

# **SDoP: A Simple DocBook Processor**

**Philip Hazel**

# **SDoP: A Simple DocBook Processor**

Author: Philip Hazel

Copyright © 2009 Philip Hazel

Revision 0.52    25 February 2009

# Contents

<b>1. Introduction .....</b>	<b>1</b>
1.1 Installing SDoP .....	1
1.2 The SDoP command line .....	1
1.3 SDoP input .....	3
1.4 SDoP output .....	3
<b>2. Processing Instructions .....</b>	<b>4</b>
2.1 Including other files .....	4
2.2 Skipping parts of the input .....	4
2.3 Changing font styles .....	5
2.4 Changing font families .....	5
2.5 Specifying individual fonts .....	5
2.6 Specifying font sets .....	5
2.7 Overall scaling .....	6
2.8 Dimensions .....	6
2.9 Changing indentation .....	6
2.10 Global processing parameters .....	6
2.11 Local processing parameters .....	13
<b>3. Using information from &lt;bookinfo&gt; or &lt;articleinfo&gt; .....</b>	<b>17</b>
<b>4. SDoP data files .....</b>	<b>19</b>
4.1 PostScript header file .....	19
4.2 Font metrics .....	19
4.3 Hyphenation dictionary .....	19
4.4 Title pages for books .....	19
4.5 Article titles .....	19
4.6 Table of contents .....	19
4.7 Headers and footers .....	20
4.8 Index sorting .....	21
<b>5. Typesetting features .....</b>	<b>22</b>
5.1 Vertical white space .....	22
5.2 Indentation for literal blocks .....	22
5.3 Pagination .....	22
5.4 Hyphenation .....	22
5.5 Fine adjustments to lines .....	23
5.6 Ligatures .....	23
5.7 Kerning .....	23
5.8 The <emphasis> element .....	23
5.9 Using a small font .....	23
5.10 Using exotic fonts .....	23
5.11 Specifying coloured text .....	24
5.12 Change bars .....	24
5.13 Itemized lists .....	24
5.14 Tables .....	24
5.15 Footnotes .....	25
5.16 Illustrations and figures .....	25
5.17 Indexes .....	25
<b>6. Making Slides .....</b>	<b>26</b>

<b>7. Supported DocBook elements .....</b>	<b>27</b>
<b>8. Supported DocBook entities .....</b>	<b>30</b>
<b>9. Supported Unicode characters .....</b>	<b>35</b>
<b>10. Maintaining the hyphenation dictionary .....</b>	<b>39</b>
<b>Index .....</b>	<b>40</b>

# 1. Introduction

SDoP is a Simple DocBook Processor. It reads DocBook XML input and writes PostScript output. The output can easily be converted into PDF form by the *ps2pdf* command that comes as part of the GhostScript package. This document describes SDoP version 0.52.

The first version of SDoP was written to support just those features of DocBook that were needed to format the Exim manual. Further features have been added, and this version has some support for almost all the elements that are part of *Simplified DocBook*, as defined in the following URL:

<http://docs.linux.cz/programming/markup/www.docbook.org/tdg/simple/en/html/sdocbook.html>

The main omissions are support for bibliographies, multiple authors, subtables within tables, and some element attributes. Chapter 7 below contains a complete list of what SDoP does support.

I wrote SDoP because at that time the available non-commercial processors for converting DocBook into PostScript or PDF were not only slow and cumbersome, but also did not produce satisfactory output (see chapter 5 below). I included “simple” in the program’s name for the following reasons:

- SDoP does not check that the input conforms to a DocBook DTD, though it does check that element starting and ending tags are correctly nested. Few people write XML by hand, and what is generated by program is likely to be correct. There are programs such as *xmllint* that can be used to validate XML if you want to check your source before giving it to SDoP.
- SDoP does not support the full set of DocBook facilities (see chapter 7). Although more features may be added, it is unlikely ever to support everything.
- SDoP is a single program, written in C, that does not rely on external information, except for a few associated data files. This makes it very fast. When it was first written it could process the Exim manual (466 pages) in under 2.5 seconds on the author’s workstation; using *xmllto* and *fop* took around 2 minutes.
- SDoP makes no use of complicated stylesheets such as XSL, though the way it formats the output can be adjusted to some extent by means of embedded processing instructions (see chapter 2).

## 1.1 Installing SDoP

You should be able to build and install SDoP using the traditional commands:

```
./configure
make
make install
```

The installation consists of one binary, one *man* page, and a directory containing SDoP’s data files (see chapter 4). This is usually installed as */usr/local/share/sdop*. SDoP is automatically compiled with support for handling JPEG images if the *libjpeg* library is installed.

## 1.2 The SDoP command line

The SDoP command line is of the following form:

```
sdop [options] [source file]
```

If no source file is given, the source is read from the standard input. Output is by default written to the standard output if the source is the standard input; otherwise it is written to a file whose name is the same as the source, but with the extension changed to *.ps*. However, the destination can be specified explicitly by the **-o** option.

### **-d**

This option causes a small amount of debugging output to be written to the standard error stream. More detailed output can be obtained by adding *+name* after **-d**; a list of debugging names can be found from the **--help** option.

**-help or --help**

This causes SDoP to list the available options and then exit.

**-o <output-file>**

This option overrides the default output destination. If the output is specified as a single hyphen character, the standard output is used.

**-p <page-list>**

This option restricts the output to just those pages that are listed. The pages are counted from the start of the body of the document, that is, after any front matter (title pages, table of contents, preface). In other words, they are a book's "normal" page numbers. SDoP usually includes bookmark information in the output, for use when the PostScript is converted to PDF. However, this is disabled when **-p** is used, because the output is incomplete.

A comma is used to separate items in a page list, and each item can be a single page number or a hyphen-separated range. The page numbers must be in numerical order. If either of the words "odd" or "even" appears anywhere in the list, the entire output is restricted to either odd or even pages, respectively. If both appear, there is no restriction. For example:

```
sdop -p odd,23,35-50,99 myfile.xml
```

If "odd" or "even" is given with no page numbers, and there is no **-pf** option, all odd or even pages (respectively), including front matter pages, are output.

**-pf <page-list>**

This option is like **-p**, except that the pages are counted from the very start of the output, that is, this option is used to select pages from the book's front matter. Unlike **-p**, "odd" and "even" may not appear in the page list for **-pf**. However, you can use **-p** (with or without a page list) in the same command as **-pf**, for example:

```
sdop -p odd -pf 5-11 myfile.xml
```

This outputs just the selected odd pages from the front matter.

**-q**

This suppresses the default warnings that SDoP outputs for unsupported DocBook elements and element attributes, all of which are ignored. It overrides any setting that may be in an SDoP processing instruction within the source document.

**-qc**

This suppresses the default warnings that SDoP output for any unsupported characters that it encounters. It overrides any setting that may be in an SDoP processing instruction within the source document. Unsupported characters are those that are not present in the PostScript fonts.

**-S <directory list>**

This option supplies a colon-separated list of directories that are to be searched, in order, when SDoP needs to read one of its data files (see chapter 4). If the required file is not found in one of these directories, the default directory (often `/usr/local/share/sdop/`) is searched.

**-w**

This requests warning messages for unsupported DocBook elements and element attributes (the inverse of **-q**). It overrides any setting that may be in an SDoP processing instruction within the source document.

**-wc**

This requests warnings for unsupported characters (the inverse of **-qc**). It overrides any setting that may be in an SDoP processing instruction within the source document. Unsupported characters are those that are not present in the PostScript fonts.

**-v**

This causes SDoP to output its version number and to list any optional facilities (such as JPEG support) that are available, and then exit.

## 1.3 SDoP input

SDoP expects its input to be XML, encoded in Unicode, using UTF-8 for characters whose values are greater than 127. SDoP interprets XML elements (for example, `<chapter>`) as specified by DocBook, though it makes no attempt to validate the input against a DocBook DTD. There are programs such as *xmllint* that can be used for validation if you want check your source before giving it to SDoP.

You can feed SDoP fragmentary input containing, for example, just a sequence of `<para>` elements, and it will try to make sense of it. However, if the nesting of elements contradicts the DocBook DTD, the results may not be what you expect.

The elements that are recognized are listed below in chapter 7. Recognition of an element does not mean that SDoP actually does any processing for it. For example, `<trademark>` is recognized, but has no effect on the output. Unrecognized elements are completely ignored, though by default SDoP outputs a list of them to the standard error stream at the end of processing.

SDoP recognizes a number of standard entities such as `&amp;` (which represents a literal ampersand character). These are listed in chapter 8 below. In addition, there are some special entities that are useful for referencing text strings for title pages, the table of contents, headers, and footers (see 3, 4.6, 4.7).

## 1.4 SDoP output

SDoP outputs an entire book or article as a single PostScript file. Except when the output is restricted to specific pages by the `-p` or `-pf` options, the PostScript contains appropriate instructions so that if it is converted into a PDF, bookmarks are created for chapters and sections. The output for a book consists of:

- (1) A title page and its reverse; this is output if there is a `<book>` element that has an associated `<title>` element, or if a title is supplied in a `<bookinfo>` element (see 4.4).
- (2) A table of contents; this is omitted if it would be empty. The **toc\_sections** parameter (see 2.11) controls what is included.
- (3) One or more “prefaces” (for example, a preface and a foreword), if present in the input.
- (4) The main body of the book; this is always output.
- (5) One or more appendices, if present in the input.
- (6) One or more indexes; these are output at the points where the appropriate `<index>` elements appear in the document. However, an index is not generated if there are no appropriate `<indexterm>` elements.
- (7) One or more colophons, if present in the input.

Although rarely needed, it is in fact possible to mix indexes, chapters, and appendices in an arbitrary order.

An article is similar to a single chapter of a book. The output for an article does not have a title page or table of contents. The article’s title material appears at the top of the first body page. An article cannot contain prefaces, indexes, or colophons. An appendix is treated as a new section (in a book it is treated as a new chapter).

## 2. Processing Instructions

XML allows for *processing instructions* to be embedded in sources. A processing instruction is of this form:

```
<?sdop table_warn_overflow="never"?>
```

After the initial `<?` the name of the application for which this instruction is intended appears. All other applications that process the input should completely ignore the item. SDoP pays attention only to those that start with `<?sdop`.

A processing instruction can contain any number of parameter settings and may extend over more than one line. Each parameter specifies a value as a quoted string, the interpretation of which depends on the particular parameter. SDoP's processing parameters are of three kinds:

- *Preprocessing* parameters modify the input before the main processing takes place.
- *Global* parameters apply throughout the document. An example might be an instruction about the format of the table of contents. SDoP scans the input for these parameters early on. If the same one appears more than once, it is the last occurrence that is used.
- *Local* parameters are processed inline with the text. They apply only to what follows. For example, the use of hyphenation can be turned on and off for different parts of the document.

There is one restriction on SDoP's processing instructions: they must not appear between a `<book>` and a `<bookinfo>` element or between an `<article>` and an `<articleinfo>` element.

### 2.1 Including other files

A preprocessing instruction is available for including an external file at a given point in the input. The format is:

```
<?sdop include="filename"?>
```

This could be used to bring in an SDoP “style” definition (a collection of processing instructions) that is held externally. The *include* instruction is processed immediately after the main input has been read.

### 2.2 Skipping parts of the input

There is a facility for skipping parts of the input. At present this feature is of little use outside the title templates (☞ 4.4, 4.5). There are two forms of conditional:

```
<?sdop ifdef="name"?>
...lines of input...
<?sdop endif=" "?>
```

```
<?sdop ifndef="name"?>
...lines of input...
<?sdop endif=" "?>
```

In the first form, the intermediate lines are processed if an entity with the given name exists and contains a non-empty string. The second form applies the opposite condition. For example, a title page template might contain this:

```
<?sdop ifdef="book_subtitle"?>
<row>
<entry>
<emphasis role="booktitle2">&book_subtitle;</emphasis>
</entry>
</row>
<?sdop endif=" "?>
```



This example inserts a row containing the book's subtitle in the table that is used for the title page, but only if a subtitle has been specified.

