous CVS access is available for those who do not require an account (such as those wishing to publish LDP documents). This repository is read-only:

```
bash$ cvs -d :pserver:cvs@anoncvs.linuxdoc.org:/cvsroot login
```

As a password, use cvs. You can then get LinuxDoc modules as above. Note that changes to the anoncvs site may be a half an hour behind the main site.

3.6.2.2. CVS Files via web

You can access the CVS repository via the web at <http://cvsweb.linuxdoc.org/index.cgi/LDP>.

3.6.2.3. Graphical access to CVS

There are graphical interfaces to CVS, and you can get a list of them at <http://freshmeat.net/appindex>. Search for CVS.

3.6.3. Updating files and CVS

CVS has a special tag, *\$Id\$*, that you can use to automatically insert the date and version directly into the document. After committing, CVS will turn this tag into *\$Id: cvs.sgml,v 1.3 2000/08/24 20:31:01 markk Exp \$*. By including this tag in your document, you can have that automatically change each time you change the file, allowing the revision mark to increment each time.

When you're ready to upload changes to the CVS server, use the command **cvs ci -m "comment" YOUR-HOWTO.sgml**. The **-m "comment"** isn't necessary, but if you don't include it, you'll be brought into the editor (usually vi, or whatever your EDITOR environment variable is) and be given the chance to add a comment about the changes.

You can follow more of the CVS discussion on the `ldp-discuss` list.

If you are using the LDP CVS tree while developing your document, the LDP will need to be notified when your document is ready to be published. E-mail should be sent to

<ldp-submit@lists.linuxdoc.org>. Indicate the title of your document and the relative path to the file(s) in the LDP CVS tree within your message.

3.7. Other/Reference

The items in this section are reference books or other utilities that can't quite be categorized (yet).

3.7.1. DocBook: The Definitive Guide

<http://www.docbook.org/>

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, and much of the emphasis is on XML, but the DocBook tags for version 3.1 are all listed in a handy format. You can pick it up at the book vendor of choice. The entire book is also available online (in HTML and SGML formats) at the above URL.

3.7.2. SGML templates

Optional – <http://www.linuxdoc.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.

3.7.3. Aspell

Optional – <http://aspell.sourceforge.net/>

This spell checking application can work around SGML tags, and only spell check the content within the tags. Default spell checkers like ispell will try to spell check the tags, causing errors at every new tag.

3.7.4. ispell

Optional – <http://www.cs.hmc.edu/~geoff/ispell.html>

The ispell program is distributed with RedHat (and possibly other distros) and also ignores markup tags.

Chapter 4. Using DocBook Tags

4.1. Introduction

DocBook defines a set of markup elements useful for marking up text so that the text can then be transformed into several different formats. TEST

It's possible to create documents in different formats HTML, XML, RTF, TeX, and others.

The idea is to write just once and reach the largest possible number of people with the information.

Digital information not stored properly tends to get lost. Due to the fact that not containing uncommon characters (such as binary formats) it's possible to index and search directly on the documents written on SGML and consequently on DocBook.

The SGML systems use markups to make their description. DocBook holds over 300 markup elements each one with several attributes which can assume several values, these can be fixed or defined by the document / style that the author has used.

Just to remind if any changes are made on the DocBook's definitions DTD, it's no longer DocBook.

4.2. Configuration needed

The identifier systems used by the SGML and by some tools are based on catalogues which perform the translation of these identifiers over to files holding the necessary definitions.

The section on tailoring a catalogue (see [Section 4.3](#)) will give more details about these files.

For such tools to be able to find the necessary catalogue(s) the value of the environment variable SGML_CATALOG_FILES should be configured, as shown in the following example:

```
$ export SGML_CATALOG_FILES="/usr/lib/sgml/catalog"
```

This is the only necessary additional configuration for the DocBook, tools and the like to work correctly on your platform.

4.3. Creating and modifying catalogues

A catalogue is a text file containing the translation rules of the public identifier to system's files.

They make easy to use the DocBook, for they allow each user to have their files installed in a different place (e.g. your home directory, /usr/local/sgml or in any other place) though no other change on the document is necessary to be processed and "compiled".

Example 4–1. Example of catalogue

```
-- Catalogue for the Conectiva Styles --

OVERRIDE YES

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

DELEGATE "-//OASIS"
         "/home/ldp/SGML/dtds/catalog.dtd"

DOCTYPE BOOK /home/ldp/SGML/dtds/docbook/db31/docbook.dtd

-- EOF --
```

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

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

Comment informing 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

4.3.1. Explaining the terminology system

Notice the identifier

```
"-//Conectiva SA//DTD books V1.0//EN"
```

Its formation is not random and follows some pre-defined conditions.

The token "--" indicates that the used identifier isn't a registered type. Only a few identifiers are registered and those usually belong to entities like ISO, IEEE, and others.

The second part of the identifier defines the name of the organization that created it. On the example above, Conectiva S.A.

The one before the last defines the contents (in this case a DTD^[2]) and the name of the identified text.

The last element indicates the language in which the document was written. Since DocBook is a DTD written in English, the language is EN. The two letter code recommended is the ISO identification of the language.

More information can be obtained at [OASIS Technical Resolution 9401:1997 \(Amendment 2 to TR 9401\)](#).

4.3.2. Useful command for catalogues

The most common commands to be used on catalogues are:

PUBLIC

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

SYSTEM

The keyword `SYSTEM` 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 charge free tools available. The benefits of this statement can be achieved somehow with multiple catalogue files.

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

CATALOG

The keyword `CATALOG` allows a catalogue to be included inside another. This is a way to make use of several different catalogues 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 catalogue to a specific type of public identifier. The clause `DELEGATE` is very similar to the `CATALOG`, except by the fact that it doesn't do anything until a specific pattern is specified.

