

## **Linuxdoc Reference**

# Table of Contents

|   |          |
|---|----------|
| <b>Linuxdoc Reference</b>                   | <b>1</b> |
| <u>A introduction to the linuxdoc dtd</u>   | 1        |
| <u>Uwe Böhme, &lt;uwe@hof.baynet.de&gt;</u> | 1        |
| <u>Appendix</u>                             | 1        |
| <u>1. Making of</u>                         | 1        |
| <u>1.1 Legal stuff</u>                      | 1        |
| <u>1.2 Genesis</u>                          | 1        |
| <u>2. Introduction</u>                      | 2        |
| <u>3. A minimalistic document</u>           | 2        |
| <u>3.1 Step By Step</u>                     | 3        |
| <u>3.2 A Startup Document</u>               | 3        |
| <u>4. Document Classes</u>                  | 4        |
| <u>4.1 Article Tag</u>                      | 5        |
| <u>Titlepage Tag</u>                        | 5        |
| <u>Title Tag</u>                            | 5        |
| <u>Author Tag</u>                           | 6        |
| <u>Date Tag</u>                             | 6        |
| <u>Abstract Tag</u>                         | 6        |
| <u>Header Tag</u>                           | 6        |
| <u>Table Of Contents Tag</u>                | 7        |
| <u>List Of Figures Tag</u>                  | 7        |
| <u>List Of Tables Tag</u>                   | 7        |
| <u>Body</u>                                 | 7        |
| <u>Appendix Tag</u>                         | 7        |
| <u>Bibliography Tag</u>                     | 7        |
| <u>Footnote Tag</u>                         | 7        |
| <u>4.2 Report Tag</u>                       | 8        |
| <u>4.3 Book Tag</u>                         | 8        |
| <u>4.4 Letter Tag</u>                       | 8        |
| <u>4.5 Telefax Tag</u>                      | 9        |
| <u>4.6 Slides Tag</u>                       | 9        |
| <u>Slide Tag</u>                            | 9        |
| <u>4.7 Note Tag</u>                         | 9        |
| <u>4.8 Manual Page Tag</u>                  | 10       |
| <u>5. Inlines</u>                           | 10       |
| <u>6. Sectioning</u>                        | 10       |
| <u>7. Paragraphs</u>                        | 11       |
| <u>7.1 Normal Paragraph</u>                 | 12       |
| <u>Paragraph tag</u>                        | 12       |
| <u>Empty Newline</u>                        | 12       |
| <u>7.2 List-like Paragraphs</u>             | 12       |
| <u>List Tag</u>                             | 12       |
| <u>Itemize Tag</u>                          | 13       |
| <u>Enum Tag</u>                             | 13       |
| <u>Descrip Tag</u>                          | 13       |
| <u>7.3 Figures and Tables</u>               | 14       |
| <u>Table Tag</u>                            | 15       |
| <u>Figure Tag</u>                           | 15       |

# Table of Contents

## **Linuxdoc Reference**

|  |    |
|--|----|
| <u>Encapsulated Postscript(TM) Tag</u> .....         | 15 |
| <u>Placeholder Tag</u> .....                         | 16 |
| <u>7.4 Tabular Tag</u> .....                         | 16 |
| <u>7.5 Mathematical Paragraph</u> .....              | 18 |
| <u>Displayed Formula Tag</u> .....                   | 18 |
| <u>Equation Tag</u> .....                            | 18 |
| <u>7.6 Theorem Paragraph</u> .....                   | 19 |
| <u>7.7 Code and verbatim Paragraphs</u> .....        | 20 |
| <u>Code Tag</u> .....                                | 20 |
| <u>Verbatim Tag</u> .....                            | 21 |
| <u>8. Inline Tags</u> .....                          | 21 |
| <u>8.1 Emphasizes</u> .....                          | 21 |
| <u>8.2 Short-quote Tag</u> .....                     | 22 |
| <u>8.3 Formula Tag</u> .....                         | 22 |
| <u>8.4 External Tag</u> .....                        | 22 |
| <u>9. Mathematical Formulas</u> .....                | 22 |
| <u>9.1 Fraction Tag</u> .....                        | 23 |
| <u>9.2 Product, Integral and Summation Tag</u> ..... | 24 |
| <u>9.3 Limited Tag</u> .....                         | 24 |
| <u>9.4 Array Tag</u> .....                           | 24 |
| <u>9.5 Root Tag</u> .....                            | 25 |
| <u>9.6 Figure Tag</u> .....                          | 25 |
| <u>9.7 Realfont Tag</u> .....                        | 25 |
| <u>9.8 Other Mathematical Tags</u> .....             | 26 |
| <u>10. Labels and References</u> .....               | 26 |
| <u>10.1 Label Tag</u> .....                          | 26 |
| <u>10.2 Reference Tag</u> .....                      | 27 |
| <u>10.3 Page reference Tag</u> .....                 | 27 |
| <u>10.4 Url Tag</u> .....                            | 27 |
| <u>10.5 Htmlurl Tag</u> .....                        | 27 |
| <u>10.6 Cite Tag</u> .....                           | 28 |
| <u>10.7 Ncite Tag</u> .....                          | 28 |
| <u>11. Indices</u> .....                             | 28 |
| <u>11.1 Including a index</u> .....                  | 29 |
| <u>Manually</u> .....                                | 29 |
| <u>Hacked</u> .....                                  | 29 |
| <u>12. Literate Programming</u> .....                | 29 |
| <u>13. Reference</u> .....                           | 30 |
| <u>14. Named Symbols</u> .....                       | 30 |
| <u>14.1 Named Characters</u> .....                   | 30 |
| <u>14.2 Named Whitespaces</u> .....                  | 31 |
| <u>15. Mathematical Figures</u> .....                | 31 |
| <u>16. Linuxdoc dtd Source</u> .....                 | 31 |

# Linuxdoc Reference

## A introduction to the linuxdoc dtd

Uwe Böhme, <uwe@hof.baynet.de>

v1.1, 30 January 2000

---

*This article is intended to be a reference for the SGML document type definition linuxdoc, which is coming along with the SGML text formatting system version 1.0. It should also be applicable to future versions which may be found at [My Homepage](#).*

---

## Appendix

### 1. Making of

#### 1.1 Legal stuff

Copyright © 1997-2000 by Uwe Böhme. This document may be distributed under the terms set forth in the Linux Documentation Project License at [LDP](#). Please contact the authors if you are unable to get the license. This is free documentation. It is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose.

This document is not part of *ldp* (even if I took their form of license). I'm not yet playing in that league.

#### 1.2 Genesis

This document was born trying to learn more about writing texts on my linux system. The one system looking like suitable to my needs was sgml-tools [SGML-Tools Organisation](#) an the linuxdoc dtd.

In [*SGML-Tools User's Guide 1.0 (\$Revision\$)*] (see section [Reference](#)) the overall structure is described nice and easy. Also [*Quick SGML Example, v1.0*] (see section [Reference](#)) was helpful, **but**:

A lot of features are not mentioned.

On the way to learn more about it, I met [*The qwertz Document Type Definition*] (see section [Reference](#)). It's as detailed as hoped, but it's not made for the linuxdoc dtd (even if linuxdoc is based on qwertz).

I tried a new approach: Look at the dtd

dtd = document type definition

file itself, and try to understand it.

As time went by I noticed that I also forgot about some stuff, or - at least - didn't point it out strong enough. This will change within the next revision.

Any feedback you might have is welcome (especially help with English spelling or grammar) by e-mail at [Uwe Böhme](#).

## 2. Introduction

The principle of any sgml'ed document (linuxdoc, docbook, html) is more or less the same:

Don't write how it should *look like*, but write what it *is*.

This is a different approach than the standard "wysiwyg"

What you see is what you (should) get (if you are a very lucky one and your computer wins the war against buggy software)

one

You might want to call it `wysiwym`, i.e. "What you see is what you mean"

. You do not tell the program that this line should be in a bigger font, **to look like** a headline. What you do is telling that this line **is** a headline. You do not try to make your document **look like** a report, but you tag it **to be** a report. So you *tag* the text with the appropriate `<tag>`.

The big advantages of this approach are:

1. You do not need to mess around with fontsetting, line gaps or anything directly connected to the layout.
2. You describe your document in a more abstract way so it's more reusable and can be mapped to different media types.

If you ever tried to reuse a document written in a specialized wysiwy layout for html then you know what I'm talking about.

In addition in all sgml-style documents you will find named symbols This is a concept to expand the charset of the document and to avoid inconsistencies in decision of the parser, how to interpret or map some special characters.

How should the parser know whether a `<` character is starting a tag or should be printed directly. This is solved by the named character *lt*. If you write `&lt;` this one will result to `<` in your text. For a list of the named symbols see [Named Symbols](#).

### Hint for the new user

It might be a good idea, to download this document not only as a dvi or ps document, but also to download the sgml source. This offers you the chance to look into the sources, if you find something within this article, which might fit your needs.

## 3. A minimalistic document

In this section you'll find what you'll need for a minimalistic linuxdoc dtd conform document. It's intended to give a first touch. Skip this section, if you already know the principles.

## 3.1 Step By Step

The steps you have to do to create a nice linuxdoc document and map it to the form you need are:

- Take a plain text editor of your choice.
- Create a file and name it (or later save it as) e.g. `start.sgml`.
- Type the document
- Save the file and close your editor.
- Run the checker by typing `sgmlcheck start.sgml`.
- If you get errors reported, reopen your document in your editor again and try to correct it

The error messages of `sgmlcheck` will give you a hint about the type of error and also line and column where it occurred.

