The nodetree package

Josef Friedrich
josef@friedrich.rocks
github.com/Josef-Friedrich/nodetree

v2.0 from 2020/05/29

Contents

1 Abstract

`nodetree` is a development package that visualizes the structure of node lists. `nodetree` shows its debug informations in the consoles' output when you compile a LuaTEX file. It uses a similar visual representation for node lists as the UNIX `tree` command does for a folder tree.

Node lists are the main building blocks of each document generated by the TEX engine *LuaTEX*. The package `nodetree` doesn't change the rendered document. The tree view can only be seen when using a terminal to generate the document.

`nodetree` is inspired by a gist from Patrick Gundlach.

2 Usage

The package `nodetree` has four usage scenarios. It can be used as a standalone Lua module, as a plain LuaTEX, a LuaLATEX package or as package to embed nodetree views in a LuaLATEX document.

2.1 As a plain LuaTEX package

Run `luatex luatex-test.tex` for example to list the nodes using LuaTEX.

```
\input{nodetree.tex}
\NodetreeRegisterCallback{postline}

Lorem ipsum dolor.
\bye
```

2.1.1 Available macros

Macro name	Reference
`\NodetreeRegisterCallback{`⟨*callbacks*⟩`}`	Page 12, Section 3.1
`\NodetreeUnregisterCallback{`⟨*callbacks*⟩`}`	Page 12, Section 3.2
`\NodetreeSetOption[`⟨*option*⟩`]{`⟨*value*⟩`}`	Page 12, Section 3.3
`\NodetreeResetOption{`⟨*option*⟩`}`	Page 12, Section 3.4
`\NodetreeReset`	Page 12, Section 3.6

2.1.2 Available options

Option name	Reference
`callback`	Page 14, Section 5.1
`channel`	Page 14, Section 5.2
`verbosity`	Page 14, Section 5.3
`color`	Page 15, Section 5.4
`unit`	Page 15, Section 5.5
`decimalplaces`	Page 17, Section 5.6

2.2 As a LuaLaTeX package

Or run `lualatex lualatex-test.tex` to show a node tree using LuaLaTeX. In LuaLaTeX you can omit `\NodetreeRegisterCallback{postline}`. `\usepackage{nodetree}` registers automatically the `post_linebreak_filter`. If you don't want debug the `post_linebreak_filter` use `\NodetreeUnregisterCallback{postline}`.

```
\documentclass{article}
\usepackage{nodetree}

\begin{document}
Lorem ipsum dolor.
\end{document}
```

2.2.1 Available macros

Macro name	Reference
`\NodetreeRegisterCallback{`⟨*callbacks*⟩`}`	Page 12, Section 3.1
`\NodetreeUnregisterCallback{`⟨*callbacks*⟩`}`	Page 12, Section 3.2
`\NodetreeSetOption[`⟨*option*⟩`]{`⟨*value*⟩`}`	Page 12, Section 3.3
`\NodetreeResetOption{`⟨*option*⟩`}`	Page 12, Section 3.4
`\NodetreeReset`	Page 12, Section 3.6
`\NodetreeSet{`⟨*kv-options*⟩`}`	Page 12, Section 3.5

2.2.2 Available options

Option name	Reference
`callback`	Page 14, Section 5.1
`channel`	Page 14, Section 5.2
`verbosity`	Page 14, Section 5.3
`color`	Page 15, Section 5.4
`unit`	Page 15, Section 5.5
`decimalplaces`	Page 17, Section 5.6

2.3 As a Lua module

Import the Lua module of the package inside `\directlua{}` with this command: `local nodetree = require('nodetree')`. Then use the Lua function `nodetree.print(head, options)` to debug nodes inside your Lua code.

```
local nodetree = require('nodetree')

local rule1 = node.new('rule')
rule1.width  = 20 * 65536
rule1.height = 10 * 65536
rule1.depth  = 10 * 65536
nodetree.print(vbox)
```

The function `nodetree.print()` takes as a second argument a Lua table to configure the output.

```
nodetree.print(vbox, { verbosity = 2, unit = 'cm' })
```

This are the default options:

```
options =  {
  callback = 'post_linebreak_filter',
  channel = 'term',
  color = 'colored',
  decimalplaces = 2,
  engine = 'luatex', -- Required for the callback registration
  unit = 'pt',
  verbosity = 1,
}
```

The following code snippet demonstrates the usage in LuaTEX. `head` is the current node.

```
  \directlua{
  local nodetree = require('nodetree')
  local test = function (head)
    nodetree.print(head)
  end
  callback.register('post_linebreak_filter', test)
}

Lorem ipsum dolor.
\bye
```

This example illustrates how the function has to be applied in LuaLATEX.

```
\documentclass{article}
\usepackage{nodetree}

\begin{document}

\directlua{
  local nodetree = require('nodetree')
  local test = function (head)
```

```
    nodetree.print(head)
  end
  luatexbase.add_to_callback('post_linebreak_filter', test, 'test')
}

Lorem ipsum dolor.
\end{document}
```

2.4 The package `nodetree-embed`

The single purpose of this auxiliary package is to provide a view similar to a terminal (console) output. This view mimics the output of `nodetree` in a terminal. The view can be embedded in a LuaLaTeX file. You have to compile documents using this embedded view with the option `--shell-escape`. The main environment of this package is `NodetreeEmbed`. Markup inside this environment is written into a temporary LaTeX file. This file is compiled in the background by `latexmk` and the `nodetree` output is embded into this view. The following list shows the single intermediate steps:

1. `jobname.tex`

```
\begin{NodetreeEmbedEnv}
nodetree
\end{NodetreeEmbedEnv}
```

2. `_nodetree-jobname/1.tex`

```
%!TEX program = lualatex
\documentclass{article}
\usepackage{nodetree}
\NodetreeSetOption[channel]{tex}
\NodetreeSetOption[verbosity]{1}
\NodetreeSetOption[unit]{pt}
\NodetreeSetOption[decimalplaces]{2}
\NodetreeUnregisterCallback{post_linebreak_filter}
\NodetreeRegisterCallback{post_linebreak_filter}
\begin{document}
nodetree
\end{document}
```

3. `_nodetree-jobname/1.nttex`: This temporary LuaLaTeXfile is compiled using `latexmk` and embed in the environment `NodetreeEmbed`

```
\par{}\par{}Callback: \textcolor{NTEred}{post\_linebreak\_filter}\par{}
----------------------------------------\par{}
\mbox{ \textcolor{NTEmagentabright}{GLUE\hspace{0.5em}}\textcolor{NTEyellow}{subtype:}
↪  baselineskip, \textcolor{NTEyellow}{width:}
↪  5.06\textcolor{NTEwhite}{pt}}\par{}
...
```

4. Finally the result:

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 5.06pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 6.94pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'n', width: 5.56pt, height: 4.42pt
    ├─GLYPH subtype: 256, char: 'o', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─KERN kern: 0.28pt
```

```
    ├─GLYPH subtype: 256, char: 'd', width: 5.56pt, height: 6.94pt, depth: 0.11pt
    │  properties: {['injections'] = {['leftkern'] = 18350.08}}
    ├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    ├─DISC subtype: regular, penalty: 50
    │ └─pre:
    │   └─GLYPH subtype: 256, char: '-', width: 3.33pt, height: 2.45pt
    ├─GLYPH subtype: 256, char: 't', width: 3.89pt, height: 6.15pt, depth: 0.11pt
    ├─GLYPH subtype: 256, char: 'r', width: 3.92pt, height: 4.42pt
    ├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    ├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

2.4.1 Available macros

Macro name	Reference
`\NodetreeRegisterCallback{`⟨*callbacks*⟩`}`	Page 12, Section 3.1
`\NodetreeUnregisterCallback{`⟨*callbacks*⟩`}`	Page 12, Section 3.2
`\NodetreeSetOption[`⟨*option*⟩`]{`⟨*value*⟩`}`	Page 12, Section 3.3
`\NodetreeResetOption{`⟨*option*⟩`}`	Page 12, Section 3.4
`\NodetreeReset`	Page 12, Section 3.6
`\NodetreeSet{`⟨*kv-options*⟩`}`	Page 12, Section 3.5
`\NodetreeEmbedCmd[`⟨*kv-options*⟩`]{`⟨*tex-markup*⟩`}`	Page 12, Section 3.7
`\NodetreeEmbedInput[`⟨*kv-options*⟩`]{`⟨*nttex-file*⟩`}`	Page 12, Section 3.8

2.4.2 Available environment

Environment name	Reference
`\begin{NodetreeEmbedEnv}[`⟨*kv-options*⟩`]`	Page 13, Section 4.1

2.4.3 Available options

Option name	Reference
`callback`	Page 14, Section 5.1
`channel`	Page 14, Section 5.2
`verbosity`	Page 14, Section 5.3
`color`	Page 15, Section 5.4
`unit`	Page 15, Section 5.5
`decimalplaces`	Page 17, Section 5.6
`theme`	Page 18, Section 5.7
`thememode`	Page 18, Section 5.7
`font`	Page 19, Section 5.8
`fontsize`	Page 19, Section 5.9

3 Macros

3.1 \NodetreeRegisterCallback

\NodetreeRegisterCallback

\NodetreeRegisterCallback{⟨*callbacks*⟩}: The argument {⟨*callbacks*⟩} takes a comma separated list of callback aliases as described in (→ 5.1).

3.2 \NodetreeUnregisterCallback

\NodetreeUnregisterCallback

\NodetreeUnregisterCallback{⟨*callbacks*⟩}: The argument {⟨*callbacks*⟩} takes a comma separated list of callback aliases as described in (→ 5.1).

3.3 \NodetreeSetOption

\NodetreeSetOption

\NodetreeSetOption[⟨*option*⟩]{⟨*value*⟩}: (→ 5) This macro sets a single [⟨*option*⟩] to {⟨*value*⟩}.

3.4 \NodetreeResetOption

\NodetreeResetOption

\NodetreeResetOption{⟨*option*⟩}: (→ 5) This macro resets a single {⟨*option*⟩} to its default value.

3.5 \NodetreeSet

\NodetreeSet

\NodetreeSet{⟨*kv-options*⟩}: This macro sets multiple options a once. It only can be used along with LuaLaTeX. {⟨*kv-options*⟩} are key value pairs.

```
\NodetreeSet{color=no,callbacks={hpack,vpack},verbosity=2}
```

3.6 \NodetreeReset

\NodetreeReset

\NodetreeReset: This macro resets multiple options to its default values.

3.7 \NodetreeEmbedCmd

\NodetreeEmbedCmd

\NodetreeEmbedCmd[⟨*kv-options*⟩]{⟨*tex-markup*⟩}:

Main macro (cmd) to evaluate some TeX markup and generate a node tree from it. See environment version. (→ 3.7). Uses the xparse +v option to grab the verbatim content. Only available in the package nodetree-embed.

3.8 \NodetreeEmbedInput

\NodetreeEmbedInput

\NodetreeEmbedInput[⟨*kv-options*⟩]{⟨*nttex-file*⟩}: The path or filename of *.nttex file without the extension. Only available in the package nodetree-embed.

4 Environments

4.1 NodetreeEmbedEnv

NodetreeEmbedEnv `\begin{NodetreeEmbedEnv}[`⟨*kv-options*⟩`] ...`*TEX markup for evaluation*`...\end{NodetreeEmbedEnv}`

Main environment (env) to evaluate some TEX markup and generate a node tree from it. See command version (→ 3.7). Uses the `\detokenize` command to grab the verbatim content. Only available in the package nodetree-embed.

5 Options

5.1 Option `callback`

The option `callback` is the most important setting of the package. It is possible to specify an alias to select the `callback`. Take a look the overview of callbacks (→ Figure 1). `nodetree` supports all node related callbacks as listed in the LuaTEXreference manual.

This macros process callback options: `\NodetreeRegisterCallback{`⟨*callbacks*⟩`}`, `\NodetreeUnregisterCallback{`⟨*callbacks*⟩`}`, `\NodetreeSet{`⟨*callback=<callbacks>*⟩`}` and `\usepackage[`⟨*callback=<callbacks>*⟩`]{`⟨*nodetree*⟩`}`.

Use commas to specify mulitple callbacks. Avoid using whitespaces:

```
\NodetreeRegisterCallback{preline,line,postline}
```

Wrap your callback aliases in curly braces for the macro `\NodetreeSet`:

```
\NodetreeSet{callback={preline,line,postline}}
```

The same applies for the macro `\usepackage`:

```
\usepackage{callback={preline,line,postline}}
```

5.2 Option `channel`

You can select the debug output channel with this option. The default value for the option `channel` is `term` which displays the node tree in the current terminal. Specify `log` and the package creates a log file named `jobname.ntlog`. Specify `tex` and a log file named `jobname.nttex` is creatred. `nt...` stands for `nodetree`. `jobname` is the basename of your file you want to debug. The debug channel is only useful for the auxiliary package `nodetree-embed`. Paste the markup in the environment `NodetreeEmbedView` and you get a terminal like view in your document.

