A Beginners' Guide to Lout

Jeffrey H. Kingston

Basser Department of Computer Science
The University of Sydney 2006
Australia

ABSTRACT

This report is a beginners' guide to the Lout document formatting system. It shows in a very practical, step-by-step way how to produce documents with paragraphs, headings, fonts, displays and lists, footnotes, numbered sections, references, tables and figures, and so on, using the DocumentLayout package (Version 2), which is designed for producing simple documents, technical reports, and books. The underlying principles of Lout are not explained here.

27 July, 1992

A Beginners' Guide to Lout

Jeffrey H. Kingston

Basser Department of Computer Science
The University of Sydney 2006
Australia

1. Introduction

The Lout document formatting system has been designed with the needs of the ordinary user very much in mind. Although the features of Lout are virtually endless, and include mathematical equations, diagrams made from lines and shapes, bibliographic databases, and so on, the system is very simple to use.

Document formatting with Lout begins with the creation of a file like the following:

```
@Doc @Text @Begin
@Heading { Introduction }
@PP
For Virginia Woolf, @I Middlemarch
was ‘the magnificent book which
for all its imperfections is one
of the few English novels written
for grown-up people.’
@End @Text
```

Ordinary words are mixed with symbols of special meaning, such as @Doc, @Text, @Heading, and @I. Most symbols begin with @, but some, like { and }, do not. It doesn't matter where the lines end: the end of a line is treated the same as a single space.

This file is then processed by the Basser Lout interpreter, for example by typing

```
lout -idoc simple > simple.ps
```

on the Unix[1] operating system, assuming that the file's name is simple. The -idoc part instructs Lout to include a special *setup file* called doc, in which the symbols are defined. The output is the PostScript[1] file simple.ps, which is suitable for printing on most laser printers and many other devices. When printed, it will appear like this:

> **Introduction**
>
> For Virginia Woolf, *Middlemarch* was ‘the magnificent book which for all its imperfections is one of the few English novels written for grown-up people.’

The meaning of each symbol is now clear. @Doc @Text @Begin and @End @Text have no visible effect, but they always bracket the document as a whole. @Heading makes the following thing into a heading; there is also @MajorHeading for producing a large heading, and @MinorHeading for a subheading. The @PP symbol starts a paragraph whose first line is indented; also available are @LP which omits the indent, @DP which leaves the amount of vertical space used for displays, and @NP, which starts a new page or column (rarely used, since Lout does this automatically when the old one fills). Finally, @I changes the font of the following thing to *Italic*.

Braces are used to group parts of the document together so that nearby symbols will affect the whole part. For example,

```
@I { Scenes of Clerical Life }
```

[1]Unix is a trademark of AT&T Bell Laboratories.

[1]PostScript is a trademark of Adobe Systems, Inc.

produces *Scenes of Clerical Life*. The braces around Introduction in the example above are not strictly necessary, since a single word does not need to be grouped; but a longer heading would require them. Braces also help when a symbol must be immediately adjacent to another symbol or to a word:

```
QuikWipe{@TradeMark}
```

has result QuikWipe™. Without them Lout would confuse the two. In a similar way, {@I Middlemarch}, produces *Middlemarch*, with the comma not set in Italic.

Characters like {, which normally are symbols, will be treated as ordinary words when enclosed in double quotes. For example, "{" produces {. The characters

```
/   |   &   {   }   #   @   ^
```

should always be so enclosed, since otherwise they will have special effects.

The symbols described above form part of the *DocumentLayout package*, a collection of symbols designed to make day-to-day document formatting easy. This package also provides symbols for producing displays and lists, numbered sections, subsections, and appendices, footnotes, figures, tables, references, and cross references. These are described in later sections of this report. This same package may also be used to produce technical reports and books, as described in Sections 13 and 14.

2. Displays

The @Display symbol displays the following thing in the centre of the page or column:

```
@Display @I Centred
```

has result

Centred

Notice that @I Centred does not have to be grouped within braces; it is already a single thing. Spaces (@DP symbols) are inserted automatically above and below the display, so no paragraph symbols are needed anywhere near the display.

The display can be made to appear at the left margin by using the @LeftDisplay symbol instead of @Display, or indented by using @IndentedDisplay. There are also @CentredDisplay and @CenteredDisplay symbols which are the same as @Display. In general, the word Centred may be spelt Centered wherever it appears.

Each display symbol has a 'raw' version, which means that no space is inserted above or below; the user must therefore add paragraph symbols:

```
... preceding text.
@DP
@RawIndentedDisplay @I Emma
@DP
@RawIndentedDisplay @I
{ Mansfield Park }
@DP
following text ...
```

has result

... preceding text.

Emma

Mansfield Park

following text ...

The point of this particular example is that two consecutive non-raw displays would be separated by two @DP symbols, which is too much. A better way to do this, using a list, will be presented in the next section.

