latex.ltx リーディング
第 1 回資料

東大 TEX 愛好会

2015 年 4 月 24 日（2015 年 5 月 18 日 版）

1　TEX のロゴたち

はじめに，ltlogos.dtx に由来するロゴ関連のコマンドの定義を記しておく．

```
1327 \def\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
1328 \DeclareRobustCommand{\LaTeX}{L\kern-.36em%
1329   {\sbox\z@ T%
1330     \vbox to\ht\z@{\hbox{\check@mathfonts
1331       \fontsize\sf@size\z@
1332       \math@fontsfalse\selectfont
1333       A}%
1334     \vss}%
1335   }%
1336   \kern-.15em%
1337   \TeX}
1338 \DeclareRobustCommand{\LaTeXe}{\mbox{\m@th
1339   \if b\expandafter\@car\f@series\@nil\boldmath\fi
1340   \LaTeX\kern.15em2$_{\textstyle\varepsilon}$}}
```

本題に入る前に断っておくと，今回は上記定義に登場するフォント関連の部分についてはあまり深く立ち入らないことにする．したがって，TEX の箱を用いた高さの調整が本節の最重要テーマとなる．

1.1　TEX 出力コマンド

\TeX の定義は（おそらく，伝統に従って）\def で行われている[*1]．その内部定義は極めてシンプルであり，ほとんど説明を必要としないであろう．念のため，いくつかのことを箇条書きで指摘しておく．

- \kern は任意長の空白を挿入するプリミティブ
- \lower は引き続く箱を任意長垂直方向に下げるプリミティブ[*2]
- \@ は latex.ltx の 1307 行目に \def\@{\spacefactor\@m} と定義されている

■疑問 1　\@ の存在意義は何か．

> \TeX の直後にピリオドが来たときに，通常の文間に挿入されるのと同じ量の空白を挿入させる．

[*1] ただし，実際の文書で \meaning\TeX と展開すると \protect\TeX となることから，別の場所で定義の書き換えが行われているらしい．

[*2] 垂直モードまたは数式モードでのみ使用可．したがって，続く \hbox 命令は必須．

1.2 LaTeX 出力コマンド

\LaTeX の定義部分のみを再掲する．

```
1328 \DeclareRobustCommand{\LaTeX}{L\kern-.36em%
1329   {\sbox\z@ T%
1330     \vbox to\ht\z@{\hbox{\check@mathfonts
1331       \fontsize\sf@size\z@
1332       \math@fontsfalse\selectfont
1333       A}%
1334     \vss}%
1335   }%
1336   \kern-.15em%
1337   \TeX}
```

\LaTeX の定義は \TeX よりも些か複雑である．また，toc 等への書き出し時にエラーとならないように，はじめから \DeclareRobustCommand で頑丈な制御綴として定義されている[*3]．

上の定義のうち LaTeX の L および TeX を出力する部分については何も難しくないと思うが，A を出力する部分（1329〜1335 行目）が少々問題である．ネストの外側から，順を追って見ていくことにする．

まず，はじめに登場する \sbox は \savebox の簡略化命令で，以下のような書式をもつ．

\sbox{〈箱名称〉}{〈内容〉}

これは文字通り「箱を保存する」ための命令で，上記 1329 行目では T という文字を入れた \z@ という名前の箱が保存されている[*4]．なお，この箱 \z@ は 1329〜1335 行目で局所的に定義されていることがわかる．

次に，1330〜1334 行目は縦箱（\vbox）の中に収められている．\vbox は

\vbox to〈長さ〉

の形で垂直方向の長さを指定することができるが，ここでは箱の高さを返すプリミティブ \ht を用いてさきほど保存した箱 \z@ の高さがその垂直方向の長さに指定されている．また，1334 行目にある \vss は垂直方向に無限に伸びるグルーを出力するプリミティブである．これにより LaTeX の A（を含む横箱）は T の高さを突き破らない最も高い位置に出力されることになる．

最後に，\hbox の中身は（最後の A を除いて）すべてフォントに関する指定を行っている．TeX/LaTeX のフォント選択機構はかなり厄介なものであるので今回は深入りしないことにするが，ごく簡単に説明すると，\fontencoding, \fontfamily, \fontseries, \fontshape, \fontsize でそれぞれの指定を行った後に，最後に \selectfont を宣言するという形をとる（これは LaTeX 流の一法にすぎない）．

