latex.ltx リーディング
第 2 回資料

東大 TEX 愛好会

2015 年 5 月 1 日

1 LATEX 流カウンター入門（朝倉）

LATEX 流カウンターの機能は非常に重要である．たった 1 回のゼミでこれを網羅的に理解することは無理があるので，今回は LATEX 流カウンター周りの基本的な事項に絞って扱うことにする．

なお，本節の執筆にあたっては『LATEX 2_ε マクロ作法』（藤田眞作）を参考にした．

1.1 TEX 流カウンター

LATEX 流カウンターについて考察を始める前に，TEX 流カウンターについて簡単に説明しておく．

1.1.1 既製のカウンター

TEX には \count0 から \count9 までの 10 個のカウンターレジスターがユーザー用に用意されている．ただし，\count0 は LATEX 2_ε ではページ番号を格納するために使用されているのでユーザーが自由に利用することはできない．ほかのカウンターも種々のパッケージで使用されている可能性があるので無闇に使用しないほうが無難である．

1.1.2 カウンターの新設

TEX 流カウンターを新設するためには \newcount 命令を用いて，次のように宣言する．

\newcount\〈カウンター名〉

ここで，**\〈カウンター名〉** はバックスラッシュから始まる制御綴の形をしていることに注意せよ．\newcount により新設されるカウンターには TEX の内部では番号が割り当てられ，\count**〈割り当て番号〉** と **\〈カウンター名〉** は同義となる．割り当て番号は，\meaning**\〈カウンター名〉** とすることで確認できる．

また，次のようにすると割り当て番号を指定して TEX 流カウンターを新設することも可能である．

```
\countdef\TestCount=200
```

このとき \TestCount と \count200 は同義となる．

1.1.3 カウンターの基本操作

以下では，一般の TEX 流カウンターを \TeXcounter と表すことにする．

■代入 \TeXcounter に対して任意の数を代入するときには\TeXcounter=100 のように直接代入したい数字を等号で結んで指定すればよい．なお，この等号は省略することが可能であり，また間にスペースがあってもなくても同じ結果となる[*1]．

[*1] ただし，\TeXcounter が \count**〈割り当て番号〉**の形をしているときは間に等号またはスペースのいずれかが必要である．

■出力　\TeXcounter が保持している数値を出力させる場合には \the\TeXcounter または \number\TeXcounter とする．また，\romannumeral\TeXcounter とすると，カウンターの出力をローマ数字の形で得ることができる．

■加法と減法　TEX 流カウンターの足し算は以下のような形で行う．

```
\advance\TeXcounter by 3
```

\TeXcounter のあとの by および空白は省略できる．また，加算する数値を負の値にすることで引き算を行うことも可能である．

■乗法と除法　掛け算と割り算については \advance と同じ要領で，それぞれ \multiply と \divide を使用することで実現できる．

1.2　LATEX 流カウンターの基本操作コマンド

1.2.1　\newcounter 命令

LATEX 流カウンターを新設する際には，\newcounter を以下の書式で使用する．

\newcounter{ **〈カウンター名〉**}[**〈親カウンター〉**]

ここで**〈カウンター名〉**にはバックスラッシュがつかないことに注意せよ．また**〈親カウンター〉**は既存の LATEX 流カウンターの名前で，新設されるカウンターはここで指定された親カウンターに従属することになる（オプション）．カウンターの親子関係については，本資料の**??**節で解説する．

TEX 流カウンターの新設命令 \newcount はカウンターの作成機能しかもたないが，LATEX 流カウンターの新設命令は様々な機能をもっている．以下に，その機能を列挙する．

1. 指定された**〈カウンター名〉**と同名のカウンターがないかどうかチェックする．
2. **〈カウンター名〉**で指定された名前の LATEX 流カウンター（以下，LaTeXcounter）を新設する．
3. LaTeXcounter の現在値を出力するための \theLaTeXcounter を自動で定義．
4. オプションで**〈親カウンター〉**が指定された場合，LaTeXcounter をこれに従属させる．
5. 相互参照用の内部（TEX 流）カウンターを新設する（今回は詳しくは扱わない）．

