As examples, here are two representative entries from the draft of the Index.

Name: btxmac.tex
Description: Provides support for using BIBTEX with plain TEX.
Keywords: plain TEX, BIBTEX, bibliography
Author: Karl Berry and Oren Patashnik ⟨opbibtex@cs.stanford.edu⟩
Supported: yes
Latest Version: v0.99j, 14 Mar 1992
Archives: labrea*, ymir

Name: longtable.sty
Description: LATEX style option defining a multi-page version of tabular.
Keywords: LATEX, array, tabular, page
Author: David Carlisle ⟨carlisle@cs.man.ac.uk⟩
Supported: yes
Latest Version: v3.1, 6 Apr 1992
Archives: shsu*
Note: Documentation requires Mittelbach's doc.sty.
See Also: supertab.sty

The current draft of the Index (dated June 1, 1992) has approximately 600 entries. I hope to increase that to 1000 by the end of June, when I plan to release the Index to the general public by making it available by anonymous ftp and mail server. Beginning in July at the Annual TEX Users Group Meeting, the Index will also be distributed through TUG. In the meantime, I'll be contacting the authors of macro packages and requesting their help in verifying the information I have. If you have written a macro package that you think should be mentioned in the Index, please contact me (preferably by electronic mail) at the address below.

◇ David M. Jones
MIT Laboratory for Computer Science
Room NE43-316
545 Technology Square
Cambridge, MA 02139
Internet:
dmjones@theory.lcs.mit.edu

Tutorial

Names of control sequences

Victor Eijkhout

1 Introduction

In the 'Lollipop' format that I wrote, first to typeset my ph.d. thesis, then to set my book 'TEX by Topic' (Addison-Wesley 1992), I try to move away a bit from the ordinary TEX syntax. For instance, declaring a `\newskip` register, and setting the value of it are done using only one command, with the syntax

```
\Distance:UnitQuad=12pt
\Distance:parindent=UnitQuad
```

The first command here declares a skip register `\UnitQuad`, and initializes it to `12pt`; the second takes the csparindent and sets it to the value of `\UnitIndent`.

In order to perform these actions correctly, we should be able to distinguish

1. whether a control sequence is already defined (`\parindent`) or not (`\UnitQuad`), and
2. whether a string is the name of a control sequence (`UnitQuad`) or a litteral string (`12pt`).

Both problems are really the same, as we shall see below.

2 Messing with \csname

The matched pair of control sequences `\csname` and `\endcsname` can be used to construct control sequences out of arbitrary characters. Ordinarily, names of control sequences are limited to letters only (or, to be more precise, to characters of category 11), but in between these two commands any character can appear. Macros and other expandable commands are also allowed, as long as they will ultimately expand to characters.

For instance

```
\csname a:b\endcsname
```

expands to a control sequence with a colon in the name, and

```
\csname \ifhmode h\else v\fi skip\endcsname
```

expands to either `\hskip` or `\vskip`.

A useful property of `\csname` is that if you form the name of a control sequence that has no definition (that is, it is no primitive, register, macro, or otherwise defined), the result is equivalent to `\relax`. Thus

```
\csname probably:not=defined!\endcsname
```

is with a high likelihood equivalent to `\relax`. We can use this property to test whether a control sequence has been defined: if it hasn't it is equivalent to `\relax`.

For this test we use `\ifx` which tests equality of control sequence definitions[1]. For instance

```
\let\HorizontalContainer=\hbox
\ifx\hbox\HorizontalContainer % is true
\def\a{<>} \def\b{<>}
\ifx\a\b % is true
```

In order to see if a control sequence has been defined, we have to compare it to `\relax`.

Suppose we want to have a macro that can be called

```
\ifUndefined{maybe:macro} .. \else .. \fi
```

We can defined this as

```
\def\ifUndefined#1{\expandafter\ifx
    \csname#1\endcsname\relax}
```

The `\expandafter` activates the `\csname` to form the control sequence name, and `\ifx` then compares it to `\relax`. Note that we have actually defined a macro that tests whether a control sequence is *un*defined.

We can now start assembling the macro `\Distance`.

```
\def\Distance:#1=#2
   {\ifUndefined{#1}\MakeNewSkip{#1}{#2}
    \else \SetOldSkip{#1}{#2}
    \fi}
```

(Note that with this definition the second parameter is a string delimited by a space, for instance the space resulting from the line end.) So far we have glossed over one point: the value that is assigned (parameter 2) can be either a value or again the name of a control sequence. It makes sense then to define

```
\def\ValueOf#1{\ifUndefined{#1}#1
    \else \csname#1\endcsname \fi}
```

which takes the argument itself if it is not the name of a control sequence, and otherwise forms that control sequence.

Now `\SetOldSkip` is easy:

```
\def\SetOldSkip#1#2{%
    \csname #1\endcsname=\ValueOf{#2}}
```

For `\MakeNewSkip` we first need to allocate a new skip:

```
\def\MakeNewSkip#1#2{%
    \expandafter\newskip
      \csname#1\endcsname
    \SetOldSkip{#1}{#2}}
```

Just a small remark here: the `\newskip` macro of plain TeX has been declared with the prefix `\outer`, so it cannot be used the way it was done above. In order to write the above code, the definition of `\newskip` has to be copied from plain TeX, but without `\outer`.

3 And now what?

The macro `\Distance` explained above is not exactly spectacular, but I hope that the readers have learned some new tricks about control sequences. Furthermore, I will be using the techniques explained here in forthcoming articles about certain parts of my Lollipop format.

◇ Victor Eijkhout
Department of Computer Science
University of Tennessee at
Knoxville
Knoxville TN 37996-1301
Internet: eijkhout@cs.utk.edu

[1] It can also be used to test characters, but that's not relevant here.

Puzzle

Where does this character come from?

Frank Mittelbach

Puzzle:

If some complex macro defined by you produces funny extra characters like "Ω" or "œ" in the output, what kind of mistake could be the reason?

◇ Frank Mittelbach
Electronic Data Systems (Deutschland)
GmbH
Eisenstraße 56 (N15)
D-6090 Rüsselsheim
Federal Republic of Germany
Mittelbach@mzdmza.zdv.Uni-Mainz.de