hiki2latex

関西学院大学・理工学部・西谷滋人

2016 年 11 月 4 日

概要

hiki format で書かれた文章を，latex format に変換する

目次

1 【documents】 1

2 【使用法】 1

2.1 sample 2

3 【仕様】 2

4 【具体的な使用例 rakefile】 2

5 【変更履歴，内容】 3

6 【開発メモ】 3

6.1 制限事項 3

6.2 【 保留項目】 4

6.3 【math】 4

6.4 【user def】 4

付録 A 詳細な使用法 4

付録 B コード内容 9

B.1 起動 9

B.2 math 環境 11

B.3 table 環境 14

B.4 source code 環境 17

1 【documents】

- http://rubygems.org/gems/hiki2latex

2 【使用法】

```
Usage: hiki2latex [options] FILE
    -v, --version                    show program Version.
```

```
    -l, --level VALUE                set Level for output section
        .
    -p, --plain FILE                 make Plain document.
    -b, --bare FILE                  make Bare document.
        --head FILE                  put headers of maketitle
            file.
        --pre FILE                   put preamble file.
        --post FILE                  put post file.
        --listings                   use listings.sty for
            preformat with style.
```

- より詳細な使用法は，hiki2latex_gemizing にある．
- 西谷研の内部で利用するときに特化したマニュアルは**こちら** (hiki2latex_manual)．

2.1 sample

幾つかのサンプルを以下に示す．

表 1　hiki2latex により作成された pdf ファイトとその元ネタ．

ソース (hiki 表示)	pdf(latex 変換後)
Shunkuntype(https://github.com/daddygongon/shunkuntype/wiki/Shunkuntype_report)	Shunkuntype **のレポー**
LPSO15 **研究会の例** (LPSO15_fall_meeting_abst)	{{attach_anchor(LP
中間発表 hand out **例** (AbstSample)	{{attach_anchor(Ab
使っている format **集** (DocumentFormatter_format_examples)	

3 【仕様】

1. hikdoc.rb に wrap
2. header 部を独立して提供
3. author, title は head に手を加えて提供
4. 図は attach_view を変換
5. 表はそのまま表示
   (a) table の multi 対応
6. caption は header で提供
7. cite は対応しない．
8. graph, table は [h] で提供．

4 【具体的な使用例 rakefile】

具体的な使用例として Shunkuntype_report を作成した時の Rakefile.rb を示す．何度も書き直す時は，このようにして自動化すべき．

```
1 task :shunkuntype do
2   dir = '~/Sites/nishitani0/Internal/data/text/'
3   targets =["Shunkuntype_manual","Shunkuntype_gemizing","
        TouchTyping_Coding"]
4   targets.each{|file|
```

```
5       p command= "rm␣-f␣#{file}.tex"
6       system command
7       p command= "hiki2latex␣-l␣2␣--listings␣-b␣#{dir}#{file}␣>␣#{
            file}.tex"
8       system command
9     }
10    file="Shunkuntype_report"
11    p command= "rm␣-f␣#{file}.tex"
12  #  p command= "hiki2latex --listings --head head.tex --post post.
        tex -p #{dir}#{file} > #{file}.tex"
13    p command= "hiki2latex␣--listings␣--post␣post.tex␣-p␣#{dir}#{
          file}␣>␣#{file}.tex"
14    system command
15
16    p command = "open␣#{file}.tex"
17    system command
18    exit
19  end
```

本文 (Shunkuntype_report.tex) と付録 (targets.texs) がある．付録は'-l 2 -b' によって subsection からの title level にして bare mode で作っている．post.tex に post で付け加える tex 部を指定して，appendix をつけている．

5 【変更履歴，内容】

変更履歴，内容を表に示す．15/8 月期で基本開発．16/2 月期に gem 化．

表 2　変更履歴，内容

memo	date	hiki
hikidoc.rb から hiki2latex.rb	15/8/4	`hiki2latex_init`
hiki2latex.rb ひな形作成	15/8/5	
@f を StringIO から String へ	15/8/5	
graph+caption	15/8/6	`LPSO15_fall_meeting_abst`
math	15/8/7	`hiki2latex_math`
table	15/8/8	`hiki2latex_table`
under_score	15/8/11	`hiki2latex_math`
gem 化	16/2/13	`hiki2latex_gemizing`
listings	16/2/16	`hiki2latex_listings`

6 【開発メモ】

6.1 制限事項

- title 中へ uri の埋め込みが未対応.
  - uri の verb が latex の title 内で使えないため.

