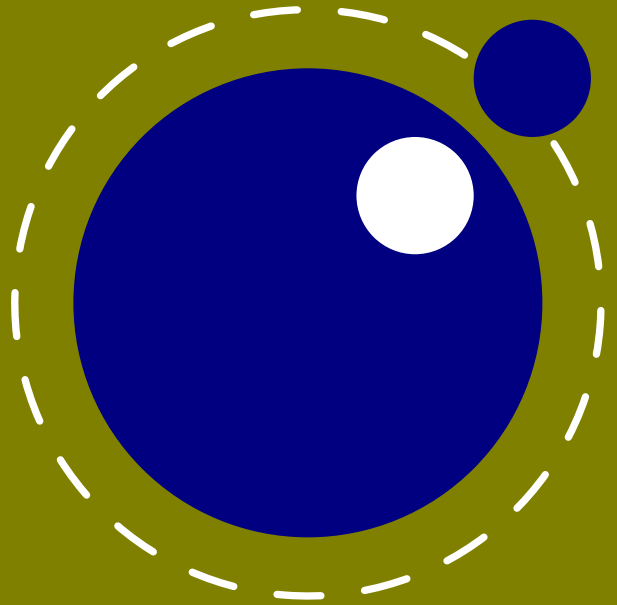


루아텍 노드와 콜백

LuaTEX Nodes and Callbacks

남수진

2020년 2월 15일 토요일

2020 한국텍학회 학술대회 및 정기총회
고려대학교

오늘 이야기할 내용

1. 루아텍 LuaTeX
2. 노드 Nodes
3. 콜백 Callbacks
4. 노드와 콜백 사용 예제
5. 맺음말

루아텍 다시 보기

2016년 학술대회, LuaTEX 활용, 프로그래밍 언어와 조판 시스템의 콜라보

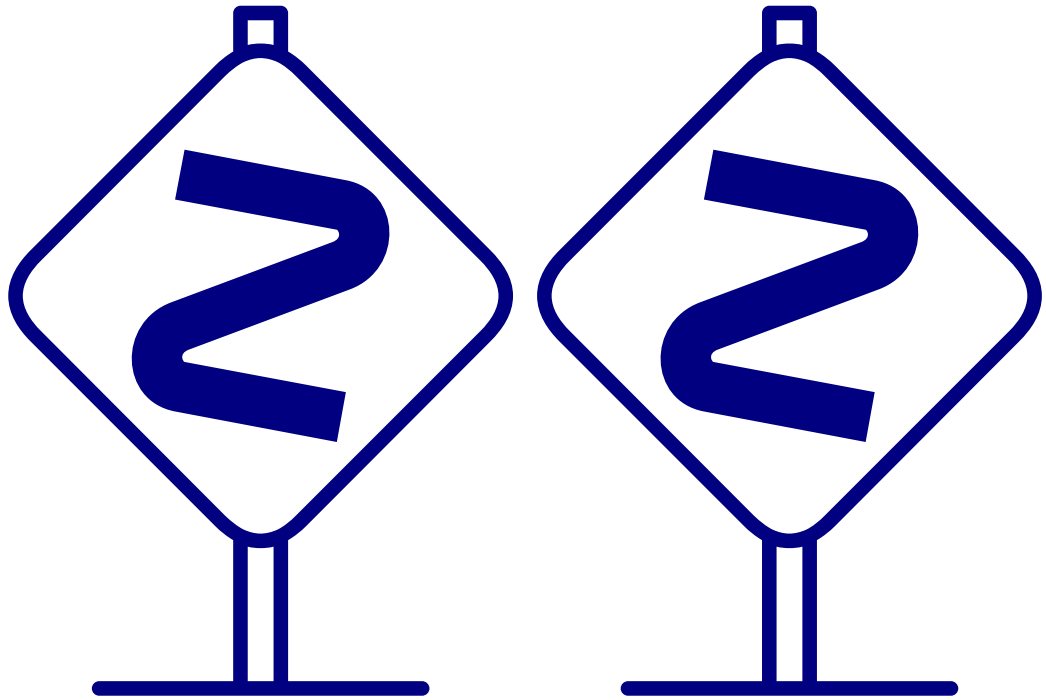
루아텍을 사용하는 이유가 단지 텍에서 루아 스크립트를 사용하기 위해서라면, 알아야 할 것은 다음 한 줄로 충분하다.

```
\directlua{tex.print("안녕하세요")}
```

루아텍은 그동안 감춰져있던 텍의 내부를 속속들이 공개하여 루아 스크립트로 그들을 다룰 수 있게 하였다.

