```
#!/usr/bin/env python
# -*- coding:utf-8 -*-
################################
# 「ttth 1.0」
# TeX文章をHTMLに変換するプログラム
# 文章はHTML、デザインはCSS
# 出力はUTF-8
# ファイル読み込みはコマンドラインから
# >python ttth.py *.tex
# 詳細はreadme.txtに記載
################################

import sys, re, datetime, string

## TeXファイル処理 ######################################
file_name = sys.argv[1]
htmlfile = file_name.replace("tex","html")
tex_file = open(file_name,"rU")
data = tex_file.read()
tex_file.close()
#####表用のcssファイルを初期化
ss = open("output/style_table_"+ file_name[:-4]+".css","w")
ss.write("")
ss.close()

############################################################
## 正規表現
############################################################
## 前処理用
re_pream = re.compile(r'\\documentclass(\[.+?\])?\{(.+?)\}(.+?)\\begin\
{document\}',re.DOTALL)
re_pers = re.compile(r'\\(begin\{)?((vlV)erb(atim)?)\}?')
re_verb = re.compile(r'\\verb(.)(.+?)\n')
#re_verbex = re.compile(r'\\verb.+?\\begin\{(V|v)erbatim\}') #non
re_verb2 = re.compile(r'\\verb-;=(.+?)-;=')
#re_v1 = re.compile(r'\\verb') #non
re_vverbatim = re.compile(r'\\begin\{Verbatim\}(.+?)\\end\{Verbatim
\}',re.DOTALL)
re_verbatim = re.compile(r'\\begin\{verbatim\}(.+?)\\end\{verbatim\}',re.DOTALL)

## コメント
```

```
40: re_comment = re.compile(r'([^\\])%(.*)')
41: re_tag_comment = re.compile(r'<!--.*?-->')
42:
43: ## title, subtitle, author, date
44: re_title = re.compile(r'\\title\{([^\{]+)\}')
45: re_subtitle = re.compile(r'\\subtitle\{([^\{]+)\}')
46: re_author = re.compile(r'\\author\{([^\{]+)\}')
47: re_date = re.compile(r'\\date\{([^\{]+)\}')
48:
49: ## documentclass
50: re_class = re.compile(r'\\documentclass(\[.+?\])?\{(.+?)\}')
51: re_document = re.compile(r'\\begin\{document\}.+\\end\{document\}',re.DOTALL)
52: re_b_document = re.compile(r'\\begin\{document\}')
53: re_e_document = re.compile(r'\\end\{document\}')
54:
55: re_maketitle = re.compile(r'\\(maketitle|タイトルページ)')
56: re_parindent = re.compile(r'\\parindent=([0-9\.]+?pt)')
57:
58: re_easylayout = re.compile(r'\\usepackage\{EasyLayout\}')
59:
60: ## 記号 ###############
61: re_onesp = re.compile(r'(\\@|\\,|[a-zA-Z0-9]~[a-zA-Z0-9])')    #-> \s
62: #re_nonsp = re.compile(r'()')
63: re_hyfn = re.compile(r'([^<][^!])?--[^>]?')
64:
65: ## TeXロゴ ####################
66: re_tex = re.compile(r'(\\TeX\{\}|\{\\TeX\}|\\TeX\s?)')
67: re_latex = re.compile(r'(\\p?LaTeX\{\}|\{\\p?LaTeX\}|\\p?LaTeX\s?)')
68: re_latexe = re.compile(r'(\\p?LaTeX2?e\{\}|\{\\p?LaTeX2?e\}|\\p?LaTeX2?e\s?)')
69:
70: ####################
71: re_newpage = re.compile(r'(\\newpage\{\}|\{\\newpage\}|\\newpage)
    {1,}',re.DOTALL)
72:
73: ## 罫線
74: re_rulefill = re.compile(r'(^.*?)(\\(hrule|dot)fill\s?|\{\\(hrule|dot)fill\}|\\(hrule|
    dot)fill\{\})(.*?$)(<br>|)')
75: re_frulef = re.compile(r'(\\hrulefill\s?|\{\\hrulefill\}|\\hrulefill\{\})')
76: re_dotf = re.compile(r'(\\dotfill\s?|\{\\dotfill\}|\\dotfill\{\})')
77: re_par = re.compile(r'(\\par\s?|\{\\par\}|\\par\{\})')
78:
```

```
## 文字サイズ !宣言
re_huge1 = re.compile(r'\{\\Huge\s?([^\{]+?)\}')
re_huge2 = re.compile(r'\{\\huge\s?([^\{]+?)\}')
re_large1 = re.compile(r'\{\\LARGE\s?([^\{]+?)\}')
re_large2 = re.compile(r'\{\\Large\s?([^\{]+?)\}')
re_large3 = re.compile(r'\{\\large\s?([^\{]+?)\}')
re_normalsize = re.compile(r'\{\\normalsize\s?([^\{]+?)\}')
re_small = re.compile(r'\{\\small\s?([^\{]+?)\}')
re_footnotesize = re.compile(r'\{\\footnotesize\s?([^\{]+?)\}')
re_scriptsize = re.compile(r'\{\\scriptsize\s?([^\{]+?)\}')
re_tiny = re.compile(r'\{\\tiny\s?([^\{]+?)\}')

## 文字の修飾 ###############
re_t_df = re.compile(r'\\text(up|md|rm)\{([^\{]+?)\}')#標準
re_t_bf = re.compile(r'\\textbf\{([^\{]+?)\}')
re_t_it = re.compile(r'\\textit\{([A-Za-z0-9\s]+)\}')
re_t_sl = re.compile(r'\\textsl\{([^\{]+?)\}')#斜体
re_t_sc = re.compile(r'\\textsc\{([^\{]+?)\}')#Smallcaps
re_t_sf = re.compile(r'\\textsf\{([^\{]+?)\}')#サンセリフ
re_t_tt = re.compile(r'\\texttt\{([^\{]+?)\}')#タイプライター
re_u_line = re.compile(r'\\(Ulu)nderline\{([^\{]+)\}')
re_emph = re.compile(r'\\emph\{([^\{]+?)\}')

## アクセント
re_acsent0 = re.compile(r'\\.\{[a-zA-Z]\}')

## img
re_img = re.compile(r'\\includegraphics\[(.+?)\]\{(.*?)([^/]+?)\}')

##position
re_raisebox = re.compile(r'\\raisebox\{(-|)(.+?)\}\{([^\{]+?)\}')
re_raisebox2 = re.compile(r'\\raisebox\{(-|)(.+?)\}([^\{]+?)')