DOCTYPE

In case of a document starts with a type of document, but has no public identifier and no system identifier the clause DOCTYPE makes the association of this document with an specific DTD.

4.4. Writing with DocBook elements

An editor capable of inserting an element according with DTD analisys helps a lot since it can allow or not the element to be used at the position where the cursor is in. This way you can be sure that no invalid element was added anywhere in your document.

In order to ensure future changes are as easy as possible, authors should try to keep as much compatibility as possible with theXML version of the DocBook DTD. This means keeping element names in upper case, using double quotes in all attributes, not using "markup minimizations" (explained below), and not omitting end tags. Most tools that automatically insert elements (like psgml+emacs) follow these rules automatically or with some fine tuning.

There are several forms of markup minimization. These include empty tags. One example of tag minimization is that instead of typing the end tag you simply type `</>`. Another example, as said before, is ommiting tags. You can see both examples below:

```
<para>I'm using <emphasis>here</>, normal text here,
and <>here</> I emphasized the text again, with empty tags.</para>
```

Each type of document created has a specific structure and example of documents can found afterwards on this document. (see [Section 4.11](#)).

Considering the explanation above we can proceed to instructions on how to write a document using DocBook.

4.4.1. Useful commands

The [Table 4–1](#) shows some commands which are useful to generate generic documents. Remember that some elements are valide only on some contexts.

Tip: Sometimes the appearance of a particular tag changes from one 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.

Note: 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 4–1. Useful commands

Description	Command	Result
-------------	---------	--------

E-mail address	<code><email>address@domain</email></code>	<code><address@domain></code>
About the author	<code><author>...</author></code>	(see example below)
Author's name	<code><firstname>First_Name</firstname> <othername>Middle_Name</othername> <surname>Surname</surname></code>	First Name Middle Name Surname
Keys' name (printings on the keyboard)	<code><keycap>F1</keycap></code>	F1
Symbol represented by the keys	<code><keysym>KEY_F1</keysym></code>	KEY_F1
Key's code	<code><keycode>0x3B</keycode></code>	0x3B
Combinations or sequences of keys	<code><keycombo> <keycap>Ctrl</keycap> <keycap>S</keycap> </keycombo></code>	Ctrl-S
Programs Menu	<code><guimenu>File</guimenu></code>	File
Menu Items	<code><guimenuitem>Save</guimenuitem></code>	Save
Menu Sequences	<code><menuchoice> <shortcut> <keycombo> <keycap>Ctrl</keycap> <keycap>S</keycap> </keycombo> </shortcut> <guimenu>File</guimenu> <guimenuitem>Save</guimenuitem> </menuchoice></code>	File→Save (Ctrl-S)
Mouse Button	<code><mousebutton>left</mousebutton></code>	left
Command Names	<code><command>comando</command></code>	command
Application Names	<code><application>application</application></code>	application
Text Bibliographical Reference	<code><citation>reference</citation></code>	[reference]
Quote	<code><blockquote> <attribution>Text Author</attribution> <para>Quote Text.</para> </blockquote></code>	Quote Text. --Text Author
Index	(NA)	See Section 4.5 .
File Names		file

	<code><filename>file</filename></code>	
Directories	<code><filename id="directory">directory</filename></code>	directory/
Emphasize Text [a]	<code><emphasis>text</emphasis></code>	<i>text</i>
Footnotes	<code><footnote> <to>Footnote text</to> </footnote></code>	(See note at the end of this table)
URLs	<code><ulink url="http://www.conectiva.com">Conectiva S.A.</></code>	Conectiva S.A.
Markups List	<code><itemizedlist> <listitem> <to>item</to> </listitem> <listitem> <to>item</to> </listitem> </itemizedlist></code>	<ul style="list-style-type: none"> • item • item
Numbered List	<code><orderedlist> <listitem> <to>item</to> </listitem> <listitem> <to>item</to> </listitem> </orderedlist></code>	<ol style="list-style-type: none"> 1. item 2. item
Segmented List	<code><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></code>	<p>Binary to Decimal Conversion</p> <p>Binary: 00</p> <p>Decimal: 0</p> <p>Binary: 01</p> <p>Decimal: 1</p> <p>Binary: 10</p> <p>Decimal: 2</p>
Variable List	<code><variablelist> <varlistentry> <term>Entry 1</term> <listitem> <to>Description</to> </listitem> </varlistentry> <varlistentry> <term>Entry 2</term> <listitem> <to>Description</to> </listitem> </varlistentry></code>	<p><i>Entry 1</i></p> <p>Description</p> <p><i>Entry 2</i></p> <p>Description</p>

	<code></variablelist></code>	
Simple Lists	<pre> <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> </pre>	1 2 3 4 5 6 A, B, C, D, E, F
Pictures	(NA)	See Section 4.6
Table	(NA)	See Section 4.7
Programs List	(NA)	See Section 4.8
Glossary	<pre> <glossentry> <glossterm>Term</glossterm> <glossdef> <to>Definition</to> </glossdef> </glossentry> </pre>	(See the glossary at the end of this document)
Crossed References	<pre> <section id="secao"> ... </section> <section id="reference the other section"> ... <to>Please, see<xref linkend="secao"> for more information. </pre>	(NA)
Notes: a. The 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 stylesheet used.		

4.5. 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 4.10.1](#)). Details about the process used to generate these indexes are shown in [Section 4.10.3](#).

The indexes have nesting levels. The markup of an index is done by the code [Example 4–2](#).