- . Run the checker again until no more errors occur.
- Now you have to decide what's your document for. Take the appropriate parser mapper combination and translate your document. To find the mappers available in the SGML-Tools see table [SGML-Tools mappers for sgml documents](#).

| type  | to produce                                 |
|---|--|
| <code>sgml2html start.sgml</code>               | Hypertext markup language for web browsers |
| <code>sgml2lyx start.sgml</code>                | Lyx or KLyx wysiwym textformat             |
| <code>sgml2info start.sgml</code>               | Info page for UN*X info                    |
| <code>sgml2latex start.sgml</code>              | DVI output                                 |
| <code>sgml2latex --output=tex start.sgml</code> | pure tex output                            |
| <code>sgml2latex --output=ps start.sgml</code>  | postscript output                          |
| <code>sgml2rtf start.sgml</code>                | rich text format                           |
| <code>sgml2txt start.sgml</code>                | pure text                                  |

SGML-Tools mappers for sgml documents

## 3.2 A Startup Document

We start with a simple document (the numbers and colon in the beginning of the line are for explanation, don't type it!):

---

```

1: <!doctype linuxdoc system>
2: <notes>
3: <title>A Small Linuxdoc Example</title>
4: <p>Hello <em>world</em>.</p>
5: <p><bf>Here</bf> we are.</p>
6: </notes>
```

---

Now we take a look at the single lines:

1. A linuxdoc document has to start, like all SGML conform documents, with the *preamble*. If you like you can take it as a piece of necessary magic, or you can try to find more information about SGML. The preamble is indicating to the SGML-parser, which dtd (document type definition) it should use for checking the syntax of the document.
2. Open the *document class*: You have to decide, wich type of document you want to write. See section [Document Classes](#) for detailed description about that *document classes*. The necessary header information, wich is depending on the *document class* is also explained there. In our case we place a `<notes>` tag forming a note, wich is indicating a simple unstructured document.
3. Even if optional it's a good idea to give a *title* to the document. That's done with the `<title>` tag.
4. A paragraph marked by the `<p>` tag, containing the word `world` wich is *inline emphasized* by the `<em>` tag.
5. Another completely tagged paragraph, with another word *inline boldfaced* by the `<bf>` tag.
6. Here we close the open *document class* tag.

The same example may be written a little bit shorter, by leaving out tags which are placed automatically by the parser, and by using shortened tags:

---

```

1: <!doctype linuxdoc system>
2: <notes>
3: <title>A Small Linuxdoc Example
4: <p>Hello <em/world/.
5:
6: <bf/Here/ we are.
7: </notes>

```

---

Now we look at the single lines again:

1. The *preamble*.
2. The document class (also unchanged).
3. The *title*. It's not closed, because the `p` tag in the next line is implicitly closing it.
4. The paragraph is implicitly closing the *title*. The *emphasize* tag is noted in short form. The short notation you can use only if your tagged text doesn't contain a litteral `/`. The *paragraph* is not explicitly closed in this line.
5. The empty line here is the reason, why you don't need to close the previous *paragraph* and don't need to open the next one. A empty line is interpreted as a end of the current paragraph and the start of a new one.
6. Another paragraph (not opened directly), with another short *inline* tag.
7. Closing the open *document class* tag, wich is implicitly also closing the still open paragraph.

Maybe now it's a little bit more clear, who you have to work with tags.

## 4. Document Classes

---

```

<!element linuxdoc o o
    (sect | chapt | article | report |
     book | letter | telefax | slides | notes | manpage ) >

```

---

This is describing the overall class of the document, so naturally it has (leave alone the doctype definition) to be the first tag enclosing your whole document. Some of the tags namely the `sect` and `chapt` (see section [Sectioning Tags](#)) doesn't make any sense taken them standalone despite being included as part of more complete classed document, so we'll describe them later as a part of the other document classes. Decide first

which of the top mentioned document classes fits the type of the document you want to write best.

To find a detailed description of the document classes see table [Document classes](#).

| Chapter            | Class tag |
|--------------------|-----------|
| <u>Article Tag</u> | <article> |
| <u>Report Tag</u>  | <report>  |
| <u>Book Tag</u>    | <book>    |
| <u>Letter Tag</u>  | <letter>  |
| <u>Telefax Tag</u> | <telefax> |
| <u>Slides Tag</u>  | <slides>  |
| <u>Notes Tag</u>   | <notes>   |
| <u>Manpage Tag</u> | <manpage> |

Document classes

To me the *article class* is the most important one. That's the reason why it's described first and most detailed.

## 4.1 Article Tag

---

```
<!element article - -
  (titlepag, header?,
   toc?, lof?, lot?, p*, sect*,
   (appendix, sect+)?, biblio?) +(footnote)>

<!attlist article
  opts cdata "null">
```

---

You can see that the *article* needs some tags included. They will be explained in consequence.

The *options* attribute (*opts*) takes a comma separated list with thy different style (LaTeX *.sty*) sheets to include within the document.

## Titlepage Tag

---

```
<!element titlepag o o (title, author, date?, abstract?)>
```

---

The *Titlepage* Tag (*titlepag*) is implicitly placed as soon a you started your *document class*. You don't need to write it explicitly. Anyway you have to note it's mandatory tags. It's purpouse is to describe the layout and elements of the titlepages.

## Title Tag

---

```
<!element title - o (%inline, subtitle?) +(newline)>
```

---



Each *document class* which owns a titlepage of course needs a *title*, which is noted down with a `<title>` tag. You don't need to close that one. A title may contain a *subtitle* started by the `<subtitle>` tag.

If you look at the headerpage of this document you'll find it to be mapped from the tags:

```
<title>Linuxdoc Reference
<subtitle>A introduction to the linuxdoc dtd
```

## Author Tag

---

```
<!element author - o (name, thanks?, inst?,
                      (and, name, thanks?, inst?)*)>
```

---

Usually you place the (your) name here. People should know who wrote the document, so you place a `<author>` tag. If you don't note the `name` tag it's implicitly placed. The *author* has also optional items which can be tagged within the `author` tag.

If you want to say thanks to anyone (might be somebody providing useful information) you place it within the `<thanks>` tag. Next, if your writing is done in your position of an *institution* staff member, place it within the `<inst>` tag.

The `<and>` tag is starting the whole story again, as if there would be a second `author` tag would have been started. Clearly this one is for coauthors.

## Date Tag

If you want to mark your document with a *date*, you can do that with the `<date>` tag.

It's not checked whether you really place a valid date here, but don't abuse it.

## Abstract Tag

This tag is intended for an *abstract* description of your document. Don't mix the `<abstract>` tag with an *introduction* which is likely to be placed inside the first *section* of your document (see section [Sectioning](#)).

## Header Tag

---

```
<!element header - - (lhead, rhead) >
<!element lhead - o (%inline)>
<!element rhead - o (%inline)>
```

---

A `<header>` tag specifies what should be printed at the top of each page. It consists of a *left heading* i.e. `<lhead>` and a *right heading* i.e. `<rhead>`). Both elements are required, if a heading is used at all, but either may be left empty, so that the effect of having only a left or right heading can be achieved easily enough.

As we will see, an initial header can be given after the title page. Afterwards, a new header can be given for each new chapter or section. The header printed on a page is the one which is in effect at the end of the current page. So that the header will be that of the last section starting on the page.

## Table Of Contents Tag

If you place the `<toc>` tag, a *table of contense* will be generated, by looking the section heading, and adding references.

In a hyperref document, this might be hyperrefs, in a LaTeX document you will come to see the pagenumbers.

Only the sections major to the `sect3` will be included.

## List Of Figures Tag

If you place the `<lof>` tag, a *list of figures* will be generated, by looking the captions of the figures, and adding references.

## List Of Tables Tag

If you place the `<lot>` tag, a *list of tables* will be generated, by looking the captions of the tables, and adding references.

## Body

Here you place various sections according section Sectioning. There is no *body tag*. The body starts with the first *chapter*, *section* or *paragraph*.

## Appendix Tag

In the end of the article you can place the `<appendix>` tag

Really you shouldn't think about people (e.g. m.d.s knifing your belly here.

, wich starts a area of appended sections. The `appendix` tag implies a different section numbering type to the following section tags.

## Bibliography Tag

It's intended to gather all the `<cites>` and `<ncites>` you used within your document. The `<biblio>` tag will be replaced by a *bibliography* according the mapping type of the document, maybe by hyperrefs maybe by section numbers or anything wich might be useful.

Until now I've not been able to create a `.bbl` file, so I wasn't able to verify.

## Footnote Tag

A *footnote* may be place in any spot of your document. Exactly the spot in yout document where you are placing the `<footnote>` tag should be the one where the reference to the tagged text shuld be rendered. It should be used for additional information, wich is not necessary for understanding the primary purpose of yor document but might be usefull, interesting, or funny.

Whereas the last one is not always true, even if you try.

anywhere within the article.

## 4.2 Report Tag

---

```
<!element report - -
  (titlepag, header?, toc?, lof?, lot?, p*,
   chapt*, (appendix, chapt+)?, biblio?) +(footnote)>
```

---

The *report* is a document class with a chapter oriented approach. So within a document clasified by a `<report>` tag the toplevel is grouped by the `<chapt>` tag (see [Sectioning](#)). The rest of the structure is identical to the *article* class [Article Tag](#).