6.2 【 保留項目】

1. includegraphics の自動提供
   （a）hiki に置かれている graph は劣化版なんでそれをいじるのはあまり筋がよろしくない．
   （b）eps ならできるかも．hiki の attach_view でサイズをどういじっているか...
2. underbar(_) が latex では全て引っかかる．escape する？-¿ 対応済み `hiki2latex_math`
   （a）snake 表記は ruby では file 名や変数名に頻繁に使われるので対処が必要かも．

6.3 【math】

$ $ での変換がうまくいかない．
`hikiconf.rb`
での設定を

```
#@style           = 'default'
@style           = 'math'
```

としたら hiki からエラーは出なくなったが．まだまだ....

6.4 【user def】

`\def\Vec#1{\mbox{\boldmath $#1$}}`
は preamble に置くことが推奨されているが，実際は，使用するまでに定義すればいい．preamble をいじるようになるころには，latex についての十分な経験があると思われるので，hiki2latex ではいじらない．ちょこっと必要なら hiki 本文中に埋め込むべし．今の仕様では，

```
1   def initialize(file_name)
2     @buf = ""
3     @buf << HEADER+"\n"
4     @buf <<  "\\begin{document}\n"
5     @buf <<  HikiDoc.to_latex(file_name)+"\n"
6     @buf <<  REFERENCE+"\n"
7     @buf << "\\end{document}\n"
8   end
9
10  def display
11    @buf
12  end
```

とするのが正しそうなので，無理に入れていない．将来は preamble を保持するような拡張機能 (option) が必要かもしれない．それは，「原典一つ主義」で，hiki を原典とするか，latex を原典とするかの選択が迫られたとき．

- – 追記 (2015/02/17) いい考察だ．解はでてないが今は hiki2latex に埋め込んで，それらの仕様をできるだけ吸収しようとしている．

付録 A　詳細な使用法

A.0.1 【概要】

hiki フォーマットの記述を latex フォーマットに変換する

A.0.2 【使用法】

gem 化して配布可能とした．したがって，install がちゃんとできていれば，

command line から

```
bob% hiki2latex Shunkuntype_manual > tmp.tex
```

library として

```
1 require 'hiki2latex'
2
3 puts HikiDoc.to_latex(File.read(ARGV[0]))
```

```
bob% ruby trans.rb ./Shunkuntype_manual > tmp.tex
```

などとして利用できる．

A.0.3 【help】

```
bob% hiki2latex
Usage: hiki2latex [options]
    -v, --version                    show program Version.
    -l, --level VALUE                set Level for output section
        .
    -p, --plain FILE                 make Plain document.
    -b, --bare FILE                  make Bare document.
        --head FILE                  put headers of maketitle
            file.
        --pre FILE                   put preamble file.
        --post FILE                  put post file.
```

A.0.4 【詳細使用例 (-p)】

以下のような SampleDoc

```
1 bob% cat SampleDoc
2 !title
3 contents
4 *list1
5 *list2
6 !!title2
```

を

```
bob% hiki2latex -p SampleDoc
\documentclass[12pt,a4paper]{jsarticle}
\usepackage[dvipdfmx]{graphicx}
\begin{document}
\section{title}
contents
\begin{itemize}
\item list1
\item list2
\end{itemize}
\subsection{title2}
\end{document}
```

