latex.ltx リーディング
第 7 回資料

東大 TEX 愛好会

2015 年 10 月 12 日

1 smash（大浦）

1.1 使用例

まず，\smash の使用例について軽く見ておこう。\smash は，引数に与えられたテキストの「高さ・深さを 0 にする」というコマンドである。つまり，

$$\int_b^a \sqrt{|x|}\,dx$$

を大かっこでくくるような場合を考えよう。ここで，\int^a_b を \smash でくくるかくくらないかで次のような出力の差を生む。ちなみに一番右の例は，\left〜\right でくくっていないもとの大かっこの大きさで出力している。

$$\left[\int_b^a \sqrt{|x|}\,dx\right] \left[\int_b^a \sqrt{|x|}\,dx\right] [\int_b^a \sqrt{|x|}\,dx]$$

インテグラルを \smash の引数に指定しておくと，インテグラルの高さは 0 とみなされ，その次に一番高さが大きかった $\sqrt{|x|}$ に大かっこのサイズが合うようになっている。

（ゼミで判明した内容：ちなみに，後述するとおり color 系のパッケージを include しておかないと使えない模様。）

1.2 定義

\smash まわりの定義をまず下に載せておく。

```
4136 \def\smash{%
4137   \relax % \relax, in case this comes first in \halign
4138   \ifmmode
4139     \expandafter\mathpalette\expandafter\mathsm@sh
4140   \else
4141     \expandafter\makesm@sh
4142   \fi}
4143 \def\makesm@sh#1{%
4144   \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finsm@sh}
4145 \def\mathsm@sh#1#2{%
4146   \setbox\z@\hbox{$\m@th#1{#2}$}\finsm@sh}
4147 \def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \box\z@}
```

この定義を理解するために，まず，数式モードの外で \smash を使うとき，\smash{**あいうえ**} というコマンドを考える。

まずこれは次のように展開される。

```
\relax\ifmmode\expandafter\mathpalette\expandafter\mathsm@sh\else\expandafter\makesm@sh\
    fi{あいうえ}
```

\relax はそのまま読みとばされ，\ifmmode の条件分岐に入る。いま，数式モードではないので，\else まで飛ばされる。その次の \expandafter によって \fi が読まれ，条件分岐が終わったと認識される。そしてその次に \makesm@sh が展開されていく。

つまり，

```
\makesm@sh{あいうえ}
```

が展開されることに相当する。ここで \makesm@sh の定義を代入して，次のように変換される。

```
\setbox\z@\hbox{\color@begingroup あいうえ\color@endgroup}\finsm@sh
```

まず，\setbox\z@\hbox{...} によって，0 番のボックスレジスタにここの \hbox を割り振る。そして \hbox が作られる。さらに \hbox の中を展開する。\color@begingroup は color 系のスタイルファイルを読み込むときに \bgroup に置き換えられ，\color@endgroup は color 系のスタイルファイルをを読み込むときに，\endgraf\endgroup になる。（ここで color 系統のパッケージが必要になる。）ここで \endgraf は，latex.ltx の 438 行目に次のように定義されている。

```
438 \let\endgraf=\par
```

また，\finsm@sh の定義を再掲すると，次のようであった。

```
4147 \def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \box\z@}
```

これらを代入すると，

```
\hbox{{あいうえ\par}}\ht\z@\z@ \dp\z@\z@ \box\z@
```

となる。つまり，0 番のボックスレジスタに入れた \hbox に対して，\ht つまり高さ，および \dp つまり深さを，0 にする操作をするコマンドとなっている。ここが \smash の肝となる部分といえよう。

いま，数式モードの外で \smash を使う例を見てきた。次は，数式モード内で \smash を使う例を見てみよう。たとえば \[\smash{1+2=3}\] というコマンドを考える。

まずこれは次のように展開される。（\[, \] については略した。）

```
\relax\ifmmode\expandafter\mathpalette\expandafter\mathsm@sh\else\expandafter\makesm@sh\
    fi{1+2=3}
```

\relax はそのまま読みとばされ，\ifmmode の条件分岐に入る。いま，数式モード内なので，次の \expandafter が読まれる。すると，\mathpalette を飛ばして，２つ目の \expandafter が読まれる。これにより，\mathsm@sh が飛ばされ，\else が読まれる。ここで，条件分岐は終了したものと判断され，その次に \mathpalette，その次に \mathsm@sh が展開されることとなる。つまり，

```
\mathpalette\mathsm@sh{1+2=3}
```