5.3 Option `verbosity`

Higher integer values result in a more verbose output. The default value for this options is `1`. At the moment only verbosity level `2` is implemented.

5.3.1 Example: `verbosity=1`

```
Callback: pre_linebreak_filter
----------------------------------------
├─LOCAL_PAR
├─HLIST subtype: indent, width: 15pt
├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
├─PENALTY subtype: linepenalty, penalty: 10000
└─GLUE subtype: parfillskip, stretch: +1fil
```

The callbacks are listed in the same order as in the LuaTEXreference manual.

Callback	Alias	Alias (longer)
contribute_filter	contribute	contributefilter
buildpage_filter	buildfilter	buildpagefilter
build_page_insert	buildinsert	buildpageinsert
pre_linebreak_filter	preline	prelinebreakfilter
linebreak_filter	line	linebreakfilter
append_to_vlist_filter	append	appendtovlistfilter
post_linebreak_filter	postline	postlinebreakfilter
hpack_filter	hpack	hpackfilter
vpack_filter	vpack	vpackfilter
hpack_quality	hpackq	hpackquality
vpack_quality	vpackq	vpackquality
process_rule	process	processrule
pre_output_filter	preout	preoutputfilter
hyphenate	hyph	
ligaturing	liga	
kerning	kern	
insert_local_par	insert	insertlocalpar
mlist_to_hlist	mhlist	mlisttohlist

Figure 1: The callback aliases

```
----------------------
```

5.3.2 Example: verbosity=2

```
Callback: pre_linebreak_filter
----------------------------------------
├─LOCAL_PAR[9] no: 460
├─HLIST[0] no: 354, subtype: indent[3], width: 15pt
├─GLYPH[29] no: 398, subtype: 256, char: '.', font: 25, width: 2.78pt, height: 1.06pt
├─PENALTY[14] no: 329, subtype: linepenalty[2], penalty: 10000
└─GLUE[12] no: 466, subtype: parfillskip[15], stretch: +1fil
----------------------
```

5.4 Option color

The default option for color is colored. Use any other string (for example none or no) to disable the colored terminal output of the package.

```
\usepackage[color=no]{nodetree}
```

5.5 Option unit

The option unit sets the length unit to display all length values of the nodes. The default option for unit is pt. See figure 2 and 3 for possible values.

Unit	Description
pt	Point 1/72.27 inch. The conversion to metric units, to two decimal places, is 1 point = 2.85 mm = 28.45 cm.
pc	Pica, 12 pt
in	Inch, 72.27 pt
bp	Big point, 1/72 inch. This length is the definition of a point in PostScript and many desktop publishing systems.
cm	Centimeter
mm	Millimeter
dd	Didot point, 1.07 pt
cc	Cicero, 12 dd
sp	Scaled point, 1/65536 pt

Figure 2: Fixed units

Unit	Description
ex	x-height of the current font
em	Width of the capital letter M

Figure 3: Relative units

5.5.1 Example: `unit=pt`

```
Callback: pre_linebreak_filter
----------------------------------------
├─LOCAL_PAR
├─HLIST subtype: indent, width: 15pt
├─GLYPH subtype: 256, char: 'L', width: 6.25pt, height: 6.83pt
├─GLYPH subtype: 256, char: 'o', width: 5pt, height: 4.48pt, depth: 0.11pt
├─GLYPH subtype: 256, char: 'r', width: 3.92pt, height: 4.42pt
├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
├─GLYPH subtype: 256, char: 'm', width: 8.33pt, height: 4.42pt
├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
├─PENALTY subtype: linepenalty, penalty: 10000
└─GLUE subtype: parfillskip, stretch: +1fil
----------------------
```

5.5.2 Example: `unit=sp`

```
Callback: pre_linebreak_filter
----------------------------------------
├─LOCAL_PAR
├─HLIST subtype: indent, width: 983040sp
├─GLYPH subtype: 256, char: 'L', width: 409600sp, height: 447611sp
├─GLYPH subtype: 256, char: 'o', width: 327680sp, height: 293601sp, depth: 7209sp
├─GLYPH subtype: 256, char: 'r', width: 256901sp, height: 289669sp
├─GLYPH subtype: 256, char: 'e', width: 290980sp, height: 293601sp, depth: 7209sp
├─GLYPH subtype: 256, char: 'm', width: 545915sp, height: 289669sp
├─GLYPH subtype: 256, char: '.', width: 182190sp, height: 69468sp
├─PENALTY subtype: linepenalty, penalty: 10000
└─GLUE subtype: parfillskip, stretch: +1fil
----------------------
```

5.5.3 Example: `unit=cm`

```
Callback: pre_linebreak_filter
----------------------------------------
├─LOCAL_PAR
├─HLIST subtype: indent, width: 0.53cm
├─GLYPH subtype: 256, char: 'L', width: 0.22cm, height: 0.24cm
├─GLYPH subtype: 256, char: 'o', width: 0.18cm, height: 0.16cm, depth: 0cm
├─GLYPH subtype: 256, char: 'r', width: 0.14cm, height: 0.16cm
├─GLYPH subtype: 256, char: 'e', width: 0.16cm, height: 0.16cm, depth: 0cm
├─GLYPH subtype: 256, char: 'm', width: 0.29cm, height: 0.16cm
├─GLYPH subtype: 256, char: '.', width: 0.1cm, height: 0.04cm
├─PENALTY subtype: linepenalty, penalty: 10000
└─GLUE subtype: parfillskip, stretch: +1fil
----------------------
```

5.6 Option `decimalplaces`

The options `decimalplaces` sets the number of decimal places for some node fields. If `decimalplaces` is set to 0 only integer values are shown.

```
\NodetreeSetOption[decimalplaces]{4}
```

5.6.1 Example: `decimalplaces=0`

```
Callback: pre_linebreak_filter
----------------------------------------
├─LOCAL_PAR
├─HLIST subtype: indent, width: 1cc
├─GLYPH subtype: 256, char: 'L', width: 0cc, height: 1cc
├─GLYPH subtype: 256, char: 'o', width: 0cc, height: 0cc, depth: 0cc
├─GLYPH subtype: 256, char: 'r', width: 0cc, height: 0cc
├─GLYPH subtype: 256, char: 'e', width: 0cc, height: 0cc, depth: 0cc
├─GLYPH subtype: 256, char: 'm', width: 1cc, height: 0cc
├─GLYPH subtype: 256, char: '.', width: 0cc, height: 0cc
├─PENALTY subtype: linepenalty, penalty: 10000
└─GLUE subtype: parfillskip, stretch: +1fil
----------------------
```

5.6.2 Example: `decimalplaces=2`

```
Callback: pre_linebreak_filter
----------------------------------------
├─LOCAL_PAR
├─HLIST subtype: indent, width: 1.17cc
├─GLYPH subtype: 256, char: 'L', width: 0.49cc, height: 0.53cc
├─GLYPH subtype: 256, char: 'o', width: 0.39cc, height: 0.35cc, depth: 0.01cc
├─GLYPH subtype: 256, char: 'r', width: 0.31cc, height: 0.34cc
├─GLYPH subtype: 256, char: 'e', width: 0.35cc, height: 0.35cc, depth: 0.01cc
├─GLYPH subtype: 256, char: 'm', width: 0.65cc, height: 0.34cc
├─GLYPH subtype: 256, char: '.', width: 0.22cc, height: 0.08cc
├─PENALTY subtype: linepenalty, penalty: 10000
└─GLUE subtype: parfillskip, stretch: +1fil
----------------------
```

5.6.3 Example: `decimalplaces=5`

```
Callback: pre_linebreak_filter
----------------------------------------
├─LOCAL_PAR
├─HLIST subtype: indent, width: 1.16821cc
├─GLYPH subtype: 256, char: 'L', width: 0.48676cc, height: 0.53193cc
├─GLYPH subtype: 256, char: 'o', width: 0.3894cc, height: 0.34891cc, depth: 0.00857cc
├─GLYPH subtype: 256, char: 'r', width: 0.30529cc, height: 0.34423cc
├─GLYPH subtype: 256, char: 'e', width: 0.34579cc, height: 0.34891cc, depth: 0.00857cc
├─GLYPH subtype: 256, char: 'm', width: 0.64875cc, height: 0.34423cc
├─GLYPH subtype: 256, char: '.', width: 0.21651cc, height: 0.08255cc
├─PENALTY subtype: linepenalty, penalty: 10000
└─GLUE subtype: parfillskip, stretch: +1fil
----------------------
```

5.7 Option theme and thememode

5.7.1 Example: theme=bwdark thememode=dark

```
Callback: pre_linebreak_filter
----------------------------------------
├─LOCAL_PAR
├─HLIST subtype: indent, width: 15pt
├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
├─PENALTY subtype: linepenalty, penalty: 10000
└─GLUE subtype: parfillskip, stretch: +1fil
----------------------
```

5.7.2 Example: theme=bwlight thememode=light

```
Callback: pre_linebreak_filter
----------------------------------------
├─LOCAL_PAR
├─HLIST subtype: indent, width: 15
├─GLYPH subtype: 256, char: '.', width: 2.78   , height: 1.06
├─PENALTY subtype: linepenalty, penalty: 10000
└─GLUE subtype: parfillskip, stretch: +1
----------------------
```

5.7.3 Example: theme=monokaisoda thememode=dark

```
Callback: pre_linebreak_filter
----------------------------------------
├─LOCAL_PAR
├─HLIST subtype: indent, width: 15pt
├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
├─PENALTY subtype: linepenalty, penalty: 10000
└─GLUE subtype: parfillskip, stretch: +1fil
----------------------
```

5.7.4 Example: theme=monokaisoda thememode=light

```
Callback: pre_linebreak_filter
----------------------------------------
```

```
├─LOCAL_PAR
├─HLIST subtype: indent, width: 15pt
├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
├─PENALTY subtype: linepenalty, penalty: 10000
└─GLUE subtype: parfillskip, stretch: +1fil
----------------------
```

5.8 Option `font`

`nodetree-embed` passes the option `font` down to the command `\setmonofont{}` of the `fontspec` package. The used font should be a monospaced and have some box drawing glyphs (See table UNICODE glyphs 4).

5.8.1 Example: `font={Liberation Mono}`

```
Callback: post_linebreak_filter
-----------------------------------------
├─GLUE subtype: baselineskip, width: 10.94pt
└─HLIST subtype: line, width: 345pt, height: 1.06pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
------------------------
```

5.8.2 Example: `font={Ubuntu Mono}`

```
Callback: post_linebreak_filter
-----------------------------------------
├─GLUE subtype: baselineskip, width: 10.94pt
└─HLIST subtype: line, width: 345pt, height: 1.06pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

5.9 Option `fontsize`

5.9.1 Example: \small

```
Callback: pre_linebreak_filter
----------------------------------------
├─LOCAL_PAR
├─HLIST subtype: indent, width: 15pt
```

```
├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
├─PENALTY subtype: linepenalty, penalty: 10000
└─GLUE subtype: parfillskip, stretch: +1fil
-----------------------
```

5.9.2 Example: \tiny

```
Callback: pre_linebreak_filter
----------------------------------------
├─LOCAL_PAR
├─HLIST subtype: indent, width: 15pt
├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
├─PENALTY subtype: linepenalty, penalty: 10000
└─GLUE subtype: parfillskip, stretch: +1fil
----------------------
```

Code	Character	Name
U+2500	─	BOX DRAWINGS LIGHT HORIZONTAL
U+2502	│	BOX DRAWINGS LIGHT VERTICAL
U+2514	└	BOX DRAWINGS LIGHT UP AND RIGHT
U+251C	├	BOX DRAWINGS LIGHT VERTICAL AND RIGHT
U+2550	═	BOX DRAWINGS DOUBLE HORIZONTAL
U+2551	║	BOX DRAWINGS DOUBLE VERTICAL
U+255A	╚	BOX DRAWINGS DOUBLE UP AND RIGHT
U+2560	╠	BOX DRAWINGS DOUBLE VERTICAL AND RIGHT

Figure 4: The UNICODE box drawings glyphs

6 Visual tree structure

6.1 Two different connections

Nodes in LuaTEX are connected. The `nodetree` package distinguishs between the `list` and `field` connections.

- `list`: Nodes, which are double connected by `next` and `previous` fields.
- `field`: Connections to nodes by other fields than `next` and `previous` fields, e. g. `head`, `pre`.

6.2 Unicode characters to show the tree view

The package `nodetree` uses the unicode box drawing symbols. Your default terminal font should contain this characters to obtain the tree view. Eight box drawing characters are necessary.
For `list` connections *light* characters are shown.