Displays may be *aligned*, which means that nominated points within a sequence of displays are made to appear directly beneath each other. Displays may also be *numbered*, which means that an automatically generated number is placed at the right-hand margin. For example, here is a first display:

$$F_n = F_{n-1} + F_{n-2} \qquad (1)$$

and here is a second display, which is aligned on its = sign with the first, and also numbered in sequence with it:

$$F_n - F_{n-1} = F_{n-2} \qquad (2)$$

Mathematical examples have been chosen because they are the most common aligned and numbered displays; but any kind of display may be aligned or numbered.

Notice that the two displays are centred as a block as well as aligned. Altogether then we have four ways in which displays vary:

- A display can be raw or not raw;
- It can be a @Display, @LeftDisplay, @IndentedDisplay, @CentredDisplay or @CenteredDisplay;
- It can be aligned or not aligned;
- It can be numbered or not numbered.

All possible combinations are allowed. The display that has everything is called

```
@RawCentredAlignedNumberedDisplay
```

By leaving out some or all of Raw, Aligned, and Numbered, and by changing or leaving out Centred, we get all these combinations.

When aligned displays are used, it is necessary to indicate where the aligned group begins and ends, by inserting @BeginAlignedDisplays just before the first, and @EndAlignedDisplays just after the last. The alignment points are indicated by preceding them by the symbol ^. Numbered displays are similarly bracketed by @BeginNumberedDisplays and @EndNumberedDisplays. So then, with the help of the @Eq equation formatting package [2], here is the input for the two displays given earlier:

```
... a first display:
@BeginNumberedDisplays
@BeginAlignedDisplays
@CentredAlignedNumberedDisplay
  @Tag { fibeq }
@Eq { F sub n ^= F sub n-1 + F sub n-2 }
and ... in sequence with it:
@CentredAlignedNumberedDisplay
@Eq { F sub n - F sub n-1 ^= F sub n-2 }
@EndNumberedDisplays
@EndAlignedDisplays
Mathematical examples ...
```

No braces need enclose @Eq { ... } because it is already a single entity. The @Tag { fibeq } part is optional and is explained in Section 9. Alignment and numbering work quite independently; they don't have to start or end together, and there can be non-aligned and non-numbered displays among the others.

@BeginNumberedDisplays has two *options*: subsidiary symbols which modify the result. For example,

```
@BeginNumberedDisplays
  style { [tag] }
  start { 12.5 }
```

will cause the associated numbered displays to be labelled [12.5], [12.6], and so on. The first label is the style option with tag replaced by the start option. Font changes and other symbols are acceptable within the style option. When omitted, the options have default values (tag) and 1 respectively.

Every symbol introduced in this section has an abbreviated form consisting of @ followed by its capital letters only. For example, @BeginNumberedDisplays can be abbreviated to @BND, and the display that has everything to @RCAND.

3. Lists

We have just seen that consecutive displays are awkward to space correctly. Provided they are not aligned or numbered, these are better treated as a *list*:

```
... preceding text.
@IndentedList
@ListItem @I Emma
@ListItem @I { Mansfield Park }
@EndList
following text ...
```

There are @LeftList, @IndentedList, and @CentredList (or @CenteredList) symbols; each item is introduced by @ListItem, and the list ends with @EndList. There may be any number of items; @DP symbols are inserted before, between, and after them.

A variety of automatically generated tags is available for indented lists. Here is the full set, showing the first tag produced:

@NumberedList	1.
@ParenNumberedList	(1)
@RomanList	i.
@ParenRomanList	(i)
@UCRomanList	I.
@ParenUCRomanList	(I)
@AlphaList	a.
@ParenAlphaList	(a)
@UCAlphaList	A.
@ParenUCAlphaList	(A)
@BulletList	•
@StarList	*
@DashList	–

The Roman numerals end at c (100), and the alphabet ends at z (26), but ordinary numbers have no limit. For example,

```
@Heading { Quiz }
@NumberedList
@ListItem { Which American
statesman owned a
two-storey clock? }
@ListItem { Which Yankee
commander cut a swathe of
destruction through the
State of Georgia? }
@EndList
```

has result

Quiz

1. Which American statesman owned a two-storey clock?

2. Which Yankee commander cut a swathe of destruction through the State of Georgia?

Alternatively, the tags may be supplied by the author, using the @TaggedList symbol and its variants @WideTaggedList and @VeryWideTaggedList, which leave a wider indent for the tags:

```
@WideTaggedList
{ 9 a.m. } @TagItem { Breakfast in
the Ipamena Lounge, served with
Irish coffee and fresh croissants. }
{ 10 a.m. } @TagItem { Prof. A. Smith
speaks on ‘The Wealth of Nations.’ }
@EndList
```

has result

9 a.m.	Breakfast in the Ipamena Lounge, served with Irish coffee and fresh croissants.
10 a.m.	Prof. A. Smith speaks on ‘The Wealth of Nations.’

Each @TagItem symbol uses the thing just preceding it as the tag.

