TEX beauties and oddities

A permanent call for TEX pearls

http://www.gust.org.pl/pearls

What is wanted:

- ▹ short TEX, METAFONT or MetaPost macro/macros (half an A4 page or half a screen at most),
- ▹ the code should be generic; potentially understandable by `plain`-oriented users,
- ▹ results need not be useful or serious, but language-specific, tricky, preferably non-obvious,
- ▹ obscure oddities, weird TEX behaviour, dirty and risky tricks and traps are also welcome,
- ▹ the code should be explainable in a couple of minutes.

The already collected pearls can be found at `http://www.gust.org.pl/pearls`. All pearl-divers and pearl-growers are kindly asked to send pearl-candidates to `pearls@gust.org.pl`, where Paweł Jackowski, our pearl-collector, is waiting impatiently. The pearl marketplace is active during the entire year, not just before the annual BachoTEX Conference.

Note: The person submitting pearl proposals and/or participating in the BachoTEX pearls session need not be the inventor. Well known hints are also welcome, unless already presented at one of our sessions.

Since some seasoned TEX programmers were indignant at calling ugly TEX constructs “Pearls of TEX programming”, we decided not to irritate them any longer. We hope they will accept “TEX beauties and oddities” as the session title.

If you yourself have something that fits the bill, please consider. If you know somebody’s work that does, please let us know, we will contact the person. We await your contributions even if you are unable to attend the conference. In such a case you are free either to elect one of the participants to present your work or “leave the proof to the gentle reader” (cf., e.g., `http://www.aurora.edu/mathematics/bhaskara.htm`).

A TEX quine (*Péter Szabó*)

The code producing itself:

```
\def\T{
\tt \hsize 32.5em\parindent 0pt\def \S {\def \S ##1>{}}\S \string
\def \string \T \string {\par \expandafter \S \meaning \T \string
}\par \expandafter \S \meaning \T \footline {} \end }
\tt \hsize 32.5em\parindent 0pt\def \S {\def \S ##1>{}}\S \string
\def \string \T \string {\par \expandafter \S \meaning \T \string
}\par \expandafter \S \meaning \T \footline {} \end
```

Multi-signed numbers (*Hans Hagen*)

TeX handles multiple signs properly:

```
    \newdimen\scratchdimen
    \scratchdimen 1pt \the\scratchdimen,
    \scratchdimen -1pt \the\scratchdimen,
    \scratchdimen --1pt \the\scratchdimen,
    \scratchdimen ---1pt \the\scratchdimen,
    \scratchdimen -+-+-+++-----+1pt \the\scratchdimen,
```

So, there is never a need to use an intermediate variable to negate a value. All digits, +/− signs and units can be faked in macros:

```
    \def\neg{-} \def\p{p}
    \scratchdimen \neg\space\neg\space\space00001\empty\p\empty\empty tttt
```

One may also notice that while whitespace characters are allowed between multiple signs (but not between digits!), leading zeros are ignored, and the unit is properly interpreted regardless of the very next character.

Double-hat trap (*Jerzy Ludwichowski*)

Is there a difference between those two cases?

```
    \number`\^^A
    \number`^^A
```

And how about this?

```
    \number`\^^@
    \number`^^@
```

In the case of `^^A` (character code 1), both lines yield the number 1, the backslash character's presence before the double-hat doesn't influence the result. In the second case, the first line yields 0, while the second results in 32. The reason is that the character of the code 0 (`^^@`) has the associated category code 'ignored' (9). Any character of the category 9 will simply be omitted, except when there is a backslash immediately before it. If there is no backslash, the very next character is considered, which is a space (code 32), and `^^@` simply disappears. This does not happen with characters of category code different from 9.

`\vbox` height vs. `\vtop` depth (*Paweł Jackowski*)

`\vbox` usually inherits its depth from the last box or rule of the vertical list it contains. Conversely, `\vtop` has usually the height of the first box or rule of the vertical list it contains. However, using whatsits as the first/last item of the box may lead to surprises.

```
    \def\what{\special{}}
    \setbox0=\vbox{Aqq \what} \the\ht0, \the\dp0 % 6.83331pt, 1.94444pt
    \setbox0=\vtop{\what Aqq} \the\ht0, \the\dp0 % 0.0pt, 8.77776pt
```

`\vbox` still obeys the rule, despite the whatsit after the very last box on the list. But `\vtop` always has zero height if its first item is a whatsit.