の \mathpalette が展開されることになる。\mathpalette は，latex.ltx 中に次のように定義されている。

```
4101 \def\mathpalette#1#2{%
4102   \mathchoice
4103     {#1\displaystyle{#2}}%
4104     {#1\textstyle{#2}}%
4105     {#1\scriptstyle{#2}}%
4106     {#1\scriptscriptstyle{#2}}}
```

よって，次のように展開される。

```
\mathchoice
  {\mathsm@sh\displaystyle{1+2=3}}%
  {\mathsm@sh\textstyle{1+2=3}}%
  {\mathsm@sh\scriptstyle{1+2=3}}%
  {\mathsm@sh\scriptscriptstyle{1+2=3}}
```

\mathchoice は引数を四つとる TEX プリミティブで，四つの引数の中は順に，displaystyle，textstyle，scriptstyle，scriptscriptstyle，内でのコードを記述する。

例えば今回のように，displaystyle の数式モード内で \smash を使った場合には，最初の引数内のコードが使われ，

```
\mathsm@sh\displaystyle{1+2=3}
```

と展開される。

次にいよいよ \mathsm@sh の展開だが，定義は次のようであった（再掲）。

```
4145 \def\mathsm@sh#1#2{%
4146   \setbox\z@\hbox{$\m@th#1{#2}$}\finsm@sh}
```

よって展開結果は次のようになる。

```
\setbox\z@\hbox{$\m@th\displaystyle{1+2=3}$}\finsm@sh
```

まず，\setbox\z@\hbox{...} によって，0 番のボックスレジスタにここの \hbox を割り振る。そして \hbox が作られる。さらに \hbox の中を展開する。

$ ハサミの効果により数式モードになっており，\m@th によって数式モード外との境界に無駄なスペースが入らないようになっている（\m@th については第 1 回 Reading 資料に説明がある）。そして，$1+2=3$ が displaystyle で表示される。

さて，最後にまた，\finsm@sh の展開である。0 番のボックスレジスタに入れた \hbox に対して，\ht つまり高さ，および \dp つまり深さを，0 にする操作を担っていることは先に述べた通りである。

これにより，数式モード中でも \smash を使うことができた。

さて，若干気になるのが，4137 行目の記述，「\relax, in case this comes first in \halign」だ。実際に展開した状態で \halign の先頭で使ってみた。つまり，次のようにソースを書き，\relax のある状態とない状態での比較を試みた。

ソース 1

```
\halign{\relax\setbox0\hbox{{$\displaystyle\int$\par}}\ht00 \dp00 \box0#1\cr
\noalign{\hrule}
www\cr
\noalign{\hrule}}
```

ソース 2

```
\halign{\setbox0\hbox{{$\displaystyle\int$\par}}\ht00 \dp00 \box0#1\cr
\noalign{\hrule}
www\cr
\noalign{\hrule}}
```

その結果が以下。

ソース 1

$$\int \mathrm{www1}$$

ソース 2

$$\int \mathrm{www1}$$

■疑問 1　コンパイルもどちらも問題なく通るし，\relax の有無の差が見えてこない。

2　LaTeX における各種レジスタの管理（大門）

今回は，TeX に本来備わっている \count，\dimen など各種レジスタの，LaTeX 流管理の実装を調べていく。

2.1 前提知識

TEX は，\count（整数），\dimen（寸法），\skip（グルー），\muskip（数式用グルー），\box（ボックス），\toks（トークンリスト）というデータ型を持ち，素の（e-TEX 拡張などされていない）TEX エンジンの場合，これらの型には 0 から 255 までの番号をもつレジスタ（変数）が用意されている[*1]。

さて，plain TEX および（2014 年以前の）LATEX では，この 0 から 255 まで用意された各種レジスタを，以下のように使用している。

番号	\count	\dimen，\skip	\box	\musikip，\toks
0 - 9	ページ番号	スクラッチ	スクラッチ	スクラッチ
10 - 20	各種レジスタの割当済み番号	割当用	割当用	割当用
21	\allocationnumber	割当用	割当用	割当用
22	定数 “-1”	割当用	割当用	割当用
23 - (/-1)	割当用	割当用	割当用	割当用
/ - 254	insert 用	insert 用	insert 用	割当用
255	スクラッチ	スクラッチ	スクラッチ	スクラッチ

ただし，/は \count20（割当済 insert 番号）の値。また，「スクラッチ」は \dimen255（\dimen@）のように，割当なしで使えるよう予め用意されたレジスタのこと
『LATEX2ε【マクロ&クラス】プログラミング基礎解説』およびマクロツイーター
(http://d.hatena.ne.jp/zrbabbler/20150503/1430658474 http://d.hatena.ne.jp/zrbabbler/20150504/1430721128)
を参照した。