## 4.3 Book Tag

---

```
<!element book - -
  (titlepag, header?, toc?, lof?, lot?, p*, chapt*,
   (appendix, chapt+)?, biblio?) +(footnote) >
```

---

You will notice that the *book* element is identical to the *report* [Report Tag](#). So anything valid there is also valid if you classify your document with a `<book>` tag.

## 4.4 Letter Tag

---

```
<!entity % addr "(address?, email?, phone?, fax?)" >

<!element letter - -
  (from, %addr, to, %addr, cc?, subject?, sref?, rref?,
   rdate?, opening, p+, closing, encl?, ps?)>
```

---

Also the purpose of the *letter* document class should be quite self explaining. Place a `<letter>` tag if you want to write one.

The letter's tags ar described in table [Tags in a letter](#)

| tag     | mandatory | what's it          |
|---------|-----------|--------------------|
| from    | yes       | from sender        |
| address | no        | sender's address   |
| email   | no        | sender's email     |
| phone   | no        | sender's phone     |
| fax     | no        | sender's fax       |
| to      | yes       | receiver           |
| address | no        | receiver's address |
| email   | no        | receiver's email   |
| phone   | no        | receiver's phone   |

|            |     |                                |
|------------|-----|--------------------------------|
| fax        | no  | receiver's fax                 |
| cc         | no  | carbon copy                    |
| subject    | no  | letters subject                |
| sref       | no  | sender's reference             |
| rref       | no  | receiver's reference           |
| rdate      | no  | received date??                |
| opening    | yes | opening                        |
| paragraphs | yes | see <a href="#">Paragraphs</a> |
| closing    | yes | closing                        |
| encl       | no  | enclosure                      |
| ps         | no  | post scriptum                  |

Tags in a letter

## 4.5 Telefax Tag

---

```
<!element telefax - -
    (from, %addr, to, address, email?,
     phone?, fax, cc?, subject?,
     opening, p+, closing, ps?)>
```

---

Overall the structure is same to the *letter* class. The only difference is that with the `<telefax>` tag the receiver's `<fax>` tag becomes mandatory.

Should be obvious why.

## 4.6 Slides Tag

---

```
<!element slides - - (slide*) >
```

---

The *slides* class is intended for overhead slides and transparencies. So the structure of a document classified by a `<slides>` tag is a very simple one. It contains single slide(s) starts by a `<slide>` tag. Nothing else. If not explicitly written the first *slide* is started implicitly.

### Slide Tag

---

```
<!element slide - o (title?, p+) >
```

---

A `<slide>` tag is only allowed within the *slides* document class. A *slide* may contain:

A *title* (see section [The Title Tag](#)) and one or more *paragraphs* (see section [Paragraphs](#)). That's all.

## 4.7 Note Tag

---

```
<!element notes - - (title?, p+) >
```

Intended as a class for personal notes the structure is even more simplified than the *slides* document class (see [The Slide Tag](#)). After classifying a document with the `<notes>` tag only a *title* (see section [The Title Tag](#)) and one or more *paragraphs* (see section [Paragraphs](#)) are allowed.

## 4.8 Manual Page Tag

```
<!element manpage - - (sect1*)
    -(sect2 | f | %mathpar | figure | tabular |
      table | %xref | %thrm )>
```

This document class is intended for writing *manual pages*, fitting the need of the `man` programm. In a document classified by a `<manpage>` tag the toplevel section tag is the `sect1` tag (see section [Sectioning](#)), for easy pasting manual pages into an *article* or *book* document class. The exception here to the normal sectioning is, that there is only one subsection level allowed (`sect2`).

## 5. Inlines

```
<!entity % inline
    " (#pcdata | f| x| %emph; |sq| %xref | %index | file ) * " >
```

*Inlines* may occur anywhere within the text, and doesn't have any influence to the textflow or logical structure of the document.

### #pcdata

*Parsed character data* is just normal written text within the flow which may contain other inlines.

### f

Inline *mathematical formulas* according to the `maths.dtd`. See [The Formula Tag](#).

### x

The *external* tag which is bypassing the parser. Tagged data walks directly into the mapped file. See chapter [The External Tag](#) for detailed information.

### %emph;

*Emphasizes* of the text. See chapter [Emphasizes](#).

### sq

*Shortquotes* within the textflow. See chapter [The Short Quote Tag](#).

### %xref

*XReferences* within the text or external references. See chapter [Labels and References](#).

### %index

Again I can't explain this one. If you can, please mail.

### file

Again I can't explain this one (I only could guess about picture files in eps). If you can, please mail.

## 6. Sectioning

```
<!element chapt - o (%sect, sect*) +(footnote)>
<!element sect  - o (%sect, sect1*) +(footnote)>
<!element sect1 - o (%sect, sect2*)>
<!element sect2 - o (%sect, sect3*)>
<!element sect3 - o (%sect, sect4*)>
<!element sect4 - o (%sect)>
```

## The *sectioning*

Also the `chapt` tag is a *sectioning* tag.

is done by the according elements, forming the section tree. They are bringing the various paragraphs within our document to follow a nice tree. The top level tag and the allowed depth is varying with the *document class* (see section [The Document Class](#)).

The normal hierarchy is

```
chapt
  sect
    sect1
      sect2
        sect3
          sect4
```

Just take a book, look the table of contents and you will see.

Each of the tags out of the *sectionings* has nearly the same syntax. All of them owe a *heading*. The `heading` tag is placed implicitly if you don't note it down. Also the each of the sectioning tags may contain a `header` tag, changing the current document header (see section [The Header Tag](#)).

Within the you may place subordinate sections and *paragraphs* (see [Paragraphs](#)).

Some of the sectioning tags may only appear in special document classes ( [Document Classes](#)).

### Hint:

It's wise to place a *label* tag after the text of the *section* tag, even if you don't want to refer to the section [Labels and references](#). Later when your document grows you might want to.

## 7. Paragraphs

---

```
<!entity % sectpar
    " %par; | figure | tabular | table | %mathpar; |
      %thrm; | %litprog; ">

<!entity % par
    " %list; | comment | lq | quote | tscreen " >

<!entity % litprog " code | verb " >
```

---

Each of the here described tags form a paragraph.

For obvious reason a paragraph is normally

The behaviour of the exceptions `figure` and `tabular` are explained there.

starting and ending with a new line.

How else you would notice it's a paragraph ?

There are some tags, which always form a paragraph, and one way to form a paragraph implicitly. There are various types of paragraphs, because not every type of paragraph is allowed to appear in every document class in every place.

The different types of paragraphs are explained in the next sections. For more details about `%litprog`; see [Literate Programming](#).

## 7.1 Normal Paragraph

Normal paragraphs can be formed in two ways:

### Paragraph tag

The `<p>` tag is starting a new *paragraph*. This tag is mandatory if you want to finish a section header without explicitly closing the `<sect>` tag. In this case `<p>` tag then closes the `<sect>` tag automatically.

### Empty Newline

A empty line between two paragraph is implicitly starting a new *paragraph*. Take care within `descriptive` lists. There a empty `<tag>` tag will not be paragraphed by an empty line.

## 7.2 List-like Paragraphs

---

```
<!entity % list
    " list | itemize | enum | descrip " >
```

---

This four tags indicate the starting of a list-like paragraph. Within each of the lists the single items are separated by an *item* tag.

---

```
<!element item o o ((%inline; | %sectpar;)*, p*) >
```

---

As you can see, a item may again contain paragraphs (and therefore also may contain other lists - even of a different type).

### List Tag

---

```
<!element list - - (item+)>
```

---

The *list* tag will be mapped to a naked list without bullets, numbers or anything else.

To see it, I place a small example:

---

```
<list>
<item>A point
<item>Another one
<item>Last
</list>
```

---

Will look (depending on the mapping) like:

- A point
- Another one
- Last

## Itemize Tag

---

```
<!element itemize - - (item+)>
```

---

The *itemize* tag will be mapped to a list with bullets, which is usually place for lists where the order of the items is not important.

A small example:

---

```
<itemize>
<item>A point
<item>Another one
<item>Last
</itemize>
```

---

Will look (depending on the mapping) like:

- A point
- Another one
- Last

## Enum Tag

---

```
<!element enum - - (item+)>
```

---

The *enum* tag will be mapped to a list with numbers.

A small example:

---

```
<enum>
<item>A point
<item>Another one
<item>Last
</enum>
```

---

Will look (depending on the mapping) like:

1. A point
2. Another one
3. Last

## Descrip Tag

---

```
<!element descrip - - (tag?, p+)+ >
```

---

The *descrip* tag will be mapped to a descriptive list. The concept here is a little bit different than with the other types of lists mentioned above.



Here you place a *tag* (this time the tag's name is really literally `tag`) wich is described later on.

A small example:

---

```
<descrip>
<tag/sgml/structured general markup language.
<tag/html - hypertext markup language/
A sgml implementation.
It contains some concepts about linking information together in a very
convenient way.
This made it to be so successful and to become the standard for documents
published by the internet.
<tag/internet/A worldwide connected internet (internet here as a
technical term)
</descrip>
```

---

Will look (depending on the mapping) like:

### **sgml**

structured general markup language.

### **html - hypertext markup language**

A sgml implementation. It contains some concepts about linking information together in a very convenient way. This made it to be so successfull and to become the standard for documents published by the internet.

