Howtos−with−LinuxDoc−mini−HOWTO

Table of Contents

Howtos−with−LinuxDoc−mini−HOWTO

David S. Lawyer

v0.01, March 2001

This is about how to write HOWTOs using the simple LinuxDoc markup. It's primarily for Linux Documentation Project authors (and future fledging authors who want to get started fast). If you want to use the more advanced DocBook markup (including XML) see the LDP Authoring Guide.

---

1. Introduction

1.1 Can you write a HOWTO ?

Do you have info about Linux that would be useful to others? Can you write clearly? Do you know how to use a word processor or editor? Do you want to help thousands of others and let them read what you write at no cost to them? Once you've written a document, are you willing to receive email suggestions from readers and selectively use this info for improving your HOWTO? Would you like to have your writings be available on hundreds of websites throughout the world? If you can answer yes to these, then you're encouraged to write something for the LDP. But be warned that all this may take more time than you expected.

1.2 Copyright (skip this if you're in a hurry)

1.3 Why I wrote this

Why did I write this when there is already an "LDP Authoring Guide"? Well, the LDP guide is a long and detailed work. If you want to get started quickly, you need something much simpler and shorter. Furthermore the LDP guide fails to even mention the simplicity of LinuxDoc. Need I say more?

Thanks to Matt Welsh for his example.sgml file which I used as a major source of info for the example sections.

---

2. The Format of HOWTOs

2.1 Introduction

Our HOWTOs are released in various formats: Plain Text, HTML, PostScript, and PDF. Instead of having to write the same HOWTO in all of these formats we write just one HOWTO in a source format which gets converted by computer into all of the others.

To get an idea of what this format looks like, take a look at the source file of a webpage (if you haven't already). You will see all sorts of words in <angle brackets>. These are called tags. These webpages (tags and all) are html: *Hypertext Markup Language*. The LDP uses something like this for its documents.

The languages LDP uses meet the requirements of the *Standard Generalized Markup Language* or *SGML* for short. The LDP now uses the following two flavors of sgml: *LinuxDoc* and *DocBook*. Originally it only used LinuxDoc (which was sometimes incorrectly just called sgml). Interestingly, it turns out that html is still another flavor of sgml. Now some people have gone from the DocBook flavor of sgml to the DocBook flavor of the XML standard.

This mini−HOWTO is all about using the simple LinuxDoc flavor of sgml. You may call it "LinuxDoc markup". It can be machine converted to html, plain text, postscript, pdf, and DocBook. It's a lot easier than html or DocBook and you don't need a special editor for it as it's easy to type in the tags (or use macros for them).

---

3. Comparing LinuxDoc to DocBook

One way to do this is to go to the LDP site http://www.linuxdoc.org click on HOWTOs and then compare the sources of the same HOWTO in the two formats: LinuxDoc and DocBook. The DocBook tags are often longer than the equivalent LinuxDoc tags and there are sometimes more of them needed to do the same task. DocBook uses <para> and </para> tags to enclose each paragraph while LinuxDoc uses only a blank line to separate paragraphs (no tags). To emphasize the word "release" the comparison is:

```
<emphasis>release</emphasis> release
<em>release</em>
```

Yes, you have to type release twice in DocBook. For a section start:

```
<sect>
<title>Introduction<title>

<sect> Introduction
```

So there's much more to type with DocBook if you're typing in tags manually. But DocBook has all sorts of tags that don't exist in LinuxDoc so it's more advanced. Just using a subset of DocBook doesn't help as you can see from the above examples. There is still a lot more tag clutter. With a more and longer tags the document becomes harder to read unless you use an editor that hides them. But hiding them has it's drawbacks since it's nice to see what tags you've used.

Still, the number of people who use DocBook greatly exceeds the number using LinuxDoc. But if you do decide to migrate to DocBook there's a program by Reuben Thomas (ld2db) which can help make the conversion. It's not 100% perfect and you may have to do some manual editing. The LDP also automatically converts a LinuxDoc HOWTO to DocBook after you submit it.

---

4. Learning LinuxDoc

4.1 Introduction

LinuxDoc is a lot easier to learn than DocBook. But most of what you learn about LinuxDoc would also be useful for DocBook. So if you eventually decide to go for DocBook, most of the effort spent on learning LinuxDoc will not be wasted.