Each of these lists also has a ‘raw’ version which omits the preceding and following space. These are mainly used when an item is itself a list:

```
@ParenNumberedList
@ListItem {
  @RawParenRomanList
  @ListItem { MV Nominees,
hereinafter called the vendor, ... }
  @EndList
}
@EndList
```

has result

(1) (i) MV Nominees, hereinafter called the vendor, ...

If @ParenRomanList had been used instead of @RawParenRomanList, (1) and (i) would have appeared on different lines.

In cases where it is desired to have a paragraph symbol following a raw list, owing to problems behind the scenes this symbol should be placed before the first @ListItem or @TagItem, not after the @EndList as would naturally be expected.

A finer control over the appearance of lists is obtained with @RawIndentedList and @RawTaggedList. These have *options*: optional subsidiary symbols which modify the result. For example,

```
@RawIndentedList
   style { [tag] }
   indent { 0.5i }
   gap { 0.3v }
   start { 5 }
@ListItem { ... }
@ListItem { ... }
@EndList
```

shows the four options available with @RawIndentedList, namely style, indent, gap, and start. It has result

[5] ...
[6] ...

The style option determines the appearance of each tag, any tag symbol within it being replaced by the number of the item. If numbers are not wanted in the tag, tag may be omitted. The other options determine the indent, the gap between items, and the number of the first item. In general, options may be given in any order, and if omitted will revert to a reasonable standard value. @RawTaggedList works in a similar way, except that tag is replaced by the tag supplied by the @TagItem symbol, and there is no start option. The symbol @DP may be used above or below the list to produce the usual amount of space.

Individual list items are kept together on one page or column. However, a new page may be started between two list items.

Every symbol introduced in this section has an abbreviated form consisting of @ followed by its capital letters only. For example, @RawNumberedList abbreviates to @RNL, and @ListItem to @LI.

4. Fonts and unusual characters

This section explains how to gain access to the many different fonts and unusual characters available with Lout. We have already seen the @I symbol, which changes the font of the following thing to *Italic*. Similarly, there is @B for **Bold**, @S for SMALL CAPITALS,[1] and @R for Roman.

The @Font symbol can be used to get many other fonts (a local expert should be able to supply the full list):

```
{ Helvetica Slope } @Font { Hi there }
```

[1] Owing to problems behind the scenes, if several words are grouped within one @S symbol they will be kept together on one line, so to get small capitals over several lines it is necessary to apply @S to each word individually. This does not happen with other font symbols.

has result

Hi there

Each font has a family name, such as Times, Helvetica, or Courier, and a face name, generally Base, Slope, or Bold. The fonts that were called Roman, Italic, and Bold above are Times Base, Times Slope, and Times Bold. When changing families the new family and a face must be specified, but when changing face within a family just the face name is enough.

The @Font symbol also changes sizes:

```
+5p @Font Hello
-3p @Font Hello
12p @Font Hello
```

has result

Hello Hello Hello

with the first Hello 5 points larger than it would have been otherwise, the second 3 points smaller, and the third in a 12 point font. There are 72 points to one inch, and most documents are set in 10 or 12 point.

There are symbols for some unusual characters that do not appear on keyboards:

``	“
''	”
--	–
---	—
@Bullet	•
@Star	∗
@ParSym	¶
@SectSym	§
@Dagger	†
@DaggerDbl	‡
@CDot	·
@Sterling	£
@Yen	¥
@Florin	ƒ
@Degree	°
@Minute	′
@Second	″
@Lozenge	◊
@Multiply	×
@Divide	÷
@CopyRight	©
@Register	®
@TradeMark	™
@Date	26 February, 1994

These may be used anywhere. The Adobe Systems Symbol font has many more such characters; the Eq equation formatting package [2] has the complete list. For example, @Eq { heart } will give ♥. Accented characters are also available (see Appendix A).

5. Paragraph breaking

Lout takes the words or other things making up a paragraph and fills lines with them. If two words are separated by one space in the input, they will be separated by one space in the output; two spaces in, two spaces out, and so on. The end of a line counts as one space, and a tab character as eight. These spaces are then enlarged to remove ragged line ends.

This process is called *paragraph breaking*, and the enlargement of spaces is *line adjustment*. The @Break symbol, which is most commonly used with displays and list items, affects paragraph breaking:

```
@IndentedDisplay ragged @Break {
This little paragraph will appear with
ragged ends to its lines. }
```

has result

This little paragraph
will appear with ragged
ends to its lines.

when placed in a four centimetre column; line adjustment is turned off. Also available are cragged @Break and rragged @Break, which centre or right-justify each line respectively after breaking.

It is also possible to have paragraphs broken in the output at the same places they are broken in the input, using lines @Break:

```
@CenteredDisplay lines @Break @I {
Teach me to hear Mermaides singing,
Or to keep off envies stinging,
   And finde
   What winde
Serves to'advance an honest minde.
}
```

has result

Teach me to hear Mermaides singing,
Or to keep off envies stinging,
And finde
What winde
Serves to'advance an honest minde.