```
│ │
│ ├─list1
│ └─list2
└─list3
```

`field` connections are visialized by *Double* characters.

```
║ ║
║ ╠═field1
║ ╚═field2
╚═field3
```

7 Examples

In this section lists some examples of the `nodetree` output.

7.1 The node list of the package name

`nodetree`

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 5.06pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 6.94pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'n', width: 5.56pt, height: 4.42pt
    ├─GLYPH subtype: 256, char: 'o', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─KERN kern: 0.28pt
    ├─GLYPH subtype: 256, char: 'd', width: 5.56pt, height: 6.94pt, depth: 0.11pt
    │  properties: {['injections'] = {['leftkern'] = 18350.08}}
    ├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    ├─DISC subtype: regular, penalty: 50
    │ └─pre:
    │   └─GLYPH subtype: 256, char: '-', width: 3.33pt, height: 2.45pt
    ├─GLYPH subtype: 256, char: 't', width: 3.89pt, height: 6.15pt, depth: 0.11pt
    ├─GLYPH subtype: 256, char: 'r', width: 3.92pt, height: 4.42pt
    ├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    ├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.2 The node list of a mathematical formula

`$1+2$`

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 5.56pt
└─HLIST subtype: line, width: 345pt, depth: 0.83pt, height: 6.44pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─MATH
    ├─GLYPH subtype: 256, char: '1', width: 5pt, height: 6.44pt
    ├─GLUE subtype: medmuskip, width: 2.22pt, stretch: 1.11pt, shrink: 2.22pt
    ├─GLYPH subtype: 256, char: '+', width: 7.78pt, height: 5.83pt, depth: 0.83pt
    ├─PENALTY subtype: noadpenalty, penalty: 700
    ├─GLUE subtype: medmuskip, width: 2.22pt, stretch: 1.11pt, shrink: 2.22pt
    ├─GLYPH subtype: 256, char: '2', width: 5pt, height: 6.44pt
    ├─MATH subtype: endmath
```

```
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.3 The node list of the word *Office*

The characters *ffi* are deeply nested in a discretionary node.
`Office`

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 4.95pt
└─HLIST subtype: line, width: 345pt, depth: 0.22pt, height: 7.05pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'O', width: 7.78pt, height: 7.05pt, depth: 0.22pt
    ├─DISC subtype: regular, penalty: 50
    │ ├─pre:
    │ ‖ ├─GLYPH subtype: 256, char: 'f', width: 3.06pt, height: 7.05pt
    │ ‖ └─GLYPH subtype: 256, char: '-', width: 3.33pt, height: 2.45pt
    │ ├─replace:
    │ ‖ └─GLYPH subtype: 258, char: '', width: 8.33pt, height: 7.05pt
    │ ‖   └─components:
    │ ‖     ├─GLYPH subtype: ghost, char: '', width: 5.83pt, height: 7.05pt
    │ ‖     │ └─components:
    │ ‖     │   ├─GLYPH char: 'f', width: 3.06pt, height: 7.05pt
    │ ‖     │   └─GLYPH char: 'f', width: 3.06pt, height: 7.05pt
    │ ‖     └─GLYPH char: 'i', width: 2.78pt, height: 6.57pt
    │ └─post:
    │   └─GLYPH subtype: 258, char: 'fi', width: 5.56pt, height: 7.05pt
    │     └─components:
    │       ├─GLYPH char: 'f', width: 3.06pt, height: 7.05pt
    │       └─GLYPH char: 'i', width: 2.78pt, height: 6.57pt
    ├─GLYPH subtype: 256, char: 'c', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    ├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4 Node types

This chapter shows some node types in a `nodetree` view.

7.4.1 Type: `hlist(0)` Subtype: `line(1)`

`Lorem`

```
Callback: post_linebreak_filter
----------------------------------------
```

```
  ├─GLUE subtype: baselineskip, width: 5.17pt
  └─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 6.83pt
    └─head:
      ├─LOCAL_PAR
      ├─HLIST subtype: indent, width: 15pt
      ├─GLYPH subtype: 256, char: 'L', width: 6.25pt, height: 6.83pt
      ├─GLYPH subtype: 256, char: 'o', width: 5pt, height: 4.48pt, depth: 0.11pt
      ├─GLYPH subtype: 256, char: 'r', width: 3.92pt, height: 4.42pt
      ├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
      ├─GLYPH subtype: 256, char: 'm', width: 8.33pt, height: 4.42pt
      ├─PENALTY subtype: linepenalty, penalty: 10000
      ├─GLUE subtype: parfillskip, stretch: +1fil
      └─GLUE subtype: rightskip
----------------------
```

7.4.2 Type: hlist(0) Subtype: box(2)

L\hbox to 40pt{ore}m

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 5.17pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 6.83pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'L', width: 6.25pt, height: 6.83pt
    ├─HLIST subtype: box, width: 40pt, depth: 0.11pt, height: 4.48pt
    │ └─head:
    │   ├─GLYPH subtype: 256, char: 'o', width: 5pt, height: 4.48pt, depth: 0.11pt
    │   ├─GLYPH subtype: 256, char: 'r', width: 3.92pt, height: 4.42pt
    │   └─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    ├─GLYPH subtype: 256, char: 'm', width: 8.33pt, height: 4.42pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.3 Type: hlist(0) Subtype: indent(3)

\setlength {\parindent }{5cm} I

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 0.18cm
└─HLIST subtype: line, width: 12.13cm, height: 0.24cm
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 5cm
    ├─GLYPH subtype: 256, char: 'I', width: 0.13cm, height: 0.24cm
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
```

```
----------------------
```

7.4.4 Type: vlist(1)

L\vbox to 40pt{O}L

```
Callback: post_linebreak_filter
- groupcode: vbox
----------------------------------------
└─HLIST subtype: line, width: 12.13cm, depth: 0.01cm, height: 0.25cm
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 0.53cm
    ├─GLYPH subtype: 256, char: 'O', width: 0.27cm, height: 0.25cm, depth: 0.01cm
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: lineskip, width: 0.04cm
└─HLIST subtype: line, width: 12.13cm, depth: 0.01cm, height: 1.41cm
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 0.53cm
    ├─GLYPH subtype: 256, char: 'L', width: 0.22cm, height: 0.24cm
    ├─VLIST width: 12.13cm, depth: 0.01cm, height: 1.41cm
    │ └─head:
    │   └─HLIST subtype: line, width: 12.13cm, depth: 0.01cm, height: 0.25cm
    │     └─head:
    │       ├─LOCAL_PAR
    │       ├─HLIST subtype: indent, width: 0.53cm
    │       ├─GLYPH subtype: 256, char: 'O', width: 0.27cm, height: 0.25cm, depth: 0.01cm
    │       ├─PENALTY subtype: linepenalty, penalty: 10000
    │       ├─GLUE subtype: parfillskip, stretch: +1fil
    │       └─GLUE subtype: rightskip
    ├─GLYPH subtype: 256, char: 'L', width: 0.22cm, height: 0.24cm
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.5 Type: rule(2)

\rule [-2mm]{10mm}{4mm}

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 2.22mm
└─HLIST subtype: line, width: 121.25mm, depth: 2mm, height: 2mm
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 5.27mm
```

```
    ├─HLIST subtype: box, width: 10㎜, depth: 2㎜, height: 2㎜
    │ └─head:
    │   └─RULE width: 10㎜, depth: 2㎜, height: 2㎜
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.6 Type: mark(4)

\mark {Lorem}.

```
Callback: pre_output_filter
- size: 36044800
- packtype: exactly
- direction: TLT
- groupcode: output
- maxdepth: 327680
----------------------------------------
├─WHATSIT subtype: write, stream: 129, data:
├─MARK mark: table: 0x599abe0
├─GLUE subtype: topskip, width: 3.14㎜
├─HLIST subtype: line, width: 121.25㎜, height: 0.37㎜
│ └─head:
│   ├─LOCAL_PAR
│   ├─HLIST subtype: indent, width: 5.27㎜
│   ├─GLYPH subtype: 256, char: '.', width: 0.98㎜, height: 0.37㎜
│   ├─PENALTY subtype: linepenalty, penalty: 10000
│   ├─GLUE subtype: parfillskip, stretch: +1fil
│   └─GLUE subtype: rightskip
└─GLUE stretch: +1fil
----------------------
Callback: pre_output_filter
- size: 36044800
- packtype: exactly
- direction: TLT
- groupcode: output
- maxdepth: 327680
----------------------------------------
├─WHATSIT subtype: write, stream: 129, data:
├─GLUE subtype: topskip, width: 3.51㎜
└─VLIST
----------------------
Callback: pre_output_filter
- size: 36044800
- packtype: exactly
- direction: TLT
- groupcode: output
- maxdepth: 327680
----------------------------------------
├─WHATSIT subtype: write, stream: 129, data:
├─GLUE subtype: topskip, width: 3.51㎜
├─HLIST width: 121.25㎜
```

```
└─GLUE stretch: +1fill
----------------------
```

7.4.7 Type: disc(7) Subtype: discretionary(0)

L\-O\-L

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 4.95pt
└─HLIST subtype: line, width: 345pt, depth: 0.22pt, height: 7.05pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'L', width: 6.25pt, height: 6.83pt
    ├─DISC penalty: 50
    │ └─pre:
    │   └─GLYPH subtype: 256, char: '-', width: 3.33pt, height: 2.45pt
    ├─GLYPH subtype: 256, char: 'O', width: 7.78pt, height: 7.05pt, depth: 0.22pt
    ├─DISC penalty: 50
    │ └─pre:
    │   └─GLYPH subtype: 256, char: '-', width: 3.33pt, height: 2.45pt
    ├─GLYPH subtype: 256, char: 'L', width: 6.25pt, height: 6.83pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.8 Type: disc(7) Subtype: regular(3)

Office

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 4.95pt
└─HLIST subtype: line, width: 345pt, depth: 0.22pt, height: 7.05pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'O', width: 7.78pt, height: 7.05pt, depth: 0.22pt
    ├─DISC subtype: regular, penalty: 50
    │ ├─post:
    │ ‖ └─GLYPH subtype: 258, char: 'fi', width: 5.56pt, height: 7.05pt
    │ ‖   └─components:
    │ ‖     ├─GLYPH char: 'f', width: 3.06pt, height: 7.05pt
    │ ‖     └─GLYPH char: 'i', width: 2.78pt, height: 6.57pt
    │ ├─pre:
    │ ‖ ├─GLYPH subtype: 256, char: 'f', width: 3.06pt, height: 7.05pt
    │ ‖ └─GLYPH subtype: 256, char: '-', width: 3.33pt, height: 2.45pt
    │ └─replace:
    │   └─GLYPH subtype: 258, char: '', width: 8.33pt, height: 7.05pt
    │     └─components:
    │       ├─GLYPH subtype: ghost, char: '', width: 5.83pt, height: 7.05pt
```

```
      |       | └─components:
      |       |   ├─GLYPH char: 'f', width: 3.06pt, height: 7.05pt
      |       |   └─GLYPH char: 'f', width: 3.06pt, height: 7.05pt
      |       └─GLYPH char: 'i', width: 2.78pt, height: 6.57pt
      ├─GLYPH subtype: 256, char: 'c', width: 4.44pt, height: 4.48pt, depth: 0.11pt
      ├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
      ├─PENALTY subtype: linepenalty, penalty: 10000
      ├─GLUE subtype: parfillskip, stretch: +1fil
      └─GLUE subtype: rightskip
----------------------
```

7.4.9 Type: whatsit(8) Subtype: pdfaction(22)

```
\usepackage{hyperref}
\begin{document}
\url{http:/luatex.org}
\end{document}
```

