2. you make a switch to an arbitrary, *but really different*, font size
3. you switch back to `\normalsize`.

2 Footnotes in a minipage

I should also like to answer the minipage footnote problem Jackie reported. The answer is given on pages 91 and 99 of the LaTeX user's guide and reference manual. Alternatively, you can delve into the comment of `latex.tex` – a bit more effort – but the answer remains the same.

In the comment in `latex.tex` (version of 13 June 1989) we can read

```
% \minipage :
%  similar to parbox, except it also
%  ...
%  changes footnotes by redefining:
%    \@mpfn        == mpfootnote
%    \thempfn      == \thempfootnote
```

Inside a `minipage` environment LaTeX doesn't use the `footnote` counter, but the `mpfootnote` counter. And so a solution is

```
\renewcommand{\thempfootnote}
  {\arabic{mpfootnote}}
```

An example to prove that this solution works. This table is formatted without any re-definitions.

`\arabic`	arabic numerals[a]
`\roman`	roman numerals (lower case)[b]
`\alph`	lower-case letters (1–26)
`\fnsymbol`	symbols (1–9)[c]

[a] Nice and simple.
[b] I don't like 'mcmlxxviii'.
[c] Old-fashioned.

This table is formatted with the re-definition I described above.

`\arabic`	arabic numerals[1]
`\roman`	roman numerals (lower case)[2]
`\alph`	lower-case letters (1–26)
`\fnsymbol`	symbols (1–9)[3]

[1] Nice and simple.
[2] I don't like 'mcmlxxviii'.
[3] Old-fashioned.

◇ Nico Poppelier
Elsevier Science Publishers
Academic Publishing Division
R&D Department
Sara Burgerhartstraat 25
Amsterdam, The Netherlands
n.poppelier@elsevier.nl

LaTeX Tree Drawer

Glenn L. Swonk

Abstract

Today, many software systems are analyzed, designed and developed in a top-down hierarchical manner. This is especially apparent with Object Oriented Programming and Design. Diagramming a hierarchical relationship can be cumbersome in TeX or LaTeX. This paper describes a MS-DOS tool that can be used to simplify the diagramming process by taking a simple input format and producing a LaTeX picture environment source which can be *input* in a LaTeX document.

Introduction

While trying to document a hierarchical directory structure, I manually placed the directory nodes' coordinates in a source file from calculations made from a rough sketch. Not only was this a tedious operation, it was prone to error and required many iterations before the exact result was achieved.

What I really wanted was a simple way specify the hierarchical relationship and an automated way to generate the output LaTeX file. Using the UNIX `find(1)` utility, it was easy to generate a list of files or directories that were to be plotted in a LaTeX picture environment. In the simplest case, `find(1)` can be used to generate a list of all files *under* a specified directory using the command `find <dirname> -print`.

What resulted from this problem is a MS-DOS tool which I call the LaTeX Tree Drawer (LTD). It takes input from a file in the form of a `find(1)` output and generates a LaTeX picture output that can be used as input to a LaTeX document. This paper describes the implementation and use of this tool.

Samples

To best illustrate what LTD can do, a few samples are in order.

Example 1 illustrates the simplest example — a single parent node and its two children. From the input file in figure 1, the resultant picture is shown in figure 2.

```
parent
parent/child1
parent/child2
```

Figure 1: Example 1 Input File

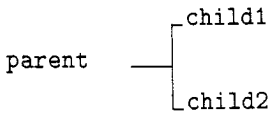

Figure 2: Example 1 Output

The second example illustrates a partial UNIX-like directory filesystem. From the input file shown in figure 3, the diagram in figure 4 is the final output.

```
root
root/bin
root/lib
root/tmp
root/usr
root/usr/bin
root/usr/lib
root/usr/acct
```

Figure 3: Example 2 Input File

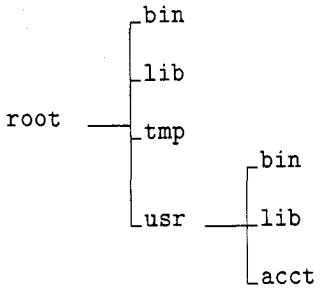

Figure 4: Example 2 Output

The third example illustrates a shape class hierarchy.

From the input file in figure 5, the diagram in figure 6 is the final output. Note the *boxing* of the nodes. LTD can box all or either branch or leaf nodes.

