The `dirtree` package
Directory Tree

Jean-Côme Charpentier*

Version 0.32 2012/12/11
Documentation revised December 11, 2012

Abstract

Package `dirtree` allows to display directory tree, like in the windows explorer.

Contents

1 Introduction 1
2 Usage 2
3 ToDo 7
4 `dirtree` \LaTeX{} Wrapper 8
5 `dirtree` Code 9

1 Introduction

During a discussion on fctt (fr.comp.text.tex) about directory tree and how display such a structure, it appeared that there wasn’t many packages which do the job.

One obvious solution is to use PSTricks but some people don’t like or don’t know this package, so I made the first release of `dirtree`.

In fact, I didn’t plan to send it in CTAN but Robin Fairbairns and Danie was very convincing!

∗Jean-Côme.Charpentier@wanadoo.fr
2 Usage

Package dirtree works both on Plain \TeX and \LaTeX. No surprise to call it:

\usepackage{dirtree}

for \LaTeX and

\input dirtree

for Plain \TeX.

Since version 0.3, dirtree has some package options. We'll see these options one by one.

The main macro is \dirtree which take one argument (the tree structure). This tree structure is a sequence of

.\level<space><text node>.<space>

Note that there is a dot in the beginning and another one at the end of each node specification. The spaces are very important: if you forgot the space before the level there will be an error and if you forgot the space after the last dot, you don’t indicate the end of the node. Since an end of line is like a space for \TeX, I recommend to write a node per line in the source file: it’s handy and more readable.

The level indicates the node depth in the tree. There is two rules you must respect:

1. The root must have the level one.

2. When you create a node, if the last node have the level \(n\), the created node must have a level between 2 and \(n + 1\).

In fact, you can indicates a level greater than \(n + 1\) if one node have a level \(n\) somewhere in the tree but the result will be strange!

A node of level \(n\) will be connected to the last node defined which has a level lesser or equal to \(n\).

For example, the code

\dirtree

.1 /.
.2 bin.
.2 home.
.3 jeancome.
.4 texmf.
.5 tex.
.6 latex.
.7 dirtree.
.3 jeancomeson.
give the result

```
{ jeancomedaughter.
  usr.
  bin.
  games.
  fortunes.
  include.
  local.
  bin.
  share.
  texmf.
  fonts.
  metapost.
  tex.
  share.
}
```

Note the `%` after the left brace in the beginning: it’s important because the first character encountered must be a dot.

\DTstyle

A text node is typeset with the command \DTstyle. Its default value is \ttfamily when you are under \LaTeX{} and \tt when you are under Plain \TeX{}. You can redefine this macro as you want, it is used with the syntax `{\DTstyle{text node}}`, so you can use both \ttfamily and \texttt{tt} for ex-
ample.

The \DTcomment command allows to put text at the right side, with leaders. The syntax is
\DTcomment{comment text}

\DTstylecomment The style of comment is defined by \DTstylecomment. Its default value is \rmfamily under $\LaTeX$ and \itshape under Plain $\TeX$, and it acts like \DTstyle. Here is an example: the code
\renewcommand*{\DTstylecomment}{\rmfamily\color{green}\textsc}
\renewcommand*{\DTstyle}{\ttfamily\textcolor{red}}
\dirtree{%
.1 /.
.2 bin.
.2 home.
.3 jeancome.
.4 texmf.
.5 tex.
.3 jeancomeson\DTcomment{Guillaume}.
.3 jeancomedaughter\DTcomment{Mathilde}.
.2 usr.
.3 bin.
}
give the result

/  
  bin  
  home  
    jeancome  
      texmf  
        tex  
          jeancomeson .......................................... Guillaume  
        tex  
          jeancomedaughter ....................................... Mathilde  
    usr  
      bin  

In this example we have used the xcolor package.
You can build complex text node. For example, the code

\dirtree{%
.1 /.
.2 bin \dots{} \begin{minipage}[t]{5cm}
  This directory holds executable files (binary
  files or link on binary files)\dots{}
\end{minipage}.
.2 home \dots{} \begin{minipage}[t]{5cm}
  jeancome\\
  guillaume\\

give the result

/.bin ...
  This directory holds
executable files (binary
  files or link on binary
  files).
/home ...
  jeancome
  guillaume
  mathilde
  texmf

We don’t encourage to try too complicated code. Package dirtree is still fragile!
Note that we pay attention to use optional parameter [t] in order to have a right
vertical alignment with horizontal rules.