One way to learn it is by examples. I've written 3 example files ranging from easy to intermediate. Start with example1. These files are included here as sections. To turn them into individual files you may cut them out and write them to files. Then you could try turning one into text by using say "sgml2txt −f example2.sgml" to see what it looks like. Make sure the file names end in .sgml.

If you want to look at some real examples you can just go to an LDP mirror site, find the HOWTOs and select LinuxDoc SGML. Or go to the main site directly: http://www.ibiblio.org/pub/Linux/docs/HOWTO/other−formats/sgml Now for the first simple example.

4.2 Example 1

```
<!doctype linuxdoc system>
<article>
<title>First Example (example1)
<author>David S.Lawyer

<sect> Introduction
<p> This is a very simple example of "source" for the LinuxDoc text
formatting system.  This paragraph begins with a paragraph tag (a "p"
enclosed in angle brackets).  Notice that there are other tags, also
enclosed in angle brackets.  If you don't see any tags, then you are
reading a converted file so find the source file: example1.sgml (which
contains the tags).

This is the next paragraph.  Note that it is separated from the above
paragraph by just a blank line.  Thus it needs no "p" tag in front of
it.  The "p" tag is only needed for the first paragraph of a section
(just after the sect-tag).  The file suffix: sgml stands for Standard
Generalized Markup Language.  You are now reading the LinuxDoc flavor
of sgml as specified in the very first line of this file.

<sect> Tags
<p> Tags are anything inside angle brackets.  The "sect" tag above
marks the start of a new section of this example document.
"Introduction" was the first section and you are now reading the
second section titled "Tags".  If this were a long document (like a
book), a section would correspond to a chapter.

Note that there are "article", "title" and "author" tags at the start
of this article.  At the end of this article is an "/article" tag
marking the end of this article.  Thus there is a pair of "article"
tags, the first being the start tag and the second being the end tag.
Thus this entire article is enclosed in "article" tags.  In later
examples you'll see that there are other tags that come in pairs like
this.   They affect whatever is between the pairs (start tag and end
tag).  Any tag name which has "/" just before it is an "end tag".

When this source code is converted to another format (such as plain
text using the program sgml2txt) the tags are removed.  Tags only help
the sgml2txt program make the conversion.  There are more tags to
learn.  So once you understand this example1, please go on to the next
example: example2.  You don't need to actually memorize the tags, as
they will be repeated (but with little or no explanation) in later
examples.
</article>
```

4.3 Example 2

```
<!-- This is a comment.  It's ignored when this source file gets
converted to other formats.  -->
<!-- The next required tag implies that this file is in LinuxDoc format -->
<!doctype linuxdoc system>

<article>

<title>Second Example (example2)
<author>David S. Lawyer
<date>v1.0, July 2000

<abstract>
This is the abstract.  This document is the second example of using
the Linuxdoc-SGML flavor of sgml.  It's more complex than the first
example (example1.sgml) but simpler than the third example
(example3.sgml).  After you digest this you'll be able to write a
simple HOWTO using LinuxDoc.  End of the abstract.
</abstract>

<!-- The "toc" = Table of Contents.   It will be created here. -->
<toc>

<!-- Begin the main part of the article (or document) here.  The part
above this is sort of a long header. -->

<sect>This Second Example (example2.sgml)

<p>Unless you're familiar with markup languages, you should first
read example1.sgml.  You may want to run these example files thru a
translator such as sgml2txt to convert them to text and notice how the
result looks different than this "source" document with all its tags.

<sect>Article Layout
<sect1>Document Header
<p> One way to create a header part is just to copy them from another
.sgml file.  Then replace everything except the tags with the correct
info for your document.  This is like using a "template".

<sect1> Document Body
<p>
After the header comes the body of the document, consisting of nested
sections marked by sect-tags.  Subsections are marked by sect1-tags.
Since this is the first subsection within the 2nd main section, it's
actually section 2.1.  Within a subsection there may be
sub-subsections marked by sect2-tags, etc.  For a sub-sub-sub-section
use "sect3".  There are even such tags as sect4 and sect5, but you are
unlikely to need them.

<sect2> This is a sub-sub-section
<p>
It's 2.2.1.  Note that a "p" tag may be on a line by itself.  This
doesn't change a thing in the resulting documents.

<sect> More Features in example3
<p> With the tags in this example2 you can write a simple short document
a few pages long.  But for longer documents or for other important
features such as putting links into documents, you need to study the
next example: example3.  It will show you how to create lists and
fonts.
```

```
</article>
```