### **internet**

A worldwide connected internet (internet here as a technical term)

## 7.3 Figures and Tables

The `<figure>` and the `<table>` tags form very special paragraphs. Not always they stay within the normal textflow. Both of the tags can hold a `loc` (*location*) attribute wich is telling how to handle the flow of this special paragraph.

The value of the `loc` attribute is a string of up to four letters, where each letter declares a location at which the figure or table **may** appear, as described in table Table Locations.

|   |        |   |
|---|--------|---|
| h | here   | At the same location as in the SGML file        |
| t | top    | At the top of a page                            |
| b | bottom | At the bottom of a page                         |
| p | page   | On a separate page only with figures and tables |

Table Locations

The default value of the `loc` attribute is `top`.

## Table Tag

---

```
<!element table    - - (tabular, caption?) >
```

---

As you can see a *table* consists of the `<table>` tag itself, including a `<tabular>` tag and a optional `<caption>` tag.

The `<tabular>` tag may also be placed without a `<table>` tag so it is described in detail in it's own section (see [Tabular Tag](#)).

The *caption* is used also to place the entry for the *list of tables* if you stated one (see [The List Of Tables Tag](#)).

A short example will show how it's working together.

```
<table loc="ht">
<tabular ca="lcr">
Look|this|table@
Isn't|it|nice@
1.234|mixed|columns
</tabular>
<caption>A sample table
</table>
```

|       |       |         |
|-------|-------|---------|
| Look  | this  | table   |
| Isn't | it    | nice    |
| 1.234 | mixed | columns |

A sample table

The *caption* "A sample table" would be the name in the *list of tables*.

## Figure Tag

---

```
<!element figure - - ((eps | ph ), img*, caption?)>
```

---

The usage of the `<figure>` tag is equivalent to the `<table>` tag. Instead of the `<tabular>` tag you place either a `<eps>` or a `<ph>` tag.

## Encapsulated Postscript(TM) Tag

---

```
<!attlist eps
    file cdata #required
    height cdata "5cm"
    angle cdata "0">
```

---

The `<eps>` tag is intended for including a external file in *encapsulated postscript(TM)* format into the document.

The attributes of the `<eps>` tag are:

#### file

The `file` attribute needs the *file name* of a encapsulated postscript(TM) file ending with a `.ps` suffix. The mandatory `.ps` suffix must not be written.

#### height

The *height* of the space the file is zoomed to. If you don't specify it defaults to 5cm. Take care that there's no space between the number and the length unit (`i`, `cm`).

#### angle

The *angle* is given in normal degrees (0-360) and as the number is increasing the file is rotated counter clockwise.

A example:

```
<figure loc="here">
<eps file="logo" height="4cm" angle="15">

<caption>A included encapsulated postscript&trade;
</figure>
```

The `img` tag is ignored by LaTeX-mapping and useful for html, 'cause most browsers don't know about eps.

A included encapsulated postscript(TM) file.

The *caption* here would go to the *list of figures* as described in section [The List Of Figures Tag](#).

## Placeholder Tag

---

```
<!attlist ph
        vspace cdata #required>
```

---

This tag doesn't place anything but keeps a clean space for good old manual picture pasting. The space kept free is destined by the `vspace` attribute. **Caveat:** The numerical argument for the `vspace` attribute needs a unit directly behind the number. Don't leave a space there (same as for the `height` attribute in [Encapsulated Postscript\(TM\) Tag](#)).

```
<figure loc="ht">
<ph vspace="5cm">
<caption>A blank space.
</figure>
```

Results to:

A blank space for gluing a photo

At this point you might want to look for your scissors and the glue.

## 7.4 Tabular Tag

---

```
<!element tabular - -
        (hline?, %tabrow, (rowsep, hline?, %tabrow)*, caption?) >
```

---

The `<tabular>` tag is interpreted as an own paragraph, if it is written standalone. Together with a `<table>` tag it gets part of the paragraph of the `<table>` tag (see [Table tag](#)).

Within the `tabular` tag you have rows and columns which are separating the text. You have to have at least one column and one row.

Wouldn't be very useful otherwise.

The `<tabular>` tag has a mandatory `ca` attribute for *column allignment*. The column allignment holds a single character for each column in their order from left to right. The characters you may place per column described in table [Columns allignements](#)

| char | alignment |
|------|-----------|
| l    | left      |
| c    | centered  |
| r    | right     |

Column alignments

In theory you should be able to place a `|` into the `ca` attribute for drawing a horizontal line for separating two columns. The problem: It doesn't work. The parser accepts it nicely, only the LaTeX output will map `|` to `{${|}$}` which is of course the set for four columns with invalid column allignment for all four columns. I'll try to figure out what to do about it.

The columns within the `<tabular>` tag are separated by a *column separator*, the `<colsep>` tag. The character `|` is translated to `<colsep>` so you can also place that one instead

Less typing, more fun.

.

What's valid for columns is also valid for rows. You separate the by a *row separator*, the `<rowsep>` tag. The character `@` is translated to `<rowsep>`.

Optional you can place a *horizontal line* with the `<hline>` tag. Take care with that one: The SGML tools will parse it nicely whether you place it in front of the row you want under the line, or behind the end of the row you want over it. But the only place to write it without causing the parser to shout "error" is to write it directly and without space or newline behind the row separator.

```
<tabular ca="lcr">
Look|this|table@<hline>
Isn't|it|nice@
1.234|mixed|columns@
</tabular>
```

Results in table [Sample table for tabular tag](#)

|       |       |         |
|-------|-------|---------|
| Look  | this  | table   |
| Isn't | it    | nice    |
| 1.234 | mixed | columns |

Sample table for tabular tag

**Attention:**

In LaTeX mapping everything works nice if you place a *tabular tag* without a *table tag*, only in the other mappings (e.g. html) it will be messed up.

## 7.5 Mathematical Paragraph

---

```
<!entity % mathpar " dm | eq " >
```

---

A *mathematical paragraph* consists either of a *displayed formula*, tagged by `<dm>`

No, sorry, not for Deutschmark! ;-)

or an *equation*, tagged by `<eq>`. They work very much the same.

Both of these tags contain a mathematical formula. See [Mathematical Formulas](#) for the tags valid here.

**Note:**

Because neither Netscape nor Microsoft has seen any need to add mathematical mappings to their browsers (like demanded and defined by *w3c*), there is no nice way of mapping, or at least displaying the math stuff in html. So if you view the online version, feel free to wonder what nonsense this man is telling here. Might be you should take a glance at the postscript version.

### Displayed Formula Tag

This tag displays a *mathematical formula* as a *paragraph*. The formula is mapped centered as a single line

No guarantee for that. You know: Mapping is a matter of taste.

.

```
<dm> (a+b) <sup/2/=a<sup/2/+2ab+b<sup/2/</dm>
```

Is mapped to:  $(a+b)^2=a^2+2ab+b^2$

### Equation Tag

```
<dm> (a+b) <sup/2/=a<sup/2/+2ab+b<sup/2/</dm>
```

Is mapped to:  $(a+b)^2=a^2+2ab+b^2$

## 7.6 Theorem Paragraph

---

```
<!entity % thrm
      " def | prop | lemma | coroll | proof | theorem " >

<!element def - - (thtag?, p+) >
<!element prop - - (thtag?, p+) >
<!element lemma - - (thtag?, p+) >
<!element coroll - - (thtag?, p+) >
<!element proof - - (p+) >
<!element theorem - - (thtag?, p+) >
```

---

As you can see the different types of *theorem* paragraphs are nearly identical. The only exception which is a little bit different is the *proof* which doesn't own a `thtag`. For all the others the `thtag` is giving the *tag* of the theorem paragraph.

You try to use that one, which is fitting the meaning of what you are typing.

```
<thrm>
<thtag>Alexander's thrm</thtag>
Let  $G$  be a set of non-trivially achievable subgoals
and  $\mu$  an order on  $G$ .  $\mu$  is abstractly
indicative if and only if it is a linearization of
 $\lim_{\mu} G$ .
</thrm>
```

The `thrm` is replaced by the adequate tag.

Maybe somebody knowing about mathematics would be shocked about my abuse of the types, but I'm lazy so I simply copied the examples:

*Definition* (`def`): Alexander's Definition

Let  $G$  be a set of nontrivially achievable subgoals and  $\mu$  an order on  $G$ .  $\mu$  is abstractly indicative if and only if it is a linearization of  $\mu G$

\*,

*Proposition* (`prop`): Alexander's Proposition

Let  $G$  be a set of nontrivially achievable subgoals and  $\mu$  an order on  $G$ .  $\mu$  is abstractly indicative if and only if it is a linearization of  $\mu G$

\*,

*Lemma* (`lemma`): Alexander's Lemma

Let  $G$  be a set of nontrivially achievable subgoals and  $\mu$  an order on  $G$ .  $\mu$  is abstractly indicative if and only if it is a linearization of  $\mu G$

\*,

*Corollation* (`coroll`): Alexander's Corollary

Let  $G$  be a set of nontrivially achievable subgoals and  $\mu$  an order on  $G$ .  $\mu$  is abstractly indicative if and only if it is a linearization of  $\mu G$

\*,

Alexander's Theorem

Let  $G$  be a set of nontrivially achievable subgoals and  $\mu$  an order on  $G$ .  $\mu$  is abstractly indicative if and only if it is a linearization of  $\mu G$

\*,

The proof is just the same without the `thtag`:

Let  $G$  be a set of nontrivially achievable subgoals and  $\mu$  an order on  $G$ .  $\mu$  is abstractly indicative if and only if it is a linearization of  $\mu G$

\*,

## 7.7 Code and verbatim Paragraphs

Both tags from a paragraph and have very similar behavior. Inside this tags most special characters don't need their named form as in section [Named Symbols](#). The exceptions are:

1. `&etago; -> </ ->` end of tag open

Maybe later the list will grow.

