expandparams の用法 第 1.02 版

gfn（Twitter: @bd_gfngfn, GitHub: gfngfn）

2014 年 10 月 10 日

1　免責

動作には最善を期していますが，expandparams パッケージをユーザーが使用したことによるいかなる損失にも作者 gfn は責任を負いません．

2　expandparams パッケージの趣旨

TEX は，その体系が Church-Rosser 性を満たさないが，コンパイラが前から順に展開していくためにしばしば展開順序の制御を手動で行なわねばならないということが運命づけられている言語です．展開順序の制御に使われる制御綴はいくつかありますが，特に重宝されている[要出典]のは\expandafter でしょう．しかしながら，展開制御のために埋め込まれるこの\expandafter の個数はしばしば皮肉な祝福性を帯びるほどに厖大となり，また，あらゆるコマンドの間に割り込んで可読性と保守性を極めて低いものにしてしまいます．一度慣れてしまえば水や空気のように何事もなく書けてしまう[要出典]\expandafter ですが，なんとかもう少し人間に歩み寄った形式が取れないものかと思案した結果考案したのが expandparams パッケージの制御綴\expandparams と\prexp です．

3　\expandparams と\prexp の使い方

3.1　一般仕様

\expandparams は，入力上先行する制御綴よりも順序的に先に引数を何度か展開したい場合に使うことができ，これにより数百個もの\expandafter を書かずに済むことになります．n を非負整数，a を n 個の引数を取って展開されるトークン，t を {，}，[，]，␣ のいずれでもないトークンとして，\expandparams は

$$
\begin{aligned}
E &\to \texttt{\textbackslash expandparams}\{a\underbrace{T\cdots T}_{n\text{ 個}}\}\\
T &\to B \mid \texttt{\textbackslash prexp}T\\
B &\to \{L\} \mid [L]\\
L &\to t \mid LL \mid B
\end{aligned}
$$

の文脈自由文法で表される表現に対して動作することを意図しています．実際にはこれよりも広い表現を受理しますが，これにあてはまらないものは\expandparams の用法として推奨しません．トークン列 L を 1 回展

開したものを $\xi(L)$ と書くとすると，\prexp{L} は {$\xi(L)$} に，\prexp[L] は [$\xi(L)$] に前から順に[*1]展開され，すべての\prexp が処理された後に先頭から窮極展開[*2]されます．また，\expandparams を入れ子にすることは現在のところ意図していません．特に開き四角括弧 [，閉じ四角括弧]，空白 ␣ はトークンとして\expandparams の引数中で使用することを非推奨としていますが，[，] を使用したい場合は

code
```
\def\openbra{[}%
\def\closebra{]}%
```

などと定義してエスケープさせ，また空白 ␣ は\␣で代替して，窮極展開が始まるまで展開しないように記述してください（例は後述）．

3.2 基本的な使用例

さて，ここからは例として，\csone，\cstwo，\separatetokens を以下のように定義しましょう：

code
```
\def\csone{GF}%
\def\cstwo{\csone N}%
\newcommand{\separatetokens}[3]{%
  \@tfor\ch:=#1\do{[\ch]}%
  \@tfor\ch:=#2\do{(\ch)}%
  \@tfor\ch:=#3\do{\{\ch\}}%
}%
```

このとき，例えば

code
```
\separatetokens{\csone V}{CD}{\cstwo W}
```

とすると出力は [GF][V](C)(D){GFN}{W}となります．さて，これを\separatetokens が展開されるより先に\csone と\cstwo をそれぞれ 1 回ずつ展開したい場合，\prexp という制御綴を使って

code
```
\expandparams{\separatetokens\prexp{\csone V}{CD}\prexp{\cstwo W}}
```

と書くことができます．比較として同等な展開制御を\expandafter を使って書くと

[*1] 正確には，連続して並ぶ\prexp は後ろのものから順に展開しています．
[*2] 展開可能な限り展開することを指します．

code

```
\expandafter\expandafter\expandafter\separatetokens%
\expandafter\expandafter\expandafter%
{\expandafter\csone\expandafter V\expandafter}\expandafter%
{\expandafter C\expandafter D\expandafter}\expandafter{\cstwo W}
```

となります．得られる結果はともに [G][F][V](C)(D){GF}{N}{W}となります．

さらに，先行する\separatetokens が展開されるより前に\cstwo を 2 回展開したい場合，\prexp を 2 個並べて

code

```
\expandparams{\separatetokens{AB}{CD}\prexp\prexp{\cstwo W}}
```

と書くことができます．こちらも\expandafter を使うと

code

```
\expandafter\expandafter\expandafter\separatetokens%
\expandafter\expandafter\expandafter{\expandafter\expandafter\expandafter A%
\expandafter\expandafter\expandafter B\expandafter\expandafter\expandafter}%
\expandafter\expandafter\expandafter{\expandafter\expandafter\expandafter C%
\expandafter\expandafter\expandafter D\expandafter\expandafter\expandafter}%
\expandafter\expandafter\expandafter{\cstwo W}
```

となります．いずれも [A][B](C)(D){G}{F}{N}{W}と出力されます．

3.3 四角括弧と空白のエスケープ

四角括弧と空白のエスケープによる表現の例を掲げます．

code

```
\expandparams{\separatetokens\prexp{\cstwo}%
  {*\openbra*\ }{*\closebra*\ }}
```

と書くと [GF][N](*)([])(*)( ){*}{]}{*}{ }のように出力されます．

3.4 引数を途中から展開する例

ここまで見てきた例はいずれも { と } で括られる引数の先頭のトークンから展開するものでしたが，

code

```
\separatetokens{\csone A}{B\cstwo}{C}
```

の第 2 引数の\cstwo を\separatetokens より先に展開したい場合のように，引数の途中から展開したいときは，\expandparams と\prexp だけで簡潔に書くのはどうしても無理があります．この場合は引数の手前に\prexp を置いておき，中の展開制御は\expandafter で行ないます．ここでは

code

```
\expandparams{\separatetokens{\csone A}\prexp{\expandafter B\cstwo}{C}}
```

とすると出力は [GF][A](B)(GF)(N){C}となり，\cstwo が展開されていることがわかります．2 回展開する場合は，

code

```
\expandparams{\separatetokens{\csone A}%
  \prexp\prexp{\expandafter\expandafter\expandafter B\cstwo}{C}}
```

とすると出力を [GF][A](B)(G)(F)(N){C}とすることができます．\prexp も 2 個書いていることに注意してください．