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 5.06pt
└─HLIST subtype: line, width: 345pt, depth: 2.29pt, height: 6.94pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─WHATSIT subtype: pdf_start_link, width: -16384pt, depth: -16384pt, height: -16384pt, objnum: 4
    | └─action:
    |   └─WHATSIT subtype: pdf_action, action_type: 3, file: , data: /Subtype/Link/A<</Type/Action/S
    ├─MATH
    ├─GLYPH subtype: 256, char: 'h', width: 5.25pt, height: 6.11pt
    ├─GLYPH subtype: 256, char: 't', width: 5.25pt, height: 5.54pt, depth: 0.06pt
    ├─GLYPH subtype: 256, char: 't', width: 5.25pt, height: 5.54pt, depth: 0.06pt
    ├─GLYPH subtype: 256, char: 'p', width: 5.25pt, height: 4.37pt, depth: 2.22pt
    ├─GLUE subtype: thickmuskip
    ├─GLYPH subtype: 256, char: ':', width: 5.25pt, height: 4.31pt
    ├─PENALTY subtype: noadpenalty, penalty: 500
    ├─GLUE subtype: thickmuskip
    ├─GLYPH subtype: 256, char: '/', width: 5.25pt, height: 6.94pt, depth: 0.83pt
    ├─GLYPH subtype: 256, char: 'l', width: 5.25pt, height: 6.11pt
    ├─GLYPH subtype: 256, char: 'u', width: 5.25pt, height: 4.31pt, depth: 0.06pt
    ├─GLYPH subtype: 256, char: 'a', width: 5.25pt, height: 4.4pt, depth: 0.06pt
    ├─GLYPH subtype: 256, char: 't', width: 5.25pt, height: 5.54pt, depth: 0.06pt
    ├─GLYPH subtype: 256, char: 'e', width: 5.25pt, height: 4.4pt, depth: 0.06pt
    ├─GLYPH subtype: 256, char: 'x', width: 5.25pt, height: 4.31pt
    ├─GLUE subtype: medmuskip
    ├─GLYPH subtype: 256, char: '.', width: 5.25pt, height: 1.25pt
    ├─PENALTY subtype: noadpenalty, penalty: 700
    ├─GLUE subtype: medmuskip
    ├─GLYPH subtype: 256, char: 'o', width: 5.25pt, height: 4.4pt, depth: 0.06pt
    ├─GLYPH subtype: 256, char: 'r', width: 5.25pt, height: 4.37pt
    ├─GLYPH subtype: 256, char: 'g', width: 5.25pt, height: 4.42pt, depth: 2.29pt
    ├─MATH subtype: endmath
```

```
    ├─WHATSIT subtype: pdf_end_link
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.10 Type: whatsit(8) Subtype: pdfcolorstack(28)

```
\usepackage{color}
\begin{document}
Lo\textcolor{red}{re}m.
\end{document}
```

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 5.17pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 6.83pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'L', width: 6.25pt, height: 6.83pt
    ├─GLYPH subtype: 256, char: 'o', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─WHATSIT subtype: pdf_colorstack, data: 1 0 0 rg 1 0 0 RG
    ├─GLYPH subtype: 256, char: 'r', width: 3.92pt, height: 4.42pt
    ├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    ├─WHATSIT subtype: pdf_colorstack, data:
    ├─GLYPH subtype: 256, char: 'm', width: 8.33pt, height: 4.42pt
    ├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.11 Type: glue(12) Subtype: baselineskip(2)

```
\baselineskip=5cm Lorem Lorem
```

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 4.76cm
└─HLIST subtype: line, width: 12.13cm, depth: 0cm, height: 0.24cm
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 0.53cm
    ├─GLYPH subtype: 256, char: 'L', width: 0.22cm, height: 0.24cm
    ├─GLYPH subtype: 256, char: 'o', width: 0.18cm, height: 0.16cm, depth: 0cm
    ├─GLYPH subtype: 256, char: 'r', width: 0.14cm, height: 0.16cm
    ├─GLYPH subtype: 256, char: 'e', width: 0.16cm, height: 0.16cm, depth: 0cm
    ├─GLYPH subtype: 256, char: 'm', width: 0.29cm, height: 0.16cm
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
```

```
----------------------
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 4.76cm
└─HLIST subtype: line, width: 12.13cm, depth: 0cm, height: 0.24cm
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 0.53cm
    ├─GLYPH subtype: 256, char: 'L', width: 0.22cm, height: 0.24cm
    ├─GLYPH subtype: 256, char: 'o', width: 0.18cm, height: 0.16cm, depth: 0cm
    ├─GLYPH subtype: 256, char: 'r', width: 0.14cm, height: 0.16cm
    ├─GLYPH subtype: 256, char: 'e', width: 0.16cm, height: 0.16cm, depth: 0cm
    ├─GLYPH subtype: 256, char: 'm', width: 0.29cm, height: 0.16cm
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.12 Type: glue(12) Subtype: parskip(3)

\parskip=5cm Lorem Lorem

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 5.17pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 6.83pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'L', width: 6.25pt, height: 6.83pt
    ├─GLYPH subtype: 256, char: 'o', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─GLYPH subtype: 256, char: 'r', width: 3.92pt, height: 4.42pt
    ├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    ├─GLYPH subtype: 256, char: 'm', width: 8.33pt, height: 4.42pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 5.06pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 6.83pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'L', width: 6.25pt, height: 6.83pt
    ├─GLYPH subtype: 256, char: 'o', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─GLYPH subtype: 256, char: 'r', width: 3.92pt, height: 4.42pt
    ├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    ├─GLYPH subtype: 256, char: 'm', width: 8.33pt, height: 4.42pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
```

```
----------------------
```

7.4.13 Type: glue(12) Subtype: spaceskip(13)

```
\spaceskip =5cm a a
```

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 7.52pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 4.48pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'a', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─GLUE subtype: spaceskip, width: 142.26pt
    ├─GLYPH subtype: 256, char: 'a', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.14 Type: glue(12) Subtype: leaders(100)

```
a \leavevmode \leaders \hbox { . }\hfill \kern 0pt a
```

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 7.52pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 4.48pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'a', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─GLUE subtype: spaceskip, width: 3.33pt, stretch: 1.66pt, shrink: 1.11pt
    ├─GLUE subtype: leaders, stretch: +1fill
    │ └─leader:
    │   └─HLIST subtype: box, width: 10.55pt, height: 1.06pt
    │     └─head:
    │       ├─GLUE subtype: spaceskip, width: 3.33pt, stretch: 1.66pt, shrink: 1.11pt
    │       ├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
    │       └─GLUE subtype: spaceskip, width: 4.44pt, stretch: 4.99pt, shrink: 0.37pt
    ├─KERN subtype: userkern
    ├─GLYPH subtype: 256, char: 'a', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.15 Type: glue(12) Subtype: cleaders(101)

```
a \leavevmode \cleaders \hbox { . }\hfill \kern 0pt a
```

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 7.52pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 4.48pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'a', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─GLUE subtype: spaceskip, width: 3.33pt, stretch: 1.66pt, shrink: 1.11pt
    ├─GLUE subtype: cleaders, stretch: +1fill
    │ └─leader:
    │   └─HLIST subtype: box, width: 10.55pt, height: 1.06pt
    │     └─head:
    │       ├─GLUE subtype: spaceskip, width: 3.33pt, stretch: 1.66pt, shrink: 1.11pt
    │       ├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
    │       └─GLUE subtype: spaceskip, width: 4.44pt, stretch: 4.99pt, shrink: 0.37pt
    ├─KERN subtype: userkern
    ├─GLYPH subtype: 256, char: 'a', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.16 Type: glue(12) Subtype: xleaders(102)

```
a \leavevmode \xleaders \hbox { . }\hfill \kern 0pt a
```

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 7.52pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 4.48pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'a', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─GLUE subtype: spaceskip, width: 3.33pt, stretch: 1.66pt, shrink: 1.11pt
    ├─GLUE subtype: xleaders, stretch: +1fill
    │ └─leader:
    │   └─HLIST subtype: box, width: 10.55pt, height: 1.06pt
    │     └─head:
    │       ├─GLUE subtype: spaceskip, width: 3.33pt, stretch: 1.66pt, shrink: 1.11pt
    │       ├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
    │       └─GLUE subtype: spaceskip, width: 4.44pt, stretch: 4.99pt, shrink: 0.37pt
    ├─KERN subtype: userkern
    ├─GLYPH subtype: 256, char: 'a', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.17 Type: glue(12) Subtype: gleaders(102)

```
a \leavevmode \gleaders \hbox { . }\hfill \kern 0pt a
```

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 7.52pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 4.48pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'a', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─GLUE subtype: spaceskip, width: 3.33pt, stretch: 1.66pt, shrink: 1.11pt
    ├─GLUE subtype: gleaders, stretch: +1fill
    │ └─leader:
    │   └─HLIST subtype: box, width: 10.55pt, height: 1.06pt
    │     └─head:
    │       ├─GLUE subtype: spaceskip, width: 3.33pt, stretch: 1.66pt, shrink: 1.11pt
    │       ├─GLYPH subtype: 256, char: '.', width: 2.78pt, height: 1.06pt
    │       └─GLUE subtype: spaceskip, width: 4.44pt, stretch: 4.99pt, shrink: 0.37pt
    ├─KERN subtype: userkern
    ├─GLYPH subtype: 256, char: 'a', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.18 Type: kern(13) Subtype: userkern(0)

a\kern 2pt

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 7.52pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 4.48pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'a', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─KERN subtype: userkern, kern: 2pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.19 Type: kern(13) Subtype: fontkern(1)

Ve

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 5.17pt
└─HLIST subtype: line, width: 345pt, depth: 0.22pt, height: 6.83pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'V', width: 7.5pt, height: 6.83pt, depth: 0.22pt
```

```
    ├─KERN kern: -0.83pt
    ├─GLYPH subtype: 256, char: 'e', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    |  properties: {['injections'] = {['leftkern'] = -54394.88}}
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.20 Type: kern(13) Subtype: accentkern(2)

\`{a}

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 5.02pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 6.98pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'à', width: 5pt, height: 6.98pt, depth: 0.11pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.21 Type: kern(13) Subtype: italiccorrection(3)

\textit {L}\/OL

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 4.95pt
└─HLIST subtype: line, width: 345pt, depth: 0.22pt, height: 7.05pt
  └─head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'L', width: 6.27pt, height: 6.83pt
    ├─KERN subtype: italiccorrection, kern: 0.17pt
    ├─GLYPH subtype: 256, char: 'O', width: 7.78pt, height: 7.05pt, depth: 0.22pt
    ├─GLYPH subtype: 256, char: 'L', width: 6.25pt, height: 6.83pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.22 Type: penalty(14)

L \penalty 23 OL

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 4.95pt
└─HLIST subtype: line, width: 345pt, depth: 0.22pt, height: 7.05pt
```

```
  ┗━head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'L', width: 6.25pt, height: 6.83pt
    ├─GLUE subtype: spaceskip, width: 3.33pt, stretch: 1.66pt, shrink: 1.11pt
    ├─PENALTY penalty: 23
    ├─GLYPH subtype: 256, char: 'O', width: 7.78pt, height: 7.05pt, depth: 0.22pt
    ├─GLYPH subtype: 256, char: 'L', width: 6.25pt, height: 6.83pt
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.23 Type: glyph(29)

abc

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 5.06pt
└─HLIST subtype: line, width: 345pt, depth: 0.11pt, height: 6.94pt
  ┗━head:
    ├─LOCAL_PAR
    ├─HLIST subtype: indent, width: 15pt
    ├─GLYPH subtype: 256, char: 'a', width: 5pt, height: 4.48pt, depth: 0.11pt
    ├─GLYPH subtype: 256, char: 'b', width: 5.56pt, height: 6.94pt, depth: 0.11pt
    ├─KERN kern: 0.28pt
    ├─GLYPH subtype: 256, char: 'c', width: 4.44pt, height: 4.48pt, depth: 0.11pt
    |  properties: {['injections'] = {['leftkern'] = 18350.08}}
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.24 Type: attribute(38)

{\attribute 0=1 A}

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 4.84pt
└─HLIST subtype: line, width: 345pt, height: 7.16pt
  ┗━head:
    ├─LOCAL_PAR attr: 0=1
    ├─HLIST subtype: indent, width: 15pt, attr: 0=1
    ├─GLYPH subtype: 256, char: 'A', width: 7.5pt, height: 7.16pt, attr: 0=1
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

7.4.25 Type: attributelist(40)

{\attribute 0=1 A}

```
Callback: post_linebreak_filter
----------------------------------------
├─GLUE subtype: baselineskip, width: 4.84pt
└─HLIST subtype: line, width: 345pt, height: 7.16pt
  └─head:
    ├─LOCAL_PAR attr: 0=1
    ├─HLIST subtype: indent, width: 15pt, attr: 0=1
    ├─GLYPH subtype: 256, char: 'A', width: 7.5pt, height: 7.16pt, attr: 0=1
    ├─PENALTY subtype: linepenalty, penalty: 10000
    ├─GLUE subtype: parfillskip, stretch: +1fil
    └─GLUE subtype: rightskip
----------------------
```

8 Implementation

8.1 The file nodetree.tex