텍의 내부와 동작 원리[1]를 이해한다면 루아텍을 통하여 텍의 진수를 맛볼 수 있다.

[1] The TEXbook과 TEX by Topic을 읽어 볼 것을 권한다.



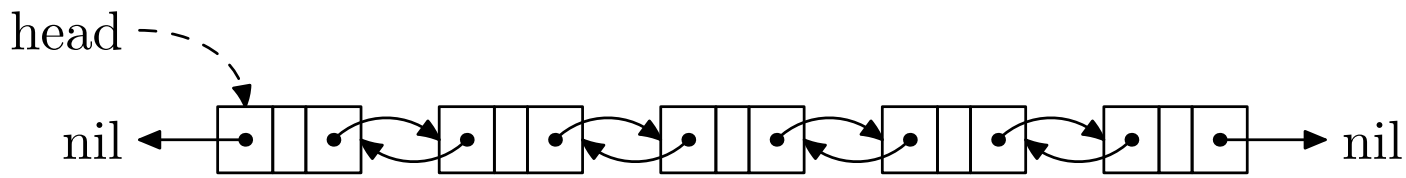
노드 Nodes

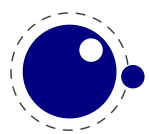
노드: 프로그래밍 관점에서

노드는 여러 속성들을 가지고 있는 객체이다.

노드는 next, prev라고 불리우는 두 개의 포인터로 서로 연결되어 노드 리스트(doubly linked list)를 이룬다.

포인터 head는 노드 리스트의 첫번째 노드를 가리킨다.



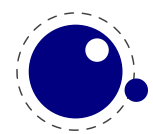


노드: 루아텍 관점에서

페이지를 구성하는 원자(atom), 텍은 노드들을 조합하여 하나의 페이지를 만든다.

하나의 페이지는 여러 개의 문단으로 구성되어있고, 하나의 문단(vbox)은 라인(hbox)들과 penalty, glue 노드들로 이루어져있고, 각 라인(hbox)은 대부분 glyph 와 glue 노드로 구성된다.

```
\hsize 90pt
A long time ago in a galaxy
far, far away $\ldots$.
\par
```

