HOWTO–HOWTO

Table of Contents

Table of Contents

HOWTO−HOWTO

Mark F. Komarinski

v1.4 12 Jun, 2000

Revision History

Revision 1.4	Jun 12, 2000	Revised by: mfk

Documented vim and sgedit. Spelling and other changes from ldp list. Also added LDP guidelines under style guide.

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

Table of Contents

About this HOWTO

Purpose / Scope of this HOWTO

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 can be found on my homepage http://www.cgipc.com/~markk in its SGML source. Other versions may be found in different formats at the LDP homepage http://www.linuxdoc.org.

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 HOWTO is designed to help you get familiar with how the LDP works, and what tools you'll need to write your own HOWTO.

About the LDP

The following is an excerpt from the LDP Manifesto (http://www.linuxdoc.org/manifesto.html)

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.

You can find out more about the Linux Documentation Project at http://www.linuxdoc.org

---

Feedback

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

Copyrights and Trademarks

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 (available at many LDP locations) 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.br>`). A great deal of thanks to both of them for their help.

Conventions

Commands that are listed have the following format. Commands are prefaced with the name of the current shell running. This is followed by a $ for commands that should be run as a normal (non−root) user. Shells followed by a # are commands that should be run as a root user.

Introduction to the LDP and SGML

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.

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. 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 <table> 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.

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.

For New Authors

If you are a new to the LDP and want to pick up an unmaintained HOWTO or write a new HOWTO or mini−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.

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>`.

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.

DSSSL

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

Norman Walsh DSSSL

http://nwalsh.com/docbook/dsssl/db152.zip

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 http://nwalsh.com/docbook/dsssl/db152d.zip.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 that provides for a table of contents.

LDP DSSSL

http://metalab.unc.edu/gferg/ldp/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.

DocBook DTD (version 3.1)

Required − http://www.oasis−open.org/docbook/sgml/3.1/docbk31.zip

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.

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.

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.

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.

Jade wrappers

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

---

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.

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

CD.

Editing tools

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

LyX

http://www.lyx.org/

LyX provides the power of writing SGML with the ease−of−use of a regular word processor. It's not a WYSIWYG program, but more WYSIWYM (What You See Is What You Mean) application, since what you see on the screen isn't necessarily what happens after the SGML processor is done with it. The display that LyX provides is similar to, but not exactly like, what the output from jade would look like. However, it's close enough for you to see the flow of the document. Sections and subsections are numbered and put in bold, and different fonts are used to signify things like <code> or <url> tags. Most tags are hidden from the main LyX window while you edit, since LyX writes in TeX, then exports the TeX to SGML.

Figure 1. LyX screen shot

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.

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.

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.

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.

Other/Reference

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

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.

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.

Getting Started with DocBook

This section covers the new method of writing LDP documentation, using the DocBook 3.1 DTD. We'll cover getting, installing, and using tools, along with an introduction to DocBook tags. Since there are over 300 DocBook tags, we won't cover them all here. Really interested readers can go to http://www.docbook.org for more information.

Downloading and installing the tools

Manual using jade/OpenJade

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`, `$_toolroot/dssl`)
3. You'll need to set the `SGML_CATALOG_FILES` environment variable to the catalogs that you have under`$_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.

---

sgmltools

Unlike previous versions of sgmltools, you will require sgmltools version 2.x for use with DocBook. Since some major distributions ship with sgmltools 1.x, you'll need to remove the sgmltools 1.x package and install either a 2.0 version, or a CVS version. To get the latest CVS source code version, you can use the following set of commands:

```
bash$ CVSROOT=:pserver:cvs@cvs.sgmltools.org:/home/cvs
bash$ export CVSROOT
bash$ cvs login
bash$ cvs -z6 get sgmltools
```

The CVS password is 'cvs'. Once downloaded, You can just use

```
bash$ ./compile
bash$ make
bash# make install
```

to install sgmltools. For Red Hat−based systems (using RPM) you can use the rpmfind command to get the latest sgmltools. The rpmfind program is available at http://www.rpmfind.net/. Make sure you get sgmltools and not sgml−tools, as the latter is sgml−tools 1.0.9 and only works with LinuxDoc documents. For Debian−based systems, running 2.2 "Potato" and above, apt−get will retrieve the right package for you:

```
bash# apt-get install sgmltools-2
```

As with Red Hat Linux, the sgml−tools package is outdated. Be sure to get sgmltools−2.

Cygnus DocBook Tools

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

- sgml−common
- docbook
- stylesheets

Red Hat has the latest version on their web site:
http://www.redhat.com/support/errata/RHBA−2000022−01.html.

Download/get/sneakernet 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.

Writing SGML by hand

Most of this is covered by Jorge Godoy's Using DocBook document. Those interested can read it athttp://metalab.unc.edu/godoy/using−docbook/using−docbook.htmlfor writing DocBook using your favorite text editor.

If you write SGML by hand: SGML has over 300 tags, and uses tags much more heavily than HTML. It's recommended that you use an existing HOWTO as a template and see how other authors have written. It's also recommended that you use a user−friendly editor like PSGML or WordPerfect for Windows, as it lists many of the tags that are available.