Example 4–2. Code for the generation of an index

```

<indexterm>
  <primary>Main level</primary>

```

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

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

Example 4–3. Use of the attributte 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 4–3](#) 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 4–4. Usage of values `startofrange` and `endofrange` on the attributeclass

```
<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>
```

4.6. Inserting Pictures

It's necessary to insert pictures for all types of media on which the document will be published.

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

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

Example 4–5. 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 text flux appears. The value 1 allows the picture to be moved to a more convenient location (this location can be described on the style sheet used or even can be controlled by the application being used).

4.6.1. Alternative Methods

The first alternative to [Example 4–5](#) is the elimination of elements `<figure>` or `<informalfigure>`.

Another interesting alternative when it's the decision to publish the text on media and pictures aren't accepted, is the use of a wrapper `<imageobject>`.

Example 4–6. 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 on 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. In case of the output format doesn't support images the element `<textobject>` will be used. However, the biggest advantage in usage of the format [Example 4–6](#) is that on the release 5.0 of the DocBook the element `<graphic>` will cease to exist.

As a disadvantage there's the need for more than one representation code of the same information. It's up to the author to decide which method he will implement illustrations and pictures on his or hers document, but for compatibility reasons with future versions *I recommend* the use of this method for pictures and graphics.

4.7. Tables

Many information are best represented when formatted as tables.

A primitive way to create tables was already presented on the [Table 4–1](#) with the use of `<simplelist>`, however, the DocBook has more sophisticated methods to deal with this information.

Example 4–7. Inserting tables

```
<table frame="all">
  <title>Sample Table</title>
  <tgroup cols="5">
    <colspec colname="column1">
    <colspec colname="column2">
    <colspec colname="column3">
    <colspec colnum="5" colname="column5">
    <spanspec namest="column1" nameend="column2" spanname="span-horiz" align="center">
    <spanspec namest="column2" nameend="column3" spanname="span-horiz-vert" align="center">
  <thead>
    <row>
      <entry spanname="span-horiz">
        <foreignphrase>Span</foreignphrase> horizontal
      </entry>
      <entry>Heading 2</entry>
      <entry>Heading 3</entry>
      <entry>Heading 4</entry>
    </row>
  </thead>
  <tfoot>
    <row>
      <entry>Footing 1</entry>
      <entry>Footing 2</entry>
      <entry>Footing 3</entry>
      <entry>Footing 4</entry>
      <entry>Footing 5</entry>
    </row>
  </tfoot>
  <tbody>
    <row>
      <entry>Data11</entry>
      <entry>Data12</entry>
      <entry>Data13</entry>
      <entry>Data14</entry>
      <entry>Data15</entry>
    </row>
    <row>
      <entry>Data21</entry>
      <entry>Data22</entry>
      <entry>Data23</entry>
      <entry>Data24</entry>
      <entry morerows="1" valign="middle">
        <foreignphrase>Span</foreignphrase> vertical
      </entry>
    </row>
    <row>
      <entry>Data31</entry>
      <entry spanname="span-horiz-vert" morerows="1" valign="bottom">
        <foreignphrase>Span</foreignphrase> duplo
      </entry>
    </row>
  </tbody>
</table>
```

```

    </entry>
    <entry>Data34</entry>
  </row>
  <row>
    <entry>Data41</entry>
    <entry>Data44</entry>
    <entry>Data45</entry>
  </row>
</tbody>
</tgroup>
</table>

```

Table 4–2. Example Table

Horizontal Span		Heading 2	Heading 3	Heading 4
Data11	Data12	Data13	Data14	Data15
Data21	Data22	Data23	Data24	Vertical Span
Data31	Double Span		Data34	
Data41	Data44	Data45		
Footing 1	Footing 2	Footing 3	Footing 4	Footing 5

4.8. Listings and program codes

An interesting feature is to show parts of code and the possibility to comment on them. The DocBook allows the insertion of the program code and also callouts to specific lines of this code.

Such a feature was used, for example, to demonstrate how a catalogue file is configured (see [Section 4.3](#)).

The used code was demonstrated below. In case the callout feature isn't desired, it's possible to eliminate the areas between `<areaspec>` and `<calloutlist>`.

```

<example id="sample-catalog">
  <title>Catalog Sample</title>
  <programlistingco>
    <areaspec>
      <area coords="1" id="ex.catalogue.comment">
      <area coords="5" id="ex.catalogue.definition">
      <area coords="11" id="ex.catalogue.eof">
    </areaspec>
    <programlisting>
-- Catalogues for the Conectiva S.A. Style --

OVERRIDE YES

PUBLIC "-//Conectiva SA//DTD books V1.0//EN" "/home/ldp/estilos/livros.dtd"

```

```

DELEGATE "-//OASIS" "/home/ldp/SGML/dtds/catalog.dtd"

DOCTYPE BOOK /home/ldp/SGML/dtds/docbook/db31/docbook.dtd

-- EOF --
    </programlisting>
  <calloutlist>
    <callout arearefs="ex.catalogue.comment">
      <to> Comment. Comments begin with <quote>--</quote>
      and follows to the end of the line. </to>
    </callout>
    <callout arearefs="ex.catalogue.definition">
      <to> The public type association
        <parameter class="option">"//Conectiva SA//DTD books V1.0//EN"</parameter>
        with the file <filename>books.dtd</filename> on the directory
        <filename class="directory">/home/ldp/estilos</filename>. </para>
      </callout>
    <callout arearefs="ex.catalogue.eof">
      <para> Comment informing the end of the file. </para>
    </callout>
  </calloutlist>
</programlistingco>
</example>

```

The listings can be directly inserted on the document's body without the need of the element `<example>` or `<para>`.

The calling coordinates specifications are done with reference to the code line which will be commented.