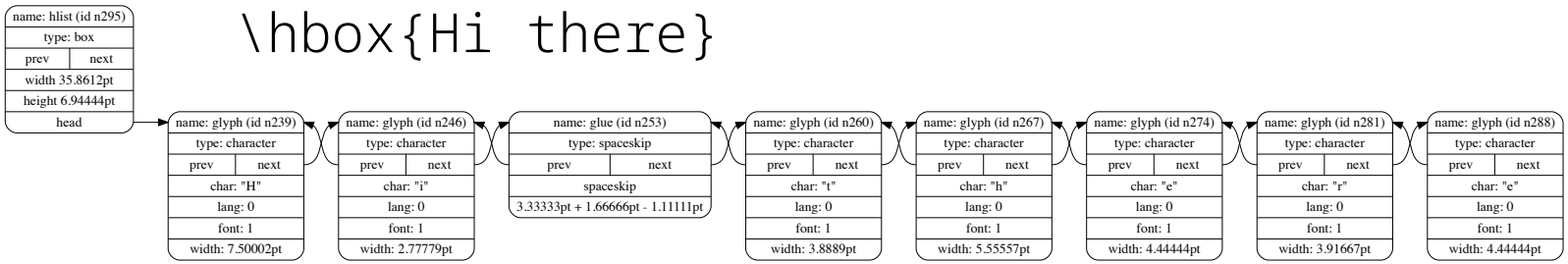


노드의 종류

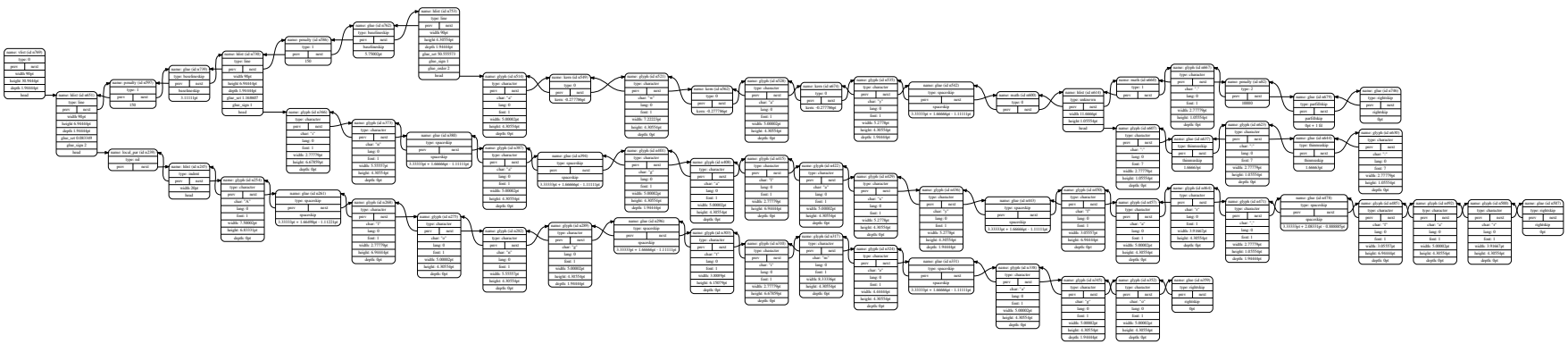
ID	NAME	ID	NAME	ID	NAME	ID	NAME
0	hlist	1	vlist	2	rule	3	ins
4	mark	5	adjust	6	boundary	7	disc
8	whatsit	9	local_par	10	dir	11	math
12	glue	13	kern	14	penalty	15	unset
16	style	17	choice	18	noad	19	radical
20	fraction	21	accent	22	fence	23	math_char
24	sub_box	25	sub_mlist	26	math_text_char	27	delim
28	margin_kern	29	glyph	30	align_record	31	pseudo_file
32	pseudo_line	33	page_insert	34	split_insert	35	expr_stack
36	nested_list	37	span	38	attribute	39	glue_spec
40	attribute_list	41	temp	42	align_stack	43	movement_stack
44	if_stack	45	unhyphenated	46	hyphenated	47	delta
48	passive	49	shape				

node.types(), This is LuaTeX, Version 1.11.2 (TeX Live 2020/dev)

노드 리스트




```
\hsize 90pt
A long time ago in a galaxy
far, far away $\ldots$.
\par
```

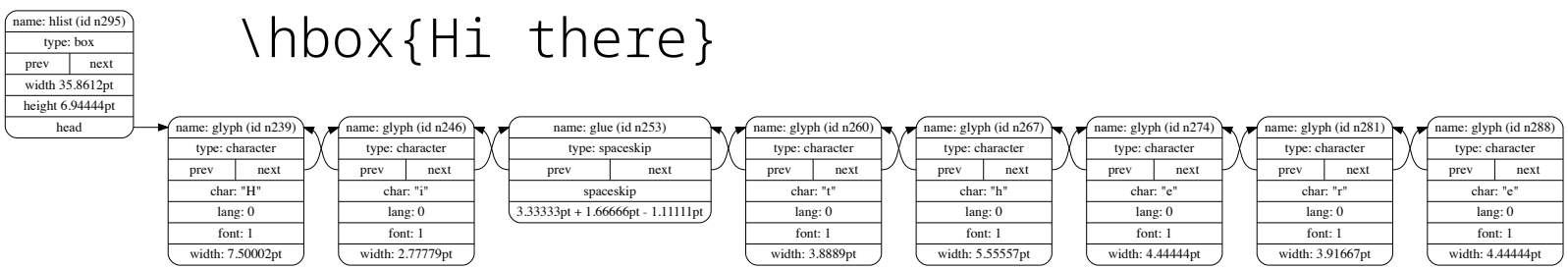


노드 리스트 순회 방법

노드 리스트 안의 노드들을 방문하면서 여러가지 일을 할 수 있다.