2.2 実装コード

```
\message{registers,}
\count10=22 % allocates \count registers 23, 24, ...
\count11=9 % allocates \dimen registers 10, 11, ...
\count12=9 % allocates \skip registers 10, 11, ...
\count13=9 % allocates \muskip registers 10, 11, ...
\count14=9 % allocates \box registers 10, 11, ...
\count15=9 % allocates \toks registers 10, 11, ...
\count16=-1 % allocates input streams 0, 1, ...
\count17=-1 % allocates output streams 0, 1, ...
\count18=3 % allocates math families 4, 5, ...
\count19=0 % allocates \language codes 1, 2, ...
\count20=255 % allocates insertions 254, 253, ...
\countdef\insc@unt=20
\countdef\allocationnumber=21
\countdef\m@ne=22 \m@ne=-1
\def\wlog{\immediate\write\m@ne}
\countdef\count@=255
\dimendef\dimen@=0
\dimendef\dimen@i=1 % global only
\dimendef\dimen@ii=2
\skipdef\skip@=0
\toksdef\toks@=0
\def\newcount{\alloc@0\count\countdef\insc@unt}
\def\newdimen{\alloc@1\dimen\dimendef\insc@unt}
\def\newskip{\alloc@2\skip\skipdef\insc@unt}
\def\newmuskip{\alloc@3\muskip\muskipdef\@cclvi}
```

[*1] その他のエンジンでは，「たくさん」（32768 または 65536 個）のレジスタが用意されている。

```
\def\newbox{\alloc@4\box\chardef\insc@unt}
\def\newhelp#1#2{\newtoks#1#1\expandafter{\csname#2\endcsname}}
\def\newtoks{\alloc@5\toks\toksdef\@cclvi}
\def\newread{\alloc@6\read\chardef\sixt@@n}
\def\newwrite{\alloc@7\write\chardef\sixt@@n}
\def\newlanguage{\alloc@9\language\chardef\@cclvi}
\def\alloc@#1#2#3#4#5{\global\advance\count1#1\@ne
  \ch@ck#1#4#2% make sure there's still room
  \allocationnumber\count1#1%
  \global#3#5\allocationnumber
  \wlog{\string#5=\string#2\the\allocationnumber}}
\def\newinsert#1{\global\advance\insc@unt \m@ne
  \ch@ck0\insc@unt\count
  \ch@ck1\insc@unt\dimen
  \ch@ck2\insc@unt\skip
  \ch@ck4\insc@unt\box
  \allocationnumber\insc@unt
  \global\chardef#1\allocationnumber
  \wlog{\string#1=\string\insert\the\allocationnumber}}
\gdef\ch@ck#1#2#3{%
  \ifnum\count1#1<#2\else
   \errmessage{No room for a new #3}%
  \fi}
\newdimen\maxdimen \maxdimen=16383.99999pt % the largest legal <dimen>
\newskip\hideskip \hideskip=-1000pt plus 1fill % negative but can grow
\newdimen\p@ \p@=1pt % this saves macro space and time
\newdimen\z@ \z@=0pt % can be used both for 0pt and 0
\newskip\z@skip \z@skip=0pt plus0pt minus0pt
\newbox\voidb@x % permanently void box register
```

まず冒頭で，\message によりレジスタ関連の定義が始まることを示してから処理に入る。非常に長いので，幾つかに分割してこれらのコードの動作を分析していく。

2.3 初期設定部（仮称）について

```
\message{registers,}
\count10=22 % allocates \count registers 23, 24, ...
\count11=9 % allocates \dimen registers 10, 11, ...
\count12=9 % allocates \skip registers 10, 11, ...
\count13=9 % allocates \muskip registers 10, 11, ...
\count14=9 % allocates \box registers 10, 11, ...
\count15=9 % allocates \toks registers 10, 11, ...
\count16=-1 % allocates input streams 0, 1, ...
\count17=-1 % allocates output streams 0, 1, ...
\count18=3 % allocates math families 4, 5, ...
\count19=0 % allocates \language codes 1, 2, ...
\count20=255 % allocates insertions 254, 253, ...
\countdef\insc@unt=20
\countdef\allocationnumber=21
\countdef\m@ne=22 \m@ne=-1
\def\wlog{\immediate\write\m@ne}
\countdef\count@=255
\dimendef\dimen@=0
\dimendef\dimen@i=1 % global only
\dimendef\dimen@ii=2
\skipdef\skip@=0
\toksdef\toks@=0
```