さて，\newcounter の機能を確認できたところで，いよいよその定義をみていくことにする．

```
1819 \def\newcounter#1{%
1820   \expandafter\@ifdefinable \csname c@#1\endcsname
1821     {\@definecounter{#1}}%
1822   \@ifnextchar[{\@newctr{#1}}{}}
```

ご覧のとおり，@を含む LATEX の内部命令だらけで，これだけ見ても何が行われているのかよくわからない．先に，一般性の高いものを説明してしまう．

まず，\@ifdefinable は与えられた制御綴（ここでは c@#1）が定義可能なものであるかを判定し，「定義不可能」であればエラーを吐いて処理を終了する．そもそも「何が定義不可能なのか」ということも含め \@ifdefinable に関する詳細は本節では扱わない．次に \@ifnextchar は文字通り直後の第一引数（ここでは [）が続いている場合は第二引数の処理（ここでは{\@newctr{#1}}），続いていない場合は第三引数の処理（ここでは{}）が実行される．\@ifdefinable は今回の場合もそうであるように，オプションの有無で分岐を行う際によく用いられる便利な命令であるが，本節ではそのメカニズムまでは扱えない．

さて，これまでの説明から先に示した機能のうち，1. は 1820 行目，4. は 1822 行目が担っているだろうことが推測できるだろう．裏を返せば，残りの 3 機能についてはすべて 1821 行目にある \@definecounter が担っていることがわかる．では，次はこの \@definecounter の内部について詳しく調べていくことにする．

```
1834 \def\@definecounter#1{\expandafter\newcount\csname c@#1\endcsname
1835   \setcounter{#1}\z@
1836   \global\expandafter\let\csname cl@#1\endcsname\@empty
1837   \@addtoreset{#1}{@ckpt}%
1838   \global\expandafter\let\csname p@#1\endcsname\@empty
1839   \expandafter
1840   \gdef\csname the#1\expandafter\endcsname\expandafter
1841     {\expandafter\@arabic\csname c@#1\endcsname}}
```

念のため確認しておくと（1821 行目から）\@definecounter に渡される#1 は LaTeXcounter である．すると，\@definecounter の定義の冒頭（1834 行目）で \c@LaTeXcounter という名の TEX 流カウンターが \newcount 命令によって新設されていることが読み取れる．この \c@LaTeXcounter が，LATEX 流カウンター LaTeXcounter の実態である．そのため，LATEX 流カウンターを TEX 流カウンターに変換するコマンドである \value は latex.ltx の 1823 行目で以下のように定義されている．

```
1823 \def\value#1{\csname c@#1\endcsname}
```

\@definecounter の定義に戻ると，1835 行目では \setcounter を用いて LaTeXcounter に 0 が代入されている．

その下の 1836 と 1838 行目はかなり似ている．これらの末尾に登場する \@empty は 122 行目にて\def\@empty{}と定義されていることからわかるように「空の制御綴」であり[*2]，それぞれ \cl@LaTeXcounter と \p@LaTeXcounter を，制御綴の代入に用いられるプリミティブ \let で空にしている．ここで，\p@LaTeXcounter は相互参照に用いる内部（TEX 流）カウンターなので今回はこれ以上扱わない．一方，\cl@LaTeXcounter と 1836・1837 行目の内容はカウンターの親子関係に関するものであるので，次節で詳しく取り上げることにする．

最後に残った 1840・1841 行目は，LaTeXcounter の現在値を出力する \theLaTeXcounter 命令の定義である．\csname hoge\endcsname の形を用いているので \expandafter が大量に必要とされ読みずらくなっているが，#1 を LaTeXcounter に置き換え，通常のバックスラッシュから始まる制御綴の形で書くのであれば

```
\gdef\theLaTeXcounter{\@arabic\c@LaTeXcounter}
```

となるはずである．なお，\@arabic は「渡された TEX 流カウンターの値をアラビア数字で出力する制御綴」だが，1849 行目にある定義は\def\@arabic#1{\number #1}と素朴である．