(Ir)relevant missing character message (*Paweł Jackowski*)

Try out the code

```
\hsize=7.3in \vsize=9.8in \leftskip=30mm \rightskip=30mm \parindent=1em
\font\LOGO=logo10
\def\MP{{\LOGO METAPOST}}
\def\MF{{\LOGO METAFONT}}

short \TeX, \MF\ or \MP\ macro/macros (half A4 page or half a~screen at most)[...]
```

The output is typeset without breaking any word at the end of a line. Try then to explain why the log file contains the line:

```
Missing character: There is no - (45) in font logo10!
```

While breaking paragraphs into lines TeX checks all feasible breakpoints and chooses the one of the smallest sum of costs (see *The TeXbook*, chapter 14). The message in the log file informs that some of the ways TeX considered of typesetting the paragraph had a discretionary break after a `META`.

Skip assignments (*Paweł Jackowski*)

Consider the code:

```
\newskip\A
\newskip\B
\A = 3pt plus 1pt minus 1pt
\B = 1\A
```

Is now the skip `\B` equal to `\A`?

No, it's not:

```
\the\A % 3pt plus 1pt minus 1pt
\the\B % 3pt
```

In an assignment of the form

```
skip = <number> skip
```

TeX eliminates the stretch and shrink of the glue. To avoid this effect one should not use a number/factor ('1' in this case) on the right hand side of the equation. When necessary, one should use the `\advance`, `\divide`, `\multiply` primitives instead, since all they preserve the glue-specific parts.

Current font global assignment (*Bogusław Jackowski*)

Font setup is normally bounded by groups. The code

```
\font\A=ec-lmr10 \A \message{\the\font}
{\font\B=ec-lmtt10 \B \message{\the\font}}
\message{\the\font}
```

gives `\A \B \A`, as one would expect. Why then does

```
\font\A=ec-lmr10 \A \message{\the\font}
{\font\B=ec-lmr10 \B \message{\the\font}}
\message{\the\font}
```

yield `\A \B \B`?

When the font used inside a group is the same as the current font in the outer grouping level, the local font assignment becomes global. In fact, font `\A` is internally mapped to `\B`. Even if we call `\A` explicitly, TeX reports `\B` as the current font.

```
\A \message{\the\font}
```

Things are intentionally different in LuaTeX ...

How to make a box disappear at a line break (*Marcin Woliński*)

Let us consider the problem of marking spaces in a paragraph with some symbol, as in the following:

Ten · typowy · testowy · akapit · tekstu · daje · przy · okazji · rodzaj
filigranowego · wysypu · hodowli · pieczarek · w · zielonym · kaszta-
nie · regloryfikacji · stanowisk · ministerialnych · i · podsypanych
minimalistom · jako · fetysz · zaduchu · studziennych · barykad.

The hard part is to make the symbol disappear when such a "space" occurs at a line break. We cannot use `\discretionary` for that purpose since we need the "space" to be stretchable to make justification possible. Moreover we want to be able to associate some penalty (e.g., 0) with our breakpoints other than `\(ex)hyphenpenalty`.

As it turns out any box can be made discardable by putting it into `\cleaders` to the exact width of the box in question. According to the rules TEX will put exactly one copy of the box in the text. So the construct will look exactly like the box itself but will behave like a glob of glue. In particular it will disappear at a line break.

Here are the macros used in the preceding passage:

```
\obeyspaces
\def {%
\setbox0\hbox{$\cdot$}%
\dimen0=\wd0\relax
\hskip1ptplus2pt%
\cleaders\box0\hskip\dimen0%
\hskip1ptplus2pt%
}
\rm\hsize9.5cm\parindent0pt
Ten typowy testowy akapit tekstu daje przy okazji rodzaj filigranowego %
wysypu hodowli pieczarek w zielonym kasztanie regloryfikacji %
stanowisk ministerialnych i podsypanych minimalistom jako fetysz %
zaduchu studziennych barykad.%
```

Stretchability is achieved with separate globs of glue so as not to allow TEX to insert more than one copy of the box in case of an overstretched space.

Note that this trick can be used in vertical mode as well (e.g., to separate paragraphs with some graphical element except the case when a paragraph boundary occurs at a page break). A discardable box can have arbitrary complexity, it can include colour, EPS graphics, and so on.