4.9. Crediting Translators and Converters

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, e.g., from HTML to DocBook SGML. Source code conversions help the LDP achieve long term goals for meta-data, and allow us to produce documentation in many different output formats.

Translators take your original document and translate it into other human-readable languages, e.g., from English to Japanese, or from German to English. Each translation allows many more people all over the world to make use of your document, and helps Linux achieve the ultimate goal – Total World Domination(tm)!

As you will see in the following sections, 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.

4.9.1. The `<othercredit>` Tag

All translators and converters 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:

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

4.9.2. The "Acknowledgements" section

Your document should have a section titled "Acknowledgements", 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.

4.9.3. The `<revremark>` tag

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. 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.

4.10. Tools & Hints

The process of producing output and generating indexes is repetitive and error prone. To make things easier some scripts are included here to facilitate this work. Customize and use them at will.

4.10.1. Compiling the sources

The script `compiles-sgml` is a set of grouped commands. As parameters the name of the document should be passed *SGML* and the output format expected.

The script version included below supports the formats *XML*, *HTML*, *TeX*, *RTF*, *PS*, *DVI* and mirrored PS, ideal for the creation of photolithographs.

The generation of the indices is made automatically by the script `collateindex.pl` [\[3\]](#), which should be installed in your system.

Besides the commands below which generate the outputs in different formats, there are other tools

from Cygnus making the direct conversion. Such tools can be obtained in [here](#).

The list below is available [here](#).

Here is also available a version of [collateindex.pl](#).

Example 4–8. Script compiles–sgml

```
#!/bin/bash
#
# Compile DocBook documents into several output formats.
#
# Godoy.
# 19991230 - Initial release.
# 20000117 - Placed the options using "case" and parameters passed
#           via command line. The pages on the Zope are already updated.
#           --- Removed to public version (/home/ldp).
# 20000120 - Placed the call to use the books.dtd.
# 20000126 - Placed the commands for the index generation.
#
# If the jade is already installed, disconsider the line bellow.
JADE=/usr/bin/jade

# If the jade package is already installed, disconsider the line bellow.
# JADE=/usr/bin/openjade

DOCUMENT=$1
shift 1
TYPE=$1

. ~/.bash_profile
. ~/.bashrc

case $TYPE in
    html)
        rm -f *.htm
        rm -f *.html
        perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -N -o index.sgml
        jade -t sgml -V html-index -d /home/ldp/SGML/style/dsssl/docbook/html/docbook.dsl $DOCUMENT
        perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -o index.sgml HTML.index
        $JADE -t sgml -i html -d /home/ldp/SGML/style/dsssl/docbook/html/docbook.dsl -d /home/ldp/
    ;;
    rtf)
        rm -f $DOCUMENT.rtf
        perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -N -o index.sgml
        jade -t sgml -V html-index -d /home/ldp/SGML/style/dsssl/docbook/html/docbook.dsl $DOCUMENT
        perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -o indice.sgml HTML.index
        $JADE -t rtf -V rtf-backend -d /home/ldp/SGML/style/dsssl/docbook/print/docbook.dsl -d /ho
    ;;
    xml)
        rm -f $DOCUMENT.xml
        perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -N -o index.sgml
        jade -t sgml -V html-index -d /home/ldp/SGML/style/dsssl/docbook/html/docbook.dsl $DOCUMENT
        perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -o indice.sgml HTML.index
        $JADE -t sgml -i xml -d /home/ldp/SGML/style/xsl/docbook/html/docbook.xsl $DOCUMENT.sgml
    ;;
    tex)
        rm -f $DOCUMENT.tex
```

```

perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -N -o indice.sgml
jade -t sgml -V html-index -d /home/ldp/SGML/style/dsssl/docbook/html/docbook.dsl $DOCUMENT
perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -o indice.sgml HTML.index
$JADE -t tex -V tex-backend -d /home/ldp/SGML/style/dsssl/docbook/print/docbook.dsl -d /ho
;;
dvi)
rm -f $DOCUMENT.tex
rm -f $DOCUMENT.dvi
perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -N -o indice.sgml
jade -t sgml -V html-index -d /home/ldp/SGML/style/dsssl/docbook/html/docbook.dsl $DOCUMENT
perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -o indice.sgml HTML.index
$JADE -t tex -V tex-backend -d /home/ldp/SGML/style/dsssl/docbook/print/docbook.dsl -d /ho
jadetex $DOCUMENT.tex
;;
mirror)
rm -f $DOCUMENT.tex
rm -f $DOCUMENT.dvi
rm -f $DOCUMENT.mirror.ps
perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -N -o indice.sgml
jade -t sgml -V html-index -d /home/ldp/SGML/style/dsssl/docbook/html/docbook.dsl $DOCUMENT
perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -o indice.sgml HTML.index
$JADE -t tex -V tex-backend -d /home/ldp/SGML/style/dsssl/docbook/print/docbook.dsl -d /ho
jadetex $DOCUMENT.tex
dvips -h /home/ldp/estilos/skel/mirr.hd -O 1.5cm,3cm -f $DOCUMENT.dvi -o $DOCUMENT.mirror.
;;
ps)
rm -f $DOCUMENT.tex
rm -f $DOCUMENT.dvi
rm -f $DOCUMENT.ps
perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -N -o indice.sgml
jade -t sgml -V html-index -d /home/ldp/SGML/style/dsssl/docbook/html/docbook.dsl $DOCUMENT
perl /home/ldp/SGML/style/dsssl/docbook/bin/collateindex.pl -o indice.sgml HTML.index
$JADE -t tex -V tex-backend -d /home/ldp/SGML/style/dsssl/docbook/print/docbook.dsl -d /ho
jadetex $DOCUMENT.tex
dvips -The 1.5cm,3cm -f $DOCUMENT.dvi -o $DOCUMENT.ps
;;
*)
echo "How to use: $0 file {html|tex|rtf|xml|ps|dvi|mirror}"
exit 1
esac

exit 0

```

Obviously such script can be modified forming a Makefile, optimizing some functions.