With lines @Break it makes sense to indent individual lines in the input (except the first), as shown. To centre or right-justify each line, use clines @Break or rlines @Break.

The usual method, where Lout fills and adjusts lines, is called adjust @Break. It has a variant called outdent @Break which inserts a small space at the beginning of each line except the first.

The @Break symbol also controls hyphenation: hyphen @Break turns it on, nohyphen @Break turns it off. For example, ragged breaking is often done without hyphenation, like this:

```
@ID { ragged nohyphen } @Break {
This little paragraph will appear with
ragged ends to its lines.
}
```

To prevent hyphenation in the entire document, see Section 12. To tell Lout where you would prefer a hyphen to be inserted (rarely necessary), use the &- symbol:

```
incent&-iv&-ate
```

If &- occurs directly after a hyphen character, hyphenation will be permitted but no extra hyphen will be inserted.

6. Conditional new page

Occasionally Lout will insert a page break directly following a heading, and this looks very poor. The solution is to precede the heading with the conditional new page symbol @CNP, which checks whether enough space is left in the page or column for a heading and at least two lines of text. If so, @CNP does nothing; if not, @CNP causes a new page or column to be begun, like @NP. The recommended arrangement is

```
end of previous part.
@DP
@CNP
@Heading { A Heading }
@PP
First paragraph of next part ...
```

The @CNP symbol should be preceded by either @DP or @LP, preferably @DP, and this determines the amount of space when the @NP action does not occur. In most cases it will be better to use the @Section symbol described below, which has @CNP included in it, rather than using @CNP directly.

7. Sections

The DocumentLayout package provides sections, like this:

```
some introductory text.
@BeginSections
@Section
   @Title { Introduction }
@Begin
@PP
...
...
@End @Section
@Section
   @Title { Displays and lists }
@Begin
@PP
...
...
@End @Section
@EndSections
```

@BeginSections and @EndSections symbols must bracket the sections, and paragraph symbols are needed at the beginning of each section but not before or after the whole group. The @Title option of @Section is made into a numbered heading, like those in the present document, preceded by a conditional new page symbol (Section 6). The body of each section is enclosed in @Begin and @End @Section, although { and } are sufficient if preferred. The former are more long-winded but less prone to error.

Within sections one may choose to have a sequence of subsections. These are introduced by @BeginSubSections and concluded by @EndSubSections:

```
in the following subsections.
@BeginSubSections
@SubSection
   @Title { ... }
@Begin
@PP
...
@End @SubSection
@SubSection
   @Title { ... }
@Begin
@PP
...
@End @SubSection
@EndSubSections
```

The first subsection of the first section will be numbered 1.1, and so on. There are no sub-subsections. DocumentLayout also provides appendices in an exactly analogous way, using the symbols @BeginAppendices, @Appendix, and @EndAppendices, and subappendices within any appendix using the symbols @BeginSubAppendices, @SubAppendix, and @EndSubAppendices.

8. Footnotes, figures, and tables

A footnote is created by typing

```
@FootNote { Like this. }
```

after the word that the footnote refers to. It will be numbered automatically and placed at the foot of the page or column;[1] or, if space there is insufficient, it may start on or run onto the following page or column. Figures are created in a similar way:

[1] Like this.

```
@Figure
   @Caption { Basser Lout }
@Fig {
   @Box Lout &
   @HArrow { 2c @Wide } &
   @Box PostScript
}
```

The @Figure symbol places the figure (which in this example is created using the Fig figure drawing package [3]) at the top of the following column or page, labelled by the @Caption option and automatically numbered. Unlike footnotes, figures will not break across several pages; each figure must fit on one page.

Tables are obtained in the same way using @Table instead of @Figure.

9. Cross references

Cross references like ‘see page 9’ are a useful feature of documents, but they are troublesome to authors, since, as the document is revised, the thing on page 9 might find itself on page 12 and the cross reference must be changed. Lout has a simple solution to this problem: instead of writing the page number directly, write

```
see page {@PageOf taxinfo}
```

instead, and at the point referred to, write

```
Taxation {@PageMark taxinfo} ...
```

{@PageMark taxinfo} will not appear in the output, but Lout makes a note of the number of the page on which the word preceding it appears, and inserts that number in place of {@PageOf taxinfo}. Any single word may be used for the tag.

Cross referencing also applies to sections, subsections, appendices, subappendices, figures, tables, and numbered displays, and can supply both the page on which the item begins and its number. First, add a @Tag option whose value is a single word:

```
@Section
   @Title { Cross references }
   @Tag { cross }
@Begin
...
```

This marks the page on which the section begins, so {@PageOf cross} will be 9 in this case. In addition, {@NumberOf cross} will be the number of the section, in this case 9.