Variable-width visible space (*Bogusław Jackowski*)

Marked spaces in a paragraph may not only disappear at a line break (as presented in the previous beauty by Marcin Woliński), but may also adjust their width, shrink and stretch, as normal interword space does.

```
\def\vispace{%
\ifdim\spaceskip=0pt
   \skip0=\fontdimen2\the\font
          plus \fontdimen3\the\font
          minus \fontdimen4\the\font
\else \skip0=\spaceskip \fi
\advance\skip0-.4pt
\cleaders\vrule width.2pt height.2ex depth.2pt\hskip.2pt
\cleaders\hrule height0pt depth.2pt\hskip\skip0
\cleaders\vrule width.2pt height.2ex depth.2pt\hskip.2pt
}

\obeyspaces\let =\vispace\def~{\nobreak\vispace}\let\ =\vispace%
% \def\^^M{\ } % plain does
Ten typowy testowy akapit tekstu daje przy okazji rodzaj filigranowego\
wysypu hodowli pieczarek w~zielonym kasztanie regloryfikacji\
stanowisk ministerialnych i~podsypanych minimalistom jako fetysz\
zaduchu studziennych barykad aglomeracji fosforescencji luminazy\
atraktywno-bajerywnej z~dodatkiem glukozy i~mineralnych bakterii\
finansowych oraz gromadzenia idei atrakcyjnych pomp prasowych z~okazji\
rozpoczynania wegetacji takich istot jak wiolonczele, napoje bazaltowe\
i~gramatyka z~okresu mezozoicznego z~jej typowym sposobem oznajmiania\
zachwytu nad bytem poprzez wycie i~popiskiwanie o~charakterystycznej\
modulacji toniczno-barycznej z~wyskokami w~kierunku reglamentacji\
zawartej immanentnie w~bagnie.
```

Ten␣typowy␣testowy␣akapit␣tekstu␣daje␣przy␣okazji␣rodzaj␣filigranowego␣wysypu␣hodowli␣pieczarek␣w␣zie-
lonym␣kasztanie␣regloryfikacji␣stanowisk␣ministerialnych␣i␣podsypanych␣minimalistom␣jako␣fetysz␣zaduchu
studziennych␣barykad␣aglomeracji␣fosforescencji␣luminazy␣atraktywno-bajerywnej␣z␣dodatkiem␣glukozy␣i␣mi-
neralnych␣bakterii␣finansowych␣oraz␣gromadzenia␣idei␣atrakcyjnych␣pomp␣prasowych␣z␣okazji␣rozpoczynania
wegetacji␣takich␣istot␣jak␣wiolonczele,␣napoje␣bazaltowe␣i␣gramatyka␣z␣okresu␣mezozoicznego␣z␣jej␣typowym
sposobem␣oznajmiania␣zachwytu␣nad␣bytem␣poprzez␣wycie␣i␣popiskiwanie␣o␣charakterystycznej␣modulacji
toniczno-barycznej␣z␣wyskokami␣w␣kierunku␣reglamentacji␣zawartej␣immanentnie␣w␣bagnie.

Do you need some stretch? (*Marcin Woliński*)

TEX's `\leaders` primitive can be used to fill arbitrary space with a stretchable line (cf. `\hrulefill`). It is also possible to have an expandable triple line:

The St. Anford Orchestra

Variations on a Theme by Tchaikovsky

```
\def\triplefil{%
  \leaders\hrule height4pt depth-3.2pt\hfil \hskip0pt plus-1fil
  \leaders\vrule height1.6pt depth0pt\hfil \hskip0pt plus-1fil
  \leaders\vrule height-.6pt depth1pt\hfil }
\def\triplefilledline#1{\hbox to\hsize{%
    \vrule height4ptdepth3ptwidth.8pt \triplefil \vrule
    height10ptdepth1ptwidth.4pt \enspace\strut#1\enspace \vrule
    height10ptdepth1ptwidth.4pt \triplefil \vrule
    height4ptdepth3ptwidth.8pt } }
\triplefilledline{The St.\ Anford Orchestra}
\triplefilledline{Variations on a Theme by Tchaikovsky}
```

To understand what happens here one needs to count stretchability of leaders and glue in `\triplefil`. It is: 1fil (from `\hfil`) + −1fil (from `\hskip`) + 1fil + −1fil + 1fil, which sums up to 1fil. So when TEX needs to set `\triplefil` to, say, 37pt it stretches each fil of glue to that length. The first leaders become 37pt wide, then comes `\hskip` to −37pt (−1fil), and so TEX overprints the second `\leaders` on the first, and the same repeats with the next glue and leaders.

This trick opens space for countless variations:

The St. Anford Orchestra

Variations on a Theme by Tchaikovsky

MetaPost tables indexed with strings (*Bogusław Jackowski*)

Converting MetaPost strings to suffixes one can implement tables indexed with strings.

```
% Definitions:
def strtosfx(expr s) =
for i:=1 upto length(s): [ASCII(substring(i-1,i) of s)] endfor
enddef;
vardef sfxtostr_ []@# =
if (str @=""): "" else: char(@) if str @#<>"": & (sfxtostr_ @#) fi fi
enddef;
def sfxtostr(suffix s) = begingroup sfxtostr_ s endgroup enddef;

% A few tests:
show sfxtostr(strtosfx("ABCABCABCABCABCABCABCABCABCABCABCABC!"));

save X; X strtosfx("ABC") =0; showvariable X;
save X;
for s:="ala", "ma", "kotakotakota", "kota": X strtosfx(s) = 0; endfor
for s:="ala", "ima", "kota": if known X strtosfx(s): show s; fi endfor
end.
```

If only there were a way to iterate over all known indexes ...