The conditional parameters are obeyed after extracting information from the `<bookinfo>` or `<article>` element, if present, and after processing any global instructions. They may be nested arbitrarily deep.

## 2.3 Changing font styles

A number of global processing parameters make it easy to change the style of font that is used for certain DocBook elements that are typically printed in a non-roman font. For example, to specify the style that is to be used for the text of `<command>` elements:

```
<?sdo command_font="bold"?>
```

The recognized values that can be specified for this parameter are: `bold`, `bolditalic`, `italic`, `mono`, `monobold`, `monobolditalic`, `monoitalic`, and `roman`.

## 2.4 Changing font families

The font families that are used by default are Times, Helvetica, and Courier. These can be changed by the processing parameters **`serif_family`**, **`sanserif_family`**, and **`monospace_family`**. For example:

```
<?sdo serif_family="Palatino"
    sanserif_family="AvantGarde"?>
```

However, the family specified must be one that SDoP knows about, so that it can generate appropriate suffixes for bold, italic, and roman fonts. The currently known families are: AvantGarde, Bookman, Courier, Helvetica, NewCenturySchlbk, Palatino, Times, and Utopia. In addition, the font metric files for the specific fonts must be available in SDoP's shared data files (☞ 4.2).

There is also a facility for using characters from individual special fonts such as a font of musical characters, described later (☞ 5.10).

## 2.5 Specifying individual fonts

There are a number of processing parameters with names starting **`font_`** that allow you to specify a font size and style for a specific type of text. For example:

```
<?sdo font_chapter="20,2,serif,bolditalic"?>
```

The first figure is the point size, and the second is the additional vertical spacing (leading) that is associated with the font. For single fonts such as the one used for chapter headings, up to four comma-separated values can be given, as in the example above. Omitting a value leaves the related parameter unchanged. The third value must be `serif`, `sanserif`, or `monospaced`, and the fourth the name of a font style.

## 2.6 Specifying font sets

Some **`font_`** parameters control sets of fonts. For example, **`font_main`** defines the set of eight fonts (four normal, four monospaced) used for the main text. In these cases only three values are used – and the third is in fact ignored for the monospaced fonts. For example:

```
<?sdo font_main="12,1,sanserif"?>
```

This specifies that the main text is to use 12-point, sanserif fonts, with 1 point of leading. There are also some parameters that affect only the monospaced set of fonts, thus allowing them to be a different size (for example). In these setting, the style should not be used, as it could lead to weird effects.

## 2.7 Overall scaling

A parameter called **scale\_typesize\_base** can be used to change all the font sizes and leading by the same factor. The value is the font size for normal text. For example:

```
<?sdoP scale_typesize_base="12"?>
```

The default size is 11 points, so the above example would multiple all fonts sizes and leadings by 12/11. The scaling is applied after any individual size changes specified by **font\_** parameters. If you change the overall type sizes, you might also need to change some of the indentation parameters.

## 2.8 Dimensions

Whenever a processing instruction sets a dimension (for example, the vertical space for a page footer), the value must be given as a fixed point number, optionally followed by one of the abbreviations **pt** (points), **pc** (picas), **in** (inches), **mm** (millimetres), or **cm** (centimetres). If no units are given, points are assumed. For example:

```
<?sdoP foot_length="0.9in"?>
```

## 2.9 Changing indentation

There are a number of processing parameters whose names end with **\_indent** that allow you to change the indentation for certain kinds of text. Up to three dimensions can be given for each one. For example:

```
<?sdoP blockquote_indent="1in,2cm,4pt"
```

The first dimension is the indent for the first lines of paragraphs; the second applies to the remaining lines, and the third is an *endent*, that is, an leftwards indent at the end of each line. The values may be negative, causing text to stick out over the normal ends of lines.

## 2.10 Global processing parameters

The global processing parameters are listed below, in alphabetical order.

### **background\_rgb**

This sets a colour for the background of each page. Its intended use is for when SDoP is being used to create slides rather than printed pages (see chapter 6). It is necessary also to set **paper\_size** so that SDoP knows the area that is to be coloured. The value is a comma-separated triple of red/green/blue values in the range 0 to 1.

### **blockquote\_indent**

This specifies the indentation parameters for block quotes (☞ 2.9). The default value is "24,24,24".

### **blockquote\_ruled**

This specifies whether a block quote should be preceded and followed by horizontal rules. The value must be **yes** or **no** (the default).

### **blockquote\_title\_justify**

This specifies the justification for block quote titles. The value must be **left**, **right**, or **centre** (the default).

### **command\_font**

This sets the style of font to be used for `<command>` elements (☞ 2.3). The default is *italic*.

### **font\_blockquote\_title**

This specifies the font for blockquote titles. The default value is "11,1,serif,bold".

**font\_booktitle1**

**font\_booktitle2**

**font\_booktitle3**

**font\_booktitle4**

These four parameters specify the fonts that can be used for different levels of title on book title pages (§ 2.5). The defaults are all sanserif bold, with respective sizes and leading "24,2", "18,1.5", "15,1.25", and "13,1".

**font\_chapter**

This specifies the font for chapter titles, which is also used for article titles. The default value is "16,0,sanserif,bold".

**font\_chapter\_subtitle**

This specifies the font for chapter subtitles, which is also used for article subtitles. The default value is "12,0,sanserif,bold".

**font\_figure\_title**

This specifies the font for figure titles. The default value is "11,1,serif,italic".

**font\_footnote**

This specifies the set of eight fonts that are used for footnotes (§ 2.6). The default value is "9,1,serif".

**font\_footnotemono**

This specifies just the set of four monospaced fonts that are used for footnotes. The default value is "9,1".

**font\_formalpara\_title**

This specifies the font for formal paragraph titles. The default value is "11,1,serif,bold".

**font\_headfoot**

This specifies the set of eight fonts that are used in heads and feet. The default value is "11,1,serif".

**font\_headfootmono**

This specifies just the set of four monospaced fonts that are used for heads and feet. The default value is "11,1".

**font\_index**

This specifies the set of eight fonts that are used for the text in indexes. The default is "9.6,1,serif".

**font\_index\_section**

This specifies the font that is used for "section titles" in indexes. The default is "11.5,0,sanserif,bold".

**font\_indexmono**

This specifies just the set of four monospaced fonts that are used for the text in indexes. The default is "9.6,1".

**font\_main**

This specifies the set of eight fonts that are used for the main body text. The default is "11,1,serif".

**font\_mainmono**

This specifies just the set of four monospaced fonts that are used for the main body text. The default is "11,1".

**font\_note\_title**

This specifies the font for note titles. The default value is "11,1,serif,italic".

**font\_section**

This specifies the font that is used for top-level section titles. The default is "13,0,sanserif,bold".

**font\_subsection**

This specifies the font that is used for section titles other than those at the top level. The default is "11,0,sanserif,bold".

**font\_sidebar\_title**

This specifies the font for sidebar titles. The default value is "11,1,serif,bold".

**font\_small\_main**

This specifies the set of eight fonts that are used for “small” main body text (§ 5.9). The default is "9,1,serif".

**font\_small\_mainmono**

This specifies just the set of four monospaced fonts that are used for “small” main body text (§ 5.9). The default is "9,1".

**font\_table\_title**

This specifies the font for table titles. The default value is "11,1,serif,italic".

**font\_title**

This specifies the set of eight fonts that are used for any text that appears on title pages, except where such text is explicitly marked as being in a title font by the use of <emphasis> (§ 4.4). Normally the fonts set by **font\_title** are used for general information on the reverse of a title page. The default value is "11,1,serif".

**font\_title\_section**

This specifies the font that is used for any section headings on title pages. These might be used to separate information on the reverse of a title page. The default value is "11.5,0,sanserif,bold".

**font\_titlemono**

This specifies just the set of four monospaced fonts that are used for text on title pages. The default value is "11,1".

**font\_toc**

This specifies the set of eight fonts that are used for the text of the table of contents. The default value is "11,1,sanserif".

**font\_toc\_fill**

This specifies the font that is used for the “filler” in table of contents lines. The default value is "11,0,serif,roman".

**font\_tocmono**

This specifies just the set of four monospaced fonts that are used for text in tables of contents. The default value is "11,0".

**foot\_length**

This sets the amount of vertical space that is reserved for the footer at the bottom of each page. The default is 24 points.

**formalpara\_indent**

This specifies the indent parameters for formal paragraphs (§ 2.9). The default value is "0,0,0".

**function\_font**

This sets the style of font to be used for <function> elements (§ 2.3). The default is italic.

**head\_length**

This sets the amount of vertical space that is reserved for the header at the top of each page. The default is zero, which causes no header to be output.

**ilist\_indent**

This specifies the indent parameters for identified lists (§ 2.9). The default value is "12,12,0".

**index\_sort\_omit**

This specifies a list of characters that are to be ignored when sorting index entries. The default is to ignore the zero-width space (U+200B) and opening and closing double quotes (U+201C and U+201D). Setting this parameter completely replaces the list of characters.

```
<?sdo index_sort_omit="[*&lsquo;"
```

This example ignores only square brackets, asterisk, and opening single quote. As the example shows, you can use entity names in the list. However, only those entities that represent a single character are permitted.

**literal\_indent**

This specifies the indent parameters for blocks of literal text (☞ 2.9). The default value is "12,12,0".

**literal\_indent\_fudge**

The value of this parameter must be *yes* (default) or *no*. It controls the handling of lines within `<literallayout>` elements. If this facility is turned on, and there is a common indent for every line, it is removed. For example:

```
<literallayout>
  abcd efgh
    ijkl
</literallayout>
```

is treated as if it were:

```
<literallayout>
abcd efgh
    ijkl
</literallayout>
```

This feature exists because SDoP automatically indents literal blocks within other indented items such as lists, whereas some other DocBook processors do not. Input that was created for other processors may therefore contain explicit indents that are not needed for SDoP and which, if heeded, lead to excessive indentation.

**main\_even\_pages**

The value of this parameter must be *yes* or *no* (default). If it is set to *yes*, SDoP ensures that the main body of the output fills an even number of pages, adding a blank page at the end if necessary.

**margin\_bottom**

This specifies the amount of space at the bottom of each page. The default is 60 points (a bit less than an inch).

**margin\_left\_recto**

This specifies the amount of space at the lefthand side of righthand pages. The default is 72 points (one inch).

**margin\_left\_verso**

This specifies the amount of space at the lefthand side of lefthand pages. The default is 72 points (one inch).

**monospace\_family**

The value of this parameter must be the name of one of the known font families (☞ 2.4). The default is Courier. It is used for monospaced text. The results are unpredictable if the fonts are not in fact monospaced.

**note\_indent**

This specifies the indentation parameters for notes (☞ 2.9). The default value is "24,24,24".

**note\_ruled**

This specifies whether a note should be preceded and followed by horizontal rules. The value must be *yes* or *no* (the default).

**note\_title\_justify**

This specifies the justification for note titles. The value must be `left`, `right`, or `centre` (the default).

**olist\_indent**

This specifies the indent parameters for ordered lists (§ 2.9). The default value is `"24, 24, 0"`.

**option\_font**

This sets the style of font to be used for `<option>` elements (§ 2.3). The default is bold.

**orderedlist\_format**

The value of this parameter controls what is printed when numbering an ordered list. The format of item numbers is controlled by the **numeration** attribute of the `<orderedlist>` element – this parameter controls the additional characters (for example, punctuation) that are added. It must contain `%s` at the point where the item number is to be inserted. The default value is `(%s)`, which outputs the number in parentheses.

**page\_foot\_line\_width**

This specifies the width of lines in footers at the bottom of each page. The default is the value of **page\_line\_width**.

**page\_full\_length**

This specifies the full length of the page that is used for printing, including any heads and feet. The default is 720 points (10 inches).

