latex.ltx リーディング
第 9 回資料

東大 TEX 愛好会

2015 年 12 月 14 日

1　エラーメッセージまわりその２（大浦）

`lterror.dtx` に由来する部分を読んでいく。定義の前半の２回目。

1.1　GenericError 続論

Error の続きをやる。定義:

```
\bgroup
\lccode`\@=`\ %
\lccode`\~=`\ %
\lccode`\}=`\ %
\lccode`\{=`\ %
\lccode`\T=`\T%
\lccode`\H=`\H%
\catcode`\ =11\relax%
\lowercase{%
\egroup%
\dimen@\ifx\@TeXversion\@undefined4\else\@TeXversion\fi\p@%
\ifdim\dimen@>3.14\p@%
\DeclareRobustCommand{\GenericError}[4]{%
\begingroup%
\immediate\write\@unused{}%
\def\MessageBreak{^^J}%
\set@display@protect%
\edef%
\@err@ %
{{#4}}%
\errhelp
\@err@ %
\let
\@err@ %
\@empty
\def\MessageBreak{^^J#1}%
\def~{\errmessage{%
#2.^^J^^J%
#3^^J%
Type H <return> for immediate help%
\@err@ %
}}%
~%
\endgroup}%
\else%
\DeclareRobustCommand{\GenericError}[4]{%
\begingroup%
```

```
930 \immediate\write\@unused{}%
931 \def\MessageBreak{^^J}%
932 \set@display@protect%
933 \edef%
934 \@err@ %
935 {{#4}}%
936 \errhelp
937 \@err@ %
938 \let
939 \@err@ %
940 \errmessage
941 \def\MessageBreak{^^J#1}%
942 \def~{\typeout{! %
943 #2.^^J^^J%
944 #3^^J%
945 Type H <return> for immediate help.}%
946 \@err@ %
947 {}}%
948 ~%
949 \endgroup}%
950 \fi}%
```

であった。903〜905 で条件分岐があり，TEX バージョンが 3.14 より大きいかどうかで場合分けされることを述べた。（ちなみに，バージョンが不明の場合にはバージョン 4 として扱われる。）今回は，それぞれの場合について，\GenericError の定義がどのようになっているかを具体的に見ていくことにする。

1.2 条件分岐 1 ：バージョンが 3.14 より大きいとき

```
905 \DeclareRobustCommand{\GenericError}[4]{%
906 \begingroup%
907 \immediate\write\@unused{}%
908 \def\MessageBreak{^^J}%
909 \set@display@protect%
910 \edef%
911 \@err@ %
912 {{#4}}%
913 \errhelp
914 \@err@ %
915 \let
916 \@err@ %
917 \@empty
918 \def\MessageBreak{^^J#1}%
919 \def~{\errmessage{%
920 #2.^^J^^J%
921 #3^^J%
922 Type H <return> for immediate help%
923 \@err@ %
924 }}%
925 ~%
926 \endgroup}%
```

905 行目で，\GenericError は引数を四つとる堅牢な制御綴として定義されることが宣言される。907 行目の，\immediate\write\@unused については前回資料で述べた通り，そのあとの内容をターミナル送りなり .log ファイル送りなりにしてくれるコマンドなのだが，ここではその引数が空なので結局どうでもよい。その次の 908 行目で，^^J つまり改行をエラーメッセージの区切りとして指定し，その次の 909 行目 \set@display@protect で，\protect を \string の別名として定義する（前回資料で扱った）。

そして 4 番目の引数を \@err@に展開した形で渡し，\errhelp\@err@を実行した上で \@err@を \@empty

に，つまり空に戻しておく[*1]。\errhelp は引数を一つ取るプリミティブで，エラーメッセージが出たあと，h キー＋ Return キーを押した場合に表示されるメッセージを指定するコマンドである。ここまででではエラーメッセージの区切りは改行のままである。

さて，ここでエラーメッセージの区切りの変更（918 行目）となる。ただの改行 ^^J に飽き足らず第一引数まで含めた ^^J#1 をエラーメッセージとして定義する。その上で~(チルダマーク）の定義を変更して，これが

```
\errmessage{%
#2.^^J^^J%
#3^^J%
Type H <return> for immediate help%
\@err@ %
}
```

の意味だというふうに定義してしまう。\errmessage は，ターミナル及び.log ファイルに送られるエラーメッセージを指定するプリミティブである。923 行目の \@err@については，すでに空になっているはずなので，考える必要はない。

そして，925 行目で，今定義した~(チルダマーク）を使ってエラーメッセージを生成し，終了となる。

では，実際の例を見てみよう。

1.3 （条件分岐１）実際例として

実際例として，latex.ltx1062 行目で定義されている \@nocounterr を取り上げる。ソース☆ 0：

```
\@nocounterr{ほげ}
```