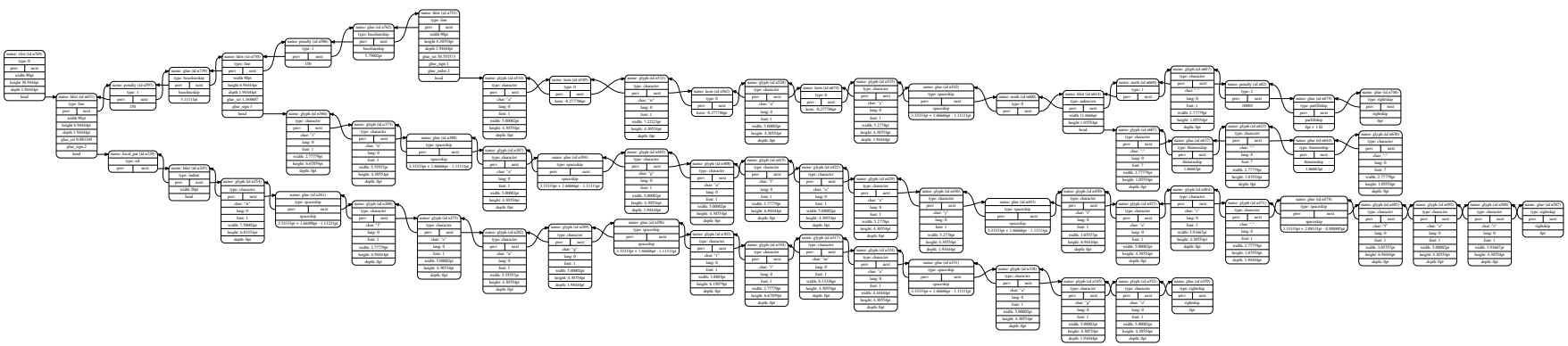
포인터 head 로 노드 리스트의 첫번째 노드를 알 수 있고, 그 노드의 next 포인터를 따라가면 다음 노드로 이동하고, 또 그 노드의 next 포인터를 따라가고 . . ..

노드_포인터 := 리스트의 시작 노드;
while 노드_포인터가 *nil* 이 아니다 **do**
    노드_포인터가 가리키는 노드를 방문한다;
    노드_포인터를 현재 노드의 next 포인터로 변경한다;
end



노드 리스트 순회 방법

노드_포인터 := 리스트의 시작 노드;
while 노드_포인터가 *nil* 이 아니다 **do**
    if 노드_포인터가 가리키는 노드가 *hlist/vlist* 노드 이다 **then**
        그 노드의 리스트를 순회 한다;
    end
    노드_포인터가 가리키는 노드를 방문한다;
    노드_포인터를 현재 노드의 next 포인터로 변경한다;
end



노드 리스트 순회 방법

```
visit_node = function (head)
   local curr = head
   while curr ~= nil do
      if curr.id == 0 or curr.id == 1 then -- hlist/vlist
         visit_node(curr.head)
      elseif curr.id == 29 then  -- glyph
         do_something_with_glyph(curr)
      elseif curr.id == 12 then  -- glue
         do_something_with_glue(curr)
      end
      curr = curr.next
   end
   return head
end
```


노드 리스트 순회 방법

```
visit_node = function (head)
   local curr = head
   while curr ~= nil do
      if curr.id == node.id("hlist")
         or curr.id == node.id("vlist") then
         visit_node(curr.head)
      elseif curr.id == node.id("glyph") then
         do_something_with_glyph(curr)
      elseif curr.id == node.id("glue") then
         do_something_with_glue(curr)
      end
      curr = curr.next
   end
   return head
end
```


노드 리스트 순회 방법

```
visit_glyph_node = function (head)
   for curr in node.traverse_id(node.id("glyph"), head) do
      do_something_with_glyph(curr)
   end
   return head
end

visit_hlist_node = function (head)
   for curr in node.traverse_id(node.id("hlist"), head) do
      do_something_with_hlist(curr)
   end
   return head
end
```


