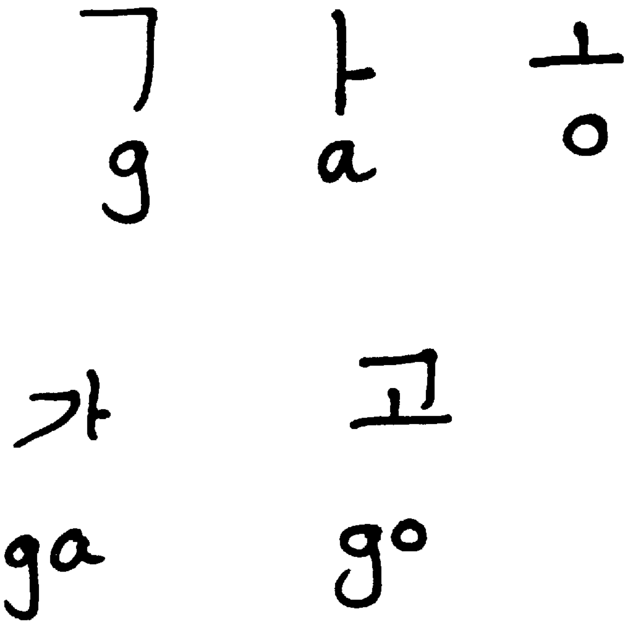

Figure 36: *Creating Korean type.*

Then Metafont can create a special purpose font for the text. There is room for 256 characters in a font, which is roughly equivalent to a page of text in a typical book. We expect, therefore, that it will take somewhere on the order of 200 special fonts to typeset a typical novel.

7 Conclusion

I have discussed a couple of ways in which when TEX and Metafont work together, the results are more than either could do singly. *That* is the point to stress, rather than the applications themselves. I hope listeners will take this lesson to heart and develop their own TEX-Metafont applications.

Getallen

David van Leeuwen

david@rulgm0.leidenuniv.nl

Inleiding

Een groot priemgetal is miljardtweehonderdvierendertigmiljoenvijfhonderdzevenenzestigduizendnegenhonderdnegenenveertig, en dat is op zich niet zo bijzonder, zij het dat ik het als `\getal 1234567949` heb ingetikt. Getallen uitspellen in het Nederlands is mogelijk, dankzij de 'logische' opzet. Natuurlijk zijn getallen tot veertien uitzondering (zoals in veel talen getallen onder de twintig), kennelijk gebruiken we die getallen vaak, en beneden de honderd hebben we die vreemde omkering van tiental en eenheid, telefoonnummers moeten we dan ook *nooit* groeperen in twee cijfers, want dat kan tot verwarring leiden. Maar voor grotere getallen groeperen we netjes in drie cijfers (die we ook eigenlijk in de notatie zouden moeten schrijven, bij voorkeur met *thinspaces*, als in 1 234 567 949), en tellen we duizendtallen, miljoenen, miljarden, etcetera.

De code

Laten we nu eens naar de code krijgen die ik in bovenstaand voorbeeldje heb gebruikt. Eerst definiëren we wat tellertjes en een slik-in macro, die er voor zorgt dat ik `\getal` kan gebruiken als plain TeX's `\number`:

```
\newcount\i\newcount\j\newif\ifhyphen              % hulp tellertjes
\def\getal{\hyphenfalse\afterassignment\nul\i=}    % slik in macro
\def\nul{\ifnum\i=0 nul\else\getall\fi}            % nul lijkt bijzonder
```

Het te kraken getal staat nu in `\i` (dit is nu even geen ı!) Het probleem is verlegd naar `\getall`, en het vervelende getal 0 is al behandeld. Als we weten dat `\i` een 1000-tal bevat, kunnen we dit 1000-tal uitschrijven, met 'duizend' erachter. Het probleem is dan verlegd tot het uitschrijven van een getal onder de duizend. Natuurlijk geldt dit ook voor 1 000 000-tallen, en zelfs voor 100-tallen. Daarom introduceren we de macro `\num` ⟨tal⟩ ⟨schrijfwijze⟩, die een ⟨tal⟩-tal afsplitst van het (hulp)getal `\j` (= `\i`), dat uitschrijft, daar ⟨schrijfwijze⟩ achteraanzet, en de rest als probleem `\i` achterlaat:

```
\def\num#1 #2 {\divide\j by #1 \ifnum\j>1 {\i=\j\getall}\fi
  #2\multiply\j by #1 \advance\i by -\j\getall}
```