Implementation

The LTD implementation comprises four basic processing steps. The following provides a short description of each.

Input

The input phase of the program interprets the command line parameters specified by the user and sets the appropriate internal variables. It then reads in the nodes from the specified file and stores the nodes in an internal data structure representing its child/parent/sibling relationship. All input parameter values are assumed to be specified in points (1/72″).

```
Shape
Shape/Ellipse
Shape/Ellipse/Circle
Shape/Triangle
Shape/Triangle/Equilateral
Shape/Triangle/Isosceles
Shape/Rectangle
Shape/Rectangle/Empty
Shape/Rectangle/Gray
Shape/Rectangle/Solid
```

Figure 5: Example 3 Input File

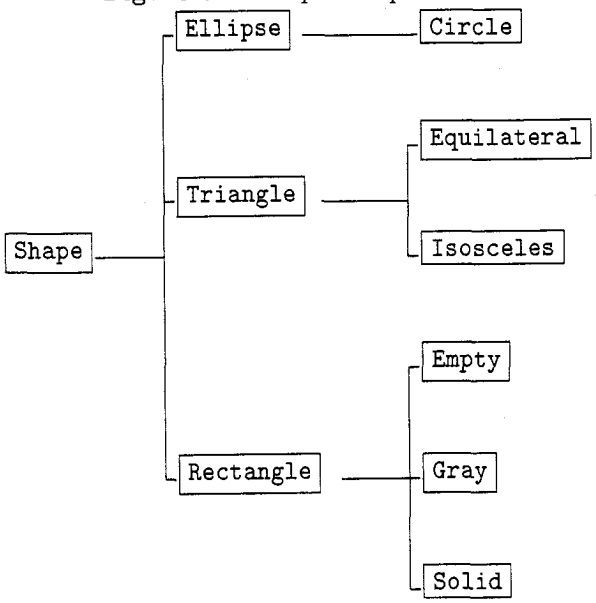

Figure 6: Example 3 Output

Layout

The layout mechanism is based on an algorithm developed by Sven Moen from Brown University and is described in his paper in the July 1990 issue of IEEE Software. The layout algorithm uses information from the command line parameters and node information (such as width, height, and border) to produce a tree layout based on the node and sub-tree contours. The layout algorithm produces horizontal trees with the following characteristics:

- A parent is drawn to the left of its children.
- Trees are drawn from left to right to accommodate varying length strings.
- Subtrees look the same, regardless of position.
- Left children are drawn above the parent, right children below.

Planting

After the layout process, each node contains relative positional information that needs to be converted to absolute positions that can be used to generate the output code. The planting process performs this on the nodes to produce absolute positions compatible with the LaTeX picture environment (x increasing left to right, y increasing bottom to top).

Output

Up to now, all processing has been device independent (with the exception of assuming a point size coordinate system). The output process now traverses the internal data structures to produce the specified output code. The output code also contains additional information such as the command line parameters, node width and height, and source input lines that can be used for debugging.

Currently two types of output formats are implemented, LaTeX and raw data.

The LaTeX data can be redirected to a file and *input* into a document via the `\input{}` command.

The raw data format may be useful to users who wish to write their own output drivers for displaying the node layout information.

Node Layout Specification

To understand how the parameters from the LTD command line affect the layout, the picture in figure 7 defines the dimensions used in the layout algorithm. The dimensions are defined by the following:

- w – Width of the node. This dimension is calculated from then length of the string times the nominal font width parameter specified by `-w <n>`.
- h – Height of the node. This dimension is specified from the `-h <n>` command line parameter.
- b – Border of the node. This dimension is specified from the `-b <n>` parameter. It is used to provide the *spacing* around a node from its parent, siblings and children. By varying this parameter you can get the layout process to spread the height of the diagram. Negative values can be used to provide *very tight* spacing.
- pd – Parent Distance (not shown). This dimension is specified from the `-pd <n>` parameter. It is a global parameter to the layout algorithm for the layout process. By varying this parameter you can get the layout process to spread the width of the diagram.