4.10.2. Inserting a summary on the initial articles page

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

To enable this it's necessary to make a modification on the stylesheet file used.

The example below describes the process and it's use is shown in [Example 4–8](#).

Example 4–9. Stylesheet 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>

```

4.10.3. Inserting indexes automatically

Although DocBook has markups for the composition of indexes, these are not generated automatically. The tool collateindex.pl allows the indexes to be generated automatically.

The way to use this script is described bellow and a practical example can be seen previously in this document (see [Example 4–8](#)).

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

```
$ jade -t sgml -d html/docbook.dsl -V html-index document.sgml
```

- Generate the document `index.sgml` with collateindex.pl.

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

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

Example 4–10. 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 4.5](#) for information on how to insert necessary information on the text.

Note: Remember that if you're trying to get Table of Contents or Indexes on PS or PDF output you'll need to run `jadetex` or `pdfjadetex` at least three times. This is a TeX requirement and not a DocBook or related application one.

4.10.4. Making notes on the text while it's being written

An interesting feature while writing a text is to be able to check if such text will be presentable or not on the final draft. It's common to have several parts of the text marked as drafts, especially when we're updating an already existent document.

DocBook allows along with an entity of parameters to include or not the insertions of specific parts of text in several places on the document based on the context. Sometimes for an upgrading we need to see how the document looks like or just have sketches of a new session or chapter, but we don't want this sketch to appear on the final draft.

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

Example 4–11. Use of parameter entities

```
<!entity % review "INCLUDE">
...
<![%review;[
<para>This paragrph 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 4–11](#). When the changes on the text are considered final, it's necessary just to remove parts of the text relative to the 3rd. and 6th. lines.

To keep the draft definitions and don't show the text on the final draft, just alter the specification of the entity from `INCLUDE` to `IGNORE`.

4.10.5. Re–using parts of documents

An important characteristic about the *external entities* is about the re–using issue of text and documents.

With those characteristics it's possible to create files with text used several times (e.g. licenses and company policies) and simply include those files in the appropriate place.

An example and application of this characteristic was found previously in [Example 4–10](#). Another example is the *SGML* code of this HOWTO.

4.11. Document samples

As stated before each type of document has a particular heading and valid commands structure. The following sub-sections will provide heading and valid commands structures on articles and books.

These examples do not cover all possibilities and they are available here only with the intention to serve as generic guides for what it's possible to do.

4.11.1. Article example

```
<article class="whitepaper" id="using -docbook" lang="pt-br"><?dbhtml filename="using-docbook.htm"

  <artheader>
    <title>Como-Fazer DocBook</title>
    <author>
      <firstname>Jorge</firstname>
      <othername>Luiz</othername>
      <surname>Godoy</surname>
      <othername>Filho</othername>
      <affiliation>
        <orgname><ulink url="http://www.conectiva.com">Conectiva S.A.</ulink></orgname>
        <orgdiv>Publishing Department</orgdiv>
        <address><email>godoy@conectiva.com</email></address>
      </affiliation>
    </author>
    <revhistory>
      <revision>
        <revnumber>1.0</revnumber>
        <date>27 de janeiro de 2000</date>
        <authorinitials>godoy</authorinitials>
        <revremark>Versão inicial.</revremark>
      </revision>
    </revhistory>

    <legalnotice>
      <para>This document can be freely translated and distributed. It's released
        under the LDP License.</para>
    </legalnotice>

    <keywordset>
      <keyword>SGML</keyword>
      <keyword>DocBook</keyword>
      <keyword>DTD</keyword>
      <keyword>XML</keyword>
      <keyword>catalogs</keyword>
      <keyword>documents</keyword>
      <keyword>Publishing</keyword>
      <keyword>Conectiva</keyword>
      <keyword>configuration</keyword>
      <keyword>use</keyword>
      <keyword>tools</keyword>
      <keyword>HOWTO</keyword>
    </keywordset>
```

```
</artheader>
```

4.11.2. Book Example

Chapter 5. LDP Style Guide

5.1. Deciding on a Subject

Before you begin writing a HOWTO, it is essential that you determine what subject area you will cover. It is best if the subject area is:

Not too broad, and not too narrow. Try to cover too much information, and you may sacrifice depth. It is better to cover a small subject area fully 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 your subject matter is very small, it might be better included as part of another HOWTO. This makes it easier for readers to find the HOWTO they need. Search the LDP repository for HOWTOs on related topics, and see if you could place your information in an existing HOWTO.

How much is too much? How little is too little? That depends on the subject you chose, your mastery of that subject, and many other factors. Just keep this admonition in mind, and use good judgement.

- **Clearly defined.** Know before you begin exactly where the boundaries of your subject area lie. You should not cover the same ground as another HOWTO, and you should try not to leave gaps between your HOWTO and related HOWTOs, either.
- **Undocumented.** Before writing on a particular subject, check other HOWTOs at the LDP, and see if the topic is already documented. If it is, refer to the other HOWTO instead of rewriting documentation that already exists. Don't reinvent the wheel.

If the HOWTO already in place is insufficient, or needs updating, contact the author and offer to help. LDP authors are usually nice folks. After all, they put in their own valuable time to help people they don't even know. And, they appreciate your help. But, please, don't duplicate work. It causes confusion for everyone.

- **Pre-approved by the LDP.** Before you proceed with your HOWTO, post to the `ldp-discuss` list and get some feedback from other LDP volunteers. Checking with the list *before* you begin can save you headaches *later*. The author speaks from experience.