となる．

A.0.5 【詳細使用例 (-b)】

上記

```
\begin{document}
...
\end{document}
```

の中身だけを生成．いくつもの tex files を include している場合の個別ファイル作成に便利．

この際，'-l 2' として付録の section とかを調整する．

A.0.6 【詳細使用例 (–head, –pre, –post】

format を標準と違うものにするために，pre,head,post がある．詳しくは sample を見よ．していしていく順番はないはずだけど，-p SampleDoc だけは順番があるのかな．あと，erb みたいにして使ったほうがいいかも．

```
bob% hiki2latex --pre preamble.tex --head head.tex --post post.
    tex -p SampleDoc
```

```
%preamble.tex
\documentclass[12pt,a4paper]{jsarticle}
\usepackage[dvipdfmx]{graphicx}
\pagestyle{empty}
```

```
\begin{document}
```

```
%head.tex
\author関西学院大学・理工学部・西谷滋人{}
\title{touch 習得環境typingshunkuntype}
\date{\today}
\maketitle
```

```
\section{title}
contents
\begin{itemize}
\item list1
\item list2
\end{itemize}
\subsection{title2}
```

```
%post.tex
\pagebreak
\appendix
\sectionマニュアル{}
\input{shunkuntype_manual}
\pagebreak
\section{化メモgem}
\input{shunkuntype_gemizing}
\pagebreak
\section初期版のコード解説{}
\input{shunkuntype_coding}
```

```
\end{document}
```

A.0.7 【lib/hiki2latex.rb】

```
require 'optparse'
require "hiki2latex/version"
#require "hiki2latex/hikidoc"
require "hiki2latex/hiki2latex"

module Hiki2latex
  class Command

    def self.run(argv=[])
      new(argv).execute
    end

    def initialize(argv=[])
      @argv = argv
      @pre=@head=@post=nil
    end

    def execute
      @argv << '--help' if @argv.size==0
      command_parser = OptionParser.new do |opt|
        opt.on('-v', '--version','show program Version.') { |v|
          opt.version = Hiki2latex::VERSION
          puts opt.ver
        }
        opt.on('-l VALUE','--level','set Level for output section.'){|level| @level=level.to_i}
        opt.on('-p FILE', '--plain','make Plain document.') { |file| plain_doc(file) }
        opt.on('-b FILE', '--bare','make Bare document.') { |file| bare_doc(file) }
        opt.on('--head FILE', 'put headers of maketitle file.') { |file| @head=file }
        opt.on('--pre FILE', 'put preamble file.') { |file| @pre=file }
        opt.on('--post FILE', 'put post file.') { |file| @post=file }
        opt.on('--listings', 'use listings.sty for preformat with style.') {@listings=true }
      end
      command_parser.banner = "Usage: hiki2latex [options] FILE"
      command_parser.parse!(@argv)
      plain_doc(@argv[0]) if @argv[0]!=nil
      exit
    end

    # pre, post, などの拡張を処理listings
    def plain_doc(file)
      if @listings==true then
        puts listings_preamble
      elsif @pre==nil then
        puts "\\documentclass[12pt,a4paper]{jsarticle}"
        puts "\\usepackage[dvipdfmx]{graphicx}"
      else
        puts File.read(@pre)
```

```
48        end
49        puts "\\begin{document}"
50        puts File.read(@head) if @head!=nil
51        plain_tex = HikiDoc.to_latex(File.read(file),{:listings=>
              @listings})
52        puts mod_abstract(plain_tex)
53        puts File.read(@post) if @post!=nil
54        puts "\\end{document}"
55      end
56
57      def bare_doc(file)
58        bare_doc = HikiDoc.to_latex(File.read(file),{:level=>@level
              ,:listings=>@listings})
59        puts kill_head_tableofcontents(bare_doc)
60      end
```

■kill_head_line　kill_head_line で付録 (bare_text) に toc が含まれている場合は削除.

```
1     def kill_head_tableofcontents(text)
2       text.gsub!(/^\\tableofcontents/,'')
3     end
```

