Hints & Tricks

Glisterings

Peter Wilson

> Seagulls scream upon the shorelines' wrack
> And seals abound
> Amid the setting sun's glistering track
> Across the Sound.
>
> *Puget Sound*

The aim of this column is to provide odd hints or small pieces of code that might help in solving a problem or two while hopefully not making things worse through any errors of mine.

Corrections, suggestions, and contributions will always be welcome. Speaking of which, David Elliott was the first[1] to point out that in the last column [6] I mistakenly attributed Matthew Arnold's poem *Dover Beach* to Tennyson. I have no idea why I should have done that.

The three topics which are the subject of this month's column have all been suggested by readers.

> They cannot scare me with their empty spaces
> Between stars — on stars where no human race is.
>
> *Desert Places*, Robert Frost

1 Empty arguments

In an earlier column [5] I talked about how to check if two strings were the same, that is, that they consisted of the same characters in the same order. A recent query on the `texhax` mailing list [2] asked about how to check if an argument was empty, which on the face of it is just a check comparing a string with an empty string. However the earlier approach does not work in this case.

In LaTeX there is often a need to check if an optional argument is present or not. The typical form for this is:

```
\newcommand{\amacro}[1][\@empty]{...
  \ifx\@empty#1 ... % no optional
  \else ...         % optional not \@empty
```

The question at hand, though, is what is the proper replacement for the pseudo code in the second line below?

```
\newcommand{\amacro}[1]{...
  \if(#1 is empty) ... % no argument
  \else ...            % argument not empty
```

where 'empty' means zero or more spaces. Thus `{}` and `{ }` both qualify as 'empty'. If you are a LaTeX user then the ifmtarg package on CTAN provides a solution. For TeX users here is the equivalent code, noting that all the macro definitions before the `\begingroup` are a regular part of LaTeX.

```
\def\makeatletter{\catcode`\@11\relax}
\def\makeatother{\catcode`\@12\relax}
\makeatletter
\long\def\@gobble #1{}
\long\def\@firstofone#1{#1}
\long\def\@firstoftwo#1#2{#1}
\long\def\@secondoftwo#1#2{#2}
\begingroup
\catcode`\Q=3
\long\gdef\@ifmtarg#1{%
  \@xifmtarg#1QQ\@secondoftwo\@firstoftwo\@nil}
\long\gdef\@xifmtarg#1#2Q#3#4#5\@nil{#4}
\long\gdef\@ifnotmtarg#1{%
  \@xifmtarg#1QQ\@firstofone\@gobble\@nil}
\endgroup
\makeatother
```

The useful parts of this are
`\@ifmtarg`{⟨*arg*⟩}{⟨*empty code*⟩}{⟨*not empty code*⟩}
`\@ifnotmtarg`{⟨*arg*⟩}{⟨*not empty code*⟩}

For example these could be used like

```
\def\isempty#1{\@ifmtarg{#1}{EMPTY}{FULL}}
\def\isnotempty#1{\@ifnotmtarg{#1}{FULL}}%
\def\mt{}
\isempty{}         -> EMPTY
\isempty{   }      -> EMPTY
\isempty{\mt}      -> FULL
\isempty{ E }      -> FULL
\isnotempty{}      ->
\isnotempty{  }    ->
\isnotempty{ \mt } -> FULL
```

The ifmtarg package originally had a much simpler approach until Donald Arseneau pointed out the error of my ways. The perils of empty were discussed in the late Michael Downes' *Around the Bend* series; the one in question is available from CTAN in `info/aro-bend/answer.002`

> Faultily faultless, icily regular, splendidly null,
> Dead perfection, no more.
>
> *Maud*, Alfred, Lord Tennyson

2 The usefulness of nothing

Another respondent on `texhax` [1] wanted an even-page version of LaTeX's `\cleardoublepage`. It might appear that a `\cleardoublepage`, which will get you to the next odd-numbered page, followed by a

[1] My wife was a close second.