■疑問 1　カウンター関連の制御綴の定義に \global（\gdef を含む）が頻出するのはなぜか．

> カウンターは原則として文書全体を通して一意的であることが望まれる．したがって，局所化された部分の中でカウンターが操作（新設や値の変更など）されたとしても，全体に対してその操作が影響をもち続けられるようにする必要があり，\global を多用することになる．

■疑問 2　1840 行目の \endcsname の前にある \expandafter が余計に感じられるが，どういうことか．

[*2] プリミティブ \relax と同じと考えて差し支えないのではなかろうか．

1.2.2 \setcounter 命令と \addtocounter 命令

よく知られている通り，\setcounter は LaTeX 流カウンターに任意の整数を代入するコマンド，\addtocounter は LaTeX 流カウンターに任意の整数を可算するコマンドである．まずはその定義を示しておく．

```
1811 \def\setcounter#1#2{%
1812   \@ifundefined{c@#1}%
1813     {\@nocounterr{#1}}%
1814     {\global\csname c@#1\endcsname#2\relax}}
1815 \def\addtocounter#1#2{%
1816   \@ifundefined{c@#1}%
1817     {\@nocounterr{#1}}%
1818     {\global\advance\csname c@#1\endcsname #2\relax}}
```

これらの定義はいずれも難しくない．だが本題に入る前に 1812・1826 行目に登場している \@ifundefined という LaTeX の便利な内部コマンドについて解説しておく．\@ifundefined は文字通り，制御綴が定義されているか否かで分岐を行うためのコマンドで，以下のような形で用いられる[*3]．

\@ifundefined{ **〈調べたい制御綴〉** } { **〈未定義だった場合の処理〉** }{ **〈定義済みだった場合の処理〉** }

このコマンドは，ユーザーがマクロを作成する際にも役立つだろうことが容易に推測できるだろう．では，どのようにしてこのようなことを実現しているのだろうか．latex.ltx の定義は以下のようになっている．

```
788 \def\@ifundefined#1{%
789   \expandafter\ifx\csname#1\endcsname\relax
790     \expandafter\@firstoftwo
791   \else
792     \expandafter\@secondoftwo
793   \fi}
```

ここで登場する \ifx は続く 2 つのトークンが同値であるかどうかで条件分岐するプリミティブである．今回のように後ろに続くトークンが制御綴トークンであった場合にはそれぞれの「意味」が同じかどうかで条件分岐する．また，ここに登場する \@firstoftwo と \@secondoftwo は文字通りの機能をもつ LaTeX の内部命令で，その定義は極めて単純である．

```
105 \long\def\@firstoftwo#1#2{#1}
106 \long\def\@secondoftwo#1#2{#2}
```

さて，\@ifundefined の定義で鍵となるのは\csname hoge\endcsname の挙動である．これは，基本的には（カテゴリーコードの問題を除けば）\hoge と同義であるが，\hoge が未定義の場合，単に \hoge とソースに記述した場合（当然，エラーになる）と異なる動きをする．すなわち，\hoge が未定義の状態で\csname hoge\endcsname が展開されると \relax として働く．そのことを把握していれば，789 行目のからくりはそれ以上詳しく説明しなくてもわかるだろう．

話を \setcounter と \addtocounter の定義に戻そう．\@ifundefined によって分岐された処理のうち，**〈定義済みだった場合の処理〉** についてはさほど追加の説明を必要としないだろう．見ての通り LaTeX 流カウンターの実態である TeX 流カウンター \c@LaTeXcounter が操作されている．また，これらの末尾に \relax があるのは，直後に数値等がきた場合にも，目的通りの処理が実行されるようにするための工夫だろうと思われる．

最後に残ったのは **〈未定義だった場合の処理〉** の中に登場する \@nocounterr である．

```
1067 \gdef\@nocounterr#1{%
1068   \@latex@error{No counter '#1' defined}\@eha}
```

[*3] 第 1 引数の **〈調べたい制御綴〉** は，正確には「調べたい制御綴から先頭のバックスラッシュを取り除いたもの」である．

今回 \@latex@error の定義までは遡らないが，そこまでしなくても「No counter 'LaTeXcounter' defined」
というエラーを出力するだろうことは簡単に推測できる．