```
\directlua{
  nodetree = require('nodetree')
}
```

\NodetreeSetOption

```
\def\NodetreeSetOption[#1]#2{
  \directlua{
    nodetree.set_option('#1', '#2')
  }
}
\let\nodetreeoption\NodetreeSetOption
```

\NodetreeResetOption

```
\def\NodetreeResetOption#1{
  \NodetreeSetOption[#1]{%
    \directlua{
      tex.print(nodetree.get_default_option('#1'))
    }%
  }%
}
```

\NodetreeReset

```
\def\NodetreeReset{
  \NodetreeResetOption{callback}
  \NodetreeResetOption{channel}
  \NodetreeResetOption{color}
  \NodetreeResetOption{decimalplaces}
  \NodetreeResetOption{engine}
  \NodetreeResetOption{unit}
  \NodetreeResetOption{verbosity}
}
\let\nodetreereset\NodetreeReset
```

\NodetreeRegisterCallback

```
\def\NodetreeRegisterCallback#1{
  \directlua{
    nodetree.set_option('callback', '#1')
    nodetree.register_callbacks()
  }
}
\let\nodetreeregister\NodetreeRegisterCallback
```

\NodetreeUnregisterCallback

```
\def\NodetreeUnregisterCallback#1{
  \directlua{
    nodetree.set_option('callback', '#1')
    nodetree.unregister_callbacks()
  }
}
\let\nodetreeunregister\NodetreeUnregisterCallback
```

8.2 The file nodetree.sty

```
\input{nodetree}

\RequirePackage{kvoptions}

\SetupKeyvalOptions{
  family=NT,
  prefix=NTK@
}

\DeclareStringOption[term]{channel}
\define@key{NT}{channel}[]{\NodetreeSetOption[channel]{#1}}

\DeclareStringOption[postlinebreak]{callback}
\define@key{NT}{callback}[]{\NodetreeSetOption[callback]{#1}}

\DeclareStringOption[1]{verbosity}
\define@key{NT}{verbosity}[]{\NodetreeSetOption[verbosity]{#1}}

\DeclareStringOption[colored]{color}
\define@key{NT}{color}[]{\NodetreeSetOption[color]{#1}}

\DeclareStringOption[1]{unit}
\define@key{NT}{unit}[]{\NodetreeSetOption[unit]{#1}}

\DeclareStringOption[1]{decimalplaces}
\define@key{NT}{decimalplaces}[]{\NodetreeSetOption[decimalplaces]{#1}}

\ProcessKeyvalOptions{NT}
\directlua{
  nodetree.register_callbacks()
}
```

\NodetreeSet

```
\newcommand{\NodetreeSet}[1]{%
  \setkeys{NT}{#1}%
}
\let\nodetreeset\NodetreeSet
```

```
\NeedsTeXFormat{LaTeX2e}[1994/06/01]
\ProvidesPackage{nodetree-embed}
  [2020/05/29 v2.0 Embed node trees into a LaTeX document]

\RequirePackage{xcolor,mdframed,expl3,xparse,fontspec}

\input{nodetree}

\RequirePackage{kvoptions}
\SetupKeyvalOptions{
  family=NTE,
  prefix=NTEK@
}

\directlua{
  nodetree = require('nodetree')
  nodetree.check_shell_escape()
}

\define@key{NTE}{callback}[]{\NodetreeSetOption[callback]{#1}}

\DeclareStringOption[1]{verbosity}
\define@key{NTE}{verbosity}[]{\NodetreeSetOption[verbosity]{#1}}

\DeclareStringOption[colored]{color}
\define@key{NTE}{color}[]{\NodetreeSetOption[color]{#1}}

\DeclareStringOption[1]{unit}
\define@key{NTE}{unit}[]{\NodetreeSetOption[unit]{#1}}

\DeclareStringOption[1]{decimalplaces}
\define@key{NTE}{decimalplaces}[]{\NodetreeSetOption[decimalplaces]{#1}}

\DeclareStringOption[monokaisoda]{theme}

\DeclareStringOption[dark]{thememode}

\DeclareStringOption[Ubuntu Mono]{font}

\DeclareStringOption[\footnotesize]{fontsize}

\DeclareBoolOption{showmarkup}

\ProcessKeyvalOptions{NTE}
```

\NTE@colors

```
\ExplSyntaxOn
\def\NTE@colors{
```

```
\str_case_e:nn{\NTEK@theme}{
  {bwdark}{
    \definecolor{NTEblack}{gray}{0}
    \definecolor{NTEred}{gray}{1}
    \definecolor{NTEgreen}{gray}{1}
    \definecolor{NTEyellow}{gray}{1}
    \definecolor{NTEblue}{gray}{1}
    \definecolor{NTEmagenta}{gray}{1}
    \definecolor{NTEcyan}{gray}{1}
    \definecolor{NTEwhite}{gray}{1}
    \definecolor{NTEblackbright}{gray}{0}
    \definecolor{NTEredbright}{gray}{1}
    \definecolor{NTEgreenbright}{gray}{1}
    \definecolor{NTEyellowbright}{gray}{1}
    \definecolor{NTEbluebright}{gray}{1}
    \definecolor{NTEmagentabright}{gray}{1}
    \definecolor{NTEcyanbright}{gray}{1}
    \definecolor{NTEwhitebright}{gray}{1}
  }
  {bwlight}{
    \definecolor{NTEblack}{gray}{0}
    \definecolor{NTEred}{gray}{0}
    \definecolor{NTEgreen}{gray}{0}
    \definecolor{NTEyellow}{gray}{0}
    \definecolor{NTEblue}{gray}{0}
    \definecolor{NTEmagenta}{gray}{0}
    \definecolor{NTEcyan}{gray}{0}
    \definecolor{NTEwhite}{gray}{1}
    \definecolor{NTEblackbright}{gray}{0}
    \definecolor{NTEredbright}{gray}{0}
    \definecolor{NTEgreenbright}{gray}{0}
    \definecolor{NTEyellowbright}{gray}{0}
    \definecolor{NTEbluebright}{gray}{0}
    \definecolor{NTEmagentabright}{gray}{0}
    \definecolor{NTEcyanbright}{gray}{0}
    \definecolor{NTEwhitebright}{gray}{1}
  }
  {monokaisoda}{
    \definecolor{NTEblack}{HTML}{1a1a1a}
    \definecolor{NTEred}{HTML}{f4005f}
    \definecolor{NTEgreen}{HTML}{98e024}
    \definecolor{NTEyellow}{HTML}{fa8419}
    \definecolor{NTEblue}{HTML}{9d65ff}
    \definecolor{NTEmagenta}{HTML}{f4005f}
    \definecolor{NTEcyan}{HTML}{58d1eb}
    \definecolor{NTEwhite}{HTML}{c4c5b5}
    \definecolor{NTEblackbright}{HTML}{625e4c}
    \definecolor{NTEredbright}{HTML}{f4005f}
    \definecolor{NTEgreenbright}{HTML}{98e024}
    \definecolor{NTEyellowbright}{HTML}{e0d561}
```

```
      \definecolor{NTEbluebright}{HTML}{9d65ff}
      \definecolor{NTEmagentabright}{HTML}{f4005f}
      \definecolor{NTEcyanbright}{HTML}{58d1eb}
      \definecolor{NTEwhitebright}{HTML}{f6f6ef}
    }
  }
  \str_case_e:nn{\NTEK@thememode}{
    {dark}{
      \definecolor{NTEbackground}{named}{NTEblack}
      \definecolor{NTEfont}{named}{NTEwhitebright}
    }
    {light}{
      \definecolor{NTEbackground}{named}{NTEwhitebright}
      \definecolor{NTEfont}{named}{NTEblack}
    }
  }
}
\ExplSyntaxOff
```

`\NTE@fonts`

```
\def\NTE@fonts{
  \bfseries%
  \NTEK@fontsize%
  \setmonofont{\NTEK@font}%
  \ttfamily%
  \setlength{\parindent}{0pt}%
  \setlength{\parskip}{-0.9pt}%
}
```

`\NodetreeSet` Same definition as in nodetree.sty. Only implement this command if not already registers.

```
\providecommand{\NodetreeSet}[1]{%
  \setkeys{NTE}{#1}%
}

\newenvironment{NodetreeEmbedView}[1][]{
  \setkeys{NTE}{#1}
  \NTE@colors
  \begin{mdframed}[
    linecolor=black,
    backgroundcolor=NTEbackground,
    fontcolor=NTEfont,
  ]%
  \NTE@fonts
}{
  \end{mdframed}%
}
```

NodetreeEmbedEnv

```
\NewDocumentEnvironment { NodetreeEmbedEnv } { O{} +b } {
  \setkeys{NTE}{#1}
  \ifNTEK@showmarkup
    \noindent
    \texttt{\detokenize{#2}}
  \else
  \fi
  \NTE@colors
  \begin{NodetreeEmbedView}
    \directlua{
      nodetree.compile_include('\luaescapestring{\unexpanded{#2}}')
    }
  \end{NodetreeEmbedView}
}{}
```

\NodetreeEmbedCmd

```
\NewDocumentCommand { \NodetreeEmbedCmd } { O{} +v } {
  \setkeys{NTE}{#1}
  \ifNTEK@showmarkup
    \noindent
    \texttt{#2}
  \else
  \fi
  \NTE@colors
  \begin{NodetreeEmbedView}
    \directlua{
      nodetree.compile_include('\luaescapestring{\unexpanded{#2}}')
    }
  \end{NodetreeEmbedView}
}
```

\NodetreeEmbedInput

```
\newcommand{\NodetreeEmbedInput}[2][]{
  \setkeys{NTE}{#1}
  \begin{NodetreeEmbedView}
  \input{#2.nttex}
  \end{NodetreeEmbedView}
}
\let\nodetreeterminalemulator\NodetreeEmbedInput
```

8.3 The file `nodetree.lua`

```
--- The nodetree package.
--
-- Nodetree uses [LDoc](https://github.com/stevedonovan/ldoc) for the
--  source code documentation. The supported tags are described on in
```

```
--  the [wiki](https://github.com/stevedonovan/LDoc/wiki).
--
-- Nodes in LuaTeX are connected. The nodetree view distinguishs
-- between the `list` and `field` connections.
--
-- * `list`: Nodes, which are double connected by `next` and
--   `previous` fields.
-- * `field`: Connections to nodes by other fields than `next` and
--   `previous` fields, e. g. `head`, `pre`.
-- @module nodetree

-- luacheck: globals node tex luatexbase lfs callback os unicode status modules

if not modules then modules = { } end modules ['nodetree'] = {
  version   = '2.0',
  comment   = 'nodetree',
  author    = 'Josef Friedrich',
  copyright = 'Josef Friedrich',
  license   = 'The LaTeX Project Public License Version 1.3c 2008-05-04'
}

--- A counter for the compiled TeX examples. Some TeX code snippets
-- a written into file, wrapped with some TeX boilerplate code.
-- This written files are compiled.
local example_counter = 0

--- The default options
local default_options = {
  callback = 'post_linebreak_filter',
  channel = 'term',
  color = 'colored',
  decimalplaces = 2,
  engine = 'luatex', -- Required for the callback registration
  unit = 'pt',
  verbosity = 1,
}

--- The current options
-- They are changed very often.
local options = {}
for key, value in pairs(default_options) do
  options[key] = value
end

if arg[0] == 'lualatex' then
  options.engine = 'lualatex'
end

--- File descriptor
local output_file

--- The lua table named `tree_state` holds state values of the current
-- tree item.
--
-- `tree_state`:
--
-- * `1` (level):
```

```
--   * `list`: `continue`
--   * `field`: `stop`
-- * `2`:
--   * `list`: `continue`
--   * `field`: `stop`
-- @table
local tree_state = {}

--- Format functions.
--
-- Low level template functions.
--
-- @section format

local format = {
  ---
  -- @treturn string
  underscore = function(string)
    if options.channel == 'tex' then
      return string.gsub(string, '_', '\\_')
    else
      return string
    end
  end,

  ---
  -- @treturn string
  escape = function(string)
    if options.channel == 'tex' then
      return string.gsub(string, [[\]], [[\string\]])
    else
      return string
    end
  end,

  -- @treturn number
  number = function(number)
    local mult = 10^(options.decimalplaces or 0)
    return math.floor(number * mult + 0.5) / mult
  end,