■mod_abstract　mod_abstract で\section{abstract}で書かれた内容を abstract 環境へ移行する. 一行づつの処理は有限状態マシン的に処理するのがいいらしい (Ruby Best Practice, p.112). 一旦 content と abstract を分けて, その後, tableofcontent の前に abstract を挿入している. delete_at(0) は一行目に\section{【abstract】}が存在するため.

```
1  # convert section to abstract
2     def mod_abstract(text)
3       abstract,section = [],[]
4       content = ""
5       text.split("\n").each do |line|
6         case line
7         when /^\\section(.+)/
8           section.push $1
9         end
10
11        case section[-1]
12        when /.+abstract.+/
13          abstract << line+"\n"
14        when 概要/.+.+/
15          abstract << line+"\n"
16        else
17          content << line+"\n"
18        end
19      end
20      abstract.delete_at(0)
21      content.gsub!(/\\tableofcontents/){|text|
22        tt="\n\\abstract\{\n#{abstract.join}\}\n\\tableofcontents
              "
23      }
24      return content
25    end
```

付録 B　コード内容

B.1　起動

B.1.1　【変更作業の手順】

- hiki を html に変換する hikidoc.rb に加えていく形で変更
- hikidoc について調べよ (澄田の宿題)

```
bob% diff hikidoc.rb master-hikidoc/lib/hikidoc.rb
39d38
< require './hiki2latex.rb'
58,61d56
<   def HikiDoc.to_latex(src, options = {})
<     new(LatexOutput.new(""), options).compile(src)
<   end
<
916,917c911
< #  puts HikiDoc.to_html(ARGF.read(nil))
<   puts HikiDoc.to_latex(ARGF.read(nil))
---
>   puts HikiDoc.to_html(ARGF.read(nil))
```

- hiki2latex.rb に変更を加えていく.

B.1.2　【設計】

■【maketitle の挿入】　latex format に仕込む際にテキストの順序が前後するコマンドが幾つかある. maketitle の前に置かれた,

- title
- author

である. これらは headers として text 部と別に返すことも可能である. 具体的には,

表 3

to_html	*f を返す
to_latex	*f と*head を返す.

なお to_html との互換性を維持するため*head を後に返すようにしている.
しかし, この部分は, to_html との整合性を考えると to_latex で処理してしまった方がよい. そこで,

```
def finish
  @f.string
end
```

となっているのを,

```
  def finish
    if @head != "" then
      @head << "\\maketitle\n"
      return @head+@f
    else
      return @f
    end
  end
```

とした.

B.1.3 【sample】

- Jihou15

B.1.4 【pdf の貼り込み】

pdfpages を使おうとすると

```
/usr/local/texlive/2015/texmf-dist/tex/latex/pdfpages/pdfpages.sty:70: LaTeX Er
ror: Missing \begin{document}.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.70 \input{pp\AM@driver.def}
```

というエラーが出る．この解決法がわからず，incudegraphics で対応.

```
¥documentclass{jsarticle}
¥usepackage[dvipdfmx,hiresbb]{graphicx}
¥begin{document}
¥includegraphics[width=5cm]{sample.pdf}
¥end{document}
```

の 4 行目を

```
¥includegraphics[page=3,width=5cm]{sample.pdf}
```

とすれば、

普通に３ページ目の画像を使うことができました。

美文書 5 版の CD-ROM から普通にインストールした TeXShop の環境です。

つまり、何も（といっても、同書 P114 にあるように、

```
 --shell--escape--commands=extractbb
```

を TeXShop 環境設定の「内部設定」タブ　pdfTex の Latex のオプションとして付け加えています。）特別なことをせずに、ページ指定をするだけで、できてしまいました。

次の、美文書の版には、ページ設定についても解説していただくといいかも知れません。＞奥村先生お願いします。

B.2 math 環境

B.2.1 【概要】

dmath,math 環境を equation に変換

B.2.2 【入力：hiki】

```
math がうまくいくかどうかの検討用サンプルです．
{{dmath 'f_x'}}
さらに inline で{{math 'x_i'}}なんかもできるといいのですが．
```

math がうまくいくかどうかの検討用サンプルです．

$$f_x \tag{1}$$

さらに inline で x_i なんかもできるといいのですが．

B.2.3 【出力：latex】

```
\begin{document}
math がうまくいくかどうかの検討用サンプルです．
\begin{equation}
f_x
\end{equation}
さらに inline で$x_i$なんかもできるといいのですが．
```