\clearpage or \newpage will then get to an even-numbered page, but this is not so as you will find that you can't move on from a page with nothing on it (excepting headers and footers). What is required is something that appears to be a nothing or a null but which is not, so far as TEX is concerned. For the purposes at hand an empty box will do. TEX has a \null command, which is shorthand for an empty horizontal box, but we can use something with wider applicability which I will name \nowt.[2]

```
\newcommand*{\nowt}{\leavevmode\hbox{}}
%% \nowt <==> \mbox{}
\newcommand{\cleartoevenpage}[1][\@empty]{%
  \clearpage
  \ifodd\c@page
    \nowt\ifx\@empty#1\else #1\fi
    \newpage
  \fi}
```

This clears the current page and if the next is not an odd one then the task is finished. Otherwise we put (the invisible) \nowt on the odd page we have reached and move on to the next one, which will be even. The optional argument can be used to put some text or illustration on the skipped over odd page. For instance:

```
\cleartoevenpage[%
 \vfill\centering THIS PAGE LEFT BLANK\vfill
 \thispagestyle{empty}]
```

where the phrase 'THIS PAGE LEFT BLANK' will be centered on the odd page, and there will be neither a header nor a footer.

If you have ever tried something like this:

```
\begin{description}
\item[Nothing] \\
  Also known as \ldots
...
```

then you probably got an error message saying:
`There's no line to end here.`
This can be resolved by putting \nowt just before the \\ newline command.

> We may be in some degree whatever character we choose.
>
> *London Journal*, James Boswell

3 Picking characters

A texhax reader [4] wanted a macro that would ensure that the first letter of a string would be in uppercase. Various answers were supplied and I'm providing a couple of my own. All the solutions depend on the fact that a TEX macro takes as a single argument either braced text or a single token, where a token is either a command name (the name of a macro) or a single character. Further, when defining a TEX macro the argument list is ended by a token, which is usually the initial opening brace of the definition.

Here's my first, long winded solution.

```
\def\gettwo#1#2\nowt{%
  \gdef\istchar{#1}\gdef\restchars{#2}}
\def\splitoff#1{\gettwo#1\nowt}
\def\Upfirst#1{\splitoff{#1}%
  \MakeUppercase{\istchar}\restchars}
```

\gettwo expects two arguments with the end of the second denoted by '\nowt' (I have chosen this on the assumption that it will not be part of either argument; any other command name that would not be in the arguments would serve as well). The macro \splitoff takes a single (string) argument and passes it on to \gettwo, which then takes the first character in the string as its first expected argument, and the rest of the string as its second argument. It globally defines \istchar and \restchars as the two arguments. \Upfirst takes a string argument, calls \splitoff, and hence \gettwo, and then ensures that \istchar is typeset in uppercase, followed by the rest of the characters.

This does not work if the argument to \Upfirst is a macro that is defined as a string (for example \def\arg{string}). This can be resolved by using TEX's \expandafter command to make sure that \Upfirst's argument is expanded[3] before being used by \splitoff:

```
\def\Upfirst#1{%
  \expandafter\splitoff\expandafter{#1}%
  \MakeUppercase{\istchar}\restchar}
```

The second version, below, is not as versatile as the first as the string is consumed internally instead of being made available in the form of the \istchar and \restchars macros.

```
\def\upperfirst#1#2\nowt{%
  \MakeUppercase{#1}\MakeLowercase{#2}}
\def\Upfirst#1{\expandafter\upperfirst#1\nowt}
```

The basic idea is the same as the first proposal. It has the added function of ensuring that only the first character in the string is uppercase (it lowercases the remainder). Neither solution can handle the case where the first character is a ligature (e.g., \oe) or accented (e.g., \^{a}), or other commands.

Uwe Lück [3] provided a more complete but more complex solution.

```
\DeclareRobustCommand{\Upfirst}[1]{%
```

[2] 'Nowt' is a Northern English dialect word meaning naught or nothing as in "Y' can't get owt fer nowt" — You can't get something for nothing.