\DTsetlength
  Some dimensions can be changed using the \DTsetlength command. The
  syntax is:

  \DTsetlength{offset}{width}{sep}{rule-width}{dot-size}

  \textbf{Text node}

  \begin{itemize}
    \item offset = 0.2em
    \item width = 1em
    \item sep = 0.2em
    \item rule-width = 0.4pt
    \item dot-size = 1.6pt
  \end{itemize}

\DTbaselineskip
  The last length parameter is \DTbaselineskip which indicates the skip be-
tween lines of the tree.

If we typeset the first example with

\setlength{\DTbaselineskip}{20pt}
\DTsetlength{1em}{3em}{0.1em}{1pt}{4pt}

we obtain the (strange) result:

```
/  
  |  
  bin  
  /  
  |  
  home  
  |  
  jeancome  
  |  
  texmf  
  |  
  tex  
  |  
  latex  
  |  
  dirtree  
  |  
  jeancomeson  
  |  
  jeancomedaughter  
  |  
  usr  
  |  
  bin  
  |  
  games  
  |  
  fortunes  
  |  
  include  
  |  
  local  
  |  
  bin  
  |  
  share  
  |  
  texmf  
  |  
  fonts  
  |  
  metapost  
  |  
  tex  
  |  
  share
```
Note that \texttt{dirtree} package is not able to split tree on several pages. If this case occurs, the result will be very strange with overfull rules. I suppose that the best is to place such trees inside floats.

3 ToDo

- Parameters with \texttt{xkeyval} syntax;
- Command \texttt{\DTsplittree} to allows a tree to be typeseted on several pages;
- Style parameters to rules (color for example) and gap between text and comment (by now it’s \texttt{\dotfill}).
- Dimension parameter \texttt{abovetreeskip} and \texttt{belowtreeskip}.

<latex-wrapper>
4 dirtree \LaTeX\ Wrapper

Nothing special here but the \DT@fromsty definition. This latter is intended to check if dirtree is called under \LaTeX\ (with \usepackage) or under Plain \TeX. 

1 \NeedsTeXFormat{LaTeX2e}[1995/06/01]
2 \ProvidesPackage{dirtree}[\filedate \space v\fileversion \space package wrapper for dirtree]
3 \newcommand*{\DT@fromsty}{}
4 \input{dirtree.tex}
5 \ProvidesFile{dirtree.tex}
6 [\filedate \space v\fileversion \space 'dirtree' (jcc)]
An “hello” message.
8 \message{'dirtree' v\fileversion, \filedate\space (jcc)}

Save at current catcode and make @@ a letter
9 \edef\DT@AtCode{\the\catcode'\@}
10 \catcode'\@=11

Define \DT@loop, \DT@repeat, and \DT@iterate like \loop, \repeat, and \iterate. The \DT@ form allows to place loop inside loop.
11 \long\def\DT@loop#1\DT@repeat{%
12 \def\DT@iterate{%#1\relax\expandafter\DT@iterate\fi}%
13 \DT@iterate
14 \let\DT@iterate\relax
15 }
16 \let\DT@repeat=\fi

Define some \LaTeX macros if we work under Plain \TeX. \@namedef-like for \edef.
17 \expandafter\ifx\csname DT@fromsty\endcsname\relax
18 \def\@namedef#1{\expandafter\def\csname #1\endcsname }
19 \def\@nameuse#1{\csname #1\endcsname}
20 \long\def\@gobble#1{}
21 \fi
22 \def\@namedef#1{\expandafter\edef\csname #1\endcsname}

Offset between vertical rule below text and text left boundary.
23 \newdimen\DT@offset \DT@offset=0.2em

Length of horizontal rule.
24 \newdimen\DT@width \DT@width=1em

Gap between horizontal rule and text.
25 \newdimen\DT@sep \DT@sep=0.2em
\DT@offset + \DT@width + \DT@sep
26 \newdimen\DT@all
27 \DT@all=\DT@offset
28 \advance\DT@all \DT@width
29 \advance\DT@all \DT@sep

Rule thickness
30 \newdimen\DT@rulewidth \DT@rulewidth=0.4pt

Size of square junction.
31 \newdimen\DT@dotwidth \DT@dotwidth=1.6pt

baselineskip inside tree.
32 \newdimen\DTbaselineskip \DTbaselineskip=\baselineskip

Max index node.
33 \newcount\DT@counti
Current index node

\newcount\DT@countii
\DT@countiii = \DT@countii - 1. That is, Previous index node.
\newcount\DT@countiii

Last node of a level lesser or equal to current one.
\newcount\DT@countiv

\DTsetlength \DTsetlength allows to define dimensions in use for the directory tree (see above).
\def\DTsetlength#1#2#3#4#5{%
\DT@offset=#1\relax
\DT@width=#2\relax
\DT@sep=#3\relax
\DT@all=
\DT@all=\DT@offset
\advance\DT@all by\DT@width
\advance\DT@all by\DT@sep
\DT@rulewidth=#4\relax
\DT@dotwidth=#5\relax
}{

\DT@all is the width of a whole column.