It is a really good idea to join the `ldp-discuss` list, and follow it regularly, even if you never post. It's a good way to stay current on the activities, needs, and policies of the LDP. Although the LDP volunteers are here to assist you, it is ultimately your responsibility to learn these policies, and to follow them.

5.2. Developing an Outline

Before you actually begin writing, prepare an outline. An outline will help you 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 logically follow the item before it, and lead into the item it precedes. For example, a HOWTO about a particular program shouldn't have a section on *configuration* before one on *installation*.

When you are comfortable with your outline, look over it once more, with a critical eye. Have you covered every relevant topic in sufficient detail? Have you wandered beyond the scope of the HOWTO? You might want to show it to someone else, and ask for feedback. It's much easier to reorganize your HOWTO at the outline stage than it will be later. Consider submitting the outline to the `ldp-discuss` list for more feedback.

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

Remember Linus' Law:

"Given enough eyeballs, all [typos] are shallow."

—Eric S. Raymond

(With apologies to Mr. Raymond.)

FIXME: Need a reference to the "standard" HOWTO layout, for topic areas such as Credits, License, Copyright, etc., etc.

5.3. Writing the Text

At this point, your HOWTO has been organized, and bits of raw information have been collected, documented, and inserted into the outline. Good news: You're past the midpoint; it's all downhill from here.

Your next challenge is to massage all of the raw data you've collected into a readable, entertaining, and understandable whole.

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 dry, boring information in your own unique way.

It is beyond the scope of this document to provide a comprehensive guide to effective writing, so I won't try to go beyond the basics. In the "[References](#)" 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.

For starters, here is some good advice from [Politics and the English Language](#):

A scrupulous writer, in every sentence that he writes, will ask himself at least four questions, thus:

1. What am I trying to say?
2. What words will express it?
3. What image or idiom will make it clearer?
4. Is this image fresh enough to have an effect?

And he will probably ask himself two more:

1. Could I put it more shortly?
2. Have I said anything that is avoidably ugly?

...One can often be in doubt about the effect of a word or phrase, and one needs rules that one can rely on when instinct fails. I think the following rules will cover most cases:

1. Never use a metaphor, simile, or other figure of speech which you are used to seeing in print.
2. Never use a long word where a short one will do.
3. If it is possible to cut a word out, always cut it out.
4. Never use the passive where you can use the active.
5. Never use a foreign phrase, a scientific word, or a jargon word if you can think of an everyday English equivalent.
6. Break any of these rules sooner than say anything outright barbarous.

—George Orwell

And, from a purely stylistic point of view, I believe that the first-person perspective of many HOWTOs adds to their charm, an attribute most documentation in other forms sorely lacks. Don't be afraid to speak for yourself, use the word "I", or describe personal experiences and opinions.

If it hasn't become painfully obvious yet, the underlying principle of all these suggestions is simplicity. Your readers are already struggling with new concepts, so don't make them struggle to understand your language. Remember the KISS principle: Keep It Simple, Stupid!

5.4. Editing and Proofing 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 doesn't 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. This is the time to correct that.

When editing and proofing your work, you must check for obvious mistakes, such as spelling errors and typos. However, you should check for deeper, but less obvious errors as well, such as "holes" in the information. Make sure that the contents of every section match the title of that section precisely.

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.

In a sense, editing is like code review in software development. Having a programmer review their own code doesn't make much sense, does it? Why does having a writer edit their own document make any more sense? So, recruit a friend, or write the ldp-discuss list to find a volunteer to proofread before submitting your HOWTO.

Note: If you are writing in a language in which you are not fluent, I strongly recommend that you seek an editor who is. Technical documentation, more than any other type of writing, must use extremely precise grammar and vocabulary. Misuse of the language, no matter how understandable and unintended, makes your HOWTO less valuable.

5.5. Maintaining Your HOWTO

Just because your HOWTO has now been published doesn't mean your job is done. HOWTOs need regular maintenance, to make sure they are up to date, and to improve them in response to readers' ideas and suggestions. A HOWTO is a living, growing body of knowledge, not just a publish-and-forget-it static document.

You put your email address in the HOWTO, and politely requested feedback from your readers, right? Once you are officially published, you will begin to receive notes with suggestions. Some will be very valuable; others will request personal assistance. You should feel free to decline personal assistance if you cannot spare the time. Writing a HOWTO for the LDP doesn't commit you to a lifetime of free support for anyone on the net. It is polite to refer questioners to another resource, if you can. And, if you see a pattern in the questions you receive, it might indicate a topic you should add to your HOWTO.

5.6. References

There are many guides to writing style available online. Here is a brief list of some of the best:

- [*Politics and the English Language*](#).
 - [*Elements of Style*](#)
 - FIXME: Please send the URL of your favorite resource on technical writing.
-

Chapter 6. Additional Style–related Items

This section contains additional style–oriented topics to consider when preparing your document.

6.1. Date formats

The `<pubdate>` tag in your header should be in the following format:

```
v1.0, 21 April 2000
```

6.2. Graphics formats

When submitting graphics to the LDP, please submit one set of graphics in `.eps`, and another in either `.gif` or `.jpg`. Be aware of the patent issues with `.gif`, but it makes slightly better pictures than `.jpg`.

6.3. DocBook Versions

The LDP supports and accepts documentation in the following formats:

- SGML DocBook versions 4.x and 3.1
- SGML LinuxDoc
- XML DocBook version 4.1.2

When writing your DocBook header, it should look like this:

```
<!doctype article public "-//OASIS//DTD DocBook V4.1//EN">
```

6.4. Depreciated Tags

Tags listed in *DocBook: The Definitive Guide* as depreciated are discouraged for use in LDP documentation. Some ways to use newer tags are listed in the tip and tricks section.