注 2016/02/20　このあたりで hiki，ruby を指定すると listings がうまく処理できない．preformatted_without_style で対応．

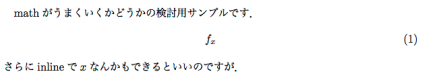

図 1

B.2.4 【コード解説】

■evaluate_plugin_block　すこしトリッキーだったので，メモです．

```
1   def evaluate_plugin_block(str, buf = nil)
2     buf ||= @output.container
3     str.split(/(\0\d+\0)/).each do |s|
4       if s[0, 1] == "\0" and s[-1, 1] == "\0"
5         buf << @output.inline_plugin(plugin_block(s[1..-2].to_i))
6       else
7         buf << @output.text(s)
8       end
9     end
10    buf
11  end
```

として inline_plugin は block_plugin と違った扱いになっています．buf に一度溜め込んでそれを@f に吐いているようです．そこで，@output.inline_plugin で行き着く先が，LatexOutput の

```
1   def inline_plugin(src)
```

```
    tmp=[]
    if ( /(\w+)\((.+)\)/ =~ src ) or ( /(\w+).\'(.+)\'/␣=~src␣)␣
        or␣(/(\w+)/␣=~␣src)
␣␣␣␣␣␣tmp␣=␣[$1,$2]
␣␣␣␣end

␣␣␣␣case␣tmp[0]
␣␣␣␣when␣'dmath'
␣␣␣␣␣␣"\\begin{equation}\n#{tmp[1]}\n\\end{equation}"
␣␣␣␣when␣'math'
␣␣␣␣␣␣"\$#{tmp[1]}\$"
␣␣␣␣else
␣␣␣␣␣␣%Q(<span␣class="plugin">{{#{src}}}</span>)
␣␣␣␣end
␣␣end
```

として math,dmath を処理しています．必要ならそれ以外の inline もここに付け足すことで対応できます．

■snake_name が latex で引っかかる　math 環境の移し替えはうまくいったが，underscore_names がすべて引っかかる equation として引っかかる．

```
\usepackage{underscore}
```

だけでもできるようだが，できれば to_latex で対応したい．ということで，paragraph に escape_snake_names を入れた．

paragraph は preformatted からは呼ばれない．しかし，$$や\begin{equation}...\end{equation}が含まれる．そこで一旦全置換してそこだけ戻すようにした．gsub のなかでなんどもできるか自信がなくて．二重の gsub にしているが...

```
  def escape_snake_names(str)
    str.gsub!(/_/,"\\_")
    str.gsub!(/\$.+?\$/){ |text| text.gsub!(/\\_/,"_")  }
    str.gsub!(/equation.+?equation/m){ |text| text.gsub!(/\\_/,"_
        ") }
  end

  def paragraph(lines)
    lines.each{|line| line = escape_snake_names(line) }
    @f << "#{lines.join("\n")}\n\n"
  end
```

gsub!で置換できなかったときには，nil が返るので対応．なんかえらく冗長．

```
  def escape_snake_names(str)
    str.gsub!(/_/,"\\_")
    str.gsub!(/\$.+?\$/){ |text|
      if text =~ /\\_/ then
        text.gsub!(/\\_/,"_")
        else
        text
      end
    }
    str.gsub!(/equation.+?equation/m){ |text|
      if text =~ /\\_/ then
        text.gsub!(/\\_/,"_")
        else
        text
      end
```

```
16      }
17      str
18    end
```

■escape_snake_names を改良 (2016/02/14)　gem への公開にあたって冗長部を簡略化．verb だけはここを切り出した検証とは違う．test に uri を埋め込んで verb 変換と snake を検証．

```
1   def escape_snake_names(str)
2     str.gsub!(/_/,"\\_")
3     patterns = [/\$(.+?)\$/ , /\\verb\|(.+?)\|/, /equation(.+?)
          equation/m ]
4     patterns.each{|pattern|
5       str.gsub!(/\\_/,"_")     if str.match(pattern)
6     }
7     str
8   end
```