\DT@all=\DT@offset
\advance\DT@all by\DT@width
\advance\DT@all by\DT@sep
\DT@rulewidth=#4\relax
\DT@dotwidth=#5\relax
}

\DTstyle is the style used to typeset nodes. \DTstylecomment is the style used to typeset comments. Since \TeX and \LaTeX are very different, we test the format used before initializations.
\expandafter\ifx\csname DT@fromsty\endcsname\relax
\def\DTstyle{\tt}
\def\DTstylecomment{\rm}
\else
\def\DTstyle{\ttfamily}
\def\DTstylecomment{\rmfamily}
\fi

\DTcomment \DTcomment places comment in a line of the tree.
\def\DTcomment#1{%
\kern\parindent\dotfill
{\DTstylecomment{#1}}%
}

In order to save some lengths we create newdimen
\newdimen\DT@indent
\newdimen\DT@parskip
\newdimen\DT@baselineskip

\dirtree \dirtree is the main package macro.
\def\dirtree#1{%
Change some parameters (save them before).

\DT@indent=\parindent
\parindent=z@
\DT@parskip=\parskip
\parskip=z@
\DT@baselineskip=\baselineskip
\baselineskip=\DTbaselineskip
\let\DT@strut=\strut
\def\strut{\vrule width z@ height0.7\baselineskip dept h0.3\baselineskip}%

Read the argument and before that, initialize counters. \DT@counti is the current index node.

\DT@counti=z@
\let\next=\DT@readarg
\next#1@nil

When \DT@readarg has done its job, the node levels and the node texts are saved in \DT@level@<index> and \DT@body@<index> respectively. \DT@counti holds the greater index. We can now display the tree.

Firstly, display the root. For that, the text is boxed.

\dimen z@=\hsize
\advance\dimen z@ -\DT@offset
\advance\dimen z@ -\DT@width
\setbox z@=\hbox to\dimen z@ of%
  \hsize=\dimen z@
  \vbox{\@nameuse{DT@body1}1}%
\ht z@=0.7\baselineskip
\dp z@=\dimen z@%

We change the height and the depth of this box in order to have the same total height and a height of 0.7\baselineskip, that is, the height of \strut.

\dimen z@=\ht z@
\advance\dimen0 by\dp z@
\advance\dimen0 by-0.7\baselineskip
\ht z@=0.7\baselineskip
\dp z@=\dimen z@%

Then we display this box with an indentation as if there had a level 0.

\par\leavevmode
\kern\DT@offset
\kern\DT@width
\box z@
\endgraf

Initialize index for the loop.

\DT@countii=\@ne
\DT@countiii=z@
\dimen3 holds the height of the node in the tree. In fact, the bottom of the node since this dimension is used to connect vertical rules.

\dimen3=\dimen z@
\DT@lastlevel\<level> holds the baseline of the last node in level \<level>.

Loop for displaying the remainder of the tree.

Exit loop when the last current index is lesser or equal to max index.

\DT@count holds current index and \DT@countii holds previous index (just current index minus one).

Horizontal offset for the text:
\( (\text{current level} - 1) \times \DT@all + \DT@offset \).

Look for last node in previous level in order to know how connect the current node.

Look for previous node

Repeat until this previous node has a level lesser or equal to current level.

Now \DT@countiv holds the index node connected to current node.

We box the text node.

Since text node is vboxed, we use a \hsize minus horizontal current offset.

Restore \hsize.
Change height and depth in such a way that height is $0.7\baselineskip$ (that is, the $\strut$ height), and total height is unchanged.

```
120 \dimen\z@=\ht\z@
121 \advance\dimen\z@ by\dp\z@
122 \advance\dimen\z@ by-0.7\baselineskip
123 \ht\z@=0.7\baselineskip
124 \dp\z@=\dimen\z@
```