re_rule = re.compile(r'\\rule(\[(-|)(.+?)\])?\{(.+?)\}\{(.+?)\}')
re_zero = re.compile(r'^\.')

re_vspace = re.compile(r'\n?\\vspace\*?\{(-|)([0-9\.]*?)\\(Cvs|baselineskip|fill)\}
\n?')
re_hspace = re.compile(r'\\hspace\*?\{([0-9\.]+?)([a-z]+?)\}')
re_hspacem = re.compile(r'\\hspace\*?\{-([0-9\.]+?)([a-z]+?)\}([^\{\\\n]+)')

```

```
119: re_makebox = re.compile(r'\\makebox\[\\(textwidth|linewidth)\]\{([^\{]+?)\}')
120:
121: #part section
122: re_part = re.compile(r'\\part\{([^\{]+?)\}')
123: re_section = re.compile(r'\\section\{([^\{]+?)\}')
124: re_subsection = re.compile(r'\\subsection\{([^\{]+?)\}')
125: re_subsubsection = re.compile(r'\\subsubsection\{([^\{]+?)\}')
126: re_paragraph = re.compile(r'\\paragraph\{([^\{]+?)\}')
127: re_subparagraph = re.compile(r'\\subparagraph\{([^\{]+?)\}')
128: re_par = re.compile(r'\\part\{[^\{]+?\}.+?(<!--endpart-->|</body>)',re.DOTALL)
129: re_sec = re.compile(r'\\section\{[^\{]+?\}.+?(<!--endsec-->|</body>)',re.DOTALL)
130: re_sub = re.compile(r'\\subsection\{[^\{]+?\}.+?(<!--end(|sub)sec-->|</
     body>)',re.DOTALL)
131:
132: re_footnote = re.compile(r'\\footnote\{([^\{]+?)\}')
133:
134: ## \begin{}\end{}
135: ## 環境
136: re_env = re.compile(r'\\begin\{.+?\}')
137:
138: re_center = re.compile(r'\\begin\{center\}(.+?)\\end\{center\}',re.DOTALL)
139: re_flushright = re.compile(r'\\begin\{flushright\}(.+?)\\end\{flushright
     \}',re.DOTALL)
140: re_flushleft = re.compile(r'\\begin\{flushleft\}(.+?)\\end\{flushleft\}',re.DOTALL)
141: re_quotation = re.compile(r'\\begin\{quotation\}(.+?)\\end\{quotation
     \}',re.DOTALL)
142:
143: re_itemize = re.compile(r'\\begin\{itemize\}(.+?)\\end\{itemize\}',re.DOTALL)
144: re_enumerate = re.compile(r'\\begin\{enumerate\}(.+?)\\end\{enumerate
     \}',re.DOTALL)
145: re_description = re.compile(r'\\begin\{description\}(.+?)\\end\{description
     \}',re.DOTALL)
146: re_item = re.compile(r'\\item\s(.+)')
147: re_item2 = re.compile(r'\\item\s\[(.+?)\](.+)')
148:
149: re_wrapfigure = re.compile(r'\\begin\{wrapfigure\}\{(l|r)\}\{(.+?)\}(.+?)\\end\
     {wrapfigure\}',re.DOTALL)
150: re_tag_img = re.compile(r'<img(.+?)>')
151:
152: re_page = re.compile(r'(.[^!][^-]{4}.\n?)?\\begin\{(boxed|)minipage\}\{0?\.
     ([0-9]+?)\\textwidth\}(.+?)\\end\{(boxed|)minipage\}(\n?.[^!][^-]{4}.)',
```

```
re.DOTALL)
re_w = re.compile(r'\\begin\{(boxedl)minipage\}\{0?\.([0-9]+?)\\textwidth\}(.+?)\\end\{(boxedl)minipage\}\n<!---->\n\\begin\{(boxedl)minipage\}\{0?\.([0-9]+?)\\textwidth\}(.+?)\\end\{(boxedl)minipage\}', re.DOTALL)
re_pcome = re.compile(r'<!---->')
re_pagepart = re.compile(r'\\end\{(lboxed)minipage\}(\n?.[^!][^-]{4}.)',re.DOTALL)

re_bb = re.compile(r'\\{2}')
re_table = re.compile(r'\\begin\{tabular\}\{(.+?)\}(.+?)\\end\{tabular\}', re.DOTALL)
re_tdand = re.compile(r'\s*&\s*')
re_mcolumn = re.compile(r'<td class="([0-9a-z]+?)">\\multicolumn\{([0-9]+)\}\{(.+?)\}\{(.+?)\}</td>(<td class="([0-9a-z]+?)">|</tr>)')
re_mrow = re.compile(r'<td class=".+?">\\multirow\{([0-9]*)\}\{(.+?)\}\{(.+?)\}')
re_thline = re.compile(r'\\hline')
re_thlinew = re.compile(r'\\hline\s{0,}\\hline')
re_trtds = re.compile(r'(</tr>|</td>)')
re_tablex = re.compile(r'\\begin\{tabularx\}(.+?)\\end\{tabularx\}', re.DOTALL)
re_tablex2 = re.compile(r'\\begin\{tabularx\}\{(.+?)\}\{(.+?)\}(.+?)\\end\{tabularx\}', re.DOTALL)
re_tablexp = re.compile(r'p\{(.+?)\}')

re_newcommand = re.compile(r'\\newcommand\{(\\[^\{\[]+?)\}\n?\s{0,}\{([^\{]+?)\}',re.DOTALL)

re_mokuji = re.compile(r'\\(tableofcontents|目次)')
re_midashi2 = re.compile(r'<h2\sclass=".+?"><a\s.+?>([0-9\.]+?\s.+?)</a></h2>')
re_midashi3 = re.compile(r'<h3\sclass=".+?"><a\s.+?>([0-9\.]+?\s.+?)</a></h3>')

re_bunken = re.compile(r'\\begin\{文献表\}(.+?)\\end\{文献表\}',re.DOTALL)
re_item3 = re.compile(r'\\item\s([^\{]+?)\\Year\{(.+?)\}\{(.+?)\}',re.DOTALL)
re_item4 = re.compile(r'\\item\s([^\{]+?)\{(.+?)\}',re.DOTALL)

re_color = re.compile(r'\{\\color\{(.+?)\}\s(.+?)\}')

######## ごみ処理及び体裁整備用
re_area = re.compile(r'(</code>.+?<code>|<body>.+?<code>|</code>.+?</html>)',re.DOTALL)
re_br = re.compile(r'(\n{2,}|\\{2}|\\par)')

re_nonbr1 = re.compile(r'(</?(h[1-6]|hr|div|ol|ul|br|li|pre|p|blockquote|dt|dd|dl)(\s.