4.4 Example 3

```
<!doctype linuxdoc system>
<!-- Note the mailto: after my name.  This allows the reader to click
on it to send me email -->

<article>
<title>Third Example (example3)
<author>David S. Lawyer <url url="mailto:dave@lafn.org">
<date>v1.0, July 2000
<abstract>
This document is the third example of using the LinuxDoc flavor of sgml.
It's more complex than the second example.
</abstract>
<!-- Comment: toc = Table of Contents  -->
<toc>

<sect> Fonts
<p>
While they will not show up in a plain text output, they will work
for other conversions.
<bf>boldface font</bf>   <em>emphasis font</em>     <sf>sans serif</sf>
<sl>slanted font</sl>    <tt>typewriter font</tt>   <it>italics font</it>
There's another way to get these same fonts by enclosing
the text in slashes like this:
<bf/boldface font/    <em/emphasis font/      <sf/sans serif/
<sl/slanted font/     <tt/typewriter font/    <it/italics font/

<sect> Links <label id="links_">
<p> You may create links (something that html browsers may click on to
go somewhere else).  They might just go to another part of this
document (cross-references) or they could get to a website on the
Internet.

<sect1> Cross-References
<p> If you click on <ref id="links_" name="Links"> you will be taken to
the start of the "Links" section above (which is labeled links_).
The label id may be any word you choose but it's a good idea to avoid
common words so that you can search for unique labels using your
editor.  That's why I use links_ (with the underline).  The name of
this link will be shown (in html format) as the name to click on.
This name (Links) will also be present in the text rendition.

<sect1> URL Links
<p> If you click on <url url="http://www.linuxdoc.org"> you will get
to the Linux Documentation Project website.  The next link adds a name
which people will click on: <url url="http://www.linuxdoc.org"
name="Linux Documentation Project">.  Using this second method, you may
not even need to explain where the link leads to since it's obvious by
the name.

<sect> Prohibited Characters
<p> Anything you type between angle bracket will be interpreted as a
tag.  But what if you want to display a tag in a document?  For this
you use a code word for the angle characters.

You may use &lt for < and &gt for >.  lt = Less Than, gt =
Greater Than.  For example, here's a p-tag: &lt;p&gt;.  Of
course it doesn't actually start any paragraph, but it will appear in
```

```
the converted document as <p>.  These codes all start with an &
character.  The ; after the lt is to separate it.  It's not needed if
there is a space after it.  For example: 3 &lt 4.  Actually, if
you know that its OK to use an unpaired > then you could have written
<p> as &lt;p>.  This will not be mistakenly recognized as a tag
since there is no opening <.

There are other characters that you can't put into the document text
directly.  For & in an AT modem command use: AT&amp;.  If other
characters cause you trouble see example3 or the "guide" that comes
with linuxdoc-tools or sgml-tools.

<sect> Verbatim, Code & Newline
<sect1> Verbatim
<p> If you want to insure that lines are not joined together during
conversion to other formats, use verbatim (verb).  Some things still
get recognized even though they are between verbatim tags.  This
includes the macros starting with & and end tags with /.

<tscreen><verb>
% sgml2txt -f example.sgml
</verb></tscreen>
The "tscreen" sets the font to typewriter and indents it nicely.

<sect1> Code
<p> This encloses computer code between two dashed lines.
<tscreen><code>
Put computer source code here
</code></tscreen>

<sect1> Newline
<p> To force a newline use <newline>
This sentence always starts at the left margin.

<sect>Lists
<p>
This puts items into a list with a marker at the start of each item.
They start with the "itemize" tag.
<itemize>
<item> This is the first item in a list.
<item> This is the second item
       <itemize>
       <item> Multiple levels (nesting) are supported.
       <item> The second item in this sublist
       </itemize>
       <enum>
       <item> Enumerated lists using <tt/enum/ also work.
       <item> This is item number 2
       </enum>
<item> The final item in the main list
</itemize>
</article>
```

4.5 LinuxDoc Quick Reference Sheet

Header Part

```
<!doctype linuxdoc system>
<article>
<title>Quick Reference Sheet
```

```
<author>David S. Lawyer
<date>v1.0, July 2000
<abstract> abstract here </abstract>
<toc> <!-- Comment: toc = Table of Contents  -->
```