10. References

The method of handling references described in this section is rather elaborate, but it has many advantages. The first step is to create a separate *database file* containing @Reference symbols. Each is enclosed in braces, and has options chosen from @Tag, @Type, @Author, @Title, @Institution, @Number, @Publisher, @Year, @Proceedings, @Journal, @Volume, @Pages, and @Comment:

```
{ @Reference
   @Tag { strunk79 }
   @Type { Book }
   @Author { Strunk and White }
   @Title { The Elements of Style }
   @Publisher { MacMillan }
   @Year { 1979 }
}

{ @Reference
   @Tag { kingston92 }
   @Type { TechReport }
   @Author { Kingston, Jeffrey H. }
   @Title { Document Formatting
with Lout }
   @Number { 449 }
   @Institution { Basser Department
of Computer Science F09, University
of Sydney 2006, Australia }
   @Year { 1992 }
}
```

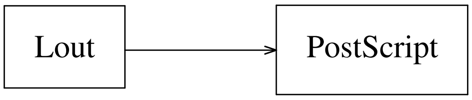

Figure 1. Basser Lout

The @Tag and @Type options are in fact compulsory; the first serves to name the reference, the second gives its type, which may be TechReport, ConferencePaper, JournalArticle, PhD, or Book, although a Lout expert will be able to add other types. The type determines which other options must be present, as shown above, and how they will be printed.

Suppose this file is given the name myrefs.ld (Basser Lout database file names must end in .ld). To tell Lout to refer to this file as needed,

```
@Database @Reference { myrefs }
```

is placed at the beginning of the document, before @Doc, @Report, or @Book.

Now, with everything prepared,

```
{@Ref kingston92}
```

for example at any point in the document will cause the reference whose tag is kingston92 to be added to the nearest following reference list, indicated by the symbol @ReferenceSection, unless it is already there. Typically one would write

```
... which ends our study.
@DP
@ReferenceSection
```

at the very end of the document (although one could equally well have several @ReferenceSection symbols scattered through the document if desired); technical reports and books add @ReferenceSection automatically when needed. The references will appear in alphabetical order by tag, and the @Ref symbol will be replaced by the number of the reference, in this case 1. @ReferenceSection has a @Title option like @Section, and it also has style, indent, gap and start options for contolling the appearance of the reference list, like @RawIndentList.

If a separate database file is not convenient for some reason, perhaps because it is desirable for the entire document to reside in one file, the @Reference symbols may be incorporated into the document itself, anywhere after @Text @Begin. Nothing will appear where they are typed in, but Lout will notice them and treat them exactly as if they had come from a database file. In this case no @Database symbol is needed.

A way to print references at any point in the document is provided by @RefPrint:

```
{@RefPrint kingston92}
```

might have result

> Kingston, Jeffrey H., *Document Formatting with Lout (Second Edition)*. Tech. Rep. 449 (1992), Basser Department of Computer Science F09, University of Sydney 2006, Australia.

depending on what references exist.

To get section numbers, footnote numbers, cross references, and references right, Basser Lout has to process the document more than once (three times for numbered references), and create special files whose names end in '.li' and '.ld'. It will print warning messages on runs when this process is incomplete, and substitute question marks for the unknown values. There should be only one document containing these features per Unix directory, otherwise Basser Lout will confuse their special files.

11. Multiple columns

DocumentLayout normally produces pages with a single column of text occupying the full width. To get multiple columns, enclose the document in

```
@Doc @ColText @Begin
...
@End @ColText
```

This will ordinarily produce two columns per page, although we will see in the next section how to produce three or more.

It is possible to start off a document with single columns and switch to multiple columns later, using the following arrangement:

```
@Doc @Text @Begin
single column part
@ColText @Begin
multiple column part
@End @ColText
@End @Text
```

This is the most useful combination, because switching from multiple back to single causes a new page to be begun, which generally looks poor. Tables and figures are always full width, but the width of footnotes depends on whether they occur within @Text or @ColText.

12. Changing the overall appearance

A few aspects of the overall appearance of the document may be changed by beginning with a @Document symbol, as in the following example:

```
@Document
   @InitialFont { Times Base 12p }
   @InitialBreak { adjust 1.2fx }
   @Hyphenate { Yes }
   @PageNumbers { Yes }
   @FirstPageNumber { 1 }
   @Columns { Double }
//
@Text @Begin
...
@End @Text
```

This shows the six options available with @Document, and their default values.

@InitialFont and @InitialBreak determine the overall font and paragraph breaking style; 1.2fx means that the inter-line spacing is to be 1.2 times the current font size, measured from baseline to baseline. To get double spacing, use 2.4fx, etc. @Hyphenate determines whether hyphenation is permitted or not, and may be Yes or No.

@PageNumbers determines whether or not page numbers will be printed (but not their format), and also may be Yes or No; @FirstPageNumber is the number given to the first page; and @Columns determines the number of columns that @ColText produces, and may be Single, Double, or Multi, the last usually meaning triple.