```

```
+?)?>)\n*<br>',re.DOTALL)
re_nonbr2 = re.compile(r'<br>\n*(</?(h[1-6]|hr|div|ol|ul|br|li|pre|p|blockquote|dt|dd|dl))',re.DOTALL)

re_dustcoment = re.compile(r'<!--(end(|sub)sec)-->\n')
re_dusttag = re.compile(r'<p>\n*</p>\n*',re.DOTALL)
re_dusttagp = re.compile(r'</p>\n(<h2 class="section">1)')

re_dustblank = re.compile(r'((\n|\t){1,}|\s{2,})')

##不要TeXコマンド
re_dust1 = re.compile(r'\\(thispagestyle|tableofcontents|narrowbaselines|setcounter)([.+?])?(\{.+?\})*')
re_dust2 = re.compile(r'\\(後注|ページ番号)(\{.*?\})?')

## HTMLtag
re_prev = re.compile(r'<pre>')
re_tag = re.compile(r'</?.+?>')
re_preamble = re.compile(r'</title>(.+?)</head>',re.DOTALL)
re_tag_title = re.compile(r'<title>.*?</title>')

############################################################
## 初期値
zwh = 9.6#pt = 1zw=1zh
today = datetime.datetime.today().strftime("%Y/%m/%d")

title = ""
subtitle = ""
author = ""
date = today
fontsize = 10

part = 0
section = 0
subsection = 0
subsubsection = 0
sec_num = ""
sec1 = ""
sec2 = ""
sec3 = ""
sec4 = ""
```

```
sec5 = ""

style = ""
note_num = 0
note_data = "<br><hr>\n<h3><a name=\"thu\">注</a></h3>\n<p class=\"thu\">
\n"

table_no = 0
td_no = 1
tr_no = 1
tdwidth = []

b_start = '</head>\n<body>\n<div id="contents">\n'
b_end = '</p>\n</div>\n</body>\n</html>'

tex = '<span class="lt">T</span><span class="le">E</span>X'
latex = '<span class="ll">L</span><span class="la">A</span>' + tex
latexe = latex + ' <span class="l2">2</span><span class="lee">&#949;</span>'

newcombefor = []
newcomafter = []

contents = "<div class=\"mokuji\"><h3>目　次</h3>\n"

############################################################
## 関数定義 パーツ
############################################################

## verbatim
def f_signal1(all):
	all = all.replace('&','&amp;')
	all = all.replace('\\','&#165;')
	all = all.replace('<','&lt;')
	all = all.replace('>','&gt;')
	all = all.replace('%','&#37;')
	return all

def f_verbase(all):
	all = f_signal1(all)
	all = '<code>' + all + '</code>'
	return all
```

```
264:
265: def f_verbatim(m):
266:         all = m.group(1)
267:         all = '<pre>' + f_verbase(all) + '</pre>\n'
268:         return all
269:
270: def f_verb1(m):
271:         key = m.group(1)
272:         all = m.group(0)
273:         all = all.replace(key,"-;=",2)
274:         all = re_verb2.sub(f_verb2,all)
275:         return all
276:
277: def f_verb2(m):
278:         all = m.group(1)
279:         all = f_verbase(all)
280:         return all
281:
282: ## comment
283: def f_comment(m):
284:     all = m.group(2)
285:     all = f_signal1(all)
286:     all = m.group(1) +"<!--" + all + "-->"
287:     return all
288:
289: re_fontsize = re.compile(r'([0-9\.]+?)pt')#フォントサイズ処理
290:
291: ## \documentclass
292: def f_header(m):
293:         global title, subtitle, author, date, style, fontsize
294:         if m.group(2) != "jsarticle":
295:                 print "This Class-Type is "+ m.group(2)
296:                 exit()
297:         if m.group(1) != None:
298:                 option = m.group(1)
299:                 fs = re_fontsize.search(option)
300:                 fontsize = float(fs.group(1))
301:
302:         header = '''<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//
    . EN"
303:         "http://www.w3.org/TR/html4/loose.dtd">
```

```
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta http-equiv="Content-Style-Type" content="text/css">
<link rel="stylesheet" type="text/css" href="./style.css" title=default>
<link rel="stylesheet" type="text/css" href="./style_table_%s.css" title=default>
'''%(file_name[:-4])
        m = re_parindent.search(data)  #parintent処理
        if m != None:
                style = '<style type="text/css">\n<!--\nbody{font-size:'+ str(fontsize)
+'pt;}\np{text-indent:'+ m.group(1) +';}\n-->\n</style>\n'
        elif m == None and re_easylayout.search(data): #parindentがないEasyLayoutがあ
る
                style = '<style type="text/css">\n<!--\nbody{font-size:'+str(fontsize)
+'pt;}\np{text-indent:0;}\n-->\n</style>\n'
        header += style +'<title></title>\n'
        return header

## 箇条書き
def f_enumerate(m):
        all = "<ol>" + m.group(1) + "</ol>"
        while re_item.search(all):
                all = re_item.sub('<li>\\1',all)
        return all

def f_itemize(m):
        all = "<ul>" + m.group(1) + "</ul>"
        while re_item.search(all):
                all = re_item.sub('<li>\\1',all)
        return all

def f_description(m):
        all = "<dl compact>" + m.group(1) + "</dl>"
        while re_item2.search(all):
                all = re_item2.sub('<dt><b>\\1</b></dt><dd>\\2</dd>',all)
        return all

##画像回り込み
def f_wrap(m):
        side = ""
        widthdata = "<!-- wrapfigure_memo:" + m.group(2) + "-->\n"
```

```
        if m.group(1) == "l":
                side = "left"
        elif m.group(1) == "r":
                side = "right"
        all = m.group(3)
        all = re_tag_img.sub('<img\\1 align="'+ side + '">',all)
        #all += '<br clear=all">\n' 回り込みクリア用
        return all

## 強調
def f_emph(m):
        all = m.group(1)
        if all.count('<em>') == 1 and all.count('<strong>'):
                all = all.replace("em","strong")
        all = "<em>"+ all + "</em>"
        return all

## 注
def f_note(m):
    global note_num,note_data
    note_num += 1
    num = str(note_num)
    note_data += '<a href="#note'+ num +'to" name="note'+ num+'">(' + num + ')</a> ' +m.group(1) + '<br>\n'
    number = '<sup><a href="#note'+ num +'" name="note'+ num +'to">(注'+ num +')</a></sup>'
    return number

## \raisebox{}{}                -> 内容が{}されていないものは未対応
def f_raisebox(m):