In difference to the normal paragraph mapping white-spaces and newlines will be mapped literally (as you write them in your source).

Also (with respect to manual layout) the font for mapping will be a non-proportional one.

See the difference between `IIWW` and `IIWW`.

### Note:

Aggain, I'm neither a native speaker not I love mathematics a lot. So I just placed some nonsense, wich might cause headache and grey hair for people who want to use this document for learning to formulate mathematical or physical theories.

Feel free to send better examples.

## Code Tag

---

```
<!element code - - rcdata>
```

---

Use the `code` tag, if you want to write sourcecode example within your text.

A code sample

`<code>`


---

```
#include <stdio.h>
int main() {
    printf("Hello world");
    return 1;
}
```

---

`</code>`

## Verbatim Tag

---

```
<!element verb - - rcddata>
```

---

Use the *verbatim tag* for anything else than sourcecode (use [Code Tag](#) for this) which needs the good old whitespace padding, like terminal hardcopy, ASCII-Graphics etc.

A verb sample

`<verb>`

```
//////////
|  *    *  |
|  |    |  |
|  <---->  |
|  _____  |
```

`</verb>`

## 8. Inline Tags

Here the abstract *inlines* are broken down until only true and usable tags will remain. Let's recall:

---

```
<!entity % inline
    " (#pcdata | f| x| %emph; |sq| %xref | %index | file ) * " >
```

---

Inlines don't have a influence to paragraphing, sectioning or document classing. Just modifying text within it's normal flow.

### 8.1 Emphasizes

---

```
<!entity % emph
    " em|i|t|b|f|s|f|s|l|t|t|c|param " >
```

---

The *emphasizes* are gathering the tags for emphasizing inline text.

The different types of emphasizes are:

#### **em -> The Emphasize Tag**

I hate to be redundant but I have to say: The *emphasize* tag you place for emphasized text. Normally it's mapped to italic letters. So if you write `<em/a emphasized text/` it will be mapped to *a*



*emphasized text.*

#### it -> The Italic Tag

The *italic* tag you place for a cursive mapping. If you write `<it/a italic text/` it will be mapped to *a italic text*.

#### bf -> The Boldface Tag

The *boldface* tag you place for a bold mapping. If you write `<bf/a bold text/` it will be mapped to **a bold text**.

#### sf -> The Swissfont Tag

I know that Tom Gordon from GMD is telling that this is the `sans serif` tag. My interpretation of the *sf* is *swissfont* wich for me is more easy to remember. This is mapping the inlined text to a font wich is out of the helvetica family. So `<sf/a swissfont text/` will be mapped to a swissfont text.

#### sl -> The Slanted Tag

I think I skip the explanation. `<sl/a slanted text/` will be mapped to *a slanted text*.

#### tt -> The Terminaltype Tag

Text tagged with *terminaltype* will be placed inline, just like all the other text within a paragraph. It will not be included into source output if you are workink as described in section [Literate Programming](#), even if it's looking like typed code. `<tt/a terminal typed text/` will be mapped to a terminal typed text.

## 8.2 Short-quote Tag

Normally this one could be viewed the same level like one of the *emphasize* tags, but the definition of the linuxdoc dtd is placing it same level like the *emphasizes*, and so I do.

The `shortquote` tag is a inline quotation, not forming an own paragraph. The text `<sq/a short quote/` is mapped to "a short quote".

## 8.3 Formula Tag

The *formula* tag allows us to note down a mathematical formula within the normal text, not appearing in an own line. So the text `<f>x=y<sup>2</sup></f>` will be displayed as  $x=y^2$ . See [Mathematical Fomulas](#) for the tags valid within the *formula*.

## 8.4 External Tag

The *external* tag is passing the tagged data directly through the parser, without modifying it. E.g. to LaTeX.

## 9. Mathematical Formulas

They can appear with in the tags listed in table [Places of Mathematical Formulas](#)

| tag | description       | see                                    |
|-----|-------------------|--|
| f   | inline formula    | <a href="#">The Formula Tag</a>        |
| dm  | displayed formula | <a href="#">Mathematical Paragraph</a> |
| eq  | equation          | <a href="#">Mathematical Paragraph</a> |

If you view this document mapped to html you will notice that html has no nice way of displaying mathematical formulas.

After a little hand parsing the contents of a *mathematical* tag looks like:

---

```
<!element  xx          - -
          ((fr|lim|ar|root) |
          (pr|in|sum) |
          (#pdata|mc| (tu|phr)) |
          (rf|v|fi) |
          (unl|ovl|sup|inf))*)>
```

---

The xx stands for f, dm or eq. All of them are the same.

**Note:**

Because neither Netscape nor Microsoft has seen any need to add mathematical mappings to their browsers (like demanded and defined by *w3c*), there is no nice way of mapping, or at least displaying the math stuff in html. So if you view the online version, feel free to wonder what nonsense this man is telling here. Might be you should take a glance at the postscript version.

## 9.1 Fraction Tag

---

```
<!element  fr          - - (nu,de) >
<!element  nu          o o ((%fbutxt;)* ) >
<!element  de          o o ((%fbutxt;)* ) >
```

---

So what we see from it is, that a *fraction* consits of a *numerator* and a *denominator* tag, wich again each one can hold a *mathematical formula*.

I think an example will tell you more:

```
<dm><fr><nu>7/<de>13/</fr></dm>
```

results to:

713

In case we want to to place 1/2 instead of the numerator without cleaning it up, we'll type:

```
<dm><fr><nu><fr><nu>1/<de>2/</fr></nu><de>13/</fr></dm>
```

Which results to:

1213

## 9.2 Product, Integral and Summation Tag

---

```

<!element  pr      - - (ll,ul,opd?) >
<!element  in      - - (ll,ul,opd?) >
<!element  sum      - - (ll,ul,opd?) >

```

---

Each of them has a *lower limit* (ll tag), a *upper limit* (ul tag) and a optional *operand*, where each of them again may consist of a formula. The tags are same in syntax like shown in table Tags with upper-, lower limit and operator.

| name      | example                 | result                  |
|-----------|-------------------------|-------------------------|
| Product   | $y = \prod_{i=1}^n x_i$ | $y = \prod_{i=1}^n x_i$ |
| Integral  | $y = \int_a^b x^2 dx$   | $y = \int_a^b x^2 dx$   |
| Summation | $y = \sum_{i=1}^n x_i$  | $y = \sum_{i=1}^n x_i$  |

Tags with upper-, lower limit and operator

## 9.3 Limited Tag

---

```

<!element  lim      - - (op,ll,ul,opd?) >
<!element  op       o o (%fcstxt;|rf|%fph;) -(tu) >
<!element  ll       o o ((%fbutxt;)* ) >
<!element  ul       o o ((%fbutxt;)* ) >
<!element  opd      - o ((%fbutxt;)* ) >

```

---

You can use that one for operators with upper and lower limits other than products, sums or integrals. The for the other types defined *operator* is destined by the `optag`, wich can contain again a mathematical formula.

Bi=0

$$n x_i$$

## 9.4 Array Tag

---

```

<!element  ar      - - (row, (arr, row)*) >
<!attlist  ar      ca cdata #required >
<!element  arr     - o empty >
<!element  arc     - o empty >
<!entity   arr     "<arr>" >
<!entity   arc     "<arc>" >

```

---

Of course a reasonable mathematical document needs a way to describe arrays and matrices. The *array* (`ar`) is noted down equivalent to a *tabular* (see section [The Tabular Tag](#)). The differences in handling are:

- No `<hline>` tag.
- The `ca` attribute character `|` is not allowed.
- Columns are not separated by `colsep` tag but with the `arc` tag (*array column*).
- Rows are not separated by `rowsep` tag but with the `arr` tag (*array row*).

Again the characters `|` and `@` are mapped to the adequate separator tag, so you really can note a array same way as a tabular.

```
<dm><ar ca="clr">
a+b+c | uv      <arc> x-y | 27      @
a+b   | u+v    | z   | 134   <arr>
a     | 3u+vw  | xyz | 2,978
</ar></dm>
```

Is mapped to:

a+b+c uv x-y 27 a+b u+v z 134 a 3u+vw xyz 2,978

## 9.5 Root Tag

```
<!element root      - - ((%fbutxt;)* ) >
<!attlist root
          n cdata ">
```

The *root* is noted down by the `root` tag, wich contains a `n` attribute, holding the value for the "n'th" root.

```
<dm><root n="3"/x+y/</dm>
```

is mapped to:

x+y

## 9.6 Figure Tag

```
<!element fi  - o (#pcdata) >
```

With the figure tag you can place mathematical figures. The tagged characters are directly mapped to a mathematical figure. Which character is mapped to which figure you'll find in [Mathematical Figures](#).

## 9.7 Realfont Tag

```
<!element rf  - o (#pcdata) >
```

This tag is placing a real font within a mathematical formula.

I'm really not sure about `rf`. What should it be?