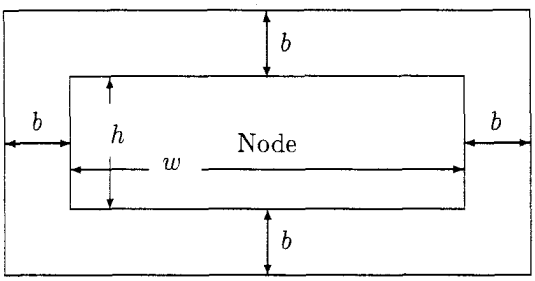

Figure 7: Node Dimensions

Command Line Parameters

The following list describes the command line parameters that can be used to modify the behavior of the LTD.

- `-h <height>` – Height of font, used for layout.
- `-b <border>` – Border distance, used for layout.
- `-w <width>` – Width of a single character, used for layout.
- `-pd <parent_distance>` – Minimum distance needed between a child and its parent, used for layout.
- `-fix` – A flag that can be used to create fixed sized widths for each level. Typically creates more visually pleasing diagrams.
- `-f <filename>` – Filename used for input.
- `-o <filename>` – Filename used for output.
- `-ba` – Switch to form box around all nodes.
- `-bb` – Switch to form box around branch nodes.
- `-bl` – Switch to form box around leaf nodes.
- `-tex` – Generate LaTeX output format (default).
- `-xy` – Generate XY output format.
- `-help` – Print help text of usage to stderr.
- `-xo <xoffset>` – a parameter that is used to shift the x position of the picture.
- `-yo <yoffset>` – a parameter that is used to shift the y position of the picture.
- `-xs <xsize>` – a parameter that is used to adjust the x size of the output picture.
- `-ys <xsize>` – a parameter that is used to adjust the y size of the output picture.

Other parameters are available in the most recent version of LTD and should be defined in the help text. The actual parameters used are printed out as comments for debugging and repeatability.

Using LTD

Invoking LTD

Typically, LTD is invoked from the DOS command line with a filename and any parameters to modify its default configuration. In order to use the output, you need to redirect the output to a filename which is eventually used for the picture environment.

In most cases, following is the simplest form:

```
ltd -f test.fil >test.ltd
```

Using LTD's Output

Once you have a file with the plotted data, you must *input* the file into a LaTeX file. I suggest using a base file that includes the output file similar to the following form:

```
%
% base.tex
%
\documentstyle{report}
\begin{document}

\begin{figure}[h]
\begin{center}
\fbox{ {\tt \input{test.ltd} } }
\end{center}
\caption{LaTeX Tree Drawer Demo}
\end{figure}

\end{document}
%% ////////////////////////////////
%% /// eof ////////////////////////
%% ////////////////////////////////
```

Note that you have full control of the font type and size that is used for the picture. By varying the font size with the command line parameters, you can vary the appearance of the final picture to provide the most aesthetic diagram.

Note that the `fbox` command is optional and is used to create an enclosure for the final output. This provides the user with an idea of the extent of the diagram. The box that borders the diagram can be shifted/sized by the optional command line parameters.

Limitations

The LTD has the following limitations:

- The input file must define the parent nodes before the child nodes can be defined. If this is not the case, some nodes may not be displayed properly.
- The '/' character is assumed to be the separator character.
- An input file character that is interpreted as a control sequence to TeX or LaTeX can cause problems, specifically the '_' (underscore) character.
- Since the font that is being used is completely under the control of the user, we can only *guess* at the width of the parent node. Therefore, some of the connectors to the parent node may fall short or exceed the desired point. To solve this, it may be necessary to specify the `-w` parameter and/or use a fixed width font. Some hand tweaking may also be required to get the optimal results.
- The program is written to use defaults for a 12pt character font.
- Maximum number of nodes is $\approx$ 500, but will vary depending on size of the strings.
- The size of the picture is calculated based on the min/max of the x and y dimensions. The width of the string is not taken into consideration, therefore some shifting of the picture may be necessary to center the picture.

Availability

Version 1.0 of LTD is in the public domain. I hope to submit the DOS executable to an archive server in uuencoded format for general availability.

Any ideas or general comments should be directed to the author at `uunet!toshais!swonk`.

References

[1] Moen, Sven. "Drawing Dynamic Trees". IEEE Software (July 1990).

[2] Lamport, Leslie. LaTeX: A Document Preparation System. Reading, Mass.: Addison-Wesley, 1986.

◇ Glenn L. Swonk
25302 Fairgreen
Mission Viejo, CA 92692
`uunet!toshais!swonk`