Hier zie je een voorbeeld van het gebruik van locale registers met TeX, binnen de haken `{...}` moeten we een deelprobleem oplossen, en veranderen tijdelijk de waarde van `\i`. Een ander probleem bij het uitschrijven van getallen is dat we van die lange woorden krijgen, we moeten ze dan zeer waarschijnlijk afbreken. Voor de veiligheid zetten we `\-` tussen elke lettergreep, maar dit mag niet als eerste, we hebben een vlag nodig die ons vertelt of we de eerste lettergreep al gezet hebben. De macro `\?` zet mogelijk een afbreukstreepje

```
\def\?{\ifhyphen\-\else\global\hyphentrue\fi}          % mag afbreukstreepje?
```

Dan nu de macro `\getall`. Bij aankomst staat het uit te schrijven getal in het register `\i`. We beginnen met de kleinste getallen:

```
\def\getall{\ifcase\i\or \?een\or \?twee\or \?drie\or \?vier\or \?vijf\or
  \?zes\or \?ze\-ven\or \?acht\or \?ne\-gen\or \?tien\or \?elf\or \?twaalf\or
  \?der\-tien\or \?veer\-tien\else \groterdanveertien \fi}
```

Wel, dat was eenvoudig maar saai. Vervolgens moeten we getallen beneden de honderd behandelen. De tigheid is voldoende onregelmatig om hem helemaal uit te schrijven. 'Omkeren' gebeurt met een `\edef`. Tenslotte moeten eenheid en tigheid aan elkaar geplakt worden, met uitzonderingen voor twee, drie en tien. (Neem aan dat `"` een goede trema-macro voorstelt.)

```
\def\groterdanveertien{\j=\i          % hulp register
 \ifnum\i<100 \divide\j by 10         % \j is tiental
  \edef\tig{\ifcase\j\or tien\or twin\-tig\or
    der\-tig\or veer\-tig\or vijf\-tig\or zes\-tig\or ze\-ven\-tig\or
    tach\-tig\or ne\-gen\-tig\fi}     % en \tig is het uitgeschreven tiental
  \multiply\j by 10 \advance\i by-\j\getall  % \i is de rest, de eenheid, <10
  \ifnum\j=10 \else\ifcase\i\or\-en\or"en\or"en\else\-en\fi\fi\?\tig % 2,3->"
 \else\groterdanhonderd\fi}
```

Wel, als we dit allemaal kunnen, kunnen we de macro `\num` gebruiken om de duizendtallen en zo uit te schrijven:

```
\def\groterdanhonderd{\ifnum\i<1000 \num100 \?hon\-derd   % 100<=\i<1000
 \else\ifnum\i<1100 \num1000 \?dui\-zend % 1000<=\i<1100, `duizend...'
  \else\ifnum\i<2000 \num100 \?hon\-derd % 1100<=\i<2000, `...tienhonderd...'
   \else\ifnum\i<1000000 \num1000 \?dui\-zend          % 1000<=\i<1000000
    \else\ifnum\i<1000000000 \num1000000 \?mil\-joen
     \else\num1000000000 \?mil\-jard
 \fi\fi\fi\fi\fi}
```

Zo kunnen we op een eenvoudige wijze grote getallen opschrijven, die we bijvoorbeeld hebben uitgerekend. Dit voorkomt fouten bij overtikken.

Een voorbeeld

Fibonacci getallen f_n zijn te vinden door $f_n = f_{n-1} + f_{n-2}$, en de eerste paar zijn in het Nederlands uitgeschreven achtereenvolgens:

0: nul; 1: een; 1: een; 2: twee; 3: drie; 5: vijf; 8: acht; 13: dertien; 21: eenentwintig; 34: vierendertig; 55: vijfenvijftig; 89: negenentachtig; 144: honderdvierenveertig; 233: tweehonderddrieëndertig; 377: driehonderdzevenenzeventig; 610: zeshonderdtien; 987: negenhonderdzevenentachtig; 1597: vijftienhonderdzevenennegentig; 2584: tweeduizendvijfhonderdvierentachtig; 4181: vierduizendhonderdeenentachtig; 6765: zesduizendzevenhonderdvijfenzestig; 10946: tienduizendnegenhonderdzesenveertig; 17711: zeventienduizendzevenhonderdelf; 28657: achtentwintigduizendzeshonderdzevenenvijftig; 46368: zesenveertigduizenddriehonderdachtenzestig; 75025: vijfenzeventigduizendvijfentwintig; 121393: honderdeenentwintigduizenddriehonderddrieënnegentig; 196418: honderdzesennegentigduizendvierhonderdachttien; 317811: driehonderdzeventienduizendachthonderdelf; 514229: vijfhonderdveertienduizendtweehonderdnegenentwintig; 832040: achthonderdtweeëndertigduizendveertig; 1346269: miljoendriehonderdzesenveertigduizendtweehonderdnegenenzestig; 2178309: tweemiljoenhonderdachtenzeventigduizenddriehonderdnegen; 3524578: driemiljoenvijfhonderdvierentwintigduizendvijfhonderdachtenzeventig; 5702887: vijfmiljoenzevenhonderdtweeduizendachthonderdzevenentachtig; 9227465: negenmiljoentweehonderdzevenentwintigduizendvierhonderdvijfenzestig; 14930352: veertienmiljoennegenhonderddertigduizenddriehonderdtweeënvijftig; 24157817: vierentwintigmiljoenhonderdzevenenvijftigduizendachthonderdzeventien; 39088169: negenendertigmiljoenachtentachtigduizendhonderdnegenenzestig; 63245986: drieënzestigmiljoentweehonderdvijfenveertigduizendnegenhonderdzesentachtig; 102334155: honderdtweemiljoendriehonderdvierendertigduizendhonderdvijfenvijftig; 165580141: honderdvijfenzestigmiljoenvijfhonderdtachtigduizendhonderdeenenveertig; 267914296: tweehonderdzevenenzestigmiljoennegenhonderdveertienduizendtweehonderdzesennegentig; 433494437: vierhonderddrieëndertigmiljoenvierhonderdvierennegentigduizendvierhonderdzevenendertig; enzovoorts.