  ---
  -- @treturn string
  whitespace = function(count)
    local whitespace
    local output = ''
    if options.channel == 'tex' then
      whitespace = '\\hspace{0.5em}'
    else
      whitespace = ' '
    end
    if not count then
      count = 1
    end
    for _ = 1, count do
      output = output .. whitespace
    end
```

```
  return output
end,

---
-- @treturn string
color_code = function(code)
  return string.char(27) .. '[' .. tostring(code) .. 'm'
end,

---
-- @treturn string
color_tex = function(color, mode)
  if not mode then mode = '' end
  return 'NTE' .. color .. mode
end,

---
-- @treturn string
node_begin = function()
  if options.channel == 'tex' then
    return '\\mbox{'
  else
    return ''
  end
end,

---
-- @treturn string
node_end = function()
  if options.channel == 'tex' then
    return '}'
  else
    return ''
  end
end,

---
-- @treturn string
new_line = function(count)
  local output = ''
  if not count then
    count = 1
  end
  local new_line
  if options.channel == 'tex' then
    new_line = '\\par{}'
  else
    new_line = '\n'
  end

  for _ = 1, count do
    output = output .. new_line
  end
  return output
end,

---
```

```
  -- @treturn string
  type_id = function(id)
    return '[' .. tostring(id) .. ']'
  end
}

--- Print the output to stdout or write it into a file (`output_file`).
-- New text is appended.
--
-- @tparam string text A text string.
--
local function nodetree_print(text)
  if options.channel == 'log' or options.channel == 'tex' then
    output_file:write(text)
  else
    io.write(text)
  end
end

--- Template functions.
-- @section template

local template = {
  node_colors = {
    hlist = {'red', 'bright'},
    vlist = {'green', 'bright'},
    rule = {'blue', 'bright'},
    ins = {'blue'},
    mark = {'magenta'},
    adjust = {'cyan'},
    boundary = {'red', 'bright'},
    disc = {'green', 'bright'},
    whatsit = {'yellow', 'bright'},
    local_par = {'blue', 'bright'},
    dir = {'magenta', 'bright'},
    math = {'cyan', 'bright'},
    glue = {'magenta', 'bright'},
    kern = {'green', 'bright'},
    penalty = {'yellow', 'bright'},
    unset = {'blue'},
    style = {'magenta'},
    choice = {'cyan'},
    noad = {'red'},
    radical = {'green'},
    fraction = {'yellow'},
    accent = {'blue'},
    fence = {'magenta'},
    math_char = {'cyan'},
    sub_box = {'red', 'bright'},
    sub_mlist = {'green', 'bright'},
    math_text_char = {'yellow', 'bright'},
    delim = {'blue', 'bright'},
    margin_kern = {'magenta', 'bright'},
    glyph = {'cyan', 'bright'},
    align_record = {'red'},
    pseudo_file = {'green'},
    pseudo_line = {'yellow'},
```

```
  page_insert = {'blue'},
  split_insert = {'magenta'},
  expr_stack = {'cyan'},
  nested_list = {'red'},
  span = {'green'},
  attribute = {'yellow'},
  glue_spec = {'magenta'},
  attribute_list = {'cyan'},
  temp = {'magenta'},
  align_stack = {'red', 'bright'},
  movement_stack = {'green', 'bright'},
  if_stack = {'yellow', 'bright'},
  unhyphenated = {'magenta', 'bright'},
  hyphenated = {'cyan', 'bright'},
  delta = {'red'},
  passive = {'green'},
  shape = {'yellow'},
},

---
-- [SGR (Select Graphic Rendition)
↪  Parameters](https://en.wikipedia.org/wiki/ANSI_escape_code#SGR_parameters)
--
-- __attributes__
--
-- | color      |code|
-- |------------|----|
-- | reset      |  0 |
-- | clear      |  0 |
-- | bright     |  1 |
-- | dim        |  2 |
-- | underscore |  4 |
-- | blink      |  5 |
-- | reverse    |  7 |
-- | hidden     |  8 |
--
-- __foreground__
--
-- | color      |code|
-- |------------|----|
-- | black      | 30 |
-- | red        | 31 |
-- | green      | 32 |
-- | yellow     | 33 |
-- | blue       | 34 |
-- | magenta    | 35 |
-- | cyan       | 36 |
-- | white      | 37 |
--
-- __background__
--
-- | color      |code|
-- |------------|----|
-- | onblack    | 40 |
-- | onred      | 41 |
-- | ongreen    | 42 |
-- | onyellow   | 43 |
```

```
-- | onblue     | 44 |
-- | onmagenta  | 45 |
-- | oncyan     | 46 |
-- | onwhite    | 47 |
--
-- @tparam string color A color name (`black`, `red`, `green`,
--   `yellow`, `blue`, `magenta`, `cyan`, `white`).
-- @tparam string mode `bright` or `dim`.
-- @tparam boolean background Colorize the background not the text.
--
-- @treturn string
color = function(color, mode, background)
  if options.color ~= 'colored' then
    return ''
  end

  local output = ''
  local code

  if mode == 'bright' then
    output = format.color_code(1)
  elseif mode == 'dim' then
    output = format.color_code(2)
  end

  if not background then
    if color == 'reset' then code = 0
    elseif color == 'red' then code = 31
    elseif color == 'green' then code = 32
    elseif color == 'yellow' then code = 33
    elseif color == 'blue' then code = 34
    elseif color == 'magenta' then code = 35
    elseif color == 'cyan' then code = 36
    else code = 37 end
  else
    if color == 'black' then code = 40
    elseif color == 'red' then code = 41
    elseif color == 'green' then code = 42
    elseif color == 'yellow' then code = 43
    elseif color == 'blue' then code = 44
    elseif color == 'magenta' then code = 45
    elseif color == 'cyan' then code = 46
    elseif color == 'white' then code = 47
    else code = 40 end
  end
  return output .. format.color_code(code)
end,

--- Format a single unicode character.
--
-- @tparam string char A single input character.
--
-- @treturn string
char = function(char)
  char = string.format('%s', unicode.utf8.char(char))
  char = '\'' .. char .. '\''
  if options.channel == 'tex' then
```

```
      char = format.escape(char)
    end
    return char
  end,

  ---
  -- @treturn string
  line = function(length)
    local output
    if length == 'long' then
      output = '----------------------------------------'
    else
      output = '----------------------'
    end
      return output .. format.new_line()
  end,

  ---
  -- @treturn string
  branch = function(connection_type, connection_state, last)
    local c = connection_type
    local s = connection_state
    local l = last
    if c == 'list' and s == 'stop' and l == false then
      return format.whitespace(2)
    elseif c == 'field' and s == 'stop' and l == false then
      return format.whitespace(2)
    elseif c == 'list' and s == 'continue' and l == false then
      return ' ' .. format.whitespace()
    elseif c == 'field' and s == 'continue' and l == false then
      return ' ' .. format.whitespace()
    elseif c == 'list' and s == 'continue' and l == true then
      return ' '
    elseif c == 'field' and s == 'continue' and l == true then
      return ' '
    elseif c == 'list' and s == 'stop' and l == true then
      return ' '
    elseif c == 'field' and s == 'stop' and l == true then
      return ' '
    end
    return ''
  end,
}

---
-- @treturn string
function template.fill(number, order, field)
  local output
  if order ~= nil and order ~= 0 then
    if field == 'stretch' then
      output = '+'
    else
      output = '-'
    end
    return output .. string.format(
      '%g%s', number / 2^16,
      template.colored_string(
```

```
        'fi' .. string.rep('l', order - 1),
        'white',
        'dim'
      )
    )
  else
    return template.length(number)
  end
end

--- Colorize a text string.
--
-- @tparam string text A text string.
-- @tparam string color A color name (`black`, `red`, `green`,
--   `yellow`, `blue`, `magenta`, `cyan`, `white`).
-- @tparam string mode `bright` or `dim`.
-- @tparam boolean background Colorize the background not the text.
--
-- @treturn string
function template.colored_string(text, color, mode, background)
  if options.channel == 'tex' then
    if mode == 'dim' then
      mode = ''
    end
    return '\\textcolor{' ..
      format.color_tex(color, mode) ..
      '}{' ..
      text ..
      '}'
  else
   return template.color(color, mode, background) .. text ..
   ↪  template.color('reset')
  end
end

--- Format a scaled point input value into dimension string (`12pt`,
--  `1cm`)
--
-- @tparam number input
--
-- @treturn string
function template.length (input)
  input = tonumber(input)
  input = input / tex.sp('1' .. options.unit)
  return string.format(
    '%g%s',
    format.number(input),
    template.colored_string(options.unit, 'white', 'dim')
  )
end

--- Convert a Lua table into a format string.
--
-- @tparam table table A table to generate a inline view of.
--
-- @treturn string
function template.table_inline(table)
```

```
  local tex_escape = ''
  if options.channel == 'tex' then
    tex_escape = '\\'
  end
  if type(table) == 'table' then
    local output = tex_escape .. '{'
    local kv_list = ''
    for key, value in pairs(table) do
        if type(key) ~= 'numbers' then
          key = '\'' ..
            template.colored_string(key, 'cyan', 'dim') .. '\''
        end
        kv_list = kv_list .. '[' .. key .. '] = ' ..
          template.table_inline(value) .. ', '
    end
    output = output .. kv_list:gsub(', $', '')
    return output .. tex_escape .. '}'
  else
    return tostring(table)
  end
end

--- Format a key value pair (`key: value, `).
--
-- @tparam string key A key
-- @tparam string|number value A value
-- @tparam string color A color name (`black`, `red`, `green`,
--   `yellow`, `blue`, `magenta`, `cyan`, `white`).
--
-- @treturn string
function template.key_value(key, value, color)
  if type(color) ~= 'string' then
    color = 'yellow'
  end
  if options.channel == 'tex' then
    key = format.underscore(key)
  end
  local output = template.colored_string(key .. ':', color)
  if value then
    output = output .. ' ' .. value .. ', '
  end
  return output
end

---
-- @treturn string
function template.type(type, id)
  local output
  if options.channel == 'tex' then
    output = format.underscore(type)
  else
    output = type
  end
  output = string.upper(output)
  if options.verbosity > 1 then
    output = output .. format.type_id(id)
  end
```

```
  return template.colored_string(
    output .. format.whitespace(),
    template.node_colors[type][1],
    template.node_colors[type][2]
  )
end

---
-- @treturn string
function template.callback(callback_name, variables)
  nodetree_print(
    format.new_line(2) ..
    'Callback: ' ..
    template.colored_string(format.underscore(callback_name), 'red', '', true) ..
    format.new_line()
  )
  if variables then
    for name, value in pairs(variables) do
      if value ~= nil and value ~= '' then
        nodetree_print(
          '- ' ..
          format.underscore(name) ..
          ': ' ..
          tostring(value) ..
          format.new_line()
        )
      end
    end
  end
  nodetree_print(template.line('long'))
end

---
-- @treturn string
function template.branches(level, connection_type)
  local output = ''
  for i = 1, level - 1  do
    output = output .. template.branch('list', tree_state[i]['list'], false)
    output = output .. template.branch('field', tree_state[i]['field'], false)
  end
-- Format the last branches
  if connection_type == 'list' then
    output = output .. template.branch('list', tree_state[level]['list'], true)
  else
    output = output .. template.branch('list', tree_state[level]['list'], false)
    output = output .. template.branch('field', tree_state[level]['field'], true)
  end
  return output
end

--- Extend the node library
-- @section node_extended

local node_extended = {}

--- Get the ID of a node.
--
```

```
-- We have to convert the node into a string and than have to extract
-- the ID from this string using a regular expression. If you convert a
-- node into a string it looks like: `<node    nil <    172 >    nil :
-- hlist 2>`.
--
-- @tparam node n A node.
--
-- @treturn string
function node_extended.node_id(n)
  return string.gsub(tostring(n), '^<node%s+%S+%s+<%s+(%d+).*', '%1')
end

--- A table of all node subtype names.
--
-- __Nodes without subtypes:__
--
-- * `ins` (3)
-- * `mark` (4)
-- * `whatsit` (8)
-- * `local_par` (9)
-- * `dir` (10)
-- * `penalty` (14)
-- * `unset` (15)
-- * `style` (16)
-- * `choice` (17)
-- * `fraction` (20)
-- * `math_char` (23)
-- * `sub_box` (24)
-- * `sub_mlist` (25)
-- * `math_text_char` (26)
-- * `delim` (27)
-- * `margin_kern` (28)
-- * `align_record` (30)
-- * `pseudo_file` (31)
-- * `pseudo_line` (32)
-- * `page_insert` (33)
-- * `split_insert` (34)
-- * `expr_stack` (35)
-- * `nested_list` (36)
-- * `span` (37)
-- * `attribute` (38)
-- * `glue_spec` (39)
-- * `attribute_list` (40)
-- * `temp` (41)
-- * `align_stack` (42)
-- * `movement_stack` (43)
-- * `if_stack` (44)
-- * `unhyphenated` (45)
-- * `hyphenated` (46)
-- * `delta` (47)
-- * `passive` (48)
-- * `shape` (49)
--
-- @treturn table
local function get_node_subtypes ()
    local subtypes = {
    -- hlist (0)
```