■escape_snake_names を再改良 (2016/02/16)　underscore では include などで snake_named_file も変換してしまい，だめ．hiki2latex で対応．

上のやり方では，うまくいかない場合が存在．元へ戻す．

```
1  def escape_snake_names(str)
2      str.gsub!(/_/,"\\_")
3      patterns = [/\$(.+?)\$/ , /\\verb\|(.+?)\|/, /equation(.+?)
           equation/m ]
4      patterns.each{|pattern|
5  #       str.gsub!(/\\_/,"_")     if str.match(pattern)
6        str.gsub!(pattern){|text|
7          if text =~ /\\_/ then
8            text.gsub!(/\\_/,'_')
9          else
10           text
11         end
12       }
13     }
14     str
15   end
```

さらに table で escape_snake_names を通ってなかった．

```
#         buf << "#{ele} &"
          buf << escape_snake_names(ele)+" &"
```

B.2.5　【copyright】

cc by Shigeto R. Nishitani, 2015

B.3 table 環境

B.3.1 【概要】

hiki2latex の table 処理部

B.3.2 【仕様】

- hiki 記法の表を tabular へ変換
- 連結作用素に対応
- 行連結では中心に表示を移動
- 縦罫は基本使わない
- 横罫は header 内枠と上下外枠のみ

B.3.3 【hiki】

```
||>A11||>A12||
||^^A21||A22||>A23||
||A11||^A22||A12||
||A21||A23||
```

表 4

A11		A12	
	A22	A23	
A21	A11	A22	A12
	A21		A23

B.3.4 【latex】

```
1  ¥
2  begin{center¥}begin{table}[htbp¥]begin{tabular}{ccccc}¥
3  hline¥
4  multicolumn{2}{c}{A11 } ¥ &multicolumn{2}{c}{A12 }  & ¥¥¥  hline
5   &A22 ¥ &multicolumn{2}{c}{A23 }  & ¥¥
6  A21  &A11  &A22  &A12  & ¥¥
7   &A21  & &A23  & ¥¥¥
8  hline¥
9  end{tabular¥}end{table¥}end{center}横罫を入れる場合は,
10 % ¥hline , ¥clineなどで. {2-3}
```

B.3.5 【コード解説】

元の HTMLOutput ではそれぞれの要素で対応していたが，LatexOutput では table_close にて

```
1     def table_close
2       @f << make_table
3     end
```

としている．make_table は下請けに make_matrix を読んでおり，ここでほぼ全ての作業をしている．作業内容は

A11		A12	
	A22	A23	
A21	A11	A22	A12
	A21		A23

図 2

- matrix を作る
- 縦連結を処理
  - 縦連結の数 (rs) だけ下行に追加
  - 連結の中心 (c_rs) に内容を表記
- 横連結を multicolumn で処理
  - ついでに最大列数 (max_col) を記録

```
1  bob% cat table.rb
2  cont = File.readlines("table")
3
4  cont.each{|line|
5    p tmp=line.split('||')
6  }
7
8  t_matrix=[]
9  cont.each{|line|
10   tmp=line.split('||')
11   tmp.slice!(0)
12   tmp.slice!(-1) if tmp.slice(-1)=="\n"
13   tmp.each_with_index{|ele,i| tmp[i] = ele.match(/\s*(.+)/)[1]}
14   t_matrix << tmp
15 }
16
17 t_matrix.each_with_index{|line,i|
18   line.each_with_index{|ele,j|
19     if ele=~/\^+/ then
20       t_matrix[i][j]=""
21       rs=$&.size
22       c_rs=rs/2
23       rs.times{|k| t_matrix[i+k+1].insert(j,"")}
24       t_matrix[i+c_rs][j]=$'
25 ␣␣␣␣end
26 ␣␣}
27 }
28 p␣t_matrix
29
30 max_col=0
31 t_matrix.each_with_index{|line,i|
32 ␣␣n_col=line.size
33 ␣␣line.each_with_index{|ele,j|
34 ␣␣␣␣if␣ele=~/>+/␣then
35 ␣␣␣␣␣␣cs=$&.size
36 ␣␣␣␣␣␣t_matrix[i][j]=␣"\\multicolumn{#{cs+1}}{c}{#{$'}} "
37 ␣␣␣␣␣␣n_col+=cs
```

```
38 ␣␣␣␣end
39 ␣␣}
40 ␣␣max_col␣=␣n_col␣if␣n_col>max_col
41 }
42 p␣t_matrix
43 p␣max_col
```