**page\_head\_line\_width**

This specifies the width of lines in headers at the top of each page. The default is the value of **page\_line\_width**.

**page\_line\_width**

This specifies the width of lines on the page. The default is 450 points (6.25 inches).

**paper\_size**

If this parameter is set, it causes a `/PageSize` setting to be included in the PostScript. It is not necessary for normal output, but can be useful if SDoP is used to make a PDF for a slideshow (see chapter 6). The value must be two numbers *in points*, separated by the letter `x`. For example:

```
<?sdo p paper_size="900x660"?>
```

The first number specifies the width and the second the depth.

**para\_indent**

This specifies the indent parameters for normal paragraphs (§ 2.9). The default value is `"0, 0, 0"`.

**replaceable\_font**

This sets the style of font to be used for `<replaceable>` elements (§ 2.3). The default is italic.

**sanserif\_family**

The value of this parameter must be the name of one of the known font families (§ 2.4). The default is Helvetica.

**scale\_typesize\_base**

This parameter can be used to scale all the font sizes at once. The value is a point size for ‘normal’ text. For example:

```
<?sdo p scale_typesize_base="13"?>
```

This changes the normal font size from 11 to 13 points, and scales other fonts (for chapter titles, etc.) in proportion. This scaling is applied after any specific font sizes have been set up by any of the **font\_xxx** parameters. It is useful for making small adjustments to the default font sizes; if the change is large, you may have to adjust other spacing such as the indents.

**sidebar\_indent**

This specifies the indentation parameters for sidebars (§ 2.9). The default value is `"24, 24, 24"`.

**sidebar\_ruled**

This specifies whether a sidebar should be preceded and followed by horizontal rules. The value must be *yes* or *no* (the default).

**sidebar\_title\_justify**

This specifies the justification for sidebar titles. The value must be *left*, *right*, or *centre* (the default).

**term\_indent**

This sets the indent parameters for *<term>* items within *<vlist>*s (☞ 2.9). The default is "0,0,0".

**toc\_chapter\_blanks**

The value of this parameter controls the spacing around chapter entries in the table of contents. It consists of two *yes/no* substrings, separated by a comma. The default value is:

```
<?sdoP toc_chapter_blanks="yes,yes"?>
```

If the first substring is *yes*, a blank line is inserted before each chapter line in the table of contents, except for the first chapter. If the second substring is *yes* a blank line is inserted before the first section line that follows a chapter line. Missing substrings are interpreted as *no*.

**toc\_even\_pages**

The value of this parameter must be *yes* (default) or *no*. If it is set to *yes*, SDoP ensures that the table of contents fills an even number of pages, adding a blank page at the end if necessary.

**toc\_fill\_leftspace**

The value of this parameter is a dimension (default 2 points). It specifies the amount of space to leave after an entry in the table of contents before starting the “filler” sequence (typically a row of dots).

**toc\_fill\_rightspace**

The value of this parameter is a dimension (default 4 points). It specifies the amount of space to leave after the “filler” sequence in an entry in the table of contents, immediately before the page number.

**toc\_fill\_string**

The value of this parameter is a literal string (default " . "). It specifies the string that is replicated to create the “filler” in table of contents lines.

**toc\_foot\_centre\_recto****toc\_foot\_centre\_verso****toc\_foot\_left\_recto****toc\_foot\_left\_verso****toc\_foot\_right\_recto****toc\_foot\_right\_verso**

These parameters set texts for use in table of contents page footers on righthand (recto) and lefthand (verso) pages (☞ 4.7). The default value for the two centre parameters is "&romanpage;", which inserts an roman page number. The others default to empty strings. These are global processing parameters because there is no way of changing them in the middle of a table of contents.

**toc\_head\_centre\_recto****toc\_head\_centre\_verso****toc\_head\_left\_recto****toc\_head\_left\_verso****toc\_head\_right\_recto****toc\_head\_right\_verso**

These parameters set texts for use in table of contents page headers on righthand (recto) and lefthand (verso) pages (☞ 4.7). They all default to empty strings. These are global processing parameters because there is no way of changing them in the middle of a table of contents.

## toc\_line\_chapter\_strings

The value of this parameter is split into six, comma-separated substrings. They control the format of lines in the table of contents that refer to chapters. If a literal comma is required in one of the substrings, it must be preceded by an ampersand. Here is a totally unrealistic example (a realistic one would be too wide for the page):

```
<?sdop toc_line_chapter_strings="A&,A,BB,CC,DD,EE,FF"?>
```

The six strings are used to construct the table of contents line as follows:

- The first string is always used at the start of the line;
- If chapter numbers are being included in the table of contents, the second string is inserted, followed by the chapter number and the third string. If chapter numbers are not being included, the second and third strings are ignored.
- Then come the chapter title and the fourth string.
- At this point, a suitable number of copies of the filler string, as defined by **toc\_fill\_string**, are inserted.
- The line ends with the fifth string, the page number, and finally the sixth string.

The default values for the strings are:

[illegible]

The entity `&nbsp;` is a “non-breaking space”, that is, it is a space that is treated as if it were a printing character.

## toc\_line\_sect1\_strings

The value of this parameter is treated in the same way as **toc\_line\_chapter\_strings**, except that it applies to first-level section lines in the table of contents. The default values for the strings are:

&nbsp; &nbsp; &nbsp; &nbsp; &nbsp;	Start of line
(empty)	Before section number
&nbsp; &nbsp; &nbsp;	After section number
(empty)	After title
(empty)	Before page number
(empty)	After page number

## toc\_line\_sect2\_strings

The value of this parameter is treated in the same way as **toc\_line\_chapter\_strings**, except that it applies to second-level (subsection) and any lower level section lines in the table of contents. The default values for the strings are the same as for first-level sections, except that the initial indent is eight spaces rather than four.

**toc\_title**

The value of this parameter is used as the title for the table of contents. The default is `Contents`.

**userinput\_font**

This sets the style of font to be used for `<userinput>` elements (☞ 2.3). The default is monospaced.

**varname font**

This sets the style of font to be used for <varname> elements (☞ 2.3). The default is italic.

**vlist\_indent**

This sets the indent parameters for the paragraphs in a <variablelist> (☞ 2.9). The default is "14, 14, 0".



**vlist\_title\_indent**

This sets the indent parameters for the title of a `<variablelist>` (☞ 2.9). The default is "0,0,0".

**warn\_unsupported**

The value of this parameter must be `yes` (default) or `no`. If it is set to `yes`, SDoP outputs a list of unrecognized DocBook elements and parameters to the standard error at the end of processing. This setting can be overridden by the use of `-q` or `-w` on the command line.

**warn\_unsupported\_characters**

The value of this parameter must be `yes` (default) or `no`. If it is set to `yes`, SDoP outputs a warning message the first time it encounters a character that is not supported. Unsupported characters are those that are not present in the PostScript fonts. This setting can be overridden by the use of `-qc` or `-wc` on the command line.

## 2.11 Local processing parameters

The local processing parameters are listed below, in alphabetical order:

**chapter\_skip\_head**

The value of this parameter must be `yes` or `no` (default). When the value is `yes`, page headings are omitted at the start of chapters. This parameter must be set before the first chapter to which it applies. If this is in the middle of the book, the best place is immediately before the relevant `<chapter>` element.

**example\_number\_format**

This parameter controls the format of example numbers that are (optionally) added to example titles, and can also be inserted as cross-references. The value is a string that can contain up to two instances of `%s`. If there is just one, the example number, counting from the start of the document, is inserted. If there are two, the first is replaced by the chapter number, and the second by the example number, counting from the start of the chapter. The default setting is `%s-%s`, leading to example numbers of the form 2-7.

**example\_title\_format**

This parameter controls the format of example titles. Its value is inserted before the text given in the `<title>` element of an example. The string may contain a single instance of `%s`, which is replaced by the example number, in the format defined by **example\_number\_format**. The default setting is the string `Example %s:` followed by a terminating space.

**example\_title\_justify**

The value of this parameter specifies how example titles are formatted. Its value must be one of the words `left`, `right`, `centre` (or `center`), or `both`, with the default being `centre`. If an example contains a `<mediaobject>`, its alignment is taken as the example's alignment. Otherwise the example is assumed to be left justified.

**example\_title\_width**

This parameter controls the width of example titles. Text that is longer than the width is split into multiple lines. The default is to take the width of a `<mediaobject>` if available, otherwise the page width. The value may be an absolute dimension, or the word `object`, signifying the default.

**extra\_leading**

This parameter's data must be a dimension. It specifies an extra amount of vertical space that is to be inserted between the lines of the next and subsequent paragraphs.

**figure\_number\_format**

This parameter controls the format of figure numbers that are (optionally) added to figure titles, and can also be inserted as cross-references. The value is a string that can contain up to two instances of `%s`. If there is just one, the figure number, counting from the start of the document, is inserted. If there are two, the first is replaced by the chapter number, and the second by the figure number, counting from the start of the chapter. The default setting is `%s-%s`, leading to figure numbers of the form 4-5.

**figure\_title\_format**

This parameter controls the format of figure titles. Its value is inserted before the text given in the `<title>` element of a figure. The string may contain a single instance of `%s`, which is replaced by the figure number, in the format defined by **figure\_number\_format**. The default setting is the string `Figure %s:` followed by a terminating space.

**figure\_title\_justify**

The value of this parameter specifies how figure titles are formatted. Its value must be one of the words `left`, `right`, `centre` (or `center`), or `both`, with the default being `centre`. If a figure contains a `<mediaobject>`, its alignment is taken as the figure's alignment. Otherwise the figure is assumed to be left justified.

The title is formatted and positioned in relation to the figure, according to the value of **figure\_title\_justify**. For example, if the figure is left aligned and the title is centred, it is centred with the figure if possible (it may not be possible for an overlong title), not with the page.

**figure\_title\_width**

This parameter controls the width of figure titles. Text that is longer than the width is split into multiple lines. The default is to take the width of a `<mediaobject>` if available, otherwise the page width. The value may be an absolute dimension, or the word `object`, signifying the default. For many figures the default works well. However, for small figures it may be necessary to specify a width that is greater than the width of the figure.

**format**

The only recognized value for this parameter at present is `newpage`. It forces a new page in the output.

**foot\_centre\_recto****foot\_centre\_verso****foot\_left\_recto****foot\_left\_verso****foot\_right\_recto****foot\_right\_verso**

These parameters set texts for use in page footers on righthand (recto) and lefthand (verso) pages (☞ 4.7). The default value for the two centre parameters is `"&arabicpage;"`, which inserts an arabic page number. The others default to empty strings. You normally need to force a new page before changing footer contents in the middle of a document.

**head\_centre\_recto****head\_centre\_verso****head\_left\_recto****head\_left\_verso****head\_right\_recto****head\_right\_verso**

These parameters set texts for use in page headers on righthand (recto) and lefthand (verso) pages (☞ 4.7). They all default to empty strings.

**hyphenate**

The value of this parameter must be `yes` (default) or `no`. When it is set to `yes`, hyphenation is enabled (☞ 5.4).

**index\_headings**

The value of this parameter must be `yes` (default) or `no`. The default is to insert headings into indexes when the leading letter of the entries changes, unless every entry starts with the same character. No headings are inserted if the value is `no`.

**kern**

The value of this parameter must be `yes` (default) or `no`. If it is set to `yes`, kerning of adjacent letters in words is enabled. Kerning adjusts the spacing between letters to make the result look nicer.

### **numbertitles**

The value of this parameter is split up into up to 10 comma-separated substrings, each of which must be `yes` or `no`. If there are fewer than 10 substrings, the remainder are assumed to be `no`. The first substring determines whether or not chapters are to be numbered. The remaining substrings control sections, subsections, subsubsections, etc. By default, only chapters, sections, and subsections are numbered in a book. In an article, there is no numbering by default. You can change this by setting this parameter, and it works even if the setting precedes the `<article>` element.