Since there are default values for the options, it is only necessary to mention those options that are to be changed. For example, to produce a document in Helvetica 10 point font with no page numbers, begin with

```
@Document
   @InitialFont { Helvetica Base 10p }
   @PageNumbers { No }
//
```

Disaster will ensue if the // symbol is forgotten. The @Doc symbol used previously is just an abbreviation for @Document //.

Of course, these five options only scratch the surface of the possible changes that could be made. Section 15 explains how to make many other and more radical changes to the overall appearance of the document.

13. Technical reports

This section describes how to use the DocumentLayout package to produce technical reports like the present document. Type -ireport in the Unix command instead of -idoc to use the package in this way. We present only the differences here; everything else works as before.

A technical report begins with a @Report symbol analogous to the @Document symbol described in the previous section. Here it is, with its nine options and their default values:

```
@Report
   @Title {}
   @Author {}
   @Institution {}
   @DateLine { @Date }
   @InitialFont { Times Base 12p }
   @InitialBreak { adjust 1.2fx }
   @Hyphenate { Yes }
   @PageNumbers { Yes }
   @Columns { Single }
//
```

The @Title, @Author, and @Institution options will be printed on the cover sheet and on the first page; they are formatted using clines @Break (see page 7). Multiple authors should be given on separate lines within the @Author option. @DateLine appears below the abstract on the cover sheet, and its default value is the current date as shown. @InitialFont, @InitialBreak, @Hyphenate, @PageNumbers and @Columns are as described in the last section, except that there is nothing analogous to @Text and @ColText: if @Columns is set to Double, the entire document after the title will be set in two columns. There is no @FirstPageNumber option. As for @Document, the symbol // must follow after, and disaster will ensue if it is omitted.

If the technical report has an abstract, it comes next:

```
@Abstract @Begin
...
...
@End @Abstract
```

The cover sheet of the present report shows how this will appear. @Abstract has a @Title option like the ones below; its default value is *ABSTRACT*.

Next come the sections of the report, each enclosed in a @Section symbol:

```
@Section
   @Title { Introduction }
@Begin
@PP
...
...
@End @Section
```

No @BeginSections or @EndSections symbols are needed. The sections may contain subsections, preceded as usual by @BeginSubSections and followed by @EndSubSections. After the sections there is opportunity for a sequence of appendices, each of the form

```
@Appendix
   @Title { ... }
@Begin
@PP
...
...
@End @Appendix
```

but these are quite optional. No @BeginAppendices and @EndAppendices symbols are needed. An appendix may contain sub-appendices via the usual symbols @BeginSubAppendices, @SubAppendix, and @BeginSubAppendices. This ends the input; there is no @End @Text, and any reference section will be added automatically.

14. Books

The DocumentLayout package may also be used to produce books. Type -ibook in the Unix command instead of -idoc to use the package in this way.

A book begins with a @Book symbol:

```
@Book
   @Title {}
   @Author {}
   @Edition {}
   @Publisher {}
   @InitialFont { Times Roman 12p }
   @InitialBreak { adjust 1.2fx }
   @Hyphenate { Yes }
//
```

The first four options are printed on the title page, the first three using clines @Break (see page 7). There are no @Columns, @PageNumbers, or @FirstPageNumber options. The last three options are as for @Document and @Report. Disaster will ensue if the // is omitted.

Next comes an optional preface:

```
@Preface
   @Tag { preface }
   @Title { About this book }
@Begin
@PP
...
@End @Preface
```

Since the title of most prefaces is simply Preface, this is the default value of the @Title option. After the preface, BookLayout will produce a table of contents listing the introduction, chapters, sections, subsections, appendices, bibliography, and index as appropriate.

The pages up to this point will be numbered in lower case Roman numerals; subsequent pages will be numbered in Arabic starting from 1.

Next comes an optional introduction, in exactly the same style as the preface except that @Introduction replaces @Preface and the default title is Introduction. After that comes a sequence of chapters in the usual style:

```
@Chapter
   @Title { Principles }
   @Tag { principles }
@Begin
@PP
...
@End @Chapter
```

No @BeginChapters or @EndChapters symbols are needed. Within a chapter, there may be a sequence of sections, each introduced by @Section in the usual way, all bracketed by @BeginSections and @EndSections. Within each section there may be subsections, each introduced by @SubSection, and the sequence as a whole bracketed by @BeginSubSections and @EndSubSections. The first subsection of the first section of the first chapter will be numbered 1.1.1, and so on. There are no sub-subsections.

Finally there is opportunity for a sequence of appendices, each introduced by @Appendix in the usual way. No @BeginAppendices or @EndAppendices symbols are needed. The appendices are numbered A, B, C, etc., and there are sub-appendices, obtained in the usual way.

Each symbol with a @Title option also has a @RunningTitle option. The more important parts of the book (preface, introduction, chapters, and appendices) have their @RunningTitle printed at the top of each even-numbered page; if omitted, the @Title option is used instead.