Save the height of the box in tree. The last node is the last node in its level!

```
125 \nameedef{\DT@lastlevel@\the\DT@countii}{\the\dimen3}
\dimen3 holds the vertical position of the bottom.
126 \advance\dimen3 by\dimen\z@
127 \advance\dimen3 by0.7\baselineskip
Display vertical rule
128 \dimen\z@=@nameuse{\DT@lastlevel@\the\DT@countii}\relax
129 \advance\dimen\z@ by-@nameuse{\DT@lastlevel@\the\DT@countiv}\relax
130 \advance\dimen\z@ by0.3\baselineskip
If this vertical rule connect two nodes which have different level, the rule must be reduced by $0.5\baselineskip$ (the rule don’t rise up to the baselineskip of the connected node).
```
131 \ifnum@nameuse{\DT@level@\the\DT@countiv} <
132 \@nameuse{\DT@level@\the\DT@countii}\relax
133 \advance\dimen\z@ by-0.5\baselineskip
134 \fi
Display vertical rule
135 \kern-0.5\DT@rulewidth
136 \hbox{\vbox to\z@{\vss\hrule width\DT@rulewidth height\dimen\z@}}%
137 \kern-0.5\DT@rulewidth
Display square junction.
138 \kern-0.5\DT@dotwidth
139 \vrule width\DT@dotwidth height0.5\DT@dotwidth depth0.5\DT@dotwidth
140 \kern-0.5\DT@dotwidth
Display horizontal rule and gap between horizontal rule and text node.
141 \vrule width\DT@width height0.5\DT@rulewidth depth0.5\DT@rulewidth
142 \kern\DT@sep
Display text node.
143 \box\z@
New paragraph for next node.
```
144 \endgraf
145 \repeat
Restore indentation, paragraph skip, and $\strut$.
```
146 \parindent=\DT@indent
147 \parskip=\DT@parskip
The first processing step is to read the whole tree. It’s a recursive macro: each call process one node, that is, a

\textnode in the source file.

\begin{verbatim}
def \DT@readarg.#1 #2. #3\@nil{%  
\DT@counti is the current index.  
\advance\DT@counti \@ne  
save level node in \DT@level@<\ti> and text node in \DT@body@<\ti>.
Two dirtree \strut are added to text node in order to ensure a right vertical alignment, according to dirtree \baselineskip.

\@namedef{DT@level@\the\DT@counti}{#1}%  
\@namedef{DT@body@\the\DT@counti}{\strut{\DTstyle{#2}\strut}}%
\end{verbatim}

If \#3 is not empty, it contains the remainder nodes and macro calls itself. Otherwise, recursive call is stopped.

\begin{verbatim}
\ifx\relax#3\relax  
\let\next\@gobble  
\fi  
\next#3\@nil  
\}
\end{verbatim}

Restore at catcode.
\catcode’\@=\DTAtCode\relax
</tex>

### Change History

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>v0.01</td>
<td>General: First release to answer a question on fctt.</td>
<td>1</td>
</tr>
<tr>
<td>v0.11</td>
<td>General: fix bug</td>
<td>1</td>
</tr>
<tr>
<td>v0.12</td>
<td>General: \DTbaselineskip, \parskip, \baselineskip, and \strut in order to fix a displaying bug.</td>
<td>1</td>
</tr>
<tr>
<td>v0.2</td>
<td>General: dtx for CTAN, code for both Plain \TeX{} and \LaTeX{}.</td>
<td>1</td>
</tr>
<tr>
<td>v0.3</td>
<td>General: xkeyval syntax, breakable tree</td>
<td>1</td>
</tr>
<tr>
<td>v0.31</td>
<td>General: bug about some lengths.</td>
<td>1</td>
</tr>
<tr>
<td>v0.32</td>
<td>General: bug about length (thanks to Philipp Kühl). Some macro names changed in order to pre-</td>
<td>10</td>
</tr>
</tbody>
</table>
vent clash with other packages.

LOOP, REPEAT and ITERATE modified to DT@ form in order to prevent some clash with other packages.

\dirtree: Inverse order of assignment between baselineskip and DT@baselineskip.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols

\@nameedef \DT@iterate 22, 125, 12, 13, 14
\DT@all \DT@loop 26, 11, 105
\DT@counti \DT@offset 23, 27, 38, 41, 74, 86, 99
\DT@countii \DT@parskip 59, 64, 147
\DT@countiii \DT@readarg 71, 151
\DT@countiv \DT@repeat 11, 16, 113
\DT@dotwidth \DT@rulewidth 30, 44, 135, 136, 137, 141
\DT@fromsty \DT@sep 25, 29, 40, 43, 142
\DT@indent \DT@strut 68, 149
\DT@parskip \DT@width 59, 64, 147
\DT@readarg \DT@width 24, 28, 39, 42, 75, 87, 141
\DT@repeat \DT@Code 9, 160
\DT@repea \DT@setlength 32, 67
\DT@setlength \DT@style 54
\DT@strut \DT@styles 37
\DT@strut \DT@styles 47, 154
\DT@styles \DT@stylecomment 47, 56