### **page\_columns**

This parameter sets the number of columns on the page (default 1). It is used automatically to print indexes in two columns, and that is the main use for which it was implemented. It is not a very sophisticated implementation. A page break is forced if the number of columns is changed when text is currently being formatted into a column other than the first.

### **page\_column\_separation**

This parameter specifies the separation between multiple columns on the page. The default value is 16 points.

### **preface\_foot\_centre\_recto**

### **preface\_foot\_centre\_verso**

### **preface\_foot\_left\_recto**

### **preface\_foot\_left\_verso**

### **preface\_foot\_right\_recto**

### **preface\_foot\_right\_verso**

These parameters set texts for use in preface page footers on righthand (recto) and lefthand (verso) pages (☞ 4.7). The default value for the two centre parameters is `"&romanpage;"`, which inserts an roman page number. The others default to empty strings. You normally need to force a new page before changing footer contents in the middle of a document.

### **preface\_head\_centre\_recto**

### **preface\_head\_centre\_verso**

### **preface\_head\_left\_recto**

### **preface\_head\_left\_verso**

### **preface\_head\_right\_recto**

### **preface\_head\_right\_verso**

These parameters set texts for use in preface page headers on righthand (recto) and lefthand (verso) pages (☞ 4.7). They all default to empty strings.

### **table\_indent**

This parameter's data is a single dimension that specifies an indent for all subsequent tables in the document.

### **table\_number\_format**

This parameter controls the format of table numbers that are (optionally) added to table titles, and can also be inserted as cross-references. The value is a string that can contain up to two instances of `%s`. If there is just one, the table number, counting from the start of the document, is inserted. If there are two, the first is replaced by the chapter number, and the second by the table number, counting from the start of the chapter. The default setting is `%s-%s`, leading to table numbers of the form 2-7.

### **table\_title\_format**

This parameter controls the format of table titles. Its value is inserted before the text given in the `<title>` element of a table. The string may contain a single instance of `%s`, which is replaced by the table number, in the format defined by **table\_number\_format**. The default setting is the string `Table %s:` followed by a terminating space.

### **table\_title\_justify**

The value of this parameter specifies how table titles are formatted. Its value must be one of the words `left`, `right`, `centre` (or `center`), or `both`, with the default being `left`. The title is formatted and positioned in relation to the table. For example, if the title is centred, it is centred with the table if possible (it may not be possible for an overlong title), not with the page.

### **table\_title\_width**

This parameter controls the width of table titles. Text that is longer than the width is split into multiple lines. The default is the width of the table. The value may be an absolute dimension, or the word `object`, signifying the default.

### **table\_warn\_overflow**

This parameter controls what happens when a cell in a table overflows (☞ 5.14). Text in table cells can be split into multiple lines if it contains any splitting points; the overflow case occurs when the text cannot be split. The value of this parameter must be `always` (default), `never`, or `overprint`. In the default case, a cell overflow is a hard error, and a warning message is always output. Conversely, if `never` is specified, the overflow is ignored.

If the value is `overprint`, a warning is generated only when the overflow actually causes overprinting of the contents of an adjacent cell, or of a separator (vertical line) between the two cells. In other words, when there is no cell separator, one cell may overflow into the next, provided that its text does not actually overprint the adjacent cell's text.

### **toc\_printed\_sections**

This parameter works in conjunction with **toc\_sections**. Its default value is the same as for **toc\_sections**, and it is altered whenever **toc\_sections** is changed. If you want **toc\_printed\_sections** to be different, you must set it after setting **toc\_sections** (each time, if there is more than one setting). The value of **toc\_printed\_sections** is in the same format as **toc\_sections**, but it can only specify fewer items. This makes it possible to limit the table of contents that is printed (that is, forms part of the document's text), while at the same time retaining a more comprehensive version in the output file for conversion to a PDF table of contents. For example:

```
<?sdoc toc_sections="yes,yes,yes,no"
  toc_printed_sections="yes,yes,no"?>
```

This example restricts the printed table of contents to chapters and sections, but includes subsections as well in a PDF version.

### **toc\_sections**

The value of this parameter must be a sequence of up to 10 comma-separated substrings. It controls which parts of the document are to be listed in the table of contents. As it is a local processing parameter, the choice can be different in different parts of the document. For example, you can include section titles in the table of contents in the body of the book, but not for the preface. If you set **toc\_sections** without setting **toc\_printed\_sections**, the selection applies both to what is printed, and to what is included in the online contents if the output is converted to PDF.

The first substring applies to chapters and chapter-like parts such as appendices. Subsequent substrings apply to successive levels of nested sections, until one of them is `no`. If the first value is `no`, no table of contents is output. In this case, no bookmark information is available in the PostScript, so if it is converted to PDF, there is no PDF table of contents either.

There are two default values, one for prefaces, and one for the body of the document. In prefaces, the default is:

```
<?sdoc toc_sections="yes"?>
```

That is, only the preface title is included by default in the table of contents. For the rest of the document, the default value is:

```
<?sdoc toc_sections="yes,yes,yes,no"?>
```

This includes chapters, sections, and subsections, but nothing deeper.

### 3. Using information from <bookinfo> or <articleinfo>

Information is extracted from the <bookinfo> or <articleinfo> elements and made available for insertion at arbitrary places in the document in two different ways:

- (1) Simple text strings are placed in special entities; for example, the data from the <surname> element inside <author> is inserted anywhere that &author\_surname; appears.
- (2) Two more complicated items that contain further entities are inserted by one of the following processing instructions:

```
<?sdo insert="legalnotice"?>
<?sdo insert="revdescription"?>
```

If there are multiple occurrences of <legalnotice> or <revdescription>, they are inserted in order of definition.

Both kinds of insert can appear anywhere in the document, though they are most commonly found in title pages. The special entities are shown in the table below. If no title, subtitle, or title abbreviation is provided in the <bookinfo> element, values from the <book> element are used if they exist. In other words, <bookinfo> overrides <book>.

The names of the entities do not change for <article>s, even though the information is taken from <articleinfo> rather than <bookinfo>. For example, you must use &book\_title; to insert the article's title.

```
&author_address;
&author_corpauthor;
&author_firstname;
&author_honorific;
&author_initials;
&author_jobtitle;
&author_lineage;
&author_orgname;
&author_othername;
&author_surname;
&book_cpyholder;
&book_cpyyear;
&book_date;
&book_edition;
&book_issuenum;
&book_pubdate;
&book_publishername;
&book_releaseinfo;
&book_revauthorinitials;
&book_revdate;
&book_revnumber;
&book_subtitle;
&book_title;
&book_titleabbrev;
&book_volumenum;
&editor_firstname;
&editor_honorific;
&editor_initials;
&editor_jobtitle;
&editor_lineage;
&editor_orgname;
&editor_othername;
&editor_surname;
&othercredit_firstname;
```

```
&othercredit_honorific;  
&othercredit_initials;  
&othercredit_jobtitle;  
&othercredit_lineage;  
&othercredit_orgname;  
&othercredit_othername;  
&othercredit_surname;
```

## 4. SDoP data files

SDoP uses a number of auxiliary data files in its processing. The distributed versions are normally installed in a public directory such as `/usr/local/share/sdop/`. The **-S** command line option can be used to supply a list of directories to search before the standard one, thereby providing a means for individual users to override the default files. This chapter discusses the uses that are made of the different files.

### 4.1 PostScript header file

At the start of SDoP's output, it writes a copy of the file *PSheader* (with PostScript comments removed). This is a typical PostScript header file that defines some PostScript functions for use by the following code. In particular, it contains a function for loading fonts and adjusting their encodings.

### 4.2 Font metrics

Any typesetting program needs to know the metrics of the fonts it is using. SDoP is distributed with AFM files for all the standard Adobe fonts in a subdirectory called *fontmetrics*. For example, the file for the Times Roman font is in *fontmetrics/Times-Roman.afm*.

### 4.3 Hyphenation dictionary

SDoP does hyphenation by means of a dictionary. A dictionary for British English is installed in the file *HyphenData*. The source of this file, and the program that prepares it for use are also distributed (see chapter 10).

### 4.4 Title pages for books

The file *titletemplate* is a template for the title pages of a book. There are some special entities and an “insert” facility that can be used in the title pages to insert data obtained from the `<bookinfo>` element (see chapter 3).

The distributed version of *titletemplate* is an input fragment that defines two tables, one for each of two pages. The tables are defined in XML, but there are also a number of SDoP conditional processing parameters that control what is output on the title pages (see 2.2), depending on what data has been supplied. If no book title is available, no title pages are output.

The `<emphasis>` element can be used in title pages with `role="booktitlen"`, where *n* is a number between 1 and 4. This selects one of four different sizes of title font: 24, 16, 18, or 13 points.

You can provide your own title page layout by copying *titletemplate* to another file, editing it to your own requirements, and passing the name of the directory where it resides to SDoP using the **-S** command line option.

### 4.5 Article titles

The file *arttemplate* is a template for the title material of an article, which appears at the top of the first page if an article title has been defined. The distributed version is an input fragment that defines a single table containing the heading material. It is similar to the first table used for a book's title page, as described in the previous section, except that the values of the special entities are obtained from the `<articleinfo>` element rather than `<bookinfo>`. Nevertheless, the same entity names are used (for example, `&book_date;` rather than `&article_date;`).

### 4.6 Table of contents

The table of contents is mostly generated dynamically as a `<table>` that is processed internally by SDoP. However, the start of the table of contents “chapter” and the beginning of the `<table>` is obtained from a file called *toctemplate*. You are unlikely to want to modify this, except perhaps to change the format of the title. The special entity `&toc_title;` contains the string set by the

**toc\_title** global processing parameter (☞ 2.10). The default is Contents. If you just want to change this text, use the processing parameter.

## 4.7 Headers and footers

A page header or footer is output only if space has been reserved for it. By default, SDoP reserves 24 points of vertical space for a footer, but no header. The **head\_length** and **foot\_length** processing parameters can be used to adjust the allocations. When headers are in use, those on pages that start chapters can be independently suppressed by setting the **chapter\_skip\_head** processing parameter to yes.

There are two files called *headtable* and *foottable* that define head and foot lines for the body of the document and any appendixes, indexes, and colophons. There are similar files called *headtable-t* and *foottable-t* for the table of contents, and a third set called *headtable-p* and *foottable-p* for the preface(s).

A two-level substitution process is used for head and foot lines. This makes it possible to have different heads and feet on left-hand and right-hand pages, while also making it easy to change their contents by means of local processing parameter settings. However, if you want to change the overall layout of heads or feet, you have to provide new template files.

In the supplied templates, an XML table is used, containing references to the following special entities:

<code>&amp;footcentre;</code>	centre foot text
<code>&amp;footleft;</code>	left foot text
<code>&amp;footright;</code>	right foot text
<code>&amp;headcentre;</code>	centre head text
<code>&amp;headleft;</code>	left head text
<code>&amp;headright;</code>	right head text

As their names suggest, these are placed in table cells that are respectively left-, centre-, and right-justified. The values of these entities for the main body of the book and the preface(s) can be set by local processing parameters (☞ 2.11). This means they can be varied during the course of the document; it is usually necessary to force a new page before changing the contents of the footer.

For the table of contents, the values are set by global processing parameters (☞ 2.10), because there is no way to change them in the middle of the table of contents.

References to other entities are permitted in the values set for the above special entities. The default for the main body pages is to print an arabic page number in the centre of the footer, equivalent to:

```
<?sdo foot_centre_recto="&arabicpage;"
      foot_centre_verso="&arabicpage;"?>
```

This setting means that, on both right-hand (recto) and left-hand (verso) pages, the value of `&footcentre;` is `&arabicpage;`. If you want to have page numbers on the left and right instead, you could use:

```
<?sdo foot_centre_recto=" "
      foot_centre_verso=" "
      foot_right_recto="&arabicpage;"
      foot_left_verso="&arabicpage;"?>
```

It is not possible to include markup when changing the values of these strings. If you want to change the markup in headers or footers, you have to provide your own template files.