With Basser Lout, a long document is best broken into a sequence of files, each containing one section, from @Section to

@End @Section inclusive. Other files are needed for the beginning of each chapter, from @Chapter to @BeginSections, and for the end of each chapter, that is the final @EndSections and @End @Chapter. On the Unix operating system a good scheme for naming these files is

ch0.00	@Book ... //
ch0.01	Preface
ch0.02	Introduction
ch1.00	Beginning of Chapter 1
ch1.01	Section 1.1
ch1.02	Section 1.2
. . .	
ch1.99	End of Chapter 1
ch2.00	Beginning of Chapter 2
ch2.01	Section 2.1
ch2.02	Section 2.2
. . .	
ch2.99	End of Chapter 2

and so on. Then the Unix command

```
lout -ibook ch0.00 ch3.??
```

will format Chapter 3 only; and

```
lout -ibook ch?.??
```

will format the entire book. The whole book must be formatted by one command to get the page numbers right, but there is no need to do this until everything is perfect.

The symbols described in Sections 1–7 above are all available as usual. The numbering of figures and tables includes a chapter or appendix number: the first figure of Appendix A will be numbered A.1, and so on. Figures and tables within the preface and introduction are numbered 1, 2, 3, etc. When figures and tables appear near the end of a chapter, the following page on which they appear may also be the first page of the next chapter, which looks very poor. The solution is to move the figure or table to an earlier point in the chapter.

Figures and tables work in a slightly different way to previously, owing to the need to attach a chapter or appendix number to each one. The first figure of each chapter or appendix must be preceded by @BeginFigures, and the last figure of each chapter or appendix must be followed by @EndFigures. These symbols must also bracket figures in the preface and introduction. The symbols are not required when there are no figures. Similarly, @BeginTables and @EndTables must bracket tables.

Cross referencing works as described in Section 9; @Tag options may be given to the preface, introduction, chapters, sections, subsections, appendices, and sub-appendices.

References work as described in Section 10, except that @ReferenceSection is added automatically as needed. A variant of the @Ref symbol called @ChapRef is provided, which causes the reference to appear at the end of the current preface, introduction, chapter, or appendix, rather than at the end of the book.

Also available are symbols for making an index. To add an entry to the index, place

```
packages @Index { Packages }
```

for example at the relevant point. The result will be something like

Packages, 27

appearing in the index in the alphabetical position determined by the left parameter, which should be a juxtaposition of simple words composed, by convention, of lowercase letters and periods only.

A variant called @SubIndex provides a small indent, as is conventional for sub-entries in indexes. For example,

```
package  @Index  { Packages }
package.r @SubIndex {ReportLayout}
package.b @SubIndex {BookLayout}
```

scattered through a document will produce something like

Packages, 27
  BookLayout, 45
  ReportLayout, 40

Note how the left parameters have been carefully chosen to produce the correct ordering. There is also a @SubSubIndex symbol with a double indent.

These symbols attach one page number to each entry. Although the best authorities recommend exactly this, many authors choose to have entries like

Fourier Transform, 576, 583–593

despite the inconvenience to their readers. @RawIndex, @RawSubIndex, and @RawSubSubIndex symbols are provided which do not add page numbers to the entry, leaving this to the user. For example, one systematic way to get page number ranges is to place

```
packages @RawIndex {
   Packages, {@PageOf packages}
   -- {@PageOf packages.end}
}
```

at the start of the range, and

```
{@PageMark packages.end}
```

at the end of the range. This works because all six index symbols include a @PageMark operation. Incidentally, this means that index tags should be different from chapter and other tags.

Another use for @RawIndex is to get blank lines into the index between the letters of the alphabet, by inserting phantom entries:

```
b @RawIndex {}
c @RawIndex {}
...
z @RawIndex {}
```

In fact there is a symbol called @IndexBlanks which creates exactly these 25 entries. Unfortunately, these blanks will occasionally appear at the top of a column, and if there are no entries beginning with x, for example, there will be two blank lines between the w and y entries. The careful user can start off with @IndexBlanks and replace it later by the appropriate subset.[1]

Owing to problems behind the scenes, the Index heading will be printed, and an entry will be made in the table of contents, even if there are no entries in the index. To prevent this, you will need to change the @MakeIndex option in the book setup file to No, using the method described in the following section.

Although the page numbers in index entries will be kept up to date automatically as the document changes, as all cross references are, the user is recommended to refrain from inserting index entries until the book is complete and an overall plan of the structure of the index can be made.

15. Making more radical changes

The DocumentLayout package makes a large number of decisions, about how large the pages will be, which fonts will ordinarily be used, and so on. The more complex decisions, such as the appearance of tables of contents in books, can only be changed by modifying the packages, but many of the simpler decisions can be changed quite easily by setting options in the setup file.

The first step is to obtain a private copy of the setup file. A local expert will know where these files are kept; for example, in

[1] For Lout to solve this problem automatically, it would need to be told which letter each index entry belongs under, perhaps by symbols @AIndex, @BIndex, etc. The author felt that this would have been too tedious.

directory /usr/lout/include. Suppose the report setup file is copied into a file called myrep; then typing

```
lout myrep myfile
```

instead of

```
lout -ireport myfile
```

runs Lout with the private copy of the setup file, myrep.