International quotations

Johannes Braams

j.l.braams@research.ptt.nl

Introduction

At the conference in Portland last year, Christina Thiele asked me if I was willing to write an article for ttn about quotation marks. She said that she suspected that there would be many people out there who would want to know how to produce quotation marks in a little piece of, say, french, text that they include in their document that is otherwise written in english. All this is of course covered in the babel language specific style files, but for those who don't want to use babel but do need the occasional quotation mark I wrote this article.

Let me start with a disclaimer: I am not familiar with all the typographic conventions that are in use in the various countries all over the world. Also I can only give you macros for those languages for which a language specific babel file exists. So, there may be more conventions. If you know of them: please inform me, so that I can enlarge my collection.

Yet another disclaimer: If you have access to fonts with the EC (or Cork) encoding, you do not need most of the hackery in this article. These fonts include the characters that are needed so they need not be constructed.

The first part of this article[1] deals with the macros needed to typeset the quotation marks needed for various languages. First I'll introduce some internal hackery, then the macros that really produce quotation marks. I also give some shorthands to make the macros easy to use. Finally I will give some examples of the use of the macros. In the examples I will show the source text as well as the typeset result.

Some help macros

Because the macros in this part of the code are not intended for use in a document we'd like to 'hide' them from the user. Usually this is done by changing the category code of an 'other' character to letter. More often than not the '@' is chosen for this purpose. Sometimes the underscore is used for this purpose as well. Here I too will use the '_'. Of course the category code needs to be reset later so we store the current value in `\uscatcode`.

```
\chardef\uscatcode=\catcode`\_
\catcode`\_=11\relax
```

In some languages we need to lower quotation marks to the baseline. For this purpose we use the macro `\set_low_box`. It has one argument and uses `\box0` to gives us its result. This macro comes from `german.tex`.

It first typesets a comma in box register 2 and its argument in box register 1. Then it computes the distance that the argument has to be lowered to reach the baseline. Finally it lowers the contents of box register 0 — using box register 0 again for the result — and adjusts the values for the height and depth of the box.

```
\def\set_low_box#1{\setbox2\hbox{,}%
  \setbox0\hbox{#1}%
  \dimen0\ht0 \advance\dimen0 -\ht2%
  \setbox0\hbox{\lower\dimen0 \box0}%
  \ht0\ht2 \dp0\dp2}
```

Using macros for quotation marks sometimes disturbes the spacefactor. Therefore we also need a macro to preserve it. This macro also stems from `german.tex`. It first checks if it is executed in horizontal mode, if that is the case the current spacefactor is stored in the macro `\_SF`. Outside horizontal mode the macro `\_SF` is empty. Then `\save_sf_q` typesets its argument and resets the spacefactor.

```
\def\save_sf_q#1{{\ifhmode
  \edef\_SF{\spacefactor\the\spacefactor}
  \else
  \let\_SF\empty \fi \leavevmode #1\_SF}}
```

Producing the quotation marks

In languages such as Dutch, German and Czech, the opening quotes are traditionally typeset at the baseline.

```
\def\loq{\protect\_loq}
\def\_loq{\save_sf_q{\set_low_box{''}%
\box0\kern-.04em}}
```

In Germany also the closing quotes are different from what is provided by TEX. They use something that looks like the english opening quotes as closing quotes. Obviously, if one didn't do anything about it, the spacing would be wrong. Therefore we need yet another macro, `\icq`.

```
\def\icq{\protect\_icq}
\def\_icq{\save_sf_q{%
           \kern-.07em``\kern.07em}}
```

[1] When the source of this article is run through LATEX it produces a file called `quoting.tex` which contains all the code that is described and used in the article.