        num = f_tani(m.group(2))
        if m.group(1) == "-":
                num = "-" + num
        text = m.group(3)
        all = '<span style="position:relative;bottom:'+ num +';">'+ text +'</span>'
        return all

## rule[]{}{}      -> ブロック要素に
def f_rule(m):
        set = ""
        if m.group(1)!=None:
```

```
381:                set = " bottom:" + m.group(2) + f_tani(m.group(3)) + ";"
382:        width = f_tani(m.group(4))
383:        height = f_tani(m.group(5))
384:        all = '<hr class="rule" style="position:relative;'+ set +' width:'+ width +';
    height:'+ height +';">'
385:        return all
386:
387: def f_tani(num):       # 単位変換 zw,zh -> pt
388:        global zwh
389:        tani = num[-2:]
390:        num = num[:-2]
391:        num = re_zero.sub("0.",num)    # .[0-9]対応
392:        if tani == "zw":
393:                tani = "pt"
394:                num = str(float(num) * zwh)
395:        elif tani == "zh":
396:                tani = "pt"
397:                num = str(float(num) * zwh)
398:        all = num + tani
399:        return all
400:
401: ### 空白#####################
402: ## \vspace{}
403: def f_vspace(m):
404:        global zwh
405:        num = m.group(2)
406:        item = ""
407:        if num == "":
408:                item = '<div class="vsn"> </div>'
409:        else:
410:                if re.search('^\.',num):
411:                        num = "0" + num
412:                num = str(float(num) * zwh)
413:                if m.group(1) == "-":
414:                        num = "-" + num
415:                else:
416:                        item = '<div style="line-height:'+ num +'pt;"> </div>'
417:        return item
418:
419: ## \hspace{}
420: def f_hspace(m):
```

```
421:         global zwh
422:         num = m.group(1)
423:         tani = m.group(2)
424:         if tani == "zw":
425:                 num = str(float(num) * zwh)
426:                 tani = "pt"
427:         elif tani == "zh":
428:                 print "未対応\n hspase:"+ tani
429:         if re.search('^\.',num):
430:                 num = "0" + num
431:         item = '<span style="letter-spacing:'+ num + tani +';"> </span>'
432:         return item
433: 
434: def f_hspacem(m):
435:         global zwh
436:         num = m.group(1)
437:         tani = m.group(2)
438:         if tani == "zw":
439:                 num = str(float(num) * zwh)
440:                 tani = "pt"
441:         if re.search('^\.',num):
442:                 num = "0" + num
443:         item = '<span style="position:relative; left:-'+ num + tani +';">' + m.group(3) +
    '</span>'# + m.group(4)
444:         print m.group(3)+"\n\n"
445:         #item = '<span style="position:relative; left:-'+ num + tani +'; letter-spacing:-'+
    num + tani +';">' + m.group(3) + '</span>'# + m.group(4)
446:         return item
447: 
448: ## \section ##############################
449: def f_section(data):
450:         data = re_part.sub(f_prepart,data)
451:         data = re_section.sub(f_presec, data)
452:         data = re_subsection.sub(f_presubsec, data)
453:         data = re_subsubsection.sub(f_presubsubsec, data)
454:         data = re_paragraph.sub(f_preparag,data)
455:         if re_part.search(data):
456:                 data = re_par.sub(f_part,data)
457:         else:
458:                 data = re_sec.sub(f_sec,data)
459:         return data
```

```
460:
461: ####section事前処理
462: def f_prepart(m):
463:     all = '</p>\n<!--endpart-->\n' + m.group(0)
464:     return all
465: def f_presec(m):
466:     all = '</p>\n<!--endsec-->\n' + m.group(0)
467:     return all
468: def f_presubsec(m):
469:     all = '</p>\n<!--endsubsec-->\n' + m.group(0)
470:     return all
471: def f_presubsubsec(m):
472:     all = '</p>\n<!--endsubsubsec-->\n' + m.group(0)
473:     return all
474: def f_preparag(m):
475:     all = '</p>\n<!--endparag-->\n' + m.group(0)
476:     return all
477: ####section本処理
478: def f_sub2(m):
479:     global subsubsection,sec3,sec4
480:     subsubsection += 1
481:     sec4 = sec3 + '.' + str(subsubsection)
482:     all = m.group(0)
483:     all = re_subsubsection.sub('<h4 class="subsubsection">' + sec4 +' \\1</h4>
    . \n<p>\n',all)
484:     #all = re_subsubsection.sub(f_sub2,all)
485:     return all
486:
487: def f_sub(m):
488:     global subsection, subsubsection ,sec2,sec3,contents
489:     subsection += 1
490:     subsubsection = 0
491:     sec3 = sec2 + '.' + str(subsection)
492:     all = m.group(0)
493:     all = re_subsection.sub('<h3 class="subsection"><a name="m'+ sec3 +'">' + sec3
    . +' \\1</a></h3>\n<p>\n',all)
494:     text = (re_midashi3.search(all)).group(1)
495:     contents += '<p> <a href="#m'+ sec3 +'">'+ text +'</a></p>\n'
496:     all = re_subsubsection.sub(f_sub2,all)
497:     return all
498:
```

```
499: def f_sec(m):
500:     global section,subsection,sec1,sec2,contents
501:     section += 1
502:     subsection = 0
503:     all = m.group(0)
504:     sec = m.group(1)
505:     sec2 = sec1 + str(section)
506:     all = re_section.sub('<h2 class="section"><a name="m'+ sec2 +'">' + sec2 + ' \
    . \1</a></h2>\n<p>\n',all)
507:     text = (re_midashi2.search(all)).group(1)
508:     contents += '<p><a href="#m'+ sec2 +'">'+ text +'</a></p>\n'
509:     all = re_sub.sub(f_sub,all)
510:     return all
511:
512: ####part処理
513: def f_part(m):
514:         global part,sec1
515:         all = m.group(0)
516:         text = m.group(1)
517:         part += 1
518:         all = re_part.sub('<h1 class="part">'+ str(part) +' \\1</h1>\n<p>\n',all)
519:         sec1 = str(part) + "."