■疑問 2　なぜ，LaTeX のうち A だけが細かいフォント指定を必要とするのだろうか．

> 少なくとも，A だけが他の文字より小さいサイズで出力されなければならないという事情がある．

[*3] \DeclareRobustCommand の定義はかなり複雑なようなので，今回は取り扱わない．

[*4] 保存された箱は \usebox 命令で出力することができる

1.3 LaTeX 2$_\varepsilon$ 出力コマンド

\LaTeXe の定義部分のみを再掲する．

```
1338 \DeclareRobustCommand{\LaTeXe}{\mbox{\m@th
1339   \if b\expandafter\@car\f@series\@nil\boldmath\fi
1340   \LaTeX\kern.15em2$_{\textstyle\varepsilon}$}}
```

まず，\mbox 内冒頭の \m@th は latex.ltx481 行目に \def\m@th{\mathsurround\z@} と定義されている命令である．\mathsurround は数式モードから出入りする際に挿入されるグルーの長さを指定するプリミティブ[*5]なので，\m@th は数式モードとの境界にスペースを空けないようにする命令ということになる．

そして，一番理解するのが難しいのが 1339 行目の内容である．まず，\@car は latex.ltx の 580 行目に \def\@car#1#2\@nil{#1} と定義されている．\f@series は現在のフォントシリーズをアルファベット 1,2 文字で返す．特に，現在のフォントシリーズがボールドである場合にはである場合には，b または bx[*6]が返る．すなわち，\@car\f@series\@nil 全体は m（標準体の場合）または b （ボールド体の場合）に展開されるはずである．プリミティブの \if は続く 2 つの文字トークンの文字コードが等しいかどうかで条件分岐を行うので 1339 行目の if 文全体としては「もし，ボールド体で LaTeX 2$_\varepsilon$ が出力されようとしている場合は，数式用のボールド体で出力せよ」という指示を行っていることになる．因みに，\@nil はどこにも定義されていない制御綴であり，\@car においては単なる区切り記号として機能している（可変長引数へ対応するため）．

1340 行目の内容は説明することがほとんどない．さきほどの \m@th 命令が効いているので，2 と ε の間にグルーが挿入されてしまう心配はない．

■疑問 3　\LaTeXe の定義全体が \mbox で囲われているのはなぜか．

途中での改行抑止または数式モードで使用された場合への対策か．また，局所化の役割も担っている．

■疑問 4　\@car の名前の由来は何か．

Lisp 言語にリストの最初の要素を取り出す car 関数というものがあり，その由来は Contents of the Address part of the Register である．LaTeX の \@car もこれと同様であると考えられる．
また，\@car の定義の前後行に \@cons と \@cdr が定義されているが，これらも同様に Lisp 由来で，それぞれ construct と Contents of the Decrement part of the Register の略である．

[*5] プリミティブであることは間違いないが，レジスタのように使用できる模様．
[*6] bold extended の略．

2 LaTeX におけるループ機能の実装

まず，\loop および \repeat の定義を示す．ltdirchk.dtx 由来となっている．

```
101 \def\loop#1\repeat{\def\iterate{#1\relax\expandafter\iterate\fi}%
102   \iterate \let\iterate\relax}
103 \let\repeat\fi
```

なお，450 行目からも，全く同じ内容で再度 \def により定義がなされている．こちらは ltplain.dtx 由来の定義である．

```
450 \long\def \loop #1\repeat{%
451   \def\iterate{#1\relax % Extra \relax
452               \expandafter\iterate\fi
453               }%
454   \iterate
455   \let\iterate\relax
456 }
457 \let\repeat=\fi
```

複数の.dtx ファイルで別々に \loop を定義しているにも関わらず，latex.ltx として 1 つのファイルに纏めてしまったため，このような状況になっているようだ．

2.1 \loop の挙動

さて，この中身についてだが，以下では次のような例を用いてコードを理解していきたいと思う．

```
\newcount\hoge
\loop
A
\advance\hoge1
\ifnum\hoge<2\repeat
```