No formula is allowed within that tag.

```
<dm><rf/Binom:/ (a+b)<sup/2/=a<sup/2/+2ab+b<sup/2/</dm>
```

is mapped to:

Binom:  $(a+b)^2=a^2+2ab+b^2$

## 9.8 Other Mathematical Tags

The remaining tags simply modify the tagged formula, without implying any other tag. The effect is shown in table [Mathematical tags without included tags](#)

| name             | tag | example  | result                        |
|------------------|-----|--|-------------------------------|
| <i>vector</i>    | v   | $\langle f \rangle \langle v/a/\&times; ; \langle v/b/=\langle v/0/\</f \rangle$ | $\rightarrow axb=0$           |
| <i>overline</i>  | ovl | $\langle f \rangle \langle ovl/1+1/=\langle ovl/2/\</f \rangle$                  | $\rightarrow 1+1=2$           |
| <i>underline</i> | unl | $\langle f \rangle \langle unl/1+1/=\langle unl/2/\</f \rangle$                  | $\rightarrow 1+1=2$           |
| <i>superior</i>  | sup | $\langle f \rangle e=m\&times;c\langle sup/2/\</f \rangle$                       | $\rightarrow e=m \times c^2$  |
| <i>inferior</i>  | inf | $\langle f \rangle x\langle inf/i/:=2x\langle inf/i-1/+3\</f \rangle$            | $\rightarrow x_i:=2x_{i-1}+3$ |

Mathematical tags without included tags

## 10. Labels and References

```
<!entity % xref
" label|ref|pageref|cite|url|htmlurl|ncite " >
```

As soon as it's a little bit more sophisticated a document will need references to other places within the document.

### 10.1 Label Tag

```
<!element label - o empty>
<!attlist label id cdata #required>
```

If you want to refer to a spot, chapter or section within your document you place a *label tag*.

A example could look like:

```
<sect1>Welcome to the article<label id="intro">
<p>...
```

## 10.2 Reference Tag

---

```
<!element ref - o empty>
<!attlist ref
      id cdata #required
      name cdata "
```

---

With this tag you can refer to a place within your document labeled as in Label Tag.

The way the reference is mapped in you document again depends to the mapper. May result to a hyper-ref (HTML) or a section number (LaTeX).

## 10.3 Page reference Tag

---

```
<!element pageref - o empty>
<!attlist pageref
      id cdata #required>
```

---

A example for a pageref:

```
<pageref id="intro">
```

---

In the HTML mapping there is no use for *pageref*, because there are no page numbers. In LaTeX mapping the tag is mapped to the pagenumber of the reffered label.

## 10.4 Url Tag

---

```
<!element url - o empty>
<!attlist url
      url cdata #required
      name cdata "
```

---

A example for a *url*:

```
<url url="http://www.gnu.org" name="GNU Organization">
```

---

GNU Organisation

The mapping to html brings up a hyper-ref in your document. The reference is the value of the *url* attribute, the text standing in the Hyperref is the *name* attribute's value.

In LaTeX mapping this one results to the name followed by the url.

## 10.5 Htmlurl Tag

---

```
<!element htmlurl - o empty>
<!attlist htmlurl
      url cdata #required
      name cdata "
```

---

A example for a `htmlurl`:

---

```
<htmlurl url="http://www.gnu.org" name="GNU Organization">
```

---

### GNU Organisation

The only difference between this tag and the Url Tag is in the LaTeX mapping.

The LaTeX mapping simply drops the `url` attribute and emphasizes the name.

In all other cases it's absolutely the same as the *url tag*.

## 10.6 Cite Tag

---

```
<!element cite - o empty>
<!attlist cite
    id cdata #required>
```

---

AFAIK this one need's bibTeX to work nicely. So I'm terribly sorry, but I was not jet able to make use of it. For that reason for sure I'm the wrong one to explain about it.

## 10.7 Ncite Tag

---

```
<!element ncite - o empty>
<!attlist ncite
    id cdata #required
    note cdata #required>
```

---

Same as Cite Tag.

## 11. Indices

---

```
<!entity % index "idx|cdx|nidx|ncdx" >

<!element idx - - (#pcdata)>
<!element cdx - - (#pcdata)>
<!element nidx - - (#pcdata)>
<!element ncdx - - (#pcdata)>
```

---

| tag  | my translation                            |
|------|---|
| idx  | index                                     |
| cdx  | code index (terminaltype index)           |
| nidx | invisible index                           |
| ncdx | invisible code index (terminaltype index) |

Index elements

The index tags serve for making a index of your document. They are only useful if you want do LaTeX mapping. They only differ very slightly as mentioned in table [Index elements](#).

## 11.1 Including a index

There are two ways to include indices into your document. Look at both and decide.

### Manually

1. Set the `opts` attribute of your document class to contain the packages *makeidx*. You do that by:  

```
<article opts="makeidx">.
```
2. Mark all the words you want to be in the index later with a *idx tag* or *cdx tag*. If the word you want to index to a location in your document is not within the text you simply write it at the location you want to index with the *nidx tag*. It's like the normal *idx* only the tagged text will be silently dropped in the normal document.
3. Process your file with `makeindex sgml2latex -m mydocument.sgml`.  
This will produce an additional `mydocument.idx`.
4. Process `mydocument.idx` with the `makeindex` command like `makeindex mydocument.idx`.  
This will produce an additional `mydocument.ind`.
5. To include the now generated index in your document you process your document with `sgml2latex -o tex -m mydocument.sgml`.  
This results in output of `mydocument.tex`.
6. Edit `mydocument.tex` with the editor of your choice.  
You look for the line `\end{document}` (should be somewhere close to the end of the file) and insert the text `\printindex` bevor this line.
7. Process the modified file with `latex mydocument.tex`.  
This gives you the final `mydocument.dvi` wich aggain you might process with `dvips` to generate a postscript document.

A lot of a mess, ain't it?

### Hacked

I'm currently working on a patch to the `sgmltools` to automate the inclusion and generation of a index. To find out the current state see <http://www.bnhof.de/~uwe/lnd/indexpatch/index.html>.

## 12. Literate Programming

---

```
<!entity % litprog " code | verb " >
```

---

This one is a funny thing. It's the idea of not to write some comment text within a program, and might be to take later some special tools, to extract the text

Think of `perlpod`.

, but to write a big document and later to extract the code from it.

People who don't like to document their code will not appreciate.



The principle is: All text within `verb` and `code` tags, will be gathered into a sourcefile.

That's it, because for now I don't remember the name of the tool doing thatone.

## 13. Reference

- *The qwertz Document Type Definition*

Norman Welsh

- *SGML-Tools User's Guide 1.0 (\$Revision\$)*

Matt Welsh and Greg Hankins and Eric S. Raymond  
November 1997

- *Quick SGML Example, v1.0*

Matt Welsh, <mdw@cs.cornell.edu>  
March 1994

## 14. Named Symbols

### 14.1 Named Characters

This is a slightly modified list taken from [*SGML-Tools User's Guide 1.0 (\$Revision\$)*]. If you miss some, don't hesitate to mail. A lot of the named characters shown in table Named Characters are same as in the html-dtd.

|        |       |         |          |        |     |        |     |        |     |        |     |
|--------|-------|---------|----------|--------|-----|--------|-----|--------|-----|--------|-----|
| AElig  | Æ     | Aacute  | Á        | Acirc  | Â   | Ae     | Ä   | Agrave | À   | Atilde | Ã   |
| Auml   | Ä     | Ccedil  | Ç        | Eacute | É   | Egrave | È   | Euml   | Ë   | Iacute | Í   |
| Icirc  | Î     | Igrave  | Ì        | Iuml   | Ï   | Ntilde | Ñ   | Oacute | Ó   | Ocirc  | Ô   |
| Oe     | Ö     | Ograve  | Ò        | Oslash | Ø   | Ouml   | Ö   | Uacute | Ú   | Ue     | Ü   |
| Ugrave | Û     | Uuml    | Ü        | Yacute | Ý   | aacute | á   | acirc  | â   | ae     | ä   |
| aelig  | æ     | agrave  | à        | amp    | &   | apos   | '   | aring  | å   | arr    | -v  |
| ast    | *     | atilde  | ã        | auml   | ä   | bsol   | \   | bull   | •   | ccedil | ç   |
| cir    | &cir; | circ    | ^        | clubs  |     | colon  | :   | comma  | ,   | commat | @   |
| copy   | ©     | darr    | -v       | deg    | °   | diams  |     | divide | ÷   | dollar | \$  |
| dquot  | "     | eacute  | é        | ecirc  | ê   | egrave | è   | equals | =   | etago  | </  |
| euml   | ë     | excl    | !        | frac12 | 1/2 | frac14 | 1/4 | frac18 | 1/8 | frac34 | 3/4 |
| frac38 | 3/8   | frac58  | 5/8      | frac78 | 7/8 | gt     | >   | half   | 1/2 | hearts |     |
| hellip | ...   | horbar  | &horbar; | hyphen | -   | iacute | í   | icirc  | î   | iexcl  | ¡   |
| igrave | ì     | iquest  | ¿        | iuml   | ï   | laquo  | «   | larr   | <-  | lcub   | {   |
| ldquo  | “     | lowbar  | _        | lpar   | (   | lsqb   | [   | lsquo  | ‘   | lt     | <   |
| mdash  | —     | micro   | μ        | middot | ·   | mu     | μ   | ndash  | –   | not    | ¬   |
| ntilde | ñ     | num     | #        | oacute | ó   | ocirc  | ô   | oe     | ö   | ograve | ò   |
| ohm    | &ohm; | ordf    | ª        | ordm   | º   | oslash | ø   | otilde | õ   | ouml   | ö   |
| para   | ¶     | percent | %        | period | .   | plus   | +   | plusmn | ±   | pound  | £   |
| quest  | ?     | quot    | "        | raquo  | »   | rarr   | ->  | rcub   | }   | rdquo  | ”   |

## Linuxdoc Reference

|       |   |        |   |        |    |       |      |        |    |        |    |
|-------|---|--------|---|--------|----|-------|------|--------|----|--------|----|
| reg   | ® | rpar   | ) | rsqb   | ]  | rsquo | '    | sect   | §  | semi   | ;  |
| sol   | / | spades |   | sup1   | ^1 | sup2  | ^2   | sup3   | ^3 | sz     | ß  |
| szlig | ß | tilde  | ~ | times  | ×  | trade | (TM) | uacute | ú  | uarr   | _^ |
| ucirc | û | ue     | ü | ugrave | ù  | uuml  | ü    | verbar |    | yacute | ý  |

Named Characters

## 14.2 Named Whitespaces

There is a small number of whatever you want to name it. The look like named characters, but will be printed not always, or not at all.