520:         all = re_sec.sub(f_sec,all)
521:         return all
522:
523: ############################################################
524: ## \minibox ######################
525: def f_page(m):
526:     all = m.group(0)
527:     if re_pcome.search(all):
528:             return all
529:     else:
530:             style = ''
531:             width = m.group(3)
532:             if m.group(2) == 'boxed':
533:                     style = ' border:1px solid black;'
534:             mae = ""
535:             ato = ""
536:             if m.group(1)!=None:
537:                 mae = m.group(1)
538:             if m.group(6)!=None:
```

```
539:                 ato = m.group(6)
540:             tag = mae +'<div class="box" style="width:' + width +'%;' + style +'">' +
    m.group(4) + '</div>'+ ato
541:             return tag
542:
543: def f_pagew(m):
544:     all = m.group(0)
545:     cont = m.group(3)
546:     if re_pagepart.search(cont):
547:             if re_page.search(all):
548:                 all = re_page.sub(f_page,all,1)
549:             return all
550:     cont = m.group(7)
551:     if re_pagepart.search(cont):
552:             if re_page.search(all):
553:                 all = re_page.sub(f_page,all,1)
554:             return all
555:     width1 = m.group(2)
556:     width2 = m.group(6)
557:     style1 = ""
558:     style2 = ""
559:     nonpre = " padding:10px"
560:     if m.group(1) == 'boxed':
561:         style1 = ' border:1px solid black;'
562:     if m.group(5) == 'boxed':
563:         style2 = ' border:1px solid black;'
564:         if not re_prev.search(m.group(7)):
565:             style2 += 'margin-top:5px; padding:10px;'
566:             width2 = str(float(width2) -2.25)
567:     tag1 = '<div style="width:' + width1 +'%; float:left;' + style1 + '">\n' + m.group(3)
    + '\n</div>\n'
568:     tag2 = '<div style="width:' + width2 +'%; float:left;' + style2 + '">\n' + m.group(7)
    + '\n</div>\n'
569:     cont = '<div class="box">' + tag1 + tag2 + '<br style="clear:both;"></div>\n'
570:     return cont
571:
572: ### 表 ##############################################
573: def f_table(m):
574:     global table_no,td_no ,test1
575:     table_no += 1
576:     td_no = 1
```

```
577:     set = m.group(1)
578:     all = m.group(2)
579:     table = '<table class="table'+ str(table_no) +'">\n'
580:     table += f_test(all,set)
581:     table += '</table>\n'
582:     return table
583: 
584: def f_tablex(m):
585:         global tdwidth
586:         tdwidth = []
587:         all = m.group(0)
588:         all = re_tablexp.sub(f_tablexp,all)
589:         table = re_tablex2.sub(f_tablex2,all)
590:         return table
591: 
592: def f_tablexp(m):
593:         global tdwidth
594:         p = m.group(1)
595:         tdwidth.append(p)
596:         return
597: 
598: def f_tablex2(m):
599:         global table_no,td_no
600:         table_no += 1
601:         td_no = 1
602:         width = m.group(1)
603:         set = m.group(2)
604:         all = m.group(3)
605:         if width[-2:] == 'cm':
606:                 width = width[:-2]
607:                 width = float(width) * 40
608:                 width = str(width) +"px"
609:         table = '<table class="table'+ str(table_no) +'" width="'+ width +'">\n'
610:         table += f_test(all,set)
611:         table += '</table>\n'
612:         return table
613: 
614: def f_test(all,set):
615:         global table_no, td_no, tr_no
616:         linum = 0
617:         tr_no = 1
```

```
618:         style_yoko = []
619:         style = ""
620:         trdata = ""
621:         style_tate = f_tate(set)
622:         # 横
623:         style_st = 'border-top:1px solid black;'
624:         style_sb = ' border-bottom:1px solid black;'
625:         style_dt = 'border-top:2px solid black;'
626:         style_db = ' border-bottom:2px solid black;'
627:         all = re_bb.sub("</td></tr>\n||",all)
628:         line = all.split("||")
629:         for it in line:
630:                 linum += 1
631:                 if re_thlinew.search(it):     #二重線
632:                         style_t = style_dt
633:                         style_b = style_db
634:                         if re_trtds.search(it):
635:                                 style = style_t
636:                                 style_yoko.append(style)
637:                                 td_no = 1
638:                                 it = re_tdand.sub(f_tdand,it)
639:                                 trdata += '<tr class="t' + str(table_no) +'tr'+ str(tr_no) +'">
    \n<td class="t' +str(table_no)+'td1">'+ it
640:                                 tr_no += 1
641:                         else:
642:                                 if linum == 1:
643:                                         style = style_t
644:                                         style_yoko.append(style)
645:                                 else:
646:                                         style = style_yoko.pop()
647:                                         style += style_b
648:                                         style_yoko.append(style)
649:                 elif re_thline.search(it):     #直線
650:                         style_t = style_st
651:                         style_b = style_sb
652:                         if re_trtds.search(it):
653:                                 style = style_t
654:                                 style_yoko.append(style)
655:                                 td_no = 1
656:                                 it = re_tdand.sub(f_tdand,it)
657:                                 trdata += '<tr class="t' + str(table_no) +'tr'+ str(tr_no) +'">
```

```
\n<td class="t' +str(table_no)+'td1">'+ it
                        tr_no += 1
                else:
                        if linum == 1:
                                style = style_t
                                style_yoko.append(style)
                        else:
                                style = style_yoko.pop()
                                style += style_b
                                style_yoko.append(style)
        trdata = re_thline.sub("",trdata)
        re_dusttable = re.compile(r'[\n\s]*(</?td.+?>)[\n\s]*')
        trdata = re_dusttable.sub('\\1',trdata)
        #横連結
        trdata = re_mcolumn.sub(f_mcolumn,trdata)
        css = ''
        i = 1
        for j in style_yoko:
                css += 'table tr.t' +str(table_no)+'tr' + str(i) +' td{\n\t'+ j +'\n}\n'
                i += 1
        i = 1
        for j in style_tate:
                css += 'table td.t' +str(table_no)+'td' + str(i) +'{\n\t'+ j +'\n}\n'
                i += 1
        ss = open("output/style_table_"+ file_name[:-4]+".css","a")
        ss.write(css)
        ss.close()
        return trdata

def f_tdand(all):   ###&->td
    global table_no, td_no
    td_no += 1
    tag = '</td><td class="t' +str(table_no)+'td'+ str(td_no) +'">'