```
hlist = {
  [0] = 'unknown',
  [1] = 'line',
  [2] = 'box',
  [3] = 'indent',
  [4] = 'alignment',
  [5] = 'cell',
  [6] = 'equation',
  [7] = 'equationnumber',
  [8] = 'math',
  [9] = 'mathchar',
  [10] = 'hextensible',
  [11] = 'vextensible',
  [12] = 'hdelimiter',
  [13] = 'vdelimiter',
  [14] = 'overdelimiter',
  [15] = 'underdelimiter',
  [16] = 'numerator',
  [17] = 'denominator',
  [18] = 'limits',
  [19] = 'fraction',
  [20] = 'nucleus',
  [21] = 'sup',
  [22] = 'sub',
  [23] = 'degree',
  [24] = 'scripts',
  [25] = 'over',
  [26] = 'under',
  [27] = 'accent',
  [28] = 'radical',
},
-- vlist (1)
vlist = {
  [0] = 'unknown',
  [4] = 'alignment',
  [5] = 'cell',
},
-- rule (2)
rule = {
  [0] = 'normal',
  [1] = 'box',
  [2] = 'image',
  [3] = 'empty',
  [4] = 'user',
  [5] = 'over',
  [6] = 'under',
  [7] = 'fraction',
  [8] = 'radical',
  [9] = 'outline',
},
-- adjust (5)
adjust = {
  [0] = 'normal',
  [1] = 'pre',
},
-- boundary (6)
boundary = {
```

```
  [0] = 'cancel',
  [1] = 'user',
  [2] = 'protrusion',
  [3] = 'word',
},
-- disc (7)
disc  = {
  [0] = 'discretionary',
  [1] = 'explicit',
  [2] = 'automatic',
  [3] = 'regular',
  [4] = 'first',
  [5] = 'second',
},
-- math (11)
math = {
  [0] = 'beginmath',
  [1] = 'endmath',
},
-- glue (12)
glue = {
  [0]   = 'userskip',
  [1]   = 'lineskip',
  [2]   = 'baselineskip',
  [3]   = 'parskip',
  [4]   = 'abovedisplayskip',
  [5]   = 'belowdisplayskip',
  [6]   = 'abovedisplayshortskip',
  [7]   = 'belowdisplayshortskip',
  [8]   = 'leftskip',
  [9]   = 'rightskip',
  [10]  = 'topskip',
  [11]  = 'splittopskip',
  [12]  = 'tabskip',
  [13]  = 'spaceskip',
  [14]  = 'xspaceskip',
  [15]  = 'parfillskip',
  [16]  = 'mathskip',
  [17]  = 'thinmuskip',
  [18]  = 'medmuskip',
  [19]  = 'thickmuskip',
  [98]  = 'conditionalmathskip',
  [99]  = 'muglue',
  [100] = 'leaders',
  [101] = 'cleaders',
  [102] = 'xleaders',
  [103] = 'gleaders',
},
-- kern (13)
kern = {
  [0] = 'fontkern',
  [1] = 'userkern',
  [2] = 'accentkern',
  [3] = 'italiccorrection',
},
-- penalty (14)
penalty = {
```

```
  [0] = 'userpenalty',
  [1] = 'linebreakpenalty',
  [2] = 'linepenalty',
  [3] = 'wordpenalty',
  [4] = 'finalpenalty',
  [5] = 'noadpenalty',
  [6] = 'beforedisplaypenalty',
  [7] = 'afterdisplaypenalty',
  [8] = 'equationnumberpenalty',
},
noad = {
  [0] = 'ord',
  [1] = 'opdisplaylimits',
  [2] = 'oplimits',
  [3] = 'opnolimits',
  [4] = 'bin',
  [5] = 'rel',
  [6] = 'open',
  [7] = 'close',
  [8] = 'punct',
  [9] = 'inner',
  [10] = 'under',
  [11] = 'over',
  [12] = 'vcenter',
},
-- radical (19)
radical = {
  [0] = 'radical',
  [1] = 'uradical',
  [2] = 'uroot',
  [3] = 'uunderdelimiter',
  [4] = 'uoverdelimiter',
  [5] = 'udelimiterunder',
  [6] = 'udelimiterover',
},
-- accent (21)
accent = {
  [0] = 'bothflexible',
  [1] = 'fixedtop',
  [2] = 'fixedbottom',
  [3] = 'fixedboth',
},
-- fence (22)
fence = {
  [0] = 'unset',
  [1] = 'left',
  [2] = 'middle',
  [3] = 'right',
  [4] = 'no',
},
-- margin_kern (28)
margin_kern = {
  [0] = 'left',
  [1] = 'right',
},
-- glyph (29)
glyph = {
```

```
      [0] = 'character',
      [1] = 'ligature',
      [2] = 'ghost',
      [3] = 'left',
      [4] = 'right',
    },
  }
  subtypes.whatsit = node.whatsits()
  return subtypes
end

---
-- @treturn string
function node_extended.subtype(n)
  local typ = node.type(n.id)
  local subtypes = get_node_subtypes()

  local output
  if subtypes[typ] and subtypes[typ][n.subtype] then
    output = subtypes[typ][n.subtype]
    if options.verbosity > 1 then
      output = output .. format.type_id(n.subtype)
    end
    return output
  else
    return tostring(n.subtype)
  end
end

--- Build the node tree.
-- @section tree

local tree = {}

---
-- @tparam node head
-- @tparam string field
--
-- @treturn string
function tree.format_field(head, field)
  local output
-- Character "0" should be printed in a tree, because in TeX fonts the
-- 0 slot usually has a symbol.
  if not head[field] or (head[field] == 0 and field ~= "char") then
    return ''
  end

  if options.verbosity < 2 and
    -- glyph
    field == 'font' or
    field == 'left' or
    field == 'right' or
    field == 'uchyph' or
    -- hlist
    field == 'dir' or
    field == 'glue_order' or
    field == 'glue_sign' or
```

```
    field == 'glue_set' or
    -- glue
    field == 'stretch_order' then
    return ''
  elseif options.verbosity < 3 and
    field == 'prev' or
    field == 'next' or
    field == 'id'
  then
    return ''
  end

  if field == 'prev' or field == 'next' then
    output = node_extended.node_id(head[field])
  elseif field == 'subtype' then
    output = format.underscore(node_extended.subtype(head))
  elseif
    field == 'width' or
    field == 'height' or
    field == 'depth' or
    field == 'kern' or
    field == 'shift' then
    output = template.length(head[field])
  elseif field == 'char' then
    output = template.char(head[field])
  elseif field == 'glue_set' then
    output = format.number(head[field])
  elseif field == 'stretch' or field == 'shrink' then
    output = template.fill(head[field], head[field .. '_order'], field)
  else
    output = tostring(head[field])
  end

  return template.key_value(field, output)
end

---
-- Attributes are key/value number pairs. They are printed as an inline
-- list. The attribute `0` with the value `0` is skipped because this
-- attribute is in every node by default.
--
-- @tparam node head
--
-- @treturn string
function tree.format_attributes(head)
  if not head then
    return ''
  end
  local output = ''
  local attr = head.next
  while attr do
    if attr.number ~= 0 or (attr.number == 0 and attr.value ~= 0) then
      output = output .. tostring(attr.number) .. '=' .. tostring(attr.value) ..
      ↪  ' '
    end
    attr = attr.next
  end
```

```
  return output
end

---
-- @tparam number level `level` is a integer beginning with 1.
-- @tparam number connection_type The variable `connection_type`
--   is a string, which can be either `list` or `field`.
-- @tparam connection_state `connection_state` is a string, which can
--   be either `continue` or `stop`.
function tree.set_state(level, connection_type, connection_state)
  if not tree_state[level] then
    tree_state[level] = {}
  end
  tree_state[level][connection_type] = connection_state
end

---
-- @tparam table fields
-- @tparam number level
function tree.analyze_fields(fields, level)
  local max = 0
  local connection_state
  for _ in pairs(fields) do
    max = max + 1
  end
  local count = 0
  for field_name, recursion_node in pairs(fields) do
    count = count + 1
    if count == max then
      connection_state = 'stop'
    else
      connection_state = 'continue'
    end
    tree.set_state(level, 'field', connection_state)
    nodetree_print(
      format.node_begin() ..
      template.branches(level, 'field') ..
      template.key_value(field_name) ..
      format.node_end() ..
      format.new_line()
    )
    tree.analyze_list(recursion_node, level + 1)
  end
end

---
-- @tparam node head
-- @tparam number level
function tree.analyze_node(head, level)
  local connection_state
  local output
  if head.next then
    connection_state = 'continue'
  else
    connection_state = 'stop'
  end
  tree.set_state(level, 'list', connection_state)
```

```
  output = template.branches(level, 'list')
    .. template.type(node.type(head.id), head.id)
  if options.verbosity > 1 then
    output = output .. template.key_value('no', node_extended.node_id(head))
  end

  -- We store the attributes output to append it to the field list.
  local attributes

  -- We store fields which are nodes for later treatment.
  local fields = {}

  -- Inline fields, for example: char: 'm', width: 25pt, height: 13.33pt,
  local output_fields = ''
  for _, field_name in pairs(node.fields(head.id, head.subtype)) do
    if field_name == 'attr' then
      attributes = tree.format_attributes(head.attr)
    elseif field_name ~= 'next' and field_name ~= 'prev' and
      node.is_node(head[field_name]) then
      fields[field_name] = head[field_name]
    else
      output_fields = output_fields .. tree.format_field(head, field_name)
    end
  end
  if output_fields ~= '' then
    output = output .. output_fields
  end

  -- Append the attributes output if available
  if attributes ~= '' then
    output = output .. template.key_value('attr', attributes, 'blue')
  end

  output = output:gsub(', $', '')

  nodetree_print(
    format.node_begin() ..
    output ..
    format.node_end() ..
    format.new_line()
  )

  local property = node.getproperty(head)
  if property then
    nodetree_print(
      format.node_begin() ..
      template.branches(level, 'field') ..
      '  ' ..
      template.colored_string('properties:', 'blue') .. ' ' ..
      template.table_inline(property) ..
      format.node_end() ..
      format.new_line()
    )
  end

  tree.analyze_fields(fields, level)
end
```

```
---
-- @tparam node head
-- @tparam number level
function tree.analyze_list(head, level)
  while head do
    tree.analyze_node(head, level)
    head = head.next
  end
end

---
-- @tparam node head
function tree.analyze_callback(head)
  tree.analyze_list(head, 1)
  nodetree_print(template.line('short') .. format.new_line())
end

--- Callback wrapper.
-- @section callbacks

local callbacks = {

  ---
  -- @tparam string extrainfo
  contribute_filter = function(extrainfo)
    template.callback('contribute_filter', {extrainfo = extrainfo})
    return true
  end,

  ---
  -- @tparam string extrainfo
  buildpage_filter = function(extrainfo)
    template.callback('buildpage_filter', {extrainfo = extrainfo})
    return true
  end,

  ---
  -- @tparam string n
  -- @tparam string i
  build_page_insert = function(n, i)
    print('lol')
    template.callback('build_page_insert', {n = n, i = i})
    return 0
  end,

  ---
  -- @tparam node head
  -- @tparam string groupcode
  pre_linebreak_filter = function(head, groupcode)
    template.callback('pre_linebreak_filter', {groupcode = groupcode})
    tree.analyze_callback(head)
    return true
  end,

  ---
  -- @tparam node head
```

```
-- @tparam boolean is_display
linebreak_filter = function(head, is_display)
  template.callback('linebreak_filter', {is_display = is_display})
  tree.analyze_callback(head)
  return true
end,

---
-- @tparam node box
-- @tparam string locationcode
-- @tparam number prevdepth
-- @tparam boolean mirrored
append_to_vlist_filter = function(box, locationcode, prevdepth, mirrored)
  local variables = {
    locationcode = locationcode,
    prevdepth = prevdepth,
    mirrored = mirrored,
  }
  template.callback('append_to_vlist_filter', variables)
  tree.analyze_callback(box)
  return box
end,