노드 라이브러리

노드 또는 노드 리스트를 쉽게 다룰 수 있도록 여러가지 유틸리티를 제공

```
node.id(), node.new(), node.remove(),
node.copy(), node.free(), node.flush_node(),
node.traverse(), node.traverse_id(),
node.traverse_char(), node.traverse_glyph(),
node.traverse_list(), node.slide(),
node.tail(),
node.insert_before(), node.insert_after(),
node.hpack(), node.vpack(),
node.ligatureing(), node.kerning(),
node.lastnode(),
...
```


콜백: 프로그래밍 관점에서

콜백은 함수인데 주로 다른 함수의 인수로 사용된다. 콜백을 넘겨받는 함수는 특정 이벤트가 발생하거나 어떤 조건을 만족하면 콜백을 호출하여 실행한다.

A nice way of imagining how a callback function works is that it is a function that is called at the back of the function it is passed into.

훅(Hooks): 프로그램의 실행 흐름에서 특정 단계의 전, 후에 추가되거나 그 단계를 대체하여 실행하는 코드(함수)에 관한 프로그래밍 기법이다.

훅에서 실행되는 코드나 함수를 핸들러(handler)라고 하고, 이때 사용되는 개념이 콜백이다.

핸들러를 특정 단계 전/후에 추가하거나 대체하는 과정을 "핸들러를 등록(register)한다" 라고 한다.

콜백: 루아텍 관점에서

루아텍의 콜백은 앞서 설명한 훅의 핸들러에 해당한다.

텍 처리 과정인 입력, 확장, 실행, 조판의 단계에서 원하는 곳에 콜백을 등록하여 텍 동작을 수정하거나 갈아 치울 수 있다.

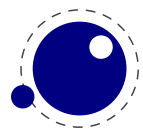
콜백은 텍이 입력 버퍼를 처리하는 데서 부터 문단을 만들거나 오프타입 폰트를 로딩하는 부분까지 다양하다.

콜백 등록과 해제

```
my_cb = function (head)
   do_something_with(head)
end
callback.register("post_linebreak_filter", my_cb)
callback.register("post_linebreak_filter", nil)
```

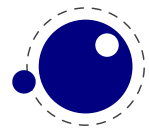

Centred last lines

This is a handbook about TEX, a new typesetting system intended for the creation of beautiful books—and especially for books that contain a lot of mathematics. By preparing a manuscript in TEX format, you will be telling a computer exactly how the manuscript is to be transformed into pages whose typographic quality is comparable to that of the world's finest printers; yet you won't need to do much more work than would be involved if you were simply typing the manuscript on an ordinary typewriter. In fact, your total work will probably be significantly less, if you consider the time it ordinarily takes to revise a typewritten manuscript, since computer text files are so easy to change and to reprocess. This manual is intended for people who have never used TEX before, as well as for experienced TEX hackers. In other words, it's supposed to be a panacea that satisfies everybody, at the risk of satisfying nobody. Everything you need to know about TEX is explained here somewhere, and so are a lot of things that most users don't care about. If you are preparing a simple manuscript, you won't need to learn much about TEX at all; on the other hand, some things that go into the printing of technical books are inherently difficult, and if you wish to achieve more complex effects you will want to penetrate some of TEX's darker corners.



Centred last lines

This is a handbook about TEX, a new typesetting system intended for the creation of beautiful books—and especially for books that contain a lot of mathematics. By preparing a manuscript in TEX format, you will be telling a computer exactly how the manuscript is to be transformed into pages whose typographic quality is comparable to that of the world's finest printers; yet you won't need to do much more work than would be involved if you were simply typing the manuscript on an ordinary typewriter. In fact, your total work will probably be significantly less, if you consider the time it ordinarily takes to revise a typewritten manuscript, since computer text files are so easy to change and to reprocess. This manual is intended for people who have never used TEX before, as well as for experienced TEX hackers. In other words, it's supposed to be a panacea that satisfies everybody, at the risk of satisfying nobody. Everything you need to know about TEX is explained here somewhere, and so are a lot of things that most users don't care about. If you are preparing a simple manuscript, you won't need to learn much about TEX at all; on the other hand, some things that go into the printing of technical books are inherently difficult, and if you wish to achieve more complex effects you will want to penetrate some of TEX's darker corners.