    return tag

def f_tate(all):   ###縦線lcrの処理
    sord = re.compile(r'(sld)')
    clr = re.compile(r'(cllr)')
    all = all.replace('ll','d')
    all = all.replace('l','s')
    all = all.replace(' ','')
```

```
698:     solid = 'border-left:1px solid black; '
699:     duble = 'border-left:2px solid black; '
700:     line = 0
701:     style_data = []
702:     k = 0
703:     for item in all:
704:         style = ''
705:         if item == 's':
706:             line = 1
707:         elif item == 'd':
708:             line = 2
709:         else:
710:             if line == 1:
711:                 style = solid
712:                 line = 0
713:             elif line == 2:
714:                 style = duble
715:                 line = 0
716:             if item == 'c':
717:                 style += 'text-align:center;'
718:             elif item == 'r':
719:                 style += 'text-align:right;'
720:             else:
721:                 style += 'text-align:left;'
722:                 if item == 'p':
723:                   global tdwidth
724:                   style += ' width:'+ tdwidth[k]
725:                   k += 1
726:             style_data.append(style)
727:     if line != 0:
728:         style = style_data.pop()
729:         if line == 1:
730:             style += ' border-right:1px solid black;'
731:         else:
732:             style += ' border-right:2px solid black;'
733:         style_data.append(style)
734:     return style_data
735: 
736: def f_mcolumn(m):
737:         sel = m.group(2)
738:         set = m.group(3)
```

```
739:        text = m.group(4)
740:        class1 = m.group(1)
741:        if m.group(5) == "</tr>":
742:                end = m.group(5)
743:        else:
744:                re_tdclass = re.compile(r'(t[0-8]+td)([0-9]+)')
745:                sel2 = re_tdclass.sub('\\2',class1)
746:                sel2 = int(sel2) + int(sel)
747:                class2 = m.group(6)
748:                class2 = re_tdclass.sub('\\1', class2)
749:                class2 = class2 + str(sel2)
750:                end = '<td class="'+ class2 +'">'
751:        set = set.replace('ll','d')
752:        set = set.replace('l','s')
753:        set = set.replace(' ','')
754:        style = 'style="'
755:        i = 0
756:        for item in set:
757:                i += 1
758:                if i == 1:
759:                        if item == 's':
760:                                style += 'border-left:1px solid black; '
761:                        elif item == 'd':
762:                                style += 'border-left:2px solid black; '
763:                        else:
764:                                style += 'border-width-left:0px; border-style-left:none;'
765:                elif i == 2:
766:                        if item == 'l':
767:                                style += 'text-align:left;'
768:                        elif item == 'c':
769:                                style += 'text-align:center;'
770:                        else:
771:                                style += 'text-align:right;'
772:                else:
773:                        if item == 's':
774:                                style += ' border-right:1px solid black;'
775:                        elif item == 'd':
776:                                style += ' border-right:2px solid black;'
777:                        else:
778:                                style += 'border-width-right:0px; border-style-left:none;'
779:        style += '"'
```

```
780:         tag = '<td class="'+ class1 +'" colspan="'+ str(sel)+'" '+ style +'>' + text + '</
    . td>'+ end
781:         return tag
782:
783: def f_mrow(m):
784:     sel = m.group(1)
785:     set = m.group(2)
786:     text = m.group(3)
787:     tag = '<td rowspan="'+ str(sel)+'">' + text
788:     return tag
789:
790: ####################################
791: ## \newcommand{}{}
792: def f_newcommand(m):
793:         global newcombefor, newcomafter
794:         befor = "{" + m.group(1) +"}"
795:         after = m.group(2)
796:         after = re_dustblank.sub("",after)
797:         after = re_tag_comment.sub("",after)
798:         newcombefor.append(befor)
799:         newcomafter.append(after)
800:         return
801:
802: ## title
803: def f_maketitle(m):
804:         global title,subtitle,author,date
805:         titles = '<div class="maketitle">\n<h1>%s<h1>\n' %(title)
806:         if subtitle != "":
807:                 titles += '<h2>%s</h2>\n' %(subtitle)
808:         titles += '<h3>%s</h3>\n<h4>%s</h4>\n</div>\n' %(author,date)
809:         return titles
810:
811: def f_title2(title,subtitle):#ヘッダー用
812:         title2 = re_tag.sub("",title)
813:         if subtitle != "":
814:                 subtitle2 = re_tag.sub("",subtitle)
815:                 title2 = title2 + subtitle2
816:         # 表記処理
817:         title2 = title2.replace("LATEX","LaTeX")
818:         title2 = title2.replace("TEX","TeX")
819:         return title2
```

```
820:
821: def f_acbody(m):     #<body>以下<pre>以外
822:         all = m.group(0)
823:         ## ソース整備
824:         all = re_br.sub("\n<br>\n",all)          # 2個以上連続する\n-><br>
825:         all = all.replace("<p> </p>","") # 空白段落を削除
826:         all = re_nonbr1.sub("\\1",all)   # 段落タグ前後の<br>の処理
827:         all = re_nonbr2.sub("\\1",all)
828:         ## ごみ処理
829:         all = re_dustcoment.sub("", all)        # 仮設コメントの削除
830:         #改行コード処理
831: #       all = all.replace("\n","\r\n")      #Linux -> Windows & Mac
832:         return all
833:
834: ## プリアンブル処理
835: def f_preamble(m):
836:         pre = file_name + "\n---------------------------------\n"
837:         pre += m.group(3) + "\n---------------------------------\n\n"
838: #       pre = pre.replace("\n","\r\n")   #Linux -> Windows & Mac
839:         pf = open("output/pre_"+ file_name[:-4]+".txt", "w")
840:         pf.write(pre)
841:         pf.close()
842:         return m.group(0)
843:
844: ############################################################
845: # EasyLayout
846:
847: ###文献表
848: def f_bunken(m):
849:         global contents
850:         all = m.group(1)
851:         all = all.replace('\\tume',"")
852:         all = re_item3.sub("<dt>\\1</dt><dd>\\3(\\2)</dd>",all)
853:         all = re_item4.sub("<dt>\\1</dt><dd>\\2</dd>",all)
854:         all = re_item.sub("<dt>\\1</dt><dd></dd>",all)
855:         all = "<h3><a name=\"bunken\">文献表</a></h3>\n<dl>"+ all +"</dl>"
856:         contents += '<p><a href="#bunken">文献表</a></p>\n'
857:         return all
858:
859:
860: ############################################################
```

```
861: ##########################################################
862: ## 実行ブロック
863: ##########################################################
864: ## 事前処理
865: def f_preaction(data):
866:         data = re_pream.sub(f_preamble,data,1)
867:         ##<pre><code>
868:         while re_pers.search(data):