まず確認だが，\loop は \if および \repeat と組み合わせて使う．上の例では，1 行目で定義されたカウンタ \hoge が（\hoge の初期値は 0），4 行目の \advance\hoge1 において 1 ずつ増加するようにしてある．5 行目において，\hoge が 2 よりも小さい間は \repeat が作用し \loop の位置に実行が戻り，ふただび文字 A を出力し，\hoge が増加する．\hoge が 2 以上になると，\repeat が作用しなくなり作業が終了する．

それでは詳しい挙動を調べていこう．1 行目については，今回は特に気にかける必要はない．\hoge というカウンタを用意しただけである．

実行が 2 行目に移ったとき，TeX は \loop を展開し始める．いま，パターンマッチより，\loop#1\repeat の引数 #1 に相当する部分は，

```
A
\advance\hoge1
\ifnum\hoge<2
```

となるから，展開後，上例は以下のようになる．

```
\def\iterate{A\advance\hoge1\ifnum\hoge<2\relax\expandafter\iterate\fi}%
  \iterate \let\iterate\relax
```

次に，TeX は \def を展開する．この作業により \iterate に，

```
A\advance\hoge1\ifnum\hoge<2\relax\expandafter\iterate\fi
```

が代入される．そのまま次に \iterate が展開されるため，結局元のコードから

```
A\advance\hoge1
\ifnum\hoge<2\relax\expandafter\iterate\fi
\let\iterate\relax
```

まで変容したことになる（見やすいように，適宜改行を施した）．

さて，このコードの展開に入るが，まず序盤の A\advance\hoge1 は特に問題ない．文字 A が出力され，カウンタ \hoge が 1 増加しただけである．次に，\ifnum\hoge<2 が読み込まれる．これは真であるから，TEX は \else または \fi までのトークン列を展開し始める．まず \relax では何もせず，次に \expandafter を読み込み，\iterate の前に \fi を展開する．

■疑問 5　\expandafter\iterate\fi の挙動（TEX プリミティブである \fi を展開するとき，TEX はどのような動作をするのか）

（推察の域を出ないが）おそらく，\fi を先に展開する（読み込む）ことにより，TEX は \ifnum より始まる条件分岐節が終了したとみなし，\if 節の外に \iterate を展開すると思われる．従って，展開終了後のコードは次の通り．

```
A\advance\hoge1 % この行は展開終了
A\advance\hoge1\ifnum\hoge<2\relax\expandafter\iterate\fi
\let\iterate\relax
```

2 行目が，先ほどの \iterate を展開したものである．

再び 2 行目の展開に入る．A\advance\hoge1 を展開することで，文字 A を出力し，カウンタ \hoge の値は 2 となった．そこで，今回は \hoge<2 が偽であるため，TEX は \ifnum から \fi の部分までの展開をスキップする．最後に，一時的な入れ子として使用した \iterate に \relax をコピーして，動作が終了する．結果として，出力は

AA

となるはずである．

2.2　\let\repeat=\fi の意味

さて，最後に定義の 103 行目，\let\repeat=\fi について考えてみたいと思う．前節で見たように，\loop が展開される際，パターンマッチにより \repeat は取り除かれてしまうため，この一文の必要性は分かりにくい．しかし，この定義は次のような例で生きてくる．

```
\newcount\fuga
\iffalse % 必ず偽の場合を実行する命令. すなわち, このコードは 7行目まで無視される.
\loop
ABC
\advance\fuga1
\ifnum\fuga<2\repeat
\fi
```

この場合，2 行目で \iffalse を展開した TEX は，以降 \else または \fi を探しながら，それらに出会うまでトークンを展開せずに読み込んでいく．従って，今回は \loop が展開されないため，\repeat も除去されずに残ったままになる．しかし，6 行目で \ifnum を読み込んでしまった（展開はしていない）TEX は，1 階層下の条件分岐に突入したと判断し，以降は 2 回 \fi が登場するまで，全てのトークンを展開せずに読み込もうとしてしまう．

従って，\let\repeat=\fi と定義されていないと，\if の個数と \fi の個数が釣り合わずに，エラーとなってしまうのだ．この定義は，\loop が展開されない場合に必要となってくる．