### thinsp

Thin space:

`d&thinsp;D ->d D`

### emsp

Emphasized space: `d&emsp;D -> d D`

### ensp

Normal space: `/d&ensp;D/ -> d D`

### nbsp

No break space: A spaces at wich the line is not allowed to be broken. Two words separated by a `nbsp` will be treated by parser and mapper to be a single long one.

### shy

Suggest Hyphen: If the mapper is up to break a word, with has the `shy` tag inside, it will probably do the wordbreak at the place of the `shy` tag and place a *hyphen* instead. If no wordbreak is necessary the `shy` expands to nothing at all.

## 15. Mathematical Figures

|  |                          |   |       |
|--|--------------------------|---|-------|
| a-ab-bc-cd-de-ef-fg-gh-hi-ij-jk-kl-lm-mn-no-<br>&horbar; | op-pq-qr-rs-<br>&horbar; | A-AB-BG-CD-DE-EE-EG-GH-HI-IJ-JK-KL-LM-MN-<br>&horbar; | NO-OI |
|--|--------------------------|---|-------|

Mathematical Figures

The special mappings for characters you might use for building up mathematical figures are shown in table [Mathematical Figures](#).

## 16. Linuxdoc dtd Source

This is the `linuxdoc.dtd` used to parse this document. The revision log, revision comments and a few redundant lines are taken out for saving paper and screenspace.

---

```
<!-- This is a DTD, but will be read as -*- sgml -*- -->
```

## Linuxdoc Reference

```
<!-- ===== -->
<!-- $Id$

This is LINUXDOC96 DTD for SGML-Tools.

This was LINUXDOC.DTD,
a hacked version of QWERTZ.DTD v1.3 by Matt Welsh,
Greg Hankins, Eric Raymond, Marc Baudoin and
Tristan Debeaupuis; modified from QWERTZ.DTD by
Tom Gordon.

<!entity % emph
    " em|it|bf|sf|sl|tt|cparam " >

<!entity % index "idx|cdx|nidx|ncdx" >

<!-- url added by HG; htmlurl added by esr -->
<!entity % xref
    " label|ref|pageref|cite|url|htmlurl|ncite " >

<!entity % inline
    " (#pcdata | f| x| %emph; |sq| %xref | %index | file )* " >

<!entity % list
    " list | itemize | enum | descrip " >

<!entity % par
    " %list; | comment | lq | quote | tscreen " >

<!entity % mathpar " dm | eq " >

<!entity % thrm
    " def | prop | lemma | coroll | proof | theorem " >

<!entity % litprog " code | verb " >

<!entity % sectpar
    " %par; | figure | tabular | table | %mathpar; |
    %thrm; | %litprog; ">
<!element linuxdoc o o
    (sect | chapt | article | report |
    book | letter | telefax | slides | notes | manpage ) >

<!-- `general' entity replaced with ISO entities - kwm -->
<!entity % isoent system"isoent">
%isoent;

<!entity urlnam sdata "urlnam" >
<!entity refnam sdata "refnam" >
<!entity tex sdata "[tex ]" >
<!entity latex sdata "[latex ]" >
<!entity latexe sdata "[latex]" >
<!entity tm sdata "[trade ]" >
<!entity dquot sdata "[quot ]" >
<!entity ero sdata "[amp ]" >
<!entity etago '</' >
<!entity Ae '&Auml;' >
<!entity ae '&auml;' >
<!entity Oe '&Ouml;' >
<!entity oe '&ouml;' >
<!entity Ue '&Uuml;' >
<!entity ue '&uuml;' >
```

```

<!entity sz '&szlig;' >
<!element p o o (( %inline | %sectpar )+) +(newline) >
<!entity ptag '<p>' >
<!entity psplit '</p><p>' >

```

```

<!shortref pmap
    "&#RS;B" null
    "&#RS;B&#RE;" psplit
    "&#RS;&#RE;" psplit
--    "'" qtag --
    "[" lsqb
    "~" nbsp
    "_" lowbar
    "#" num
    "%" percent
    "^" circ
    "{" lcub
    "}" rcub
    "|" verbar >

```

```

<!usemap pmap p>
<!element em - - (%inline)>
<!element bf - - (%inline)>
<!element it - - (%inline)>
<!element sf - - (%inline)>
<!element sl - - (%inline)>
<!element tt - - (%inline)>
<!element sq - - (%inline)>
<!element cparam - - (%inline)>

```

```

<!entity ftag '<f>' -- formula begin -- >
<!entity qendtag '</sq>'>

```

```

<!shortref sqmap
    "&#RS;B" null
--    "'" qendtag --
    "[" lsqb
    "~" nbsp
    "_" lowbar
    "#" num
    "%" percent
    "^" circ
    "{" lcub
    "}" rcub
    "|" verbar >

```

```

<!usemap sqmap sq >

```

```

<!element lq - - (p*)>
<!element quote - - ((%inline; | %sectpar;)*, p*)+ >
<!element tscreen - - ((%inline; | %sectpar;)*, p*)+ >
<!element itemize - - (item+)>
<!element enum - - (item+)>
<!element list - - (item+)>

```

```

<!shortref desmap
    "&#RS;B" null
    "&#RS;B&#RE;" ptag
    "&#RS;&#RE;" ptag
    "~" nbsp
    "_" lowbar
    "#" num

```

```

    "%" percent
    "^" circ
    "[" lsqb
    "]" rsqb
    "{" lcub
    "}" rcub
    "|" verbar >

<!element descrip - - (tag?, p+)+ >
<!usemap desmap descrip>

<!element item o o ((%inline; | %sectpar;)*, p*) >

<!element tag - o (%inline)>
<!usemap desmap tag>

<!usemap global (list,itemize,enum)>
<!entity space " ">
<!entity null "">

<!--
<!shortref bodymap
    "&#RS;B&#RE;" ptag
    "&#RS;&#RE;" ptag
    "'" qtag
    "[" lsqb
    "~" nbsp
    "_" lowbar
    "#" num
    "%" percent
    "^" circ
    "{" lcub
    "}" rcub
    "|" verbar>
-->

<!element figure - - ((eps | ph ), img*, caption?)>
<!attlist figure
    loc cdata "tbp"
    caption cdata "Caption">

<!-- eps attributes added by mb and td -->
<!element eps - o empty >
<!attlist eps
    file cdata #required
    height cdata "5cm"
    angle cdata "0">

<!element ph - o empty >
<!attlist ph
    vspace cdata #required>

<!element img - o empty>
<!attlist img
    src cdata #required>

<!element caption - o (%inline)>

<!shortref oneline
    "B&#RE;" space
    "&#RS;&#RE;" null
    "&#RS;B&#RE;" null

```

```

--      "'" qtag --
      "[" ftag
      "~" nbsp
      "_" lowbar
      "#" num
      "%" percent
      "^" circ
      "{" lcub
      "}" rcub
      "|" verbar>

<!usemap oneline tag>
<!usemap oneline caption>

<!entity % tabrow "(%inline, (colsep, %inline)*)" >
<!element tabular - -
      (hline?, %tabrow, (rowsep, hline?, %tabrow)*, caption?) >

<!attlist tabular
      ca cdata #required>

<!element rowsep - o empty>
<!element colsep - o empty>
<!element hline - o empty>

<!entity rowsep "<rowsep>">
<!entity colsep "<colsep>">

<!shortref tabmap
      "&#RE;" null
      "&#RS;&#RE;" null
      "&#RS;B&#RE;" null
      "&#RS;B" null
      "B&#RE;" null
      "BB" space
      "@" rowsep
      "|" colsep
      "[" ftag
--      "'" qtag --
      "_" thinsp
      "~" nbsp
      "#" num
      "%" percent
      "^" circ
      "{" lcub
      "}" rcub >

<!usemap tabmap tabular>
<!element table - - (tabular, caption?) >
<!attlist table
      loc cdata "tbp">

<!element code - - rcddata>
<!element verb - - rcddata>

<!shortref ttmap -- also on one-line --
      "B&#RE;" space
      "&#RS;&#RE;" null
      "&#RS;B&#RE;" null
      "&#RS;B" null
      '#' num
      '%' percent