869:                 m = (re_pers.search(data)).group(2)
870:                 if m == "verb":
871:                         data = re_verb.sub(f_verb1,data,1)
872:                 elif m == "verbatim":
873:                         data = re_verbatim.sub(f_verbatim,data,1)
874:                 elif m == "Verbatim":
875:                         data = re_vverbatim.sub(f_verbatim,data,1)
876:                 else:
877:                         print "ERROR:verbatim->" + m
878:         ##<!---->
879:         data = re_comment.sub(f_comment, data)
880:         ##記号
881:         data = data.replace('\\$','&#36;')
882:         data = data.replace('\\&','&amp;')
883:         data = data.replace('\\%','&#37;')
884:         data = data.replace('\\#','&#35;')
885:         data = data.replace('\\_','&#95;')
886:         data = data.replace('\\{','&#123;')
887:         data = data.replace('\\}','&#125;')
888:         data = data.replace('$\\backslash$','&#47;')
889:         data = data.replace("{\\char'176}",'&#94;')#~
890:         data = data.replace("{\\char'136}",'&#126;')#^
891:         data = data.replace('$<$','&lt;')
892:         data= data.replace('$>$','&gt;')
893:         ##HTML形成
894:         data = re_class.sub(f_header, data)
895:         data = re_b_document.sub(b_start,data)
896:         data = re_e_document.sub(b_end,data)
897:         return data
898:
899: def f_action1(data):
900:         global tex,latex,tex,today
901:         ## 単純処理
```

```
902:         ## TeXロゴ ####################
903:         data = re_latexe.sub('<span class="logo_tex">' + latexe + '</span>',data)
904:         data = re_latex.sub('<span class="logo_tex">' + latex + '</span>',data)
905:         data = re_tex.sub('<span class="logo_tex">' + tex + '</span>', data)
906:         ## 記号
907:         data = data.replace("'''",'"')
908:         data = data.replace('\\ldots','...')
909:         data = data.replace('\\dag','&#134;')
910:         data = data.replace('\\ddag','&#135;')
911:         data = data.replace('\\S','&#167;')
912:         data = data.replace('\\P','&#182;')
913:         data = data.replace('\\copyright','&#169;')
914:         data = data.replace('\\pounds','&#163;')
915:         data = data.replace('\\textregistered','&#174;')
916:         data = data.replace('\\texttrademark','&#153;')
917:         data = data.replace('\\textasciitilde','〜')
918:         ## \today
919:         data = data.replace('\\today',today)
920:         ## 改ページ -> <hr>
921:         data = re_newpage.sub('<hr class="newpage">',data)
922:         #スペース制御
923:         data = re_onesp.sub(" ",data)
924:         #data = re_nonsp.sub("",data)
925:         #その他
926:         data = data.replace('{[','[')
927:         data = data.replace(']}',']')
928:         data = data.replace('','')
929:         return data
930: 
931: def f_action2(data):
932:         ###### 引数を取る
933:         ## 文字サイズ
934:         data = re_huge1.sub('<span class="huge1">\\1</span>',data)
935:         data = re_huge2.sub('<span class="huge2">\\1</span>',data)
936:         data = re_large1.sub('<span class="large1">\\1</span>',data)
937:         data = re_large2.sub('<span class="large2">\\1</span>',data)
938:         data = re_large3.sub('<span class="large3">\\1</span>',data)
939:         data = re_normalsize.sub('<span class="normalsize">\\1</span>',data)
940:         data = re_small.sub('<span class="small">\\1</span>',data)
941:         data = re_footnotesize.sub('<span class="footnotesize">\\1</span>',data)
942:         data = re_scriptsize.sub('<span class="scriptsize">\\1</span>',data)
```

```
943:         data = re_tiny.sub('<span class="tiny">\\1</span>',data)
944:         ## 文字の修飾 ###############
945:         data = re_t_df.sub('<span class="t_df">\\2</span>',data)
946:         data = re_t_bf.sub("<b>\\1</b>", data)
947:         data = re_t_it.sub('<i>\\1</i>', data)
948:         data = re_t_sl.sub('<span class="t_sl">\\1</span>',data)
949:         data = re_t_sc.sub('<span class="t_sc">\\1</span>',data)
950:         data = re_t_sf.sub('<span class="t_sf">\\1</span>',data)
951:         data = re_t_tt.sub("<tt>\\1</tt>",data)
952:         data = re_u_line.sub('<u>\\2</u>',data)
953:         ## color ###############
954:         data = re_color.sub("<span style=\"color:\\1;\">\\2</span>",data)
955:         ## 画像 #################
956:         data = re_img.sub('<img src="img/\\3" alt=""><!--img_memo:\\1-->',data)
957:         return data
958: 
959: def f_action3(data):
960:         ###### 引数を取る&内包する
961:         data = re_emph.sub(f_emph,data)    ## emph
962:         data = re_makebox.sub('<div class="makebox">\\2</div>',data)    ##
    . \makebox
963:         data = re_footnote.sub(f_note,data)## \footnote
964:         data = re_raisebox.sub(f_raisebox,data)   ## \risebox
965:         data = re_raisebox.sub(f_raisebox,data)
966:         data = re_raisebox2.sub(f_raisebox,data)
967:         data = re_rule.sub(f_rule, data) ## \rule
968:         data = f_section(data)      ## \section
969:         return data
970: 
971: def f_blank(data):
972:         ###### \vspace, \hspace
973:         data = re_vspace.sub(f_vspace,data)
974:         data = re_vspace.sub(f_vspace,data)
975:         data = re_hspace.sub(f_hspace,data)
976:         data = re_hspacem.sub(f_hspacem,data)
977:         data = re_vspace.sub(f_vspace,data)
978:         data = re_vspace.sub(f_vspace,data)
979:         data = re_hspace.sub(f_hspace,data)
980:         data = re_hspacem.sub(f_hspacem,data)
981:         return data
982: 
```

```
983: ## 環境
984: def f_env(m):
985:         data = m
986:         #寄せ
987:         data = re_center.sub('<div class="center">\\1</div>\n', data)
988:         data = re_flushright.sub('<div class="flushright">\\1</div>\n', data)
989:         data = re_flushleft.sub('<div class="flushleft">\\1</div>\n', data)
990:         data = re_quotation.sub('<blockquote>\\1</blockquote>\n', data)
991:         #箇条書き
992:         data = re_itemize.sub(f_itemize,data)
993:         data = re_enumerate.sub(f_enumerate,data)
994:         data = re_description.sub(f_description,data)
995:         #画像の回り込み
996:         data = re_wrapfigure.sub(f_wrap,data)
997:         #表
998:         data = re_table.sub(f_table,data)
999:         data = re_tablex.sub(f_tablex,data)
1000:         #minibox
1001:         data = re_w.sub(f_pagew, data)
1002:         data = re_w.sub(f_pagew, data)
1003:         data = re_page.sub(f_page,data)
1004:         return data
1005:
1006: ############################################################
1007: ## 実行処置
1008: ############################################################
1009: ## 事前処理
1010: data = f_preaction(data) ## verb % 特殊記号 HTML形成
1011: ## ごみ処理 前
1012: data = re_dust1.sub('',data)
1013: data = re_dust2.sub('',data)     #EasyLayout用コマンド
1014: data = data.replace('\\vspace*{-\\intextsep}', "")    #ゴミ処理?