6.5. Tag Minimization

Tag minimization is using `</>` instead of the full end of a tag (such as `</para>`). Since this makes the document more confusing for future authors and LDP members, and is not allowed in XML DocBook, please refrain from this practice.

6.6. Conventions

Conventions for different kinds of text is as follows:

If you're going to show the use of a command, format the command so it looks like a user's command line. The prompt must contain the shell type (bash, tcsh, zsh, etc) followed by a \$ for commands to be run as a normal (non-root) user or a # for a root user.

A command would then look like this:

```
bash$ command "run as a normal user"
bash# command "run as a root user"
tcsh# setenv DISPLAY :0.0
```

Chapter 7. Tips and Tricks with DocBook

This section covers a few quirks of DocBook that you may run into when writing your documents.

7.1. Including Images

If you plan on including images in your HOWTOs, you can now do this, as LinuxDoc didn't support images. Here's a sample way of including an image in your HOWTOS:

```
<figure>
  <title>LyX screen shot</title>
  <mediaobject>
    <imageobject>
      <imagedata fileref="lyx_screenshot.eps" format="eps">
    </imageobject>
    <imageobject>
      <imagedata fileref="lyx_screenshot.jpg" format="jpg">
    </imageobject>
    <textobject>
      <phrase>Screen shot of the LyX document processing program</phrase>
    </textobject>
  </mediaobject>
</figure>
```

This is a better way than using `<graphic>` for two reasons. First, `<graphic>` will be removed in DocBook 5.0 in favor of the `<mediaobject>` tag. So you may as well get started with the right way now. Second, `<mediaobject>` allows for different kinds of media based on what the output is. In this example, the first `<imageobject>` is an encapsulated PostScript(eps) file for use with formats derived from TeX such as DVI, PS, and PDF. The second `<imageobject>` is a JPEG image for visual display, mostly for HTML output. The `<textobject>` is presented if the output doesn't support graphics (TXT). Think of it as an `<alt>` tag.

7.2. Naming separate HTML files

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, or so you know what files are what. 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">
```

On the first `<chapter>` tag, do not modify it, 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. Names should include only alphanumeric characters, and it should be short enough to understand what it is.

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

7.3. Using ldp.dsl

The LDP uses its own DSSSL file, which adds things like a white background and automatic generation of the table of contents you see at the beginning of HOWTOs. You can find the latest copy of the file at <http://www.linuxdoc.org/authors/tools/ldp.dsl>.

Once you have the file, you may need to do some editing of the first few lines based on the location of your DocBook DSSSL files. My example uses the Cygnus tool set.

Place the `ldp.dsl` file in `/usr/lib/sgml/stylesheets` and bring it up under your favorite text editor. You should see something like this:

```
<!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN" [  
<!ENTITY % html "IGNORE">  
<![%html;[  
<!ENTITY % print "IGNORE">  
<!ENTITY docbook.dsl SYSTEM "docbook.dsl" CDATA dsssl>  
]]>  
<!ENTITY % print "INCLUDE">  
<![%print;[  
<!ENTITY docbook.dsl SYSTEM "docbook.dsl" CDATA dsssl>  
]]>  
>
```

Change the first "docbook.dsl" to read

`/usr/lib/sgml/stylesheets/nwalsh-modular/html/docbook.dsl`

Change the second "docbook.dsl" to read

`/usr/lib/sgml/stylesheets/nwalsh-modular/print/docbook.dsl`

If you're using another DSSSL, point those two files to the location of the HTML and print DSSSL files. They're usually in directories called `html` and `print`.

With that complete, you can now generate HTML files:

```
bash$ mkdir HOWTO-HOWTO ; cd HOWTO-HOWTO  
bash$ jade -t sgml -i html -d /usr/lib/sgml/stylesheets/ldp.dsl\#html ../HOWTO-HOWTO.sgml
```

The first command creates a new directory to put your files into. The second command (the `jade` one) generates individual HTML files for each section of your document. If you are going to something like RTF, you can do this:

```
bash$ jade -t rtf -d /usr/lib/sgml/stylesheets/ldp.dsl ../HOWTO-HOWTO.sgml
```

Chapter 8. Distributing your documentation

8.1. Before you distribute

Before you distribute your code to millions of potential readers there are a few things you should do.

First, be sure to spell-check your document. Most utilities that you would use to write SGML have plug-ins to perform a spell check. If not, there's always the `aspell` program.

Second, get someone to review your documentation for comments and factual correctness. The documentation that is published by the LDP needs to be as factually correct as possible, as there are millions of Linux users that may be reading it. If you're part of a larger mailing list talking about the subject, ask others from the list to help you out.

Third, create a web site where you can distribute your documentation. This isn't required, but is helpful for people to find the original location of your document.

8.1.1. Validate your SGML code

Using `jade`, or really the `nsgmls` command, you can validate your `.sgml` code against the DTD to make sure there aren't any errors.

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

If there are no issues, you'll just get your command prompt back.

8.2. Copyright and Licensing issues

In order for an LDP document to be accepted by the LDP, it must be licensed to conform to the "LICENSE REQUIREMENTS" section of the LDP Manifesto located at <http://www.linuxdoc.org/manifesto.html>. As an author, you may retain the copyright and add other restrictions (for example, you must approve any translations or derivative works).

We recommend using the [GNU Free Documentation License \(GFDL\)](#) or the [Open Publication License \(OPL\)](#) *without* options A and B. If you choose, you can get DocBook markups up 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. Due to its length, you may just want to provide a link to the source.

If you choose to use a boilerplate copyright, simply copy it into your source code under a section called "Copyright and Licenses" or similar. Also include a copyright statement of your own (since you still own it). If you are a new maintainer for an already-existing HOWTO, you must include the previous copyright statements of the previous author(s) and the dates they maintained that document.

You'll note that the licensing for the LDP Authoring Guide requires notification to the author of any derivative works or translations. I also explicitly place any source code (aside from the SGML the Guide was written in) under the GPL. If your HOWTO includes bits of source code that you want others to use, you may

do the same.

8.3. Submission to LDP

Once your LDP document has been carefully reviewed, you can release your document to the LDP. Send an e-mail with the SGML source code as an attachment (you may gzip it if you like) to [<ldp-submit@lists.linuxdoc.org>](mailto:ldp-submit@lists.linuxdoc.org).

Be sure to include the name of your HOWTO in the subject line, and use the body to outline changes you've made and attach your HOWTO. This allows the maintainers to do their jobs faster, so you don't have to wait for your HOWTO to be updated on the LDP web site. If you don't hear anything in 5 calendar days, please follow up with an e-mail to make sure things are still in process.

If your HOWTO contains extras, such as graphics or a special catalog, create a tar.gz file with all the files in it including the .sgml source code and mail it as an attachment to the ldp-submit list.

If you are using the LDP CVS tree while developing your document, the LDP will still need to be notified when your document is ready to be published. E-mail should be sent to [<ldp-submit@lists.linuxdoc.org>](mailto:ldp-submit@lists.linuxdoc.org). Indicate the title of your document and the relative path to the file(s) in the LDP CVS tree within your message.

8.4. Maintaining Your HOWTO

Just because your HOWTO has now been published doesn't mean your job is done. HOWTOs need regular maintenance, to make sure they are up to date, and to improve them in response to readers' ideas and suggestions. A HOWTO is a living, growing body of knowledge, not just a publish-and-forget-it static document.

You put your email address in the HOWTO, and politely requested feedback from your readers, right? Once you are officially published, you will begin to receive notes with suggestions. Some will be very valuable; others will request personal assistance. You should feel free to decline personal assistance if you cannot spare the time. Writing a HOWTO for the LDP doesn't commit you to a lifetime of free support for anyone on the net. It is polite to refer questioners to another resource, if you can. And, if you see a pattern in the questions you receive, it might indicate a topic you should add to your HOWTO.

Chapter 9. FAQs about the LDP

Q: [I want to help the LDP. How can I do this?](#)

Q: [I want to publish a collection of LDP documents in a book. How is the LDP content licensed?](#)

Q: [I found an error in an LDP document. Can I fix it?](#)

Q: [But I don't know SGML/Can't get the tools working/Don't like SGML](#)

Q: I want to help the LDP. How can I do this?

A: The easiest way is to find something and document it. Also check the unmaintained HOWTOs and see if there is a subject there that you know about and can continue documenting.

Q: I want to publish a collection of LDP documents in a book. How is the LDP content licensed?

A: Please see <http://www.linuxdoc.org/COPYRIGHT.html>. Note that this is only a guideline to authors. However, the licensing cannot be more restrictive than what is listed in that URL.

Q: I found an error in an LDP document. Can I fix it?

A: Contact the author of the document, or the LDP coordinator at [<ldp-discuss@lists.linuxdoc.org>](mailto:ldp-discuss@lists.linuxdoc.org) and mention the problem and how you think it needs to be fixed.

Q: But I don't know SGML/Can't get the tools working/Don't like SGML

A: That's okay. You have the option of writing your first draft of the HOWTO in the format of your choice, then submit that to the LDP. An LDP volunteer will review the document, then convert it into DocBook for you. Once that's done, it will be easier for you to maintain the HOWTO. If you run into questions, you can always drop a line to the LDP volunteer or the LDP Docbook list at [<ldp-docbook@lists.linuxdoc.org>](mailto:ldp-docbook@lists.linuxdoc.org).

Glossary

attribute

One 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. Example of an attribute is `id="identification"`, which gives to the attribute `id` the value `identification`.

Document Type Definition (DTD)

Group of statements that define element names and their attributes specifying the rules for combinations and sequences. It's the DTD that define which elements can or cannot be inserted in the context on which the cursor is in.

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 stylesheets pages for SGML.

element

The elements define the hierarchical structure of a document. The majority of elements have opening and closing pointers. Among these pointers, pieces of text or even the whole document being written can be found. There are empty elements which contains only opening pointers without any content.

entity

Entity is a name designated for some part of data so that it can be referenced by a name. These designations are made by a statement and the stored data might hold from simple characters to chapters or set of statements of a DTD. There are parameter entities generic, external, internal and of data on the SGML.

external entity

An external entity points to an external document. External entities are used to include texts on certain locations of a SGML document. Suggestions for its use includes sample screens, legal notes and chapters.

generic entities

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

internal entity

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

parameter entity

An entity often used on the DTD. The entity's name starts with a percent sign (%) and it ends with a semicolon.

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 a printed text, the chart can appear either at the top or at the bottom of the page. It can also be placed on the next page if that's too large.

processing instruction

A processing instruction is a command passed to the document formatting tool. It starts with "<?". A sample of instruction is used on this document for the generated file's choice when the file is converted to HTML: `<?dbhtml filename="file.html">`

SGML

Standard Generalized Markup Language. It's also an international standard (ISO8879) which specifies rules for the creation of electronic documents in markup systems regardless the work platform used.

tag

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

XML

eXtensible Markup Language. A subproduct of SGML created specifically for Internet use.

XSL

XML Style Language. XSL is to a XML document what a DSSSL style is for a SGML document. In fact, the style is a document XML.