Body Layout

```
<sect> Chapter 1            Note: Put a <p> on the first line of
<sect1> Subsection 1.1      each section (or subsection, etc.)
<sect1> Subsection 1.2
<sect> Chapter 2            Choose title names to replace "Chapter"
<sect1> Subsection 2.1      "Subsection", etc.
<sect2> Sub-subsection 2.1.1
<sect2> Sub-subsection 2.1.2
<sect1> Subsection 2.2
</article>
```

Fonts

There are two ways to get these:

```
<bf>boldface font</bf>   <em>emphasis font</em>     <sf>sans serif font</sf>
<sl>slanted font</sl>    <tt>typewriter font</tt>   <it>italics font</it>
<bf/boldface/    <em/emphasis/     <sf/sans serif/
<sl/slanted/     <tt/typewriter/   <it/italics/
```

Lists (nesting is OK)

```
Ordinary unnumbered list:               Numbered list:
<itemize>                               <enum>
<item> First item                       <item> First item
<item> Second item                      <item> Second item
<item> etc.                             <item> etc.
</itemize>                              </enum>
```

Links

```
   Cross-References:                            An Email Link:
<ref id="links_" name="Links">          <url url="mailto:bob@linuxdoc.org">
```

Newline, Verbatim, URLs

```
To force a newline <newline>
<tscreen><verb>
<url url="http://www.linuxdoc.org">
<url url="http://www.linuxdoc.org" name="Linux Documentation Project">.
</verb></tscreen>
```

Character Codes (macros).

- Use `&` for the ampersand (&),
- Use `<` for a left bracket (<),
- Use `>` for a right bracket (>),
- Use `&etago;` for a left bracket with a slash (</)
- Use `$` for a dollar sign ($),

- Use `#` for a hash (#),
- Use `%` for a percent (%),
- Use `˜` for a tilde (~),
- Use ` `` ` and `''` for quotes, or use `&dquot;` for ".
- Use `` for a soft hyphen (that is, an indication that this is a good place to break a word for horizontal justification).

---

5. Getting/Using the LinuxDoc Software

You could write a LinuxDoc document without having any LinuxDoc software. However it's likely that it would contain some errors in the tags (or their use) so that it would be returned to you for correction. Even if there were no errors, the results likely would not look quite right. So it's best for you to have the software to convert your source code on your computer.

The Debian distribution of Linux has a linuxdoc−tools package. There is also a rpm package for non−Debian distributions. It was formerly called sgml−tools and version 1.0.9 (the final one) may still be useful if you can't get linuxdoc−tools. Don't use the sgmltools−2 package which is primarily for DocBook.

To use it, you run converter programs on the *.sgml files. For example, type: "sgml2txt −f my−HOWTO.sgml". To get html type: "sgml2html my−HOWTO.sgml". If it shows errors, it will show the line number and the column number where the error is in the source file. Typing "man −k sgml" should show you a number of other programs with a one−line description of each.

---

6. Writing the HOWTO

6.1 Before you start writing

First email the HOWTO coordinator at mailto:linux−howto@metalab.unc.edu or mailto:feedback@linuxdoc.org. If you're taking over an unmaintained HOWTO, contact the former author. This may required by the copyright−license but you should do it out of courtesy even it it's not required.

6.2 Guidelines

These are mostly by Tim Bynum (a former HOWTO coordinator).

- Be sure and use an accepted format (such as LinuxDoc :−).
- Try to use meaningful structure and organization, and write clearly. Remember that many of the people reading HOWTOs do not speak English as their first language.
- Make sure that all of the information is correct. I can't stress this enough. When in doubt, speculate, but make it clear that you're only guessing. I use ?? if I'm not sure.
- Make sure that you are covering the most recent version of the available software.
- Consider including a "FAQ" section. It might also be called "Common Problems" or "Trouble Shooting".
- Be sure to copyright it in your name and include a license which meets the requirements stated in the LDP manifesto.
- Use the standard header with title, author, and date (including a version number). See Example 3

- Lastly, be prepared to receive email questions and comments from readers. How much you help people is up to you but you should make use of good suggestions and reports of errors. You may also get some "thank you" email as well as well as mail from people asking for help who never even looked at your HOWTO.

6.3 Submitting the HOWTO, etc.

After you have written the HOWTO, email the SGML source to submit@linuxdoc.org. Then all you need to do is to keep the HOWTO up−to date by submitting periodic updates to the same email as you used for the first edition.

---

7. More Information

There's a forthcoming HOWTO about LinuxDoc that covers it in much greater detail than this mini−HOWTO. To be released sometime in April 2001 ??