1015: ## 未対応処理    data = data.replace('','')
1016: data = data.replace('\\hfill',"") # -> 横方向バネの空白
1017: data = data.replace('\\vspace\{\\fill\}',"")       # -> 縦方向バネの空白
1018:
1019: i = 0
1020: while i < 5:
1021:         data = f_action1(data)     ##TeXロゴ、\today,\newpage
1022:         data = f_action2(data)     ##文字サイズ&修飾, 画像
1023:         data = f_action3(data)     ##\makebox, \footnote
```

```
1024:         data = f_blank(data) ##空白
1025:         data = f_env(data)    ## 環境
1026:         data = f_env(data)
1027:         ## コマンド定義
1028:         data = re_newcommand.sub(f_newcommand,data)
1029:         for befor,after in zip(newcombefor,newcomafter):
1030:                 befor = befor.replace("{","")
1031:                 befor = befor.replace("}","")
1032:                 data = data.replace(befor,after)
1033:         i += 1
1034:
1035: ###############################################################
1036: ###############################################################
1037: ## title
1038: if re_title.search(data):
1039:         title = (re_title.search(data)).group(1)
1040: if re_subtitle.search(data):
1041:         subtitle = (re_subtitle.search(data)).group(1)
1042:         subtitle = " — "+ subtitle +" — "
1043: if re_author.search(data):
1044:         author = (re_author.search(data)).group(1)
1045:         author = author.replace("\\and "," ")
1046: if re_date.search(data):
1047:         date =(re_date.search(data)).group(1)
1048: #maketitle
1049: data = re_maketitle.sub(f_maketitle,data)
1050: #<title>
1051: data = data.replace('<title></title>','<title>'+ f_title2(title,subtitle) +'</title>')
1052: ###################################
1053: ##EasyLayout 文献表
1054: data = re_bunken.sub(f_bunken,data)
1055: ## 挿入部分 ######################
1056: end = '<div class="note">\n'
1057: # 注
1058: if note_data != '<br><hr>\n<h3><a name="thu">注</a></h3>\n<p class="thu">\n':
1059:         note_data += "</p>\n"
1060:         end += note_data
1061:         contents += '<p><a href="#thu">注釈</a></p>\n'
1062: end += '</div>\n'
1063: data = data.replace('</body>',end + '</body>')
1064: ##目次
```

```
1065: data = re_mokuji.sub(contents,data)
1066: ############################################################
1067: ############################################################
1068: ## HTML体裁
1069: #プリアンブル処理
1070: data = re_preamble.sub("</title></head>",data)
1071: #ごみ処理後
1072: data = re_area.sub(f_acbody,data) #<body>以下<pre>以外
1073: ############################################################
1074: ## 出力
1075: #print data
1076: h = open("output/"+ htmlfile,"w")
1077: h.write(data)
1078: h.close()
1079:
```

1079:
1080: 対応コマンド一覧
1081:
1082: \documentclass[]{jsarticle}
1083: \verb
1084: %
1085: \title{}
1086: \subtitle{}
1087: \author{}
1088: \and
1089: \date
1090: \today
1091: \maketitle
1092: \parindent
1093: \@
1094: \,
1095: ~
1096: \\
1097: \par
1098: \TeX
1099: \LaTeX
1100: \LaTeXe
1101: \newpage
1102: {\Huge }
1103: {\huge }
1104: {\LARGE }
1105: {\Large }
1106: {\large }
1107: {\normalsize }
1108: {\small }
1109: {\footnotesize }
1110: {\scriptsize }
1111: {\tiny }
1112: \textup{}
1113: \textmd{}
1114: \textrm{}
1115: \textbf{}
1116: \textit{}
1117: \textsl{}
1118: \textsc{}
1119: \textsf{}

```
\texttt{}
\underline{}
\Underline{}
\emph{}
\includegraphics[]{}
\raisebox{}{}
\rule[]{}
\vspace{}
\vspace*{}
\hspace{}
\hspace*{}
\makebox[]{}
\part{}
\section{}
\subsection{}
\subsubsection{}
\footnote{}
\newcommand{}{}
\item
\$
\&
\%
\#
\_
\{
\}
$\backslash$
{\char'136}
{\char'176}
$<$
$>$
\hline
\multicolumn{}{}{}
\multirow{}{}{}
p{}
''
\ldots
\dag
\ddag
\S
\P
```

```
\copyright
\pounds
\textregistered
\texttrademark
\textasciitilde
{[
]}
\begin{document}\end{document}
\begin{verbatim}\end{verbatim}
\begin{Verbatim}\end{Verbatim}
\begin{center}\end{center}
\begin{flushright}\end{flushright}
\begin{flushleft}\end{flushleft}
\begin{quotation}\end{quotation}
\begin{itemize}\end{itemize}
\begin{enumerate}\end{enumerate}
\begin{description}\end{discription}
\begin{wrapfigure}{}{}\end{wrapfigure}
\begin{minipage}{}\end{minipage}
\begin{boxedminipage}\end{boxedminipage}
\begin{tabular}{}\end{tabular}
\begin{tabularx}{}\end{tabularx}
{\color{} }
\tableofcontents

パッケージ EasyLayout
\後注
\タイトルページ
\目次
\begin{文献表}\Year{}{}\end{文献表}
```