目录

1 说明 2

2 结点的utf、序列化、id、计数 3

3 遍历结点 5
3.1 node.traverse()遍历结点 5
3.2 node.traverse_id()遍历特定类型/id的结点 5
3.3 用n.netx遍历结点 5

4 打印结点详细信息 6

5 打印结点字体信息 9

6 在回调中遍历结点 10

1 说明

本项目的目的在于探索LuaTeX的基本数据结构——结点（node）。实际的运行环境是LuaMetaTex(LuaTeX的后继者，简称LMTX)和ConTeXt。

- 安装ConTeXt LMTX
- 运行`context luametatex-node-playground.lmtx`，编译
- 如果控制台输出汉字乱码，可用命令`chcp 65001`临时改变代码页
- 观察控制台和pdf输出
- play `luametatex-node-playground.lmtx` 并编译
- 观察并 play again, and on and on

2 结点的utf、序列化、id、计数

```
n=\hbox{abc\hbox{d}}
```

-- nodes.toutf()把当前结点后(包括嵌套中)所有字模结点/字符结点转换成utf:
- nodes.toutf(n):
abc
- nodes.toutf(n.head):
abc
- nodes.toutf(n.head.next):
bc
- nodes.toutf(n.head.next.next):
c

-- nodes.listtoutf()把当前结点后列表本级（不含嵌套中）结点转换成utf:
- nodes.listtoutf(n):
[-]
- nodes.listtoutf(n.head):
ab[-]
- nodes.listtoutf(n.head.next):
b[-]
- nodes.listtoutf(n.head.next.next):
[-]

-- nodes.tosequence()把当前结点后列表本级（不含嵌套中）结点序列化:
- nodes.tosequence(n):
hlist
- nodes.tosequence(n.head):
U+0061:a U+0062:b hlist
- nodes.tosequence(n.head.next):
U+0062:b hlist
- nodes.tosequence(n.head.next.next):
hlist

-- nodes.idstostring()本结点以后结点id转文本（同id合并）:
[hlist]
[2*glyph] [hlist]

-- 本节点以后结点计数(包括嵌套中):
5
4
-- 结点id与类型名称互转:
hlist
0

3 遍历结点

3.1 node.traverse()遍历结点

控制台输出:

```
-----node.traverse()-----
ab[-]
<node :    nil <=   7598 =>   7586 : glyph unset>
b[-]
<node :   7598 <=   7586 =>   4110 : glyph unset>
[-]
<node :   7586 <=   4110 =>    nil : hlist box>
```

3.2 node.traverse_id()遍历特定类型/id的结点

控制台输出:

```
-----node.traverse_id()-----
ab[-]
<node :    nil <=   8688 =>   8676 : glyph unset>
b[-]
<node :   8688 <=   8676 =>   5703 : glyph unset>
```

3.3 用n.netx遍历结点

U+0061:a U+0062:b hlist
U+0062:b hlist
hlist
这个方法的优点在于，可以在遍历过程中增删结点，如果在`node.traverse_id()`和`node.traverse_id()`过程中增删则会导致引用混乱。

4 打印结点详细信息

```
----nodes.tosequence(n)----
U+0061:a U+0062:b hlist
----k, v, n[v]----
1       id      28
2       subtype 32768
3       attr    <node :    nil <=   7042 =>   6905 : attribute
list>
4       char    97
5       font    1
6       language        56
7       lhmin   2
8       rhmin   2
9       uchyph  1
10      state   0
11      left    0
12      right   0
13      xoffset 0
14      yoffset 0
15      xscale  1000
16      yscale  1000
17      width   319116
18      height  313614
19      depth   12576
20      total   326190
21      expansion       0
22      data    0
23      script  1
24      hyphenate       499519
25      options 128
0       next    <node :   2543 <=   2335 =>   7538 : glyph unset>
```

```
-1      prev    nil
----nodes.tosequence(n)----
U+0062:b hlist
----k, v, n[v]----
1       id      28
2       subtype 32768
3       attr    <node :    nil <=   7042 =>   6905 : attribute
list>
4       char    98
5       font    1
6       language        56
7       lhmin   2
8       rhmin   2
9       uchyph  1
10      state   0
11      left    0
12      right   0
13      xoffset 0
14      yoffset 0
15      xscale  1000
16      yscale  1000
17      width   400860
18      height  518760
19      depth   16506
20      total   535266
21      expansion       0
22      data    0
23      script  1
24      hyphenate       499519
25      options 128
0       next    <node :   2335 <=   7538 =>    nil : hlist box>
-1      prev    <node :    nil <=   2543 =>   2335 : glyph unset>
----nodes.tosequence(n)----
```

```
hlist
----k, v, n[v]----
1       id      0
2       subtype 2
3       attr    <node :    nil <=   7042 =>   6905 : attribute
list>
4       width   327762
5       depth   12576
6       height  313614
7       direction       0
8       shift   0
9       glueorder       0
10      gluesign        0
11      glueset 0.0
12      list    <node :    nil <=   2915 =>    nil : glyph unset>
13      orientation     0
14      source  0
15      target  0
16      woffset 0
17      hoffset 0
18      doffset 0
19      xoffset 0
20      yoffset 0
21      state   1
22      class   nil
0       next    nil
-1      prev    <node :   2543 <=   2335 =>   7538 : glyph unset>
```

5 打印结点字体信息

控制台输出:

```
U+0061:a U+0062:b hlist
c:/windows/fonts/stsong.ttf
::::::n.char font desc:::::
depth   16
vheight 1301
unicode 97
height  399
width   406
tsb     587
boundingbox     table: 000005b8777200c0
index   68
:::::BoundingBox:::::
1       43
2       -16
3       412
4       399
```

6 在回调中遍历结点

脚本：

```
a\hbox{b\hbox{c}}\hbox{d\hbox{e}}f

A\hbox{B\hbox{C}}\hbox{D\hbox{E}}F
```

排版：
abcdef
ABCDEF
回调输出
从这里可以看出"processors"回调
- 每个结点列表回调一次（每个盒子包含一个结点列表）；
- 从嵌套的最底层开始；
- 从第一个嵌套开始；
- 系统自动注入了段落结点par和首尾胶结点glue：
U+0063:c
U+0062:b hlist
U+0065:e
U+0064:d hlist
par glue U+0061:a hlist hlist U+0066:f glue
U+0043:C
U+0042:B hlist
U+0045:E
U+0044:D hlist
par glue U+0041:A hlist hlist U+0046:F glue