を展開してみることにしよう。つまり，「ほげ」という名前のカウンタが定義されていなかったのに，

```
\setcounter{ほげ}{10}
```

のようにあたかも「ほげ」という名前のカウンタがあるかのように書いてしまったときに生成されるエラーの生成機序を追っていこうということである。まず，\@nocounterr は次のように定義されている。

```
1062 \gdef\@nocounterr#1{%
1063   \@latex@error{No counter '#1' defined}\@eha}
```

これより，ソース☆ 0 は，一段階展開されて次のソース☆ 1 となる。

```
\@latex@error{No counter 'ほげ' defined}\@eha}
```

ここで，\@latex@error が展開される。\@latex@error は，1003 行目に，次のように定義されている。

```
1003 \gdef\@latex@error#1#2{%
1004    \GenericError{%
1005       \space\space\space\@spaces\@spaces\@spaces
1006    }{%
1007       LaTeX Error: #1%
1008    }{%
1009       See the LaTeX manual or LaTeX Companion for explanation.%
1010    }{#2}%
1011 }
```

これより，ソース☆ 1 はもう一段階展開されて次のソース☆ 2 となる。

```
\GenericError{%
   \space\space\space\@spaces\@spaces\@spaces
}{%
   LaTeX Error: No counter 'ほげ' defined%
}{%
   See the LaTeX manual or LaTeX Companion for explanation.%
}{\@eha}%
```

[*1] 念のため。\@empty は latex.ltx の 122 行目に空の綴りとして定義されている。

そしてお待ちかね \GenericError の展開となる。展開結果は次のようなソース☆ 3。

```
\begingroup%
\immediate\write\@unused{}%
\def\MessageBreak{^^J}%
\set@display@protect%
\edef%
\@err@ %
{{\@eha}}%
\errhelp
\@err@ %
\let
\@err@ %
\@empty
\def\MessageBreak{^^J\space\space\space\@spaces\@spaces\@spaces}%
\def~{\errmessage{%
LaTeX Error: No counter 'ほげ' defined.^^J^^J%
See the LaTeX manual or LaTeX Companion for explanation.^^J%
Type H <return> for immediate help%
\@err@ %
}}%
~%
\endgroup
```

つまり，こういうこと。

h キー＋ Return キーを押したときに表示されるヘルプメッセージの内容は，\@eha になり，その区切り記号は改行である。

エラーメッセージの内容は，LaTeX Error〜immediate help であり，その区切り記号は改行＋ \space×3 ＋ \@spaces×3 である。

さて，\@eha，\space，\@spaces という三つの初出コマンドが出てきたので一つずつ解説を加えよう。

\@eha については，別途 latex.ltx に定義がある。

```
1041 \gdef\@eha{%
1042   Your command was ignored.\MessageBreak
1043   Type \space I <command> <return> \space to replace it %
1044   with another command,\MessageBreak
1045   or \space <return> \space to continue without it.}
```

この時点では \MessageBreak はただの改行なので，\MessageBreak は改行に読み替えておけば良いし，\space については後述する通りただの半角スペースなのでそう読み替えておけば良い。

\space についても，別途 latex.ltx に定義がある。

```
97 \def\space{ }
```

ただの半角スペースじゃないか。

\@spaces も latex.ltx 由来のコマンドで，\@eha を定義したすぐ上流の 1040 行目に定義されている。

```
1040 \def\@spaces{\space\space\space\space}
```

ただの半角スペース 4 こ連続じゃないか。

1.4 条件分岐２：バージョンが 3.14 より小さいとき

\GenericError の定義はこんな感じに変わる。

```
928 \DeclareRobustCommand{\GenericError}[4]{%
929 \begingroup%
930 \immediate\write\@unused{}%
931 \def\MessageBreak{^^J}%
```

```
\set@display@protect%
\edef%
\@err@ %
{{#4}}%
\errhelp
\@err@ %
\let
\@err@ %
\errmessage
\def\MessageBreak{^^J#1}%
\def~{\typeout{! %
#2.^^J^^J%
#3^^J%
Type H <return> for immediate help.}%
\@err@ %
{}}%
~%
\endgroup}%
```

\@err@に第 4 引数を代入して，それをヘルプメッセージとして指定するところまでは同じである。そこからさきが若干異なる。\let を使って，\@err@を空にするのではなく \errmessage の別名として定義し，次の \typeout 命令の中で使えるようにしてしまう。

さて 942 行目からは先ほどの \errmessage コマンドにかわり \typeout 命令に変わっている。これは latex.ltx に定義されたコマンドで，

```
\def\typeout{\immediate\write17}
```

となっている。\write のあとに続く数が 15 より大きいので，\typeout の引数は，.log ファイル送りになりかつ，ターミナルに表示される。また，\@err@は空ではないので今回は考える必要が生じるように思われるものの，今それは \errmessage の別名であり，その引数もないので，結局考えなくて済んでしまう。