In french typography, a very different kind of quotes is used, the so called 'guillemets'. To realise these guillemets, various macros have been floating around the net. According to the `french.sty` package by Bernard Gaulle, one of the oldest definitions is:

```
\def\oog{\protect\_oog}
\def\_oog{\leavevmode\ifdim\lastskip>0pt%
          \unskip\penalty-9\hskip0.35em %
                          minus 0.35em\fi
          \raise .27ex\hbox{%
             $\scriptscriptstyle\ll$}%
          $\,$\nobreak\ignorespaces}
\def\ocg{\protect\_ocg}
\def\_ocg{\leavevmode\ifdim\lastskip>0pt%
          \unskip\penalty10000\fi
          \nobreak$\,$\leavevmode
          \raise .27ex\hbox{%
          $\scriptscriptstyle\gg$}}
```

But, this does not *really* give the result that the french would like. Therefore, if you have access to the LaTeX-symbol fonts it is better to use the following definition for the guillemets:

```
\chardef\lg='050
\chardef\rg='051
\def\og{\protect\_og}
\def\_og{\hbox{\ly\lg\kern-0.2em\lg%
                        \kern+0.2em}}
\def\cg{\protect\_cg}
\def\_cg{\hbox{\ly\kern+0.2em\rg%
                    \kern-0.2em\rg}}
```

In the code above the control sequence `\ly` is used. It is an internal macro from the old font selection schem in LaTeX. When you use the nfss you will have to define it:

```
\ifx\undefined\selectfont
\else
  \def\ly{\fontfamily{lasy}\fontseries{m}
          \fontshape{n}\selectfont}
\fi
```

Easy usage

In the previous section a couple of macros have been defined to make it possible to use various quotes. But it would be nice if one didn't have to do so much typing each time they are used. To provide easy acces, it is common use in language specific files to introduce active characters. For the macros presented here, we could introduce three active characters, the `"`, < and the >.

```
\def\dq{"}\catcode`\"=\active
\def\lt{<}\catcode`\<=\active
\def\gt{>}\catcode`\>=\active
```

As you may have noticed, I saved a copy of the non active version of each character in a control sequence. These are needed later on, when the active character has inspected its argument and decides that it needs to insert the non-active version of itself.

We use the active `"` to acces the low opening quotes and the german closing quotes, the other two are used to produce the guillemets. Here is the definition of the active characters:

```
\def"#1{\ifx#1`\loq{}\else
        \ifx#1'\icq{}\else\dq#1
        \fi\fi}
\def<#1{\ifx#1<\ifmmode\lt\lt\else%
                 \og{}\fi\else\lt#1\fi}
\def>#1{\ifx#1>\ifmmode\gt\gt\else%
                 \cg{}\fi\else\gt#1\fi}
```

But, be carefull when introducing new active characters. You have to make sure that they get deactivated at the right moment. Therefor we need to add them to macros such as `\dospecials` and — in case you use LaTeX — `\@sanitize`. A safe way of doing this was found by Bernd Raichle. It involves using an extra macro `\add_special`.

```
\chardef\atcatcode=\catcode`\@
\catcode`\@=11\relax
\def\add_special#1{\begingroup
    \def\do{\noexpand\do\noexpand}
    \def\@makeother{%
        \noexpand\@makeother\noexpand}
    \edef\x{\endgroup
      \def\noexpand\dospecials{%
                     \dospecials\do#1}
      \expandafter\ifx\csname%
               @sanitize\endcsname\relax
                   \else
        \def\noexpand\@sanitize{%
                \@sanitize\@makeother#1}
      \fi}
    \x}
\catcode`\@=\atcatcode\relax
```

Once that macro is defined we use it to tell TeX to treat our active characters with caution.

```
\add_special\"
\add_special\<
\add_special\>
```

Wrapping up

These macros were defined while the category code of the '_' was changed. It must not be forgotten to 'undo' that change:

```
\catcode`\_=\uscatcode\relax
```

Examples

A quotation in Dutch might look like this:

```
Hij zei: "`Ga je mee?''.
```

Hij zei: „Ga je mee?”.

But in French (using the definition that uses the LaTeX symbol font) it would be:

```
Il disait: << Tu va? >>.
```

Il disait: « Tu va? ».

Whereas in Italian it might look like:

```
Lui dice: >>Andiamo?<<.
```

Lui dice: »Andiamo? « .

To show the difference between the two definitions given for the guillemets, this is wat the 'old' version looks like:

```
Il disait: \oog\ Tu va? \ocg.
```

Il disait: $\ll$ Tu va? $\gg$.