---
-- @tparam node head
-- @tparam string groupcode
post_linebreak_filter = function(head, groupcode)
  template.callback('post_linebreak_filter', {groupcode = groupcode})
  tree.analyze_callback(head)
  return true
end,

---
-- @tparam node head
-- @tparam string groupcode
-- @tparam number size
-- @tparam string packtype
-- @tparam string direction
-- @tparam node attributelist
hpack_filter = function(head, groupcode, size, packtype, direction,
↪  attributelist)
  local variables = {
    groupcode = groupcode,
    size = size,
    packtype = packtype,
    direction = direction,
    attributelist = attributelist,
  }
  template.callback('hpack_filter', variables)
  tree.analyze_callback(head)
  return true
end,

---
-- @tparam node head
-- @tparam string groupcode
-- @tparam number size
```

```
-- @tparam string packtype
-- @tparam number maxdepth
-- @tparam string direction
-- @tparam node attributelist
vpack_filter = function(head, groupcode, size, packtype, maxdepth, direction,
↪  attributelist)
  local variables = {
    groupcode = groupcode,
    size = size,
    packtype = packtype,
    maxdepth = template.length(maxdepth),
    direction = direction,
    attributelist = attributelist,
  }
  template.callback('vpack_filter', variables)
  tree.analyze_callback(head)
  return true
end,

---
-- @tparam string incident
-- @tparam number detail
-- @tparam node head
-- @tparam number first
-- @tparam number last
hpack_quality = function(incident, detail, head, first, last)
  local variables = {
    incident = incident,
    detail = detail,
    first = first,
    last = last,
  }
  template.callback('hpack_quality', variables)
  tree.analyze_callback(head)
end,

---
-- @tparam string incident
-- @tparam number detail
-- @tparam node head
-- @tparam number first
-- @tparam number last
vpack_quality = function(incident, detail, head, first, last)
  local variables = {
    incident = incident,
    detail = detail,
    first = first,
    last = last,
  }
  template.callback('vpack_quality', variables)
  tree.analyze_callback(head)
end,

---
-- @tparam node head
-- @tparam number width
-- @tparam number height
```

```
process_rule = function(head, width, height)
  local variables = {
    width = width,
    height = height,
  }
  template.callback('process_rule', variables)
  tree.analyze_callback(head)
  return true
end,

---
-- @tparam node head
-- @tparam string groupcode
-- @tparam number size
-- @tparam string packtype
-- @tparam number maxdepth
-- @tparam string direction
pre_output_filter = function(head, groupcode, size, packtype, maxdepth,
↪  direction)
  local variables = {
    groupcode = groupcode,
    size = size,
    packtype = packtype,
    maxdepth = maxdepth,
    direction = direction,
  }
  template.callback('pre_output_filter', variables)
  tree.analyze_callback(head)
  return true
end,

---
-- @tparam node head
-- @tparam node tail
hyphenate = function(head, tail)
  template.callback('hyphenate')
  nodetree_print('head:')
  tree.analyze_callback(head)
  nodetree_print('tail:')
  tree.analyze_callback(tail)
end,

---
-- @tparam node head
-- @tparam node tail
ligaturing = function(head, tail)
  template.callback('ligaturing')
  nodetree_print('head:')
  tree.analyze_callback(head)
  nodetree_print('tail:')
  tree.analyze_callback(tail)
end,

---
-- @tparam node head
-- @tparam node tail
kerning = function(head, tail)
```

```
    template.callback('kerning')
    nodetree_print('head:')
    tree.analyze_callback(head)
    nodetree_print('tail:')
    tree.analyze_callback(tail)
  end,

  ---
  -- @tparam node local_par
  -- @tparam string location
  insert_local_par = function(local_par, location)
    template.callback('insert_local_par', {location = location})
    tree.analyze_callback(local_par)
    return true
  end,

  ---
  -- @tparam node head
  -- @tparam string display_type
  -- @tparam boolean need_penalties
  mlist_to_hlist = function(head, display_type, need_penalties)
    local variables = {
      display_type = display_type,
      need_penalties = need_penalties,
    }
    template.callback('mlist_to_hlist', variables)
    tree.analyze_callback(head)
    return node.mlist_to_hlist(head, display_type, need_penalties)
  end,
}

--- Set a single option key value pair.
--
-- @tparam string key The key of the option pair.
-- @tparam number|string value The value of the option pair.
local function set_option(key, value)
  if not options then
    options = {}
  end
  if key == 'verbosity' or key == 'decimalplaces' then
    options[key] = tonumber(value)
  else
    options[key] = value
  end
end

--- Set multiple key value pairs using a table.
--
-- @tparam table opts Options
local function set_options(opts)
  if not options then
    options = {}
  end
  for key, value in pairs(opts) do
    set_option(key, value)
  end
end
```

```
--- Check if the given callback name exists.
--
-- Throw an error if it doen't.
--
-- @tparam string callback_name The name of a callback to check.
--
-- @treturn string The unchanged input of the function.
local function check_callback_name(callback_name)
  local info = callback.list()
  if info[callback_name] == nil then
    tex.error(
      'Package "nodetree": Unkown callback name or callback alias: "' ..
      callback_name ..
      '"'
    )
  end
  return callback_name
end

--- Get the real callback name from an alias string.
--
-- @tparam string alias The alias of a callback name or the callback
-- name itself.
--
-- @treturn string The real callback name.
local function get_callback_name(alias)
  local callback_name
  -- Listed as in the LuaTeX reference manual.
  if alias == 'contribute' or alias == 'contributefilter' then
    callback_name = 'contribute_filter'

  -- Formerly called buildpage, now there is a build_page_insert.
  elseif alias == 'buildfilter' or alias == 'buildpagefilter' then
    callback_name = 'buildpage_filter'

  -- Untested: I don't know how to invoke this filter.
  elseif alias == 'buildinsert' or alias == 'buildpageinsert' then
    callback_name = 'build_page_insert'

  elseif alias == 'preline' or alias == 'prelinebreakfilter' then
    callback_name = 'pre_linebreak_filter'

  elseif alias == 'line' or alias == 'linebreakfilter' then
    callback_name = 'linebreak_filter'

  elseif alias == 'append' or alias == 'appendtovlistfilter' then
    callback_name = 'append_to_vlist_filter'

  -- postlinebreak is not documented.
  elseif alias == 'postline' or alias == 'postlinebreak' or alias ==
  ↪  'postlinebreakfilter' then
    callback_name = 'post_linebreak_filter'

  elseif alias == 'hpack' or alias == 'hpackfilter' then
    callback_name = 'hpack_filter'
```

```
  elseif alias == 'vpack' or alias == 'vpackfilter' then
    callback_name = 'vpack_filter'

  elseif alias == 'hpackq' or alias == 'hpackquality' then
    callback_name = 'hpack_quality'

  elseif alias == 'vpackq' or alias == 'vpackquality' then
    callback_name = 'vpack_quality'

  elseif alias == 'process' or alias == 'processrule' then
    callback_name = 'process_rule'

  elseif alias == 'preout' or alias == 'preoutputfilter' then
    callback_name = 'pre_output_filter'

  elseif alias == 'hyph' or alias == 'hyphenate' then
    callback_name = 'hyphenate'

  elseif alias == 'liga' or alias == 'ligaturing' then
    callback_name = 'ligaturing'

  elseif alias == 'kern' or alias == 'kerning' then
    callback_name = 'kerning'

  elseif alias == 'insert' or alias == 'insertlocalpar' then
    callback_name = 'insert_local_par'

  elseif alias == 'mhlist' or alias == 'mlisttohlist' then
    callback_name = 'mlist_to_hlist'

  else
    callback_name = alias
  end
  return check_callback_name(callback_name)
end

--- Register a callback.
--
-- @tparam string cb The name of a callback.
local function register_callback(cb)
  if options.engine == 'lualatex' then
    luatexbase.add_to_callback(cb, callbacks[cb], 'nodetree')
  else
    callback.register(cb, callbacks[cb])
  end
end

--- Unregister a callback.
--
-- @tparam string cb The name of a callback.
local function unregister_callback(cb)
  if options.engine == 'lualatex' then
    luatexbase.remove_from_callback(cb, 'nodetree')
  else
    register_callback(cb, nil)
  end
end
```

```lua
--- Exported functions.
-- @section export

local export = {
  set_option = set_option,
  set_options = set_options,

  ---
  register_callbacks = function()
    if options.channel == 'log' or options.channel == 'tex' then
      -- nt = nodetree
      -- jobname.nttex
      -- jobname.ntlog
      local file_name = tex.jobname .. '.nt' .. options.channel
      io.open(file_name, 'w'):close() -- Clear former content
      output_file = io.open(file_name, 'a')
    end
    for alias in string.gmatch(options.callback, '([^,]+)') do
      register_callback(get_callback_name(alias))
    end
  end,

  ---
  unregister_callbacks = function()
    for alias in string.gmatch(options.callback, '([^,]+)') do
      unregister_callback(get_callback_name(alias))
    end
  end,

  --- Compile a TeX snippet.
  --
  -- Write some TeX snippets into a temporary LaTeX file, compile this
  -- file using `latexmk` and read the generated `*.nttex` file and
  -- return its content.
  --
  -- @tparam string tex_markup
  --
  -- @treturn string
  compile_include = function(tex_markup)
    -- Generate a subfolder for all tempory files: _nodetree-jobname.
    local parent_path = lfs.currentdir() .. '/' .. '_nodetree-' .. tex.jobname
    lfs.mkdir(parent_path)

    -- Generate the temporary LuaTeX or LuaLaTeX file.
    example_counter = example_counter + 1
    local filename_tex = example_counter .. '.tex'
    local absolute_path_tex = parent_path .. '/' .. filename_tex
    output_file = io.open(absolute_path_tex, 'w')

    local format_option = function (key, value)
      return '\\NodetreeSetOption[' .. key .. ']{' .. value .. '}' .. '\n'
    end

    -- Process the options
    local options =
      format_option('channel', 'tex') ..
```

```
    format_option('verbosity', options.verbosity) ..
    format_option('unit', options.unit) ..
    format_option('decimalplaces', options.decimalplaces) ..
    '\\NodetreeUnregisterCallback{post_linebreak_filter}' .. '\n' ..
    '\\NodetreeRegisterCallback{' .. options.callback .. '}'

  local prefix = '%!TEX program = lualatex\n' ..
                 '\\documentclass{article}\n' ..
                 '\\usepackage{nodetree}\n' ..
                 options .. '\n' ..
                 '\\begin{document}\n'
  local suffix = '\n\\end{document}'
  output_file:write(prefix .. tex_markup .. suffix)
  output_file:close()

  -- Compile the temporary LuaTeX or LuaLaTeX file.
  os.spawn({ 'latexmk', '-cd', '-pdflua', absolute_path_tex })
  local include_file = assert(io.open(parent_path .. '/' .. example_counter ..
  ↪  '.nttex', 'rb'))
  local include_content = include_file:read("*all")
  include_file:close()
  include_content = include_content:gsub('[\r\n]', '')
  tex.print(include_content)
end,

--- Check for `--shell-escape`
--
check_shell_escape = function()
  local info = status.list()
  if info.shell_escape == 0 then

    ↪  tex.error('Package "nodetree-embed": You have to use the --shell-escape option')
  end
end,

--- Print a node tree.
---
-- @tparam node head The head node of a node list.
-- @tparam table opts Options as a table.
print = function(head, opts)
  if opts and type(opts) == 'table' then
    set_options(opts)
  end
  nodetree_print(format.new_line())
  tree.analyze_list(head, 1)
end,

--- Format a scaled point value into a formated string.
--
-- @tparam number sp A scaled point value
--
-- @treturn string
format_dim = function(sp)
  return template.length(sp)
end,

--- Get a default option that is not changed.
```

```
  -- @tparam string key The key of the option.
  --
  -- @treturn string|number|boolean
  get_default_option = function(key)
    return default_options[key]
  end
}

--- Use export.print
-- @tparam node head
export.analyze = export.print

return export
```

Change History

v0.1

General: Converted to DTX file . 36

v1.0

General: Inital release 36

v1.1

General: Fix the registration of same callbacks 36

v1.2

General: Fix difference between README.md in the upload and that from nodetree.dtx . . 36

v2.0

General: * Switch from lowercase macro names to PascalCase names for better readability. * The Lua code is no longer developed inside the DTX file, instead in a separate file named nodetree.lua. * Add a sub package named nodetree-embed.sty for embedding nodetree views into a LaTeX document. * Add support for new node subtype names. * Add support for a new LuaTeX node callback. * Add support for node properties. * Less verbose representation of node attributes. * Minor tree output adjustments. 36

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.