Other useful entities that are available in header and footer lines are `&chapternumber;`, `&chaptertitle;`, `&sectionnumber;`, and `&sectiontitle;`. In the case of titles, the value is taken from a `<titleabbrev>` element if present, otherwise from `<title>`. Here is an example of a processing instruction that adds the chapter title and number to footer lines, on the right and left, respectively:



```
<?sdop
  foot_right_recto="&chaptertitle; (&chapternumber;)"
  foot_left_verso="&chaptertitle; (&chapternumber;)"
?>
```

During chapter-like components such as prefaces, appendices, indexes, and colophons, the title is placed in `&chaptertitle`.

In an article, `&chaptertitle` contains the title of the article. In an appendix in an article, the title is placed in `&sectiontitle`, because it behaves like a section and does not start a new page.

You can use `&romanpage;` to obtain the page number as a roman numeral. This is used in the default settings for the table of contents and the preface(s), where the page number is printed in italic.

If you want to use a rule (a horizontal line) in a header or a footer, you can do so by providing your own template file containing a suitable table with a top or bottom frame line.

## 4.8 Index sorting

The file called *indexcollate* controls the order in which index items are sorted. There are comments in the file that explain its format. You can change the index sorting order by overriding this file.

## 5. Typesetting features

This chapter discusses some of the typesetting features of SDoP. The lack of some of these facilities in other processors was one of the reasons for writing SDoP.

### 5.1 Vertical white space

Additional vertical white space can be obtained by inserting a paragraph containing just a single “hard space” character (U+00A0). A smaller amount of vertical space can be obtained by inserting a paragraph containing just a single “zero-width space” character (U+200B). SDoP treats this case specially, removing the line containing the space so that only the pre- and post-paragraph spaces remain.

### 5.2 Indentation for literal blocks

SDoP automatically indents literal blocks within indented items. For example, paragraphs in a list are indented; if a `<literallayout>` block is part of a list item, SDoP indents it by the list item’s indent, in addition to any indent for the literal block itself.

Some other DocBook processors do not do this. They always set literal blocks with reference to the lefthand edge of the page. Consequently, input that was created for such other processors may contain extra indents for literal blocks that are not needed when it is processed by SDoP. To avoid excessive indentation, SDoP automatically removes an indent that is common to every line in a literal block. This feature can be turned off by setting the global processing parameter **literal\_indent\_fudge** to no (☞ 2.10).

### 5.3 Pagination

SDoP never outputs a section title as the last line on a page (chapter titles are always at the head of a page). A table that contains just a single row is treated as a sort of heading and is also not printed at the end of a page. SDoP generally avoids printing just one line of a multi-line paragraph as either the first or last line of a page (such lines are known as “widow” and “orphan” lines, respectively).

### 5.4 Hyphenation

Some other applications do overly-aggressive hyphenation. If a word that does not fit into a line can be hyphenated, they always split it. This leads to far too much hyphenation. SDoP attempts hyphenation only when the line is very “loose” – that is, the ratio of space to text in the line is high, which would make it look ugly if no more text were added.

When it encounters a very loose line, SDoP first looks for an explicit hyphen within the next word. There are three explicit hyphen characters:

- A “hard” hyphen is the normal hyphen character (U+002D). It is output whether or not the word is split immediately after it.
- The “soft” hyphen character (U+00AD) indicates a potential hyphenation point, but is output only if the word is actually split at that point. It is omitted when there is no split.
- The “zero width space” character (U+200B) also indicates a potential hyphenation point. Unlike the soft hyphen, however, nothing is inserted if the word is split at that point. It is useful, for example, for allowing technical “words” containing underscores to be split after the underscore without any insertion. When the spaces in a line are stretched to justify a line, zero width spaces are not affected.

If no explicit hyphen is found, SDoP tries automatic hyphenation. Both forms of hyphenation can be disabled by a processing instruction (☞ 2.11). This is a local processing parameter, so it can be turned on and off as often as you like throughout the document. SDoP uses a dictionary for automatic hyphenation (☞ 4.3).

## 5.5 Fine adjustments to lines

Once it has formatted a paragraph, SDoP does a second pass over the lines to look for instances where a very tight, but not hyphenated, line is followed by a very loose line. If it finds such a pair of lines, it checks to see whether it is possible to move the last word from the first line onto the second line without making the first line unacceptably loose. This is a rare occurrence, but when it works it can make the paragraph look a lot nicer.

## 5.6 Ligatures

SDoP automatically uses the single character “fi” (U+FB01) whenever the letter “f” is followed by the letter “i” and the font is not monospaced and does contain the ligature character.

## 5.7 Kerning

SDoP uses the kerning information in a font’s AFM file to move certain pairs of characters closer together or further apart, in order to improve the appearance of the text. There is a local processing setting that can suppress this for all or part of a document (☞ 2.11).

## 5.8 The <emphasis> element

The <emphasis> element with no attributes specifies italic; the attribute `role="bold"` specifies a boldface font. Also supported is `role="roman"`, which can be useful for overriding a bold or italic font caused by an element such as <varname>.

The `role` attribute can also be used to specify special fonts or coloured text. In the template for the title page (the data file *titletemplate*) the form `role="booktitlen"`, where *n* is a number between 1 and 4, can appear. This selects one of four different sizes of title font.

## 5.9 Using a small font

A normal font, but at a smaller than normal size, can be specified by:

```
<emphasis role="smallfont">
```

The actual size of font can be set by the **font\_small\_main** processing parameter.

## 5.10 Using exotic fonts

In the main text, the `role` attribute can be used to specify the use of a special font. The format for this is:

```
<emphasis role="exotic-font-name/size/leading">
```

The leading (“ledding”, not “leeding”) is a minimum amount of extra vertical white space that is inserted above each line that contains characters from this font. (The default font size is 11 points, with 1 point of leading.)

If the size and leading are omitted, the values from the current font are used. The size can be given as zero to mean ‘the current size’ if just additional leading is needed. Fonts with standard encoding are re-encoded as Unicode; for others, only characters in the range 0–255 are supported, in their native encoding. For example, to include a treble clef at a 9-point size from the musical font that is part of *Philip’s Music Writer*, you could use:

```
<emphasis role="PMW-Music/9">!/</emphasis>
```

The treble clef is encoded as character 33 in that font, which is an exclamation mark in Unicode.

If you use a special font in this way, you must ensure that its AFM file is available in the *fontmetrics* subdirectory of a directory that you pass to SDoP via the **-S** command line option.

## 5.11 Specifying coloured text

You can specify that text is to be coloured like this:

```
<emphasis role="rgb=r,g,b">...</emphasis>
```

Where the colour is defined by the three red-green-blue values, in the range 0 to 1. For example:

```
This is a <emphasis role="rgb=1,0,0">red</emphasis> word.
```

## 5.12 Change bars

SDoP supports the `revisionflag` option on elements, but the only value that is recognized is “changed”. The effect is to print a vertical change bar at the right hand side of the affected text.

## 5.13 Itemized lists

If no `mark` option is present on an `<itemizedlist>` element, SDoP uses a bullet (U+00B7) for top level lists, a dash (U+2212) for the first nested level, and a small circle (the letter “o”) for the next level. SDoP recognizes the words `bullet`, `dash`, and `opencircle` as values for `mark`. You can also give a Unicode code point, for example, U+261E.

## 5.14 Tables

Space is left at the beginning and end of the entries in each column of a table. This reduces the column widths as specified in the XML. Vertical space is left between rows, above the top of the frame, and below the bottom of the frame (in cases where a top and bottom frame line is drawn). The thickness of the frame is in principle variable, but none of these values are yet adjustable (though they may be in a later release):

Left entry space	5pt
Right entry space	5pt
Row separation space	5pt
Top frame space	0pt
Bottom frame space	4pt
Frame thickness	0.5pt

Tables are by default set flush left on the page, but they can be indented by setting the **`table_indent`** processing parameter. There are a number of other processing parameters that can be used to control the way table titles are handled (☞ 2.11). A further parameter can be used to alter the way SDoP diagnoses table cells that overflow. Setting:

```
<?sdoP table_warn_overflow="never"?>
```

suppresses all cell overflow warnings, whereas setting:

```
<?sdoP table_warn_overflow="overprint"?>
```

suppresses cell overflow warnings in two circumstances when there is no actual text collision, provided that no separator is being printed between the relevant columns:

- (1) The overflowing cell is not the last on the line, is left justified, and the next cell is not left justified.
- (2) The overflowing cell is not the first on the line, is right justified, and the previous cell is not right justified.

Both of these settings affect only what follows; they can be changed during the course of a document.

There is support for aligning cells by reference to a specific character (for example, to align monetary values on the decimal point). This takes the form of support for `align="char"` and the `char` and `charoff` attributes of the `<tgroup>`, `<colspec>`, and `<entry>` elements. The default values for `char` and `charoff` are "." and "50", respectively. The value of `char` can be an entity name. If the aligning character is not found in a cell's text, SDoP behaves as if it were present at the end of

the string. If character alignment would cause overflow or underflow, an error is generated and right or left alignment (respectively) is used instead.

## 5.15 Footnotes

Up to nine footnotes per page are fully supported, keyed by the numbers 1–9. If there are more than nine footnotes on a page, the lines containing footnote references greater than nine may be poorly spaced. A warning message is output. This is another example of the “simple” approach that SDoP takes.<sup>1</sup> If a `<footnote>` element is at the start of an input line, the newline character at the end of the previous line is ignored, so that the footnote key is hard up against the preceding word.

## 5.16 Illustrations and figures

The `<textobject>` and `<imageobject>` elements within a `<mediaobject>` element are supported. For images, the `<imagedata>` element supports only the `fileref` attribute for the source of the image data and the `format` attribute to specify a format. If no format is given, the file name extension is used. The only formats currently recognized are EPS for encapsulated PostScript and JPEG or JPG for a JPEG image (when JPEG support is available).

SDoP recognizes the `align`, `depth`, `width`, `scale`, and `scalefit` attributes on an `<imagedata>` element. If either of the width and depth are not given, they are taken from the bounding box data of the image, if available. If an explicit depth is greater than the bounding box depth, the image is vertically centred in the given depth.

If `scalefit` is set to 1, the image is scaled to fit the explicit width, if given, otherwise to the explicit depth, if given, otherwise to the page width.

Occasionally, an image may not appear exactly where you want it; this can happen if the bounding box values in the data are not quite right. A facility for adjusting the position of an image is therefore provided. If the `<imageobject>` element has a `role` attribute that consists of one or two dimensions, comma separated, they are used to adjust the position of the image. The values may be negative. The first dimension moves the image right or left, and the second, if present, up or down.

There is support for the `<figure>` element, and there are a number of processing parameters that can be used to control the format and alignment of figure titles (☞ 2.11). However, figures are always set inline; there is no support for floating figures.

## 5.17 Indexes

A document may contain multiple indexes, interspersed with chapters and appendices if required. The entries for a particular index are identified by the use of the `role` option on `<indexterm>` and `<index>` elements. Unlike some other DocBook processors, SDoP respects font-changing elements such as `<emphasis>` and `<filename>` in index entries. It is possible to alter the order in which index elements sort by overriding the default collation file (☞ 4.8).

Indexes are printed in two columns, using slightly smaller fonts than the body of the document by default. An index is omitted if there are no relevant entries. If there are at least two different leading characters in the index entries, headings are inserted before those that start with a symbol, those that start with a digit, and those that start with each individual letter. If all the entries start with the same character, no heading is inserted (this might be the case in an index of variables, for example, where they all start with \$). Headings can be manually disabled by setting the processing parameter **index\_headings** to `no`.

Apart from the index sorting order, which is controlled by one of SDoP’s data files (☞ 4.8), and the list of characters that are ignored when sorting (set by the **index\_sort\_omit** processing instruction), no other characteristics of indexes are at present configurable.