[3] To one level only.

```
  \protected@edef\upfirst@rg{#1}%
  \expandafter\upit\upfirst@rg\nowt}
```

Using `\DeclareRobustCommand` instead of `\def` or `\newcommand` ensures that `\Upfirst` can be used in a moving argument without having to be protected. The `\protected@edef` is used to expand the argument while maintaining any `\protect`s. In order to handle an accented initial character the string has to be split into three parts: the first element (which may be a character or an accent command), the second (which may be the argument to an accent command), and the third is the remainder of the string. The string, by the way, must have at least two characters.

```
\def\upit#1#2#3\nowt{%
  \let\@uptokone#1%
  \let\@xuptoktwo\@empty
  \def\@yuptoktwo{#2}%
  \expandafter\test@ccent\@ccentlist\@sentinel
  \MakeUppercase{#1\@xuptoktwo}%
  \MakeLowercase{\@yuptoktwo#3}}
```

The `\upit` macro takes three arguments, which are then the three portions of the initial string, and stores the first two in `\@uptokone` and `\@yuptoktwo` respectively. The macro `\test@accent` determines if the first token is an accent, changing `\@xuptoktwo` and `\@yuptoktwo` if it is.

```
\def\@ccentlist{‘>\"\‘\’\b\c}% plus the rest
\def\test@ccent#1{%
  \ifx#1\@sentinel\else
    \ifx\@uptokone#1
      \let\@xuptoktwo\@yuptoktwo
      \let\@yuptoktwo\@empty
    \fi
    \expandafter\test@ccent
  \fi}
```

`\@ccentlist` is a list of the accent commands; if a string is likely to start with an analphabetic character, such as an opening quote (‘), then these characters should also be included in the list.

The `\test@ccent` macro iterates through the list of accent commands and characters supplied as its argument and if there is a match with `\@uptokone` then it swaps the `\@xuptoktwo` and `\@yuptoktwo` values. The end result is that if the initial string starts with an accent then `\@xuptoktwo` has the accented character and `\@yuptoktwo` is empty, otherwise `\@xuptoktwo` is empty and `\@yuptoktwo` has the second character in the string.

Following are some examples using the last definition of `\Upfirst`.

```
low UP \& \Upfirst{low UP} ->
```
low UP & Low up

```
\def\stuff{rAnDoM 26 sTuFf}
\stuff{} \& \Upfirst{\stuff} ->
```
rAnDoM 26 sTuFf & Random 26 stuff

```
\oe{}rstead \& \Upfirst{\oe{}rstead} ->
```
œrstead & Œrstead

```
\c{c}edilla \& \Upfirst{\c{c}edilla} ->
```
çedilla & Çedilla

```
\emph{strong} \& \emph{\Upfirst{strong}} ->
```
strong & *Strong*

```
‘quote’ \& \Upfirst{‘quote’} ->
```
‘quote’ & ‘Quote’

```
>que? \& \Upfirst{>que?} ->
```
¿que? & ¿Que?

As always, if you are doing things with macros that include `@` in their name, either put the code into a package (`.sty`) file or enclose the code in a `\makeatletter ... \makeatother` pair.

Perhaps next time I'll take a look at traversing a string character by character and other kinds of looping macros but on the other hand, perhaps not.

References

[1] Susan Dittmar. Variant of `\cleardoublepage` starting on even page numbers. Post to `texhax` mailing list, 18 August 2005.

[2] Adam Fenn. Empty arguments. Post to `texhax` mailing list, 17 August 2005.

[3] Uwe Lück. Re: [texhax] read and process single characters. Post to `texhax` mailing list, 24 June 2005.

[4] Torsten Wagner. Read and process single characters. Post to `texhax` mailing list, 24 June 2005.

[5] Peter Wilson. Glisterings. *TUGboat*, 22(4):339–341, December 2001.

[6] Peter Wilson. Glisterings. *TUGboat*, 25(2):201–202, 2004.

◇ Peter Wilson
18912 8th Ave. SW
Normandy Park, WA 98166
USA
herries.press (at) earthlink.net