

Bernd Raichle

Global assignments done locally

Sometimes it is necessary to define a macro under a changed input regime, e.g, using different category codes or another end line character. Usually this is done inside a group to keep the changes locally and the macro or the token is defined `\global'y`.

In `plain.tex` you can find the following examples:

```
\catcode'\@=11
```

1. Helper macro for `\newif`:

```
{\uccode'1='i \uccode'2='f \uppercase{\gdef\if@12{}} % 'if' is required
```

2. Definitions of `\obeylines` and `\obeyspaces`:

```
{\catcode'\^^M=\active % these lines must end with %  
 \gdef\obeylines{\catcode'\^^M\active \let^^M\par}%  
 \global\let^^M\par} % this is in case ^^M appears in a \write  
{\obeyspaces\global\let =\space}
```

3. Definition of `\getf@ctor`:

```
{\catcode'p=12 \catcode't=12 \gdef\#1pt{#1}} \let\getf@ctor=\
```

4. Math definitions for primes and underscore:

```
{\catcode'\=' \active \gdef'\bgroup\prim@s}  
{\catcode'\_=\active \global\let_=\_} % _ in math is either subscript or \_
```

By placing the begin and end of group tokens in a bit “unusual” way using a temporary token register assignment or a macro definition, all these assignments can be done locally:

1. Helper macro for `\newif`:

```
\begingroup \uccode'1='i \uccode'2='f  
\uppercase{\endgroup\def\if@12{}}% 'i'+ 'f' as delimited arguments are required
```

2. Definitions of `\obeylines` and `\obeyspaces`:

```
\begingroup \endlinechar=-1 \catcode'\^^M=\active  
 \toks0={\endgroup  
 \def\obeylines{\catcode'\^^M\active \let^^M\par}%  
 \let^^M=\par % this is in case ^^M appears in a \write  
 } \the\toks0 \relax  
\begingroup \obeyspaces\def\x{\endgroup\let =\space}\x
```

3. Definition of `\getf@ctor`:

```
\begingroup \catcode'P=12 \catcode'T=12  
 % \lccode'P='p \lccode'T='t % is default setting  
 \lowercase{\endgroup\def\getf@ctor#1PT{#1}}% 'p'+ 't' with catcode 'other'
```

4. Math definitions for primes and underscore:

```
\begingroup \catcode'\=' \active  
 \def\x{\endgroup\def'\bgroup\prim@s}\x  
\begingroup \catcode'\_=\active  
 \def\x{\endgroup \let_=\_}\x % _ in math is either subscript or \_
```

There is no advantage of a local definition for the shown `plain.tex` cases but if you want to do similar definitions within a group, the shown technique can be very helpful.

Note: If you want to define macros with arguments it is better to use a token register assignment because you have to double the hash mark as macro parameter character inside the macro definition text.