Exactly what the setup file contains will depend upon the local situation, but it will be something like this:

```
# Setup file for reports.
# J. H. Kingston, July 1991

@SysInclude { ft }
@SysInclude { dl }
# @SysInclude { eq }
# @SysInclude { pas }
# @SysInclude { fig }
# @SysInclude { tab }

@Use { @DocumentLayout
  # @InitialFont { Times Base 12p }
  # @InitialBreak { adjust 1.2fx }
  # @Hyphenate { Yes }
  # @PageNumbers { Yes }
  # @Columns { Single }
  # @HeadingFont { Bold }
  # @ParaIndent { 2.0f }
  # @PageTop { |0.5rt - @PageNum - }
  # @PageFoot { @Null }
}
```

Whenever Lout encounters a # character not enclosed in quotes, it ignores it and everything following it up to the end of the line. The first two lines of the setup file, then, are comments for the human reader. After them come lines which cause Lout to read the file ft of font definitions and the file dl which contains the definition of the DocumentLayout package.

The next four lines are comments and will be ignored, but if the initial # is deleted they cause Lout to read the definitions of the Eq equation formatting package and the Pas Pascal program formatting package [2], the Fig diagram drawing package [3], and the Tab table formatting package [4].

Next comes a @DocumentLayout symbol within a @Use clause. It is this symbol whose options may be changed so as to affect the overall layout. These options are listed as comments on the following lines, together with the default value of each. To change an option, delete the # and change the value. For example, the normal paragraph indent produced by @PP is 2.0f (twice the current font size). To change it, say to 3.0f, change the line to

```
@ParaIndent { 3.0f }
```

The display indent option (not shown here) should probably be changed as well.

The first five options have the same name as five of the @Document symbol's options, and they determine the default value of those options. The @PageTop and @PageFoot options determine the appearance of the page header and footer lines (where the page numbers appear), and are best left to experts; but, for example,

```
@PageTop { |1rt  @PageNum }
```

will make the page number appear at the top right of each page, without the - characters.

The setup file is also the place to add your own definitions. They should be placed just before the @Use clause, as shown for example in Section 4 of the report describing the Eq equation formatting package [2].

Those who wish to make more radical changes will have to copy the dl file and change the definition of the DocumentLayout package. This requires knowledge of the principles of Lout, and the primitive features from which others are built, as described in the Lout user manual [1].

Appendix A. Accented characters

Accented characters are available in Lout. If you have a Latin-1 keyboard with these characters on them, just use them; otherwise you have to name the character using the @Char symbol. For example, {@Char eacute} produces é. The complete list of names of accented characters in Basser Lout is

À	Agrave	à	agrave
Á	Aacute	á	aacute
Â	Acircumflex	â	acircumflex
Ã	Atilde	ã	atilde
Ä	Adieresis	ä	adieresis
Å	Aring	å	aring
Æ	AE	æ	ae
Ç	Ccedilla	ç	ccedilla
È	Egrave	è	egrave
É	Eacute	é	eacute
Ê	Ecircumflex	ê	ecircumflex
Ë	Edieresis	ë	edieresis
Ì	Igrave	ì	igrave
Í	Iacute	í	iacute
Î	Icircumflex	î	icircumflex
Ï	Idieresis	ï	idieresis
Ð	Eth	ð	eth
Ñ	Ntilde	ñ	ntilde
Ò	Ograve	ò	ograve
Ó	Oacute	ó	oacute
Ô	Ocircumflex	ô	ocircumflex
Õ	Otilde	õ	otilde
Ö	Odieresis	ö	odieresis
Ø	Oslash	ø	oslash
Ù	Ugrave	ù	ugrave
Ú	Uacute	ú	uacute
Û	Ucircumflex	û	ucircumflex
Ü	Udieresis	ü	udieresis
Ý	Yacute	ý	yacute
Þ	Thorn	þ	thorn
ß	germandbls	ÿ	ydieresis

References

1. Kingston, Jeffrey H., *Document Formatting with Lout (Second Edition)*. Tech. Rep. 449 (1992), Basser Department of Computer Science F09, University of Sydney 2006, Australia.

2. Kingston, Jeffrey H., *Eq – a Lout package for typesetting mathematics*. Tech. Rep. 452 (1992), Basser Department of Computer Science F09, University of Sydney 2006, Australia. Contains an appendix describing the Pas Pascal formatter.

3. Kingston, Jeffrey H., *Fig – a Lout package for drawing figures*. Tech. Rep. 453 (1992), Basser Department of Computer Science F09, University of Sydney 2006, Australia.

4. Kingston, Jeffrey H., *Tab – a Lout package for formatting tables*. Tech. Rep. 451 (1992), Basser Department of Computer Science F09, University of Sydney 2006, Australia.