Writing SGML using LyX

New documents

You can easily start a new HOWTO using LyX. Use the File−>New From Template... menu command to bring up the template listings. Select *Templates* on the right side of the screen, then select *docbook_template.lyx* in the file listing. Select OK, and you'll have a new document. Fill in the items, such as title, abstract, and author name, then start writing.

Figure 2. DocBook Template screen from LyX

Existing documents

If you have an already−existing LyX, TeX, or text document, you can import it into LyX with the File−>import command. Once the file is imported, go to Layout−>Document... In the popup window, under Style, select *SGML (DocBook Article)*. You'll be asked if you want to convert all text over, and say Yes. You will need to re−apply most tags, but it's a fairly simple matter of selecting text and changing the style. Many LyX functions have a keyboard shortcut to assist you.

Figure 3. Document Layout screen

Exporting documents to SGML

Once your document is written or converted, save it in LyX format. This will allow you to edit future versions easily. Then, go to File−>Export−>as DocBook... and the file will be exported in DocBook.

Writing SGML using PSGML

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/.

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

SGML Smoke Test

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

```
60;!DOCTYPE test [
60;!ELEMENT test - - (#PCDATA)62;
]62;
```

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:

```
60;!DOCTYPE test [
60;!ELEMENT test - - (#PCDATA)62;
]62;
60;test62;60;/test62;
```

Writing a New HOWTO in DocBook

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

```
60;!DOCTYPE ARTICLE PUBLIC "-//OASIS//DTD DocBook V3.1//EN"62;
```

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.

Quick Reference for Emacs with PSGML

See Nik Clayton's primer for FreeBSD documentation:
http://www.freebsd.org/tutorials/docproj−primer/psgml−mode.html

Style guides

This section contains notes on conventions that the LDP has agreed to in order to give all LDP documents a similar look and feel. You should keep these guides in mind when writing.

Date formats

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

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

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 then .jpg.

DocBook Versions

Only DocBook 3.1 is supported by the LDP at this time. DocBook 4.0 is under consideration. Many 3.1 documents can be converted to 4.0 easily by avoiding the use of depreciated tags.

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

```
60;!doctype article public "-//OASIS//DTD DocBook V3.1//EN"62;
```

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.

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.

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
```

Tips and Tricks with DocBook

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

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:

```
60;figure62;
60;title62;LyX screen shot60;/title62;
60;mediaobject62;
60;imageobject62;60;imagedata fileref="lyx_screenshot.eps" format="eps"62;60;/imageobject62;
60;imageobject62;60;imagedata fileref="lyx_screenshot.jpg" format="jpg"62;60;/imageobject62;
60;textobject62;60;phrase62;Screen shot of the LyX document processing program60;/phrase62;60;/textobject62;
60;/mediaobject62;
60;/figure62;
```

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.

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:

```
60;article id="index"62;
```

On the first <sect1> tag, do not modify it, as it's usually an introduction and you want that on the first page. For each other <sect> 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.

```
60;sect1 id="tips"62;
```

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://metalab.unc.edu/gferg/ldp/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:

```
60;!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN" [
60;!ENTITY % html "IGNORE"62;
60;![%html;[
60;!ENTITY % print "IGNORE"62;
60;!ENTITY docbook.dsl SYSTEM "docbook.dsl" CDATA dsssl62;
]]62;
60;!ENTITY % print "INCLUDE"62;
60;![%print;[
60;!ENTITY docbook.dsl SYSTEM "docbook.dsl" CDATA dsssl62;
]]62;
]62;
```

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 -ihtml -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
```

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.sourcegear.com/CVS/Docs/blandy
- https://wroclaw.art.pl/~ser/docs/cvs.html

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(60;62;,\
join '',('.', '/', 0..9, 'A'..'Z', 'a'..'z')[rand 64, rand 64]),\"\n\""
```

Take the output of this command, and send it with your proposed userid to `<cvsadmin@cvslist.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@cvs.linuxdoc.org:/cvsroot
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 `.cvsroot` 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 howto/YOUR−HOWTO.sgml
bash$ cvs get minihowto/YOURDOC.sgml
```

Other CVS repository notes

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.

CVS Files via web

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

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.

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: HOWTO−HOWTO.sgml,v 1.4 2000/06/12 20:49:54 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. For the time being, LDP submissions should still be sent to `<ldp-submit@lists.linuxdoc.org>`.

Distributing your documentation

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.

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.

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). A sample license is available in the Manifesto or at http://www.linuxdoc.org/COPYRIGHT.html. If you choose to use the 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 HOWTO−HOWTO requires notification to the author of any derivative works or translations. I also explicitly place any source code (aside from the SGML the HOWTO 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.

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>`.

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 7 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.

HOWTO maintenance

Now that you're a HOWTO author, you should maintain the document and update it when new versions of software are released. You should also respond to reasonable comments and questions from your readers. You don't have to help them all, especially if their question is already answered in your HOWTO. However, a good experience with the LDP from readers is one of our goals and a great way of increasing the popularity of Linux

FAQs about the LDP

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

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.

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

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.

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

Contact the author of the document, or the LDP coordinator at `<ldp-discuss@lists.linuxdoc.org>`and mention the problem and how you think it needs to be fixed.

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

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>`.