この部分は特にどうと言うことも無く，特殊な役割をもつレジスタの使用を準備しているだけである。

2.4 \newcount 系の定義部（仮称）について

```
\def\newcount{\alloc@0\count\countdef\insc@unt}
\def\newdimen{\alloc@1\dimen\dimendef\insc@unt}
\def\newskip{\alloc@2\skip\skipdef\insc@unt}
\def\newmuskip{\alloc@3\muskip\muskipdef\@cclvi}
\def\newbox{\alloc@4\box\chardef\insc@unt}
\def\newhelp#1#2{\newtoks#1#1\expandafter{\csname#2\endcsname}}
\def\newtoks{\alloc@5\toks\toksdef\@cclvi}
\def\newread{\alloc@6\read\chardef\sixt@@n}
\def\newwrite{\alloc@7\write\chardef\sixt@@n}
\def\newlanguage{\alloc@9\language\chardef\@cclvi}
\def\alloc@#1#2#3#4#5{\global\advance\count1#1\@ne
  \ch@ck#1#4#2% make sure there's still room
  \allocationnumber\count1#1%
  \global#3#5\allocationnumber
  \wlog{\string#5=\string#2\the\allocationnumber}}
\def\newinsert#1{\global\advance\insc@unt \m@ne
  \ch@ck0\insc@unt\count
  \ch@ck1\insc@unt\dimen
  \ch@ck2\insc@unt\skip
  \ch@ck4\insc@unt\box
  \allocationnumber\insc@unt
  \global\chardef#1\allocationnumber
  \wlog{\string#1=\string\insert\the\allocationnumber}}
\gdef\ch@ck#1#2#3{%
  \ifnum\count1#1<#2\else
   \errmessage{No room for a new #3}%
  \fi}
```

ここでは \newcount，\newdimen，\newskip，\newmuskip，\newbox，\newtoks をまとめて \newcount 系と呼ぶことにする（他の \newinsert，\newhelp，\newread，\newwrite，\newlanguage については，今回は措いておく）。これら \newcount 系の実装はどれも似通っているので，代表例として \newcount の動作を追跡していくことにしよう。

2.4.1 具体例

以下の様なソースコードを実行する事を考える。

```
\newcount\ほげ
```

これを上に従って適宜展開・実行していくと，次のようになる。

```
   \newcount\ほげ

-> \alloc@0\count\countdef\insc@unt\ほげ

-> \global\advance\count10\@ne
   \ch@ck0\insc@unt\count% make sure there's still room
   \allocationnumber\count10%
   \global\countdef\ほげ\allocationnumber
   \wlog{\string\ほげ=\string\count\the\allocationnumber}

   （\advance が実行され，\count10の値が 1増える）

-> \ifnum\count10<\insc@unt\else
    \errmessage{No room for a new \count}%
   \fi
   \allocationnumber\count10%
   \global\countdef\ほげ\allocationnumber
```

```
\wlog{\string\ほげ=\string\count\the\allocationnumber}
```

ここの \ifnum で，現在割り当てようとしているレジスタの番号が，割り当て済みの insert 番号（先の表中の “/” に相当）より小さいかを判別し，これを超過するようであればエラーが出る。このチェックを通り抜けた場合，\allocationnumber に今から割り当てるレジスタの番号を格納し，

```
\global\countdef\ほげ\allocationnumber
```

によってレジスタに **\ほげ**を割り当てる。末尾の \wlog は，*source2e.pdf* に “Write on log file (only)” とあるように，.log ファイルに **\ほげ**が定義された事を書き込む役割をもつ。

2.5 その他の定義部（仮称）について

```
351 \newdimen\maxdimen \maxdimen=16383.99999pt % the largest legal <dimen>
352 \newskip\hideskip \hideskip=-1000pt plus 1fill % negative but can grow
353 \newdimen\p@ \p@=1pt % this saves macro space and time
354 \newdimen\z@ \z@=0pt % can be used both for 0pt and 0
355 \newskip\z@skip \z@skip=0pt plus0pt minus0pt
356 \newbox\voidb@x % permanently void box register
```

\maxdimen は寸法レジスタの最大値を設定し，\hideskip は伸びることも可能な負のスペース（アラインメントに使用されるようだが詳細は不明）を設定する。\p@，\z@，\z@skip はお馴染みのメモリ削減テクニックの産物である。\voidb@x は常に「空の」box レジスタであり，\leavevmode の定義に利用されたり，（普通のプログラム言語における Null 値のように）レジスタを初期化するために利用されたりする。