---

<sup>1</sup> Footnote numbering is a chicken-and-egg problem. Paragraphs have to be formatted before they can be assigned to pages. SDoP assumes that a footnote reference will be a single digit. Only after the page breaks are determined can the footnotes be numbered, and an extra digit alters the length of the line. If the line is justified and the extra digit can be accommodated by reducing the stretch for each space, all is well, but this cannot be guaranteed.

## 6. Making Slides

As well as generating PostScript for printed page images, SDoP can be used to generate PostScript that, when converted to a PDF, can be used for slideshows. To do this you need to adjust some of the processing parameters. The following is an example:

```
<?sdop background_rgb="0.8,0.8,1.0"
      page_foot_line_width="840"
      page_full_length="620"
      page_line_width="810"
      paper_size="900x660"
      font_main="26"
      font_mainmono="24"
      font_chapter="32"
      font_toc="18,1,serif"
      foot_centre_recto=""
      foot_centre_verso=""
      foot_right_recto="&arabicpage;"
      foot_right_verso="&arabicpage;"
      ilist_indent="24,24,0"
      margin_bottom="20"
      margin_left_recto="45"
      margin_left_verso="45"
      numbertitles="no"
      toc_sections="no"
?>
```

This sets up a pale background, adjusts sizes and widths, and arranges that slide (page) numbers are in the bottom lefthand corner. Each slide can then be defined as a “chapter”.

## 7. Supported DocBook elements

The named entities and Unicode characters that SDoP supports are listed in the following two chapters. As well as elements and entities, SDoP recognizes these special XML constructs:

<code>&lt;!. . . !&gt;</code>	XML heading: ignored
<code>&lt;!--. . .--&gt;</code>	XML comment: ignored
<code>&lt;![CDATA[. . .]]&gt;</code>	Literal character data

The following table lists the DocBook elements that are supported, in whole or in part, by SDoP. In some cases, though the element is recognized, it causes no change of processing. Examples of elements of this type are `<abbrev>`, `<acronym>`, and `<trademark>`. Elements that are defined for Simplified DocBook are marked with a dagger.

Element	Comment
† abbrev	Ignored
† abstract	Ignored
† acronym	Ignored
address	
† affiliation	
† appendix	Supports id
† article	
† articleinfo	
† attribution	
† audiodata	Ignored
† audioobject	Ignored
† author	
† authorblurb	Ignored
† authorinitials	
† blockquote	
book	
bookinfo	
† caption	
chapter	Supports id
† citetitle	Treated as <code>&lt;emphasis&gt;</code>
colophon	
† colspec	Supports align, char, charoff, colwidth, colsep
† command	
† computeroutput	Treated as <code>&lt;literal&gt;</code>
† copyright	Supported in <code>&lt;articleinfo&gt;</code> and <code>&lt;bookinfo&gt;</code>
† corpauthor	
† date	
† edition	
† editor	
† email	Italic by default
† emphasis	Supports role
† entry	Supports align, char, charoff
† epigraph	Works like <code>&lt;blockquote&gt;</code>
† example	Supports id
† figure	Supports id
† filename	
† firstname	
† footnote	
† footnoteref	Only on same page as the footnote
formalpara	
function	
† holder	
† honorific	

† imagedata	Supports align, depth, fileref, format, scale, scalefit, width
† imageobject	
index	Supports role
indexterm	Supports class, id, role
informalfigure	
† informaltable	Supports frame, colsep, rowsep
† inlinemediaobject	Treated as <mediaobject>
† issuenum	
† itemizedlist	Supports mark
† jobtitle	
† keyword	Ignored
† keywordset	Ignored
† legalnotice	
† lineage	
† lineannotation	Uses a small italic font
† link	Ignored
† listitem	
† literal	
† literallayout	Supports class
† mediaobject	
† note	Works like <blockquote>
† objectinfo	Ignored
† option	
† orderedlist	Supports numeration
† orgname	
† othercredit	
† othername	
† para	
† phrase	
preface	
primary	
† programlisting	Treated as <screen>
† pubdate	
† publishername	
† quote	
† releaseinfo	
† replaceable	Italic by default
† revdescription	
† revhistory	
† revision	
† revnumber	
† revremark	Ignored
† row	Supports rowsep
screen	
secondary	
† section	Supports id
† sectioninfo	Ignored
sectn	Treated as <section>
see	
seealso	
† sidebar	Works like <blockquote>
simpara	
† subject	Ignored
† subjectset	Ignored
† subjectterm	Ignored
subscript	
† subtitle	For articles, chapters, appendixes, and indexes



superscript	
† surname	
† systemitem	Ignored
† table	Supports frame, colsep, rowsep
† tbody	
† term	
tertiary	
† textobject	
† tfoot	
† tgroup	Supports align, char, charoff, cols, colsep, rowsep
† thead	
† title	
† titleabbrev	
† trademark	Ignored
† ulink	Supports url
† userinput	Monospaced by default
† variablelist	
† varlistentry	
varname	
† videodata	Ignored
† videoobject	Ignored
† volumenum	
† xref	Supports linkend
† year	

In an <article> element, <id> and <class> attributes are recognized, but ignored.

The Simplified DocBook elements that are not supported are <authorgroup>, <bibliobox>, <entrytbl>, and <spanspec>.

## 8. Supported DocBook entities

Individual data characters can always be specified by their numerical code points using the entity notation `&#n;` (where *n* is a decimal number) or `&#xn;` (where *n* is a hexadecimal number). The latter form is more common because Unicode code points are normally listed in hexadecimal. For example, a copyright symbol can be specified as `&#xA9;`.

SDoP also recognizes a number of named entities that represent special characters. They are shown in the table below. Some of these have no corresponding character in the standard PostScript fonts. Such characters are output as `␣` and by default a warning message is output the first time any one of them is encountered. These warnings can be suppressed by the **-qc** command line option, or by setting the **warn\_unsupported\_characters** processing parameter to “no”.

Additional special entity names are recognized in header, footer, title page, and table of contents templates (see 4.7, 4.4, 4.6).

Name	Code point	Character
<code>&amp;AElig;</code>	U+00C6	Æ
<code>&amp;Aacute;</code>	U+00C1	Á
<code>&amp;Abreve;</code>	U+0102	
<code>&amp;Acirc;</code>	U+00C2	Â
<code>&amp;Agrave;</code>	U+00C0	À
<code>&amp;Amacr;</code>	U+0100	
<code>&amp;Aogon;</code>	U+0104	
<code>&amp;Aring;</code>	U+00C5	Å
<code>&amp;Atilde;</code>	U+00C3	Ã
<code>&amp;Auml;</code>	U+00C4	Ä
<code>&amp;Cacute;</code>	U+0106	
<code>&amp;Ccaron;</code>	U+010C	
<code>&amp;Ccedil;</code>	U+00C7	Ç
<code>&amp;Ccirc;</code>	U+0108	␣
<code>&amp;Cdot;</code>	U+010A	␣
<code>&amp;Dagger;</code>	U+2021	†
<code>&amp;Dcaron;</code>	U+010E	
<code>&amp;Dstrok;</code>	U+0110	
<code>&amp;ENG;</code>	U+014A	␣
<code>&amp;ETH;</code>	U+00D0	Ð
<code>&amp;Eacute;</code>	U+00C9	É
<code>&amp;Ecaron;</code>	U+011A	
<code>&amp;Ecirc;</code>	U+00CA	Ê
<code>&amp;Edot;</code>	U+0116	
<code>&amp;Egrave;</code>	U+00C8	È
<code>&amp;Emacr;</code>	U+0112	
<code>&amp;Eogon;</code>	U+0118	
<code>&amp;Euml;</code>	U+00CB	Ë
<code>&amp;Euro;</code>	U+20AC	€
<code>&amp;Gbreve;</code>	U+011E	
<code>&amp;Gcedil;</code>	U+0122	
<code>&amp;Gcirc;</code>	U+011C	␣
<code>&amp;Gdot;</code>	U+0120	␣
<code>&amp;Hcirc;</code>	U+0124	␣
<code>&amp;Hstrok;</code>	U+0126	␣
<code>&amp;IJlig;</code>	U+0132	␣
<code>&amp;Iacute;</code>	U+00CD	Í
<code>&amp;Icirc;</code>	U+00CE	Î
<code>&amp;Idot;</code>	U+0130	
<code>&amp;Igrave;</code>	U+00CC	Ì
<code>&amp;Imacr;</code>	U+012A	

&Iogon;	U+012E	
&Itilde;	U+0128	ı
&Iuml;	U+00CF	İ
&Jcirc;	U+0134	ı
&Kcedil;	U+0136	
&Lacute;	U+0139	
&Lcaron;	U+013D	
&Lcedil;	U+013B	
&Lmidot;	U+013F	ı
&Lstrok;	U+0141	Ł
&Nacute;	U+0143	
&Ncaron;	U+0147	
&Ncedil;	U+0145	
&Ntilde;	U+00D1	Ñ
&OElig;	U+0152	Œ
&Oacute;	U+00D3	Ó
&Ocirc;	U+00D4	Ô
&Odblac;	U+0150	
&Ograve;	U+00D2	Ò
&Omacr;	U+014C	
&Oslash;	U+00D8	Ø
&Otilde;	U+00D5	Õ
&Ouml;	U+00D6	Ö
&Racute;	U+0154	
&Rcaron;	U+0158	
&Rcedil;	U+0156	
&Sacute;	U+015A	
&Scaron;	U+0160	Š
&Scedil;	U+015E	
&Scirc;	U+015C	ı
&THORN;	U+00DE	Þ
&Tcaron;	U+0164	
&Tcedil;	U+0162	ı
&Tstrok;	U+0166	ı
&Uacute;	U+00DA	Ú
&Ubreve;	U+016C	ı
&Ucirc;	U+00DB	Û
&Udblac;	U+0170	
&Ugrave;	U+00D9	Ù
&Umacr;	U+016A	
&Uogon;	U+0172	
&Uring;	U+016E	
&Utilde;	U+0168	ı
&Uuml;	U+00DC	Ü
&Wcirc;	U+0174	ı
&Yacute;	U+00DD	Ý
&Ycirc;	U+0176	ı
&Yuml;	U+0178	Ÿ
&Zacute;	U+0179	
&Zcaron;	U+017D	Ž
&Zdot;	U+017B	
&aacute;	U+00E1	á
&abreve;	U+0103	
&acirc;	U+00E2	â
&aelig;	U+00E6	æ
&agrave;	U+00E0	à
&amacr;	U+0101	
&amp;	U+0026	&

&aogon;	U+0105	
&aring;	U+00E5	å
&atilde;	U+00E3	ã
&auml;	U+00E4	ä
&brvbar;	U+00A6	
&acute;	U+0107	
&ccaron;	U+010D	
&ccedil;	U+00E7	ç
&ccirc;	U+0109	ç
&cdot;	U+0A0B	¤
&cent;	U+00A2	¢
&check;	U+2713	✓
&clubs;	U+2663	♣
&copy;	U+00A9	©
&cross;	U+2717	✕
&curren;	U+00A4	¤
&dagger;	U+2020	†
&darr;	U+2193	↓
&dcaron;	U+010F	
&deg;	U+00B0	°
&diams;	U+2666	◆
&divide;	U+00F7	÷
&dstrok;	U+0111	
&eacute;	U+00E9	é
&ecaron;	U+011B	
&ecirc;	U+00EA	ê
&edot;	U+0117	
&egrave;	U+00E8	è
&emacr;	U+0113	
&eng;	U+014B	¤
&eogon;	U+0119	
&eth;	U+00F0	ð
&euml;	U+00EB	ë
&filig;	U+FB01	fi
&fllig;	U+FB02	fl
&frac12;	U+00BD	½
&frac14;	U+00BC	¼
&frac34;	U+00BE	¾
&gbreve;	U+011F	
&gcirc;	U+011D	¤
&gdot;	U+0121	¤
&gt;	U+003E	>
&half;	U+00BD	½
&hcirc;	U+0125	¤
&hearts;	U+2665	♥
&hellip;	U+2026	...
&hstrok;	U+0127	¤
&iacute;	U+00ED	í
&icirc;	U+00EE	î
&iexcl;	U+00A1	¡
&igrave;	U+00EC	ì
&ijlig;	U+0133	¤
&imacr;	U+012B	
&inodot;	U+0131	ı
&iogon;	U+012F	
&iquest;	U+00BF	¿
&itilde;	U+0129	¤
&iuml;	U+00EF	ï