Centred last lines

```
centred_lastline = function (head)
   local myglue = node.new(node.id("glue"))
   myglue.width = 0                  -- 0pt plus 1fil
   myglue.stretch = 1*2^16
   myglue.stretch_order = 2

   local lline = node.tail(head) -- last line
   node.insert_before(lline.head, lline.head, myglue)
   lline.head = myglue

   lline.head = node.hpack(lline.head, lline.width, "exactly")
   return head
end
luatexbase.add_to_callback("post_linebreak_filter",
                           centred_lastline,
                          "centred_lastline")
```


Centred last lines

TEX by Topic 18.3.1 Centred last lines

```
\leftskip=0cm plus 0.5fil \rightskip=0cm plus -0.5fil
\parfillskip=0cm plus 1fil
```


Raggedright `\raggedright`

This is a handbook about TEX, a new typesetting system intended for the creation of beautiful books—and especially for books that contain a lot of mathematics. By preparing a manuscript in TEX format, you will be telling a computer exactly how the manuscript is to be transformed into pages whose typographic quality is comparable to that of the world's finest printers; yet you won't need to do much more work than would be involved if you were simply typing the manuscript on an ordinary typewriter. In fact, your total work will probably be significantly less, if you consider the time it ordinarily takes to revise a typewritten manuscript, since computer text files are so easy to change and to reprocess. This manual is intended for people who have never used TEX before, as well as for experienced TEX hackers. In other words, it's supposed to be a panacea that satisfies everybody, at the risk of satisfying nobody. Everything you need to know about TEX is explained here somewhere, and so are a lot of things that most users don't care about. If you are preparing a simple manuscript, you won't need to learn much about TEX at all; on the other hand, some things that go into the printing of technical books are inherently difficult, and if you wish to achieve more complex effects you will want to penetrate some of TEX's darker corners.

Justified

This is a handbook about TEX, a new typesetting system intended for the creation of beautiful books—and especially for books that contain a lot of mathematics. By preparing a manuscript in TEX format, you will be telling a computer exactly how the manuscript is to be transformed into pages whose typographic quality is comparable to that of the world's finest printers; yet you won't need to do much more work than would be involved if you were simply typing the manuscript on an ordinary typewriter. In fact, your total work will probably be significantly less, if you consider the time it ordinarily takes to revise a typewritten manuscript, since computer text files are so easy to change and to reprocess. This manual is intended for people who have never used TEX before, as well as for experienced TEX hackers. In other words, it's supposed to be a panacea that satisfies everybody, at the risk of satisfying nobody. Everything you need to know about TEX is explained here somewhere, and so are a lot of things that most users don't care about. If you are preparing a simple manuscript, you won't need to learn much about TEX at all; on the other hand, some things that go into the printing of technical books are inherently difficult, and if you wish to achieve more complex effects you will want to penetrate some of TEX's darker corners.

Between raggedright and justified

This is a handbook about TEX, a new typesetting system intended for the creation of beautiful books—and especially for books that contain a lot of mathematics. By preparing a manuscript in TEX format, you will be telling a computer exactly how the manuscript is to be transformed into pages whose typographic quality is comparable to that of the world's finest printers; yet you won't need to do much more work than would be involved if you were simply typing the manuscript on an ordinary typewriter. In fact, your total work will probably be significantly less, if you consider the time it ordinarily takes to revise a typewritten manuscript, since computer text files are so easy to change and to reprocess. This manual is intended for people who have never used TEX before, as well as for experienced TEX hackers. In other words, it's supposed to be a panacea that satisfies everybody, at the risk of satisfying nobody. Everything you need to know about TEX is explained here somewhere, and so are a lot of things that most users don't care about. If you are preparing a simple manuscript, you won't need to learn much about TEX at all; on the other hand, some things that go into the printing of technical books are inherently difficult, and if you wish to achieve more complex effects you will want to penetrate some of TEX's darker corners.