2 \c@page 周辺に関して（谷口）

まず以下に latex.ltx 中の \c@page 周辺の定義を掲載する．これは ltpageno.dtx 由来となっている．

```
3851 \countdef\c@page=0 \c@page=1
3852 \def\cl@page{}
3853 \def\pagenumbering#1{\%
3854   \global\c@page \@ne \gdef\thepage{\csname @#1\endcsname
3855    \c@page}}
```

2.1 \c@page に関して

まずページ数を表すレジスターである \count0 で \c@page を定義した上で，ここに初期値 1 を代入している．

■疑問 3 \count0 と \cl@page を動かしている命令はどこのファイルに記述されているのか．また，どのように制御されているのか．

2.2 \pagenumbering に関して

\pagenumbering を用いることで，ページ番号のスタイルを変更することが可能である．
ここで例えばページ番号のスタイルとして \roman を指定したとすれば，まず\csname @#1\endcsnamec@page により以下の \@roman が呼び出されてその引数として \c@page がとられる．

```
1850 \def\@roman#1{\romannumeral #1}
```

■疑問 4 なぜ直接 TEX プリミティブの \romannumeral を呼び出さずに，一旦 \@roman を介するのか．

書き換えを楽にするため．

3 mod（谷口）

整数論で用いられる mod には 2 種類存在し，一方は \pmod で，もう一方は \bmod で記述することができる．

3.1 mod の使用例

高校等で扱われる合同式は \pmod を用いて以下のように記述し，次の表示を得ることができる．

```
a\equiv b\pmod{p}
```

$$a \equiv b \pmod{p}$$

この式で整数 a と b を p で割った余りが等しいことを示している．
また，二項演算子としての mod も存在し，\pmod を用いて以下のように記述し，次の表示を得ることができる．

```
a\bmod{p}b
```

$$a \bmod pb$$

この式で整数 a を p で割った余りが b であることを示している.[*4]

3.2 \pmod の定義

\pmod は ltmath.dtx 由来で以下のように定義されている.

```
4098 \def\pmod#1{%
4099   \allowbreak\mkern18mu({\operator@font mod}\,\,#1)}
```

本定義で用いられている命令を一つずつ見ていこう.

まず 1 つ目に用いられている命令は \allowbreak であり，これは latex.ltx 上で定義されている.

```
470 \def\allowbreak{\penalty \z@}
```

ここで，\penalty は TEX プリミティブであり，続くまた，\z@は latex.ltx 上で以下のように定義された寸法であり，0pt を表している.

```
354 \newdimen\z@ \z@=0pt % can be used both for 0pt and 0
```

続いて \mkern は TEX プリミティブであり，引き続く長さの kern を挿入する．ここで，mu は 18em に等しく，em 単位ではなく mu 単位で長さを指定しなければならない.

続いて，\operator@font は\mathgroup\symoperators で定義されていて，更に，\mathgroup は \fam で，\symoperators は \char"0 で定義されている．また，\fam は TEX プリミティブであり，数式モード中で使うフォントファミリーを指定している.

3.3 一般のオペレーターの定義

ltmath.dtx 中で定義された他の大抵のオペレーターはほぼ同様に定義されている．その例として \log の定義を掲載する.

```
4062 \def\log{\mathop{\operator@font log}\nolimits}
```

ここで，\mathop は TEX プリミティブであり，\mathop の引数の前後に適切な余白を空ける．また，\nolimits は TEX プリミティブであり，添字を下ではなく右下につけるよう設定している.

■疑問 5 \pmod の定義に関して，\log 等の定義を踏襲して，以下のように記述することで，\pmod と同様な出力を得ることができる.

```
\allowbreak\mkern18mu(\mathop{\operator@font mod}\,p)
```

ここで，余白調節のために \, を一つ抜いてある．なぜこのように \mathop を用いずに定義しているのか.

[*4] $a \bmod pb \iff a \equiv b \pmod{p}$ は成立するが，逆は成り立たない.