&jcirc;	U+0135	Ƶ	
&kcedil;	U+0137		
&kgreen;	U+0138	ƶ	
&lacute;	U+013A		
&laquo;	U+00AB	«	
&larr;	U+2190	←	
&lcaron;	U+013E		
&lcedil;	U+013C		
&ldquo;	U+201C	“	
&ldquor;	U+201E	”	
&lmidot;	U+0140	ƴ	
&loz;	U+25CA		
&lsquo;	U+2018	‘	
&lsquor;	U+201A	,’	
&lstrok;	U+0142	‡	
&lt;	U+003C	<	
&malt;	U+2720	⌘	
&mdash;	U+2014	—	
&micro;	U+00B5	μ	
&middot;	U+00B7	•	
&mldr;	U+2026	...	
&napost;	U+0144		
&napos;	U+0149	Ƶ	
&nbsp;	U+00A0		hard space
&ncaron;	U+0148		
&ncedil;	U+0146		
&ndash;	U+2013	—	
&not;	U+00AC	¬	
&ntilde;	U+00F1	ñ	
&oacute;	U+00F3	ó	
&ocirc;	U+00F4	ô	
&odblac;	U+0151		
&oelig;	U+0153	œ	
&ograve;	U+00F2	ò	
&omacr;	U+014D		
&ordf;	U+00AA	ª	
&ordm;	U+00BA	º	
&oslash;	U+00F8	ø	
&otilde;	U+00F5	õ	
&ouml;	U+00F6	ö	
&para;	U+00B6	¶	
&phone;	U+260E	☎	
&plusmn;	U+00B1	±	
&pound;	U+00A3	£	
&racute;	U+0155		
&raquo;	U+00BB	»	
&rarr;	U+2192	→	
&rcaron;	U+0159		
&rcedil;	U+0157		
&rdquo;	U+201D	”	
&rdquor;	U+201D	”	(same as &rdquo;)
&reg;	U+00AE	®	
&rsquo;	U+2019	,	
&rsquor;	U+2019	,	(same as &rsquo;)
&sacute;	U+015B		
&scaron;	U+0161	š	
&scedil;	U+015F		
&scirc;	U+015D	ƶ	

&sect;	U+00A7	§
&sect;	U+2736	✱
&shy;	U+00AD	-
&spades;	U+2660	♠
&sup1;	U+00B9	¹
&sup2;	U+00B2	²
&sup3;	U+00B3	³
&szlig;	U+00DF	ß
&tcaron;	U+0165	ř
&tcedil;	U+0163	ţ
&thorn;	U+00FE	þ
&times;	U+00D7	×
&trade;	U+2122	™
&tstrok;	U+0167	ř
&uacute;	U+00FA	ú
&uarr;	U+2191	↑
&ubreve;	U+016D	ř
&ucirc;	U+00FB	û
&udblac;	U+0171	ű
&ugrave;	U+00F9	ù
&umacr;	U+016B	ŕ
&uogon;	U+0173	
&uring;	U+016F	
&utilde;	U+0169	ř
&uuml;	U+00FC	ü
&wcirc;	U+0175	ř
&yacute;	U+00FD	ý
&ycirc;	U+0177	ř
&yen;	U+00A5	¥
&yuml;	U+00FF	ÿ
&zacute;	U+017A	
&zcaron;	U+017E	ž
&zdot;	U+017C	

## 9. Supported Unicode characters

SDoP supports all the Unicode characters that are in the common Adobe PostScript fonts, plus some additional characters from the *Symbol* and *ZapfDingbats* fonts. You do not have to specify anything to cause SDoP to use the special fonts; they are used automatically for characters that are not in the ordinary fonts. When it encounters a character for which it does not have a PostScript equivalent, SDoP prints the rarely-used “currency symbol”, ¤.

All printing characters in the *C0 Controls and Basic Latin* code page (U+0020–U+007E, corresponding to ASCII) and in the *C1 Controls and Latin-1 Supplement* code page (U+00A0–U+00FF, corresponding to ISO 8859-1) are supported, including the “hard space” (U+00A0) and the “soft hyphen” (U+00AD). SDoP also supports the “zero width space” (U+200B). Most characters in the *Latin Extended-A* page (U+0100–U+017F) are supported. The exceptions (characters not in the PostScript fonts) are shown in the following table:

U+0108	C with circumflex	U+013F	L with middle dot
U+0109	c with circumflex	U+0140	l with middle dot
U+010A	C with dot above	U+0149	n with leading apostrophe
U+010B	c with dot above	U+014A	capital “eng” (N with hook)
U+0114	E with breve	U+014B	lower case “eng” (n with hook)
U+0115	e with breve	U+014E	O with breve
U+011C	G with circumflex	U+014F	o with breve
U+011D	g with circumflex	U+015C	S with circumflex
U+0120	G with dot above	U+015D	s with circumflex
U+0121	g with dot above	U+0162	T with cedilla
U+0124	H with circumflex	U+0163	t with cedilla
U+0125	h with circumflex	U+0166	T with stroke
U+0126	H with stroke	U+0167	t with stroke
U+0127	h with stroke	U+0168	U with tilde
U+0128	I with tilde	U+0169	u with tilde
U+0129	i with tilde	U+016C	U with breve
U+012C	I with breve	U+016D	u with breve
U+012D	i with breve	U+0174	W with circumflex
U+0132	IJ ligature	U+0175	w with circumflex
U+0133	ij ligature	U+0176	Y with circumflex
U+0134	J with circumflex	U+0177	y with circumflex
U+0135	j with circumflex	U+017F	long s
U+0138	small “kra” (Greenlandic)		

The following tables show all the characters that are supported by SDoP. The first table shows those from the first three Unicode pages (U+0000–U+017F):

U+0020	space	U+00A8	¨	U+00E1	á	U+012A
U+0021	!	U+00A9	©	U+00E2	â	U+012B
U+0022	"	U+00AA	ª	U+00E3	ã	U+012E
U+0023	#	U+00AB	«	U+00E4	ä	U+012F
U+0024	\$	U+00AC	¬	U+00E5	å	U+0130
U+0025	%	U+00AD	-	U+00E6	æ	U+0131
U+0026	&	U+00AE	®	U+00E7	ç	U+0136
U+0027	'	U+00AF	¯	U+00E8	è	U+0137
U+0028	(	U+00B0	°	U+00E9	é	U+0139
U+0029	)	U+00B1	±	U+00EA	ê	U+013A
U+002A	*	U+00B2	²	U+00EB	ë	U+013B
U+002B	+	U+00B3	³	U+00EC	ì	U+013C
U+002C	,	U+00B4	´	U+00ED	í	U+013D
U+002D	-	U+00B5	µ	U+00EE	î	U+013E
U+002E	.	U+00B6	¶	U+00EF	ï	U+0141
U+002F	/	U+00B7	•	U+00F0	ð	U+0142
U+0030	0	U+00B8	¸	U+00F1	ñ	U+0143

U+0031	1	U+00B9	¹	U+00F2	ð	U+0144	
U+0032	2	U+00BA	º	U+00F3	ó	U+0145	
U+0033	3	U+00BB	»	U+00F4	ô	U+0146	
U+0034	4	U+00BC	¼	U+00F5	õ	U+0147	
U+0035	5	U+00BD	½	U+00F6	ö	U+0148	
U+0036	6	U+00BE	¾	U+00F7	÷	U+014C	
U+0037	7	U+00BF	¿	U+00F8	ø	U+014D	
U+0038	8	U+00C0	À	U+00F9	ù	U+0150	
U+0039	9	U+00C1	Á	U+00FA	ú	U+0151	
U+003A	:	U+00C2	Â	U+00FB	û	U+0152	Œ
U+003B	;	U+00C3	Ã	U+00FC	ü	U+0153	œ
U+003C	<	U+00C4	Ä	U+00FD	ý	U+0154	
U+003D	=	U+00C5	Å	U+00FE	þ	U+0155	
U+003E	>	U+00C6	Æ	U+00FF	ÿ	U+0156	
U+003F	?	U+00C7	Ç	U+0100		U+0157	
U+0040	@	U+00C8	È	U+0101		U+0158	
U+0041	A	U+00C9	É	U+0102		U+0159	
	...	U+00CA	Ê	U+0103		U+015A	
U+005A	Z	U+00CB	Ë	U+0104		U+015B	
U+005B	[	U+00CC	Ì	U+0105		U+015E	
U+005C	\	U+00CD	Í	U+0106		U+015F	
U+005D	]	U+00CE	Î	U+0107		U+0160	Š
U+005E	^	U+00CF	Ï	U+010C		U+0161	š
U+005F	~	U+00D0	Ð	U+010D		U+0164	
U+0060	`	U+00D1	Ñ	U+010E		U+0165	
U+0061	a	U+00D2	Ò	U+010F		U+016A	
	...	U+00D3	Ó	U+0110		U+016B	
U+007A	z	U+00D4	Ô	U+0111		U+016E	
U+007B	{	U+00D5	Õ	U+0112		U+016F	
U+007C		U+00D6	Ö	U+0113		U+0170	
U+007D	}	U+00D7	×	U+0116		U+0171	
U+007E	~	U+00D8	Ø	U+0117		U+0172	
U+00A0	space	U+00D9	Ù	U+0118		U+0173	
U+00A1	¡	U+00DA	Ú	U+0119		U+0178	Ÿ
U+00A2	¢	U+00DB	Û	U+011A		U+0179	
U+00A3	£	U+00DC	Ü	U+011B		U+017A	
U+00A4	¤	U+00DD	Ý	U+011E		U+017B	
U+00A5	¥	U+00DE	Þ	U+011F		U+017C	
U+00A6	¦	U+00DF	ß	U+0122		U+017D	Ž
U+00A7	§	U+00E0	à	U+0123		U+017E	ž

The next table shows the other supported characters that are taken from the normal PostScript fonts if possible (some older fonts lack some of these characters):

U+0218		U+0328	ˆ	U+2020	†	U+2211	
U+0219		U+0394	ˆ	U+2021	‡	U+2212	–
U+021A		U+2013	—	U+2026	...	U+221A	
U+021B		U+2014	—	U+2027	·	U+2260	
U+0302	^	U+2018	‘	U+2031	‰	U+2264	
U+0303	~	U+2019	’	U+2039	<	U+2265	
U+0306	˘	U+201A	,	U+203A	>	U+25CA	
U+0307	˙	U+201C	“	U+2044	/	U+FB01	fi
U+030B	˘	U+201D	”	U+20AC	€	U+FB02	fl
U+0326		U+201E	„	U+2122	™		

The next table shows characters that are supported by using the *Symbol* font (in some cases, only if the normal font lacks them):