Between raggedright and justified

```
eatlines = function (head)
   for line in node.traverse_id(node.id("hlist"), head) do
      if line.glue_order == 0    -- finite glue
         and line.glue_sign == 1 -- stretching
         and line.glue_set > .2  -- ratio
      then
         line.head = node.hpack(line.head)
      end
   end
   return head
end
luatexbase.add_to_callback("post_linebreak_filter",
                           eatlines, "eatlines")
```


Between raggedright and justified

TEX by Topic 5.9.6 Dissecting paragraphs with \lastbox

```
\newbox\linebox \newbox\snapbox
\def\eatlines{
  \setbox\linebox\lastbox    % check the last line
  \ifvoid\linebox
  \else                      % if it's not empty
  \unskip\unpenalty          % take whatever is
  {\eatlines}                % above it;
                             % collapse the line
  \setbox\snapbox\hbox{\unhcopy\linebox}
                     % depending on the difference
  \ifdim\wd\snapbox<.98\wd\linebox
        \box\snapbox % take the one ore the other,
  \else \box\linebox \fi
  \fi}
```


Fadelines

This is a handbook about TEX, a new typesetting system intended for the creation of beautiful books—and especially for books that contain a lot of mathematics. By preparing a manuscript in TEX format, you will be telling a computer exactly how the manuscript is to be transformed into pages whose typographic quality is comparable to that of the world's finest printers; yet you won't need to do much more work than would be involved if you were simply typing the manuscript on an ordinary typewriter. In fact, your total work will probably be significantly less, if you consider the time it ordinarily takes to revise a typewritten manuscript, since computer text files are so easy to change and to reprocess. This manual is intended for people who have never used TEX before, as well as for experienced TEX hackers. In other words, it's supposed to be a panacea that satisfies everybody, at the risk of satisfying nobody. Everything you need to know about TEX is explained here somewhere, and so are a lot of things that most users don't care about. If you are preparing a simple manuscript, you won't need to learn much about TEX at all; on the other hand, some things that go into the printing of technical books are inherently difficult, and if you wish to achieve more complex effects you will want to penetrate some of TEX's darker corners.

Fadelines

```
fadelines = function (head)
   local graycolor = node.new("whatsit", "pdf_colorstack")
   local gvalue = 0
   for line in node.traverse_id(node.id("hlist"), head) do
      graycolor.data = gvalue .. " g"
      node.insert_before(head, line, node.copy(graycolor))
      gvalue = math.min(gvalue+0.06, 1)
   end
   return head
end
luatexbase.add_to_callback("post_linebreak_filter",
                           fadelines, "fadelines")
```


다양한 예제

- CHICKENIZE
- Three things you can do with LuaTEX that would be extremely painful otherwise
- gist.github.com/dohyunkim/drawcharbox.lua
- github.com/dohyunkim/colorjamo
- github.com/dohyunkim/luatexko
- https://tex.stackexchange.com/search?q=%5Bluatex%5D+call-back

LuaTEXnician이 되는 방법

간단하다. 아래의 책들을 순서대로 읽는다.

1. The TEXbook by Donald E. Knuth
2. TEX by Topic, A TEXnician's Reference by Victor Eijkhout
3. The ε-TEX manual by The NTS Team
4. The pdfTEX user manual by Han The Thanh and friends
5. Programming in Lua by Roberto Ierusalimschy
6. The LuaTEX Reference Manual by LuaTEX deveopment team

루아테크니션이 되었다면, LuaTEX-ko 개발에 참여한다.

맺음말

- 루아텍은 노드와 콜백으로 그동안 다루기 힘들었던 텍 내부에 접근 할 수 있게 되었다.
- 루아텍을 통하여 텍의 진수를 맛볼 수 있다.
- 패키지 개발할때 루아텍을 이용하는 것이 더욱 직관적이고 코드도 이해하기 쉽고, 버그를 줄일 수 있다.
- 루아텍은 패키지 개발자 뿐만 아니라, 일반 사용자에게도 많은 이로움을 제공한다. 특히 한글 문서는 LuaTEX-ko 를 사용하는 것을 권한다.
- 일반 (라)텍 사용자를 넘어 루아테크니션이 되자.