Multiple expansions triggered with a single `\expandafter` (*Marcin Woliński*)

This pearl (coded on October 18, 1996) is the most useless one I could think of. Nonetheless it is an example of a really curious expansion of macros.

Let us imagine that we have a list of non-space tokens and we want to assign this list to a token register without expanding the tokens and in reversed order. Here is a simple macro that reverses a list in an expand-only way:

```
\def\afterfi#1#2\fi{\fi#1}

\def\reverse#1{\reverseX{}#1\stopreverse}
\def\stopreverse{\noexpand\stopreverse}

\def\reverseX#1#2{\ifx\stopreverse#2%
    \afterfi{#1}%
  \else
    \afterfi{\reverseX{#2#1}}%
  \fi}
```

Now we can write

```
\reverse{abcdefg}
```

and TEX will respond with writing `gfedcba` on the terminal.

To put the result of reversing the list `abc\foo def\bar ghi` in a token register we do the following:

```
\toks0=\expandafter{\if0\reverse{abc\foo def\bar ghi0}}\fi
\showthe\toks0
```

With the use of `\expandafter` we introduce a single expansion to the region where expansion is suppressed. The token being expanded is the `\if`. To expand an `\if` TEX needs to find the next two non-expandable tokens to compare them. The first token is `0`, but then TEX sees the macro `\reverse`. So the macro gets expanded. An interesting feature of `\reverse` is that no non-expandable tokens are emitted until the list is fully reversed. So only then does TEX stop expansion. The first non-expandable token TEX will see is the second `0`, which we have devilishly inserted at the end of the list. At this point the condition turns out to be true and the next tokens get assigned as contents to the token register.

Hacking verbatim (*Grzegorz Murzynowski*)

How do you get *italics* inside a verbatim? By a 'verbatim' I mean a LATEX environment that changes the catcodes of special chars and thus allows typesetting them verbatim (the tricks below apply to TEX in general, though). LATEX's `\begin{verbatim}` expands mostly to `\begingroup\csname verbatim\endcsname` and `\verbatim` acts mostly like DEK's `\ttverbatim`, `\end{verbatim}` is needed to delimit `\verbatim`'s argument.

Let's recall that the chars of codes 1–32 (except the end of line, etc.) are catcoded as 'invalid' in LATEX. Therefore I dare to assume they are neither used nor present in decent (LA)TEX files. The verbatim environments do not recatcode them, so I can use them for my wicked purpose:

```
\catcode`\^^E\active
\def^^E{\bgroup\it}
\let^^F\egroup
\begin{verbatim*}
How do you get ⟨char5⟩italics⟨char6⟩ inside a~verbatim?
\end{verbatim*}
```

Gives

```
How do you get italics inside a~verbatim?
```

Note that we should use explicit ⟨*char*5⟩ and ⟨*char*6⟩ since verbatims recatcode `^` to category 'other' so '`^^E`' would produce just `^^E`.

Now, how to input selected lines of a file verbatim?

```
\long\def\firstofone#1{#1}
\catcode`\@=11
\newread\my@file
\openin\my@file=bachotex2007-grzegorz-murzynowski-pearl1.src
\def\my@reading#1 #2{%
  \loop\ifnum\count\z@<#1%
    \advance\count\z@\@ne\read\my@file to\@tempa
  \ifx.#2\@tempa\endgraf\fi\repeat}%
\firstofone{%
  \begin{verbatim}%
  \count\z@\z@
  \my@reading1 -%
  \my@reading2 .%
  \my@reading22 -%
  \my@reading26 .%
}\end{verbatim}
```

The given code results in the following:

```

\def^^E{\bgroup\it}
\let^^F\egroup
\begin{verbatim*}
How do you get ΠitalicsΣ inside a~verbatim?
```

What is the most fundamental trick? The `\firstofone` macro (I learnt it from my TEX Guru who did not invent it either). Apparently it doesn't do anything: it has one parameter and expands exactly to it. But there is one very important thing it does: it 'freezes' the catcodes in the argument. Therefore all the commands and their arguments cannot be recatcoded by `\verbatim` and they are expanded and executed.

Custom overfull text (*Paweł Jackowski*)

How to replace a black overfull rule at the end of too long lines of a paragraph?

Well, there is no direct way to do so, but one should never underestimate TEX's bells and whistles. First of all, we can test if the last (