U+0391	A	U+03BA	κ	U+21B5	↵	U+2283	⊃
U+0392	B	U+03BB	λ	U+21D0	⇐	U+2284	⊄



U+0393	Γ	U+03BC	μ	U+21D1	↑↑	U+2286	⊆
U+0394		U+03BD	ν	U+21D2	⇒	U+2287	⊂
U+0395	E	U+03BE	ξ	U+21D3	⇅	U+2295	⊕
U+0396	Z	U+03BF	ο	U+21D4	↔	U+2297	⊗
U+0397	H	U+03C0	π	U+2200	∀	U+22A5	]
U+0398	Θ	U+03C1	ρ	U+2202		U+22C0	^
U+0399	I	U+03C2	ς	U+2203	∃	U+22C1	∨
U+039A	K	U+03C3	σ	U+2205	∅	U+22C5	.
U+039B	Λ	U+03C4	τ	U+2207	∇	U+2320	{
U+039C	M	U+03C5	υ	U+2208	∈	U+2321	}
U+039D	N	U+03C6	φ	U+2209	∉	U+2329	<
U+039E	Ξ	U+03C7	χ	U+220D	ə	U+232A	>
U+039F	O	U+03C8	ψ	U+220F	Π	U+239B	{
U+03A0	Π	U+03C9	ω	U+2211		U+239C	}
U+03A1	P	U+03D1	ϑ	U+2215	/	U+239D	{
U+03A3	Σ	U+03D2	Υ	U+221A		U+239E	}
U+03A4	T	U+03D5	φ	U+221D	∞	U+239F	{
U+03A5	Y	U+03D6	ϖ	U+221E	∞	U+23A0	}
U+03A6	Φ	U+12AA		U+2220	∠	U+23A1	{
U+03A7	X	U+2032	'	U+2229	∩	U+23A2	}
U+03A8	Ψ	U+2033	"	U+222A	∪	U+23A3	{
U+03A9	Ω	U+20AC	€	U+222B	∫	U+23A4	}
U+03B1	α	U+2111	ℑ	U+2234	∴	U+23A5	{
U+03B2	β	U+2118	℔	U+223C	{	U+23A6	}
U+03B3	γ	U+211C	℔	U+2245	≡	U+23A7	{
U+03B4	δ	U+2135	℔	U+2248	≈	U+23A8	}
U+03B5	ε	U+2190	←	U+2260		U+23A9	{
U+03B6	ζ	U+2191	↑	U+2261	≡	U+23AB	}
U+03B7	η	U+2192	→	U+2264		U+23AC	{
U+03B8	θ	U+2193	↓	U+2265		U+23AD	}
U+03B9	ι	U+2194	↔	U+2282	⊂	U+23AE	

The final table shows characters that are supported by using the *ZapfDingbats* font:

U+25A0	■	U+2722	✂	U+2751	☐	U+2791	⑧
U+25B2	▲	U+2723	✂	U+2752	☐	U+2792	⑨
U+25BC	▼	U+2724	⌘	U+2756	❖	U+2793	⑩
U+25C6	◆	U+2725	✂	U+2758		U+2794	➔
U+25CA		U+2726	◆	U+2759		U+2798	➤
U+25D7	◐	U+2727	◇	U+275A	■	U+2799	➔
U+260E	☺	U+2729	☆	U+275B	‘	U+279A	➤
U+261B	☞	U+272A	☼	U+275C	’	U+279B	➔
U+261E	☜	U+272B	☆	U+275D	“	U+279C	➔
U+2660	♠	U+272C	☆	U+275E	”	U+279D	➔
U+2663	♣	U+272D	☆	U+2761	☾	U+279E	➔
U+2665	♥	U+272E	★	U+2762	☼	U+279F	➔
U+2666	♦	U+272F	☆	U+2763	☼	U+27A0	➔
U+26AB	●	U+2730	☆	U+2764	♥	U+27A1	➔
U+2701	✂	U+2731	★	U+2765	♣	U+27A2	➤
U+2702	✂	U+2732	★	U+2766	⑧	U+27A3	➤
U+2703	✂	U+2733	*	U+2767	☼	U+27A4	➤
U+2704	✂	U+2734	*	U+2776	①	U+27A5	➔
U+2706	☞	U+2735	☼	U+2777	②	U+27A6	➔
U+2707	☞	U+2736	*	U+2778	③	U+27A7	➔
U+2708	☞	U+2737	*	U+2779	④	U+27A8	➔
U+2709	☞	U+2738	*	U+277A	⑤	U+27A9	➔
U+270C	☞	U+2739	*	U+277B	⑥	U+27AA	➔
U+270D	☞	U+273A	☼	U+277C	⑦	U+27AB	➔
U+270E	☞	U+273B	*	U+277D	⑧	U+27AC	➔

U+270F	☯	U+273C	✱	U+277E	⑨	U+27AD	↩
U+2710	✍	U+273D	✱	U+277F	⑩	U+27AE	↪
U+2711	✍	U+273E	✱	U+2780	①	U+27AF	↩
U+2712	✍	U+273F	✱	U+2781	②	U+27B1	↩
U+2713	✓	U+2740	✱	U+2782	③	U+27B2	▶
U+2714	✓	U+2741	✱	U+2783	④	U+27B3	↔
U+2715	✕	U+2742	✱	U+2784	⑤	U+27B4	↘
U+2716	✕	U+2743	✱	U+2785	⑥	U+27B5	➡
U+2717	✕	U+2744	✱	U+2786	⑦	U+27B6	↗
U+2718	✕	U+2745	✱	U+2787	⑧	U+27B7	↙
U+2719	⊕	U+2746	✱	U+2788	⑨	U+27B8	➡
U+271A	⊕	U+2747	✱	U+2789	⑩	U+27B9	↙
U+271B	⊕	U+2748	✱	U+278A	①	U+27BA	➡
U+271C	⊕	U+2749	✱	U+278B	②	U+27BB	➡
U+271D	†	U+274A	✱	U+278C	③	U+27BC	➡
U+271E	†	U+274B	✱	U+278D	④	U+27BD	➡
U+271F	†	U+274D	○	U+278E	⑤	U+27BE	➡
U+2720	✱	U+274F	□	U+278F	⑥		
U+2721	✱	U+2750	□	U+2790	⑦		

## 10. Maintaining the hyphenation dictionary

SDoP is distributed with a British English hyphenation dictionary in a file called *HyphenData*. This is installed with the other SDoP data files (often in */usr/local/share/sdop/*). The distribution also contains tools that enable you to modify this dictionary, or create entirely different dictionaries. These tools are not installed as user commands – they are built by “make”, but remain in the build directory.

The *buildhy* program reads a file of hyphenated words and adds indexing information on the front so that SDoP can then use it as a hyphenation dictionary. Use it like this:

```
buildhy <inputfile> <outputfile>
```

The input file is a list of words, one per line, in alphabetical order (not counting the hyphens in the sort). The list that is used to create the distributed dictionary is supplied in the file *hyphenlist*. It starts like this:

```
aba-cus
aba-lone
aban-don
aban-doned
aban-doner
aban-don-ment
abase-ment
```

The index that is added to the start of the output file (which is otherwise a copy of the input) is in the form of four-letter entries with the offset of the relevant point in the dictionary in digits after them. The total number of entries in the index is output on the first line. The distributed dictionary starts like this:

```
1541
abdi148
abdu225
abid309
abol413
abor528
abra576
```

You can test a newly-built dictionary using the *hytest* program:

```
hytest <new dictionary> [<input file> [<output file>]]
```

If no input (output) file is given, the standard input (output) is used. A good test for a new dictionary is to take the input that you originally gave to *buildhy*, remove all the hyphens, feed it to *hytest* and check that you get back the original, hyphenated file. If you do not, the most likely cause is that the input file is not in the correct alphabetic order.

# Index

## Symbols

- d option 1
- help option 2
- o option 2
- p option 2, 3
- pf option 2, 3
- q option 2
- qc option 2
- S option 2, 19
- v option 2
- w option 2
- wc option 2

## A

- article
  - title material 3
  - title template 19
- articleinfo** data 17

## B

- background colour 6
- block quotes
  - changing parameters 6
- book
  - title page template 19
  - title pages 3
  - title pages, fonts for 7, 8
- bookinfo** data 17

## C

- change bars 24
- chapter
  - font for title 7
  - skipping heading at start 13
- character alignment 24
- colour
  - background 6
  - text 24
- columns, specifying 15
- command line 1
- contents *see* table of contents

## D

- data files 19
  - specifying location 2
- debugging option 1
- dimensions, format of 6
- DocBook
  - omitted features 1
  - Simplified DocBook 1

## E

- elements
  - supported 27–29
- emphasis** 23–24
- entities
  - for heads and feet 20
  - set from **bookinfo** and **articleinfo** 17
  - supported 30–34

- example
  - format of numeration 13
  - format of title 13

## F

- feet *see* heads and feet
- figures
  - font for titles 7
  - format of numeration 13
  - format of titles 14
  - parameters for 25
- files
  - data for SDoP 19
  - font metrics 19
  - hyphenation dictionary 19
  - including 4
  - index collation sequence 21
  - PostScript header 19
  - template for article title 19
  - template for book title 19
  - templates for heads and feet 20
- font metric files 19
- fonts
  - bold 23
  - changing families 5
  - changing styles 5
  - exotic 23
  - italic 23
  - main text 7
  - metric files 5
  - overall scaling 6
  - roman 23
  - sanserif family 10
  - scaling 10
  - small 23
  - small main text 8
  - specifying for headings 5
  - specifying sets of 5
  - specifying special 23
- footnotes 25
  - fonts for 7
- formal paragraphs
  - font for titles 7
  - indent 8

## H

- headings
  - specifying fonts for 5
- heads and feet 20–21
  - entities for use in 20
  - fonts, specifying 7
  - line width 10
  - templates for 20
  - text for, contents pages 11
  - text for, main pages 14
  - text for, preface pages 15
  - vertical space 8
- hyphen
  - hard 22
  - invisible 22
  - soft 22

- hyphenation
  - dictionary file 19
  - dictionary maintenance 39
  - disabling 14
  - processing details 22
- I**
- identified lists, indent 8
- illustrations, parameters for 25
- including other files 4
- indent
  - block quotes 6
  - formal paragraph 8
  - identified lists 8
  - literal layout 9
  - note 9
  - ordered list 10
  - paragraph 10
  - sidebar 10
  - tables 15
  - term 11
  - variable list 12
  - variable list title 13
- indentation
  - format for changing 6
  - literal blocks 22
- index
  - collating sequence 21
  - disabling headings 14
  - fonts 7
  - parameters for 25
  - sort, omitting characters 9
- input
  - format of 3
  - skipping parts of 4
  - specifying 1
- insertions from **bookinfo** and **articleinfo** 17
- installing SDoP 1
- itemized lists, marks 24

- J**
- JPEG support 1

- K**
- kerning 23
  - disabling 14

- L**
- leading
  - extra 13
- ligatures 23
- literal layout indent 9

- M**
- margin
  - bottom of page 9
  - left of page 9

- N**
- new page, forcing 14
- note
  - indent 9

- note (*continued*)
  - parameters 9
  - title, fonts for 7

- O**
- ordered list
  - format of numeration 10
  - indent 10
- output
  - format of 3
  - specifying 1
- output, selecting pages 2

- P**
- page
  - forcing new 14
  - length 10
  - padding 9
  - selection option 2
  - width 10
- paper size 10
- paragraph indent 10
- PDF bookmarks 3
- PostScript
  - /PageSize setting 10
  - header file 19
- preface, heads and feet for 20
- processing instructions 4–16
  - parameter types 4
- processing parameters
  - global, list of 6–13
  - local, list of 13–16

- S**
- section titles, fonts for 7
- sidebar
  - indent 10
  - parameters 10
  - title, fonts for 8
- Simplified DocBook 1
- slides, parameters for 26
- space
  - extra between lines 13
  - hard 22
  - vertical 22
  - zero-width 22
- subsection titles, fonts for 8

- T**
- table of contents
  - fonts for 8
  - heads and feet for 20
  - parameters 11–12
  - specifying what is included 16
  - specifying what is printed 16
  - template for 19
- tables
  - cell overflow warnings 16
  - indentation 15
  - numeration 15
  - parameters for 24
  - title format 15
  - titles, fonts for 8

- term indent 11
- title numbering, specifying 15
- title page template 19
- typesetting features 22–25

## **U**

- Unicode 3
  - supported characters 35–38

## **V**

- variable list
  - indent 12
  - title indent 13
- version number, output of 2

## **W**

- warnings
  - controlling 13
  - requesting 2
  - suppressing 2