B.4 source code 環境

B.4.1 【概要】

latex の listings スタイルを使って，source code の色付き表示が可能に.

B.4.2 【hiki2latex の変更】

hikidoc2tex で使われているのを参照して，

```
1   def block_preformatted(str, info)
2     if (@listings==true and info!=nil) then
3       style='customRuby' if info=='ruby'
4       style='customCsh' if (info=='tcsh' or info=='csh')
5       style='customTeX' if info=='tex'
6       style='customJava' if info=='java'
7       preformatted_with_style(str,style)
8     else
9       preformatted(text(str))
10    end
11  end
12
13  def preformatted(str)
14    @f.slice!(-1)
15    @f << "\\begin{quote}\\begin{verbatim}\n"
16    @f << str+"\n"
17    @f << "\\end{verbatim}\\end{quote}\n"
18  end
19
20  def preformatted_with_style(str,style)
21    @f.slice!(-1)
22    @f << "\\begin{lstlisting}[style=#{style}]\n"
23    @f << str+"\n"
24    @f << "\\end{lstlisting}\n"
25  end
```

opt 周りは,

```
1         opt.on('--listings', 'use␣listings.sty␣for␣preformat␣with
              ␣style.') {@listings=true }
```

としている．これを HikiDoc へ渡す時は,

```
1     def plain_doc(file)
2       if @listings==true then
3         puts listings_preamble
4       elsif @pre==nil then
5         puts "\\documentclass[12pt,a4paper]{jsarticle}"
6         puts "\\usepackage[dvipdfmx]{graphicx}"
7       else
8         puts File.read(@pre)
9       end
10      puts "\\begin{document}"
11      puts File.read(@head) if @head!=nil
12      puts HikiDoc.to_latex(File.read(file),{:listings=>@listings
            })
13      puts File.read(@post) if @post!=nil
```

```
14          puts "\\end{document}"
15        end
```

後ろの options=の中に hash で登録している．tex の style は listing_preamble で打ち出すようにしている．

listings の設定は以下の通り．

```
\documentclass[10pt,a4paper]{jsarticle}
\usepackage[dvipdfmx]{graphicx}
\usepackage[dvipdfmx]{color}
\usepackage{listings}
% to use japanese correctly, install jlistings.
\lstset{
  basicstyle={\small\ttfamily},
  identifierstyle={\small},
  commentstyle={\small\itshape\color{red}},
  keywordstyle={\small\bfseries\color{cyan}},
  ndkeywordstyle={\small},
  stringstyle={\small\color{blue}},
  frame={tb},
  breaklines=true,
  numbers=left,
  numberstyle={\scriptsize},
  stepnumber=1,
  numbersep=1zw,
  xrightmargin=0zw,
  xleftmargin=3zw,
  lineskip=-0.5ex
}
\lstdefinestyle{customCsh}{
  language={csh},
  numbers=none,
}
\lstdefinestyle{customRuby}{
  language={ruby},
  numbers=left,
}
\lstdefinestyle{customTex}{
  language={tex},
  numbers=none,
}
\lstdefinestyle{customJava}{
  language={java},
  numbers=left,
}
```

```
\begin{lstlisting}[style=customRuby]
 def block_preformatted(str, info)
   if (@listings==true and info!=nil) then
     style='customRuby' if info=='ruby'
     style='customCsh' if (info=='tcsh' or info=='csh')
     style='customTeX' if info=='tex'
...
\end{lstlisting}
```