```

```
'~'      tilde
'_'      lowbar
'^'      circ
'{'      lcub
'}'      rcub
'|'      verbar >
```

```
<!usemap ttmap tt>
<!element mc -- cdata >
<!entity % sposs "tu" >
<!entity % fcs "%sposs;|phr" >
<!entity % fcstxt "#pcdata|mc|%fcs;" >
<!entity % fscs "rf|v|fi" >
<!entity % limits "pr|in|sum" >
<!entity % fbu "fr|lim|ar|root" >
<!entity % fph "unl|ovl|sup|inf" >
<!entity % fbutxt "(%fbu;) | (%limits;) |
(%fcstxt;)|(%fscs;)|(%fph;)" >
<!entity % fphtxt "p|#pcdata" >
<!element f -- ((%fbutxt;)*) >
```

```
<!entity fendtag '</f>' -- formula end -- >
```

```
<!shortref fmap
"&#RS;B" null
"&#RS;B&#RE;" null
"&#RS;&#RE;" null
"_" thinsp
"~" nbsp
"]" rsqb
"#" num
"%" percent
"^" circ
"{" lcub
"}" rcub
"|" verbar>
```

```
<!usemap fmap f >
```

```
<!element dm -- ((%fbutxt;)*)>
<!element eq -- ((%fbutxt;)*)>
```

```
<!shortref dmmmap
"&#RE;" space
"_" thinsp
"~" nbsp
"]" rsqb
"#" num
"%" percent
"^" circ
"{" lcub
"}" rcub
"|" verbar>
```

```
<!usemap dmmmap (dm,eq)>
<!element fr -- (nu,de) >
<!element nu o o ((%fbutxt;)*) >
<!element de o o ((%fbutxt;)*) >
<!element ll o o ((%fbutxt;)*) >
<!element ul o o ((%fbutxt;)*) >
<!element opd - o ((%fbutxt;)*) >
<!element pr -- (ll,ul,opd?) >
```

## Linuxdoc Reference

```

<!element in - - (ll,ul,opd?) >
<!element sum - - (ll,ul,opd?) >
<!element lim - - (op,ll,ul,opd?) >
<!element op o o (%fcstxt;|rf|%fph;) -(tu) >
<!element root - - ((%fbutxt;)* >
<!attlist root
    n cdata ">
<!element col o o ((%fbutxt;)* >
<!element row o o (col, (arc, col)* >

<!element ar - - (row, (arr, row)* >
<!attlist ar
    ca cdata #required >
<!element arr - o empty >
<!element arc - o empty >
<!entity arr "<arr>" >
<!entity arc "<arc>" >

<!shortref arrmap
    "&#RE;" space
    "@" arr
    "|" arc
    "_" thinsp
    "~" nbsp
    "#" num
    "%" percent
    "^" circ
    "{" lcub
    "}" rcub >

<!usemap arrmap ar >
<!element sup - - ((%fbutxt;)* -(tu) >
<!element inf - - ((%fbutxt;)* -(tu) >
<!element unl - - ((%fbutxt;)* >
<!element ovl - - ((%fbutxt;)* >
<!element rf - o (#pcdata) >
<!element phr - o ((%fphtxt;)* >
<!element v - o ((%fcstxt;)*
    -(tu|limits;|%fbu;|%fph;) >
<!element fi - o (#pcdata) >
<!element tu - o empty >

<!usemap global (rf,phr)>
<!element def - - (thtag?, p+) >
<!element prop - - (thtag?, p+) >
<!element lemma - - (thtag?, p+) >
<!element coroll - - (thtag?, p+) >
<!element proof - - (p+) >
<!element theorem - - (thtag?, p+) >
<!element thtag - - (%inline)>

<!usemap global (def,prop,lemma,coroll,proof,theorem)>
<!usemap oneline thtag>
<!entity qtag '<sq>' >

<!shortref global
    "&#RS;B" null -- delete leading blanks --
    -- "'" qtag --
    "[" ftag
    "~" nbsp
    "_" lowbar
    "#" num

```



## Linuxdoc Reference

```
"%" percent
"^" circ
 "{" lcub
 "}" rcub
 "|" verbar>

<!usemap global linuxdoc>
<!element label - o empty>
<!attlist label id cdata #required>

<!-- ref modified to have an optional name field HG -->
<!element ref - o empty>
<!attlist ref
      id cdata #required
      name cdata "&refnam">

<!-- url entity added to have direct url references HG -->
<!element url - o empty>
<!attlist url
      url cdata #required
      name cdata "&urlnam" >

<!-- htmlurl entity added to have quieter url references esr -->
<!element htmlurl - o empty>
<!attlist htmlurl
      url cdata #required
      name cdata "&urlnam" >

<!element pageref - o empty>
<!attlist pageref
      id cdata #required>
<!element comment - - (%inline)>
<!element x - - ((#pcdata | mc)*) >
<!usemap #empty x >

<!-- Hacked by mdw to exclude abstract; abstract now part of titlepag -->
<!element article - -
      (titlepag, header?,
       toc?, lof?, lot?, p*, sect*,
       (appendix, sect+)?, biblio?) +(footnote)>

<!attlist article
      opts cdata "null">

<!-- Hacked by mdw to exclude abstract; abstract now part of titlepag -->
<!element report - -
      (titlepag, header?, toc?, lof?, lot?, p*,
       chapt*, (appendix, chapt+)?, biblio?) +(footnote)>

<!attlist report
      opts cdata "null">
<!element book - -
      (titlepag, header?, toc?, lof?, lot?, p*, chapt*,
       (appendix, chapt+)?, biblio?) +(footnote) >

<!attlist book
      opts cdata "null">

<!-- Hacked by mdw, abstract now part of titlepag -->
<!element titlepag o o (title, author, date?, abstract?)>
<!element title - o (%inline, subtitle?) +(newline)>
<!element subtitle - o (%inline)>
```

## Linuxdoc Reference

```
<!usemap oneline titlepag>
<!element author - o (name, thanks?, inst?,
                        (and, name, thanks?, inst?)*)>
<!element name o o (%inline) +(newline)>
<!element and - o empty>
<!element thanks - o (%inline)>
<!element inst - o (%inline) +(newline)>
<!element date - o (#pcdata) >

<!usemap global thanks>

<!element newline - o empty >
<!entity nl "<newline>">

<!-- Hacked by mdw -->
<!element abstract - o (%inline)>
<!usemap oneline abstract>

<!element toc - o empty>
<!element lof - o empty>
<!element lot - o empty>
<!element header - - (lhead, rhead) >
<!element lhead - o (%inline)>
<!element rhead - o (%inline)>
<!entity % sect "heading, header?, p* " >
<!element heading o o (%inline)>
<!element chapt - o (%sect, sect*) +(footnote)>
<!element sect - o (%sect, sect1*) +(footnote)>
<!element sect1 - o (%sect, sect2*)>
<!element sect2 - o (%sect, sect3*)>
<!element sect3 - o (%sect, sect4*)>
<!element sect4 - o (%sect)>
<!usemap oneline (chapt,sect,sect1,sect2,sect3,sect4)>
<!element appendix - o empty >
<!element footnote - - (%inline)>
<!usemap global footnote>
<!element cite - o empty>
<!attlist cite
            id cdata #required>

<!element ncite - o empty>
<!attlist ncite
            id cdata #required
            note cdata #required>

<!element file - - (#pcdata)>

<!element idx - - (#pcdata)>
<!element cdx - - (#pcdata)>
<!element nidx - - (#pcdata)>
<!element ncdx - - (#pcdata)>

<!element biblio - o empty>
<!attlist biblio
            style cdata "linuxdoc"
            files cdata ">
<!element slides - - (slide*) >

<!attlist slides
            opts cdata "null">
<!element slide - o (title?, p+) >
```

## Linuxdoc Reference

```
<!entity % addr "(address?, email?, phone?, fax?)" >

<!element letter - -
    (from, %addr, to, %addr, cc?, subject?, sref?, rref?,
     rdate?, opening, p+, closing, encl?, ps?)>

<!attlist letter
    opts cdata "null">

<!element from          - o (#pcdata) >
<!element to            - o (#pcdata) >

<!usemap oneline (from,to)>

<!element address      - o (#pcdata) +(newline) >
<!element email        - o (#pcdata) >
<!element phone        - o (#pcdata) >
<!element fax          - o (#pcdata) >

<!element subject      - o (%inline;) >
<!element sref         - o (#pcdata) >
<!element rref         - o (#pcdata) >
<!element rdate        - o (#pcdata) >

<!element opening      - o (%inline;) >
<!usemap oneline opening>

<!element closing      - o (%inline;) >
<!element cc - o (%inline;) +(newline) >
<!element encl - o (%inline;) +(newline) >

<!element ps - o (p+) >

<!element telefax - -
    (from, %addr, to, address, email?,
     phone?, fax, cc?, subject?,
     opening, p+, closing, ps?)>

<!attlist telefax
    opts cdata "null"
    length cdata "2">

<!element notes - - (title?, p+) >
<!attlist notes
    opts cdata "null" >
<!element manpage - - (sect1*)
    -(sect2 | f | %mathpar | figure | tabular |
     table | %xref | %thrm )>

<!attlist manpage
    opts cdata "null"
    title cdata ""
    sectnum cdata "1" >
<!shortref manpage
    "&#RS;B" null
--    "'" qtag --
    "[" ftag
    "~" nbsp
    "_" lowbar
    "#" num
    "%" percent
```

```
"^" circ  
"{" lcub  
"}" rcub  
"|" verbar>
```

```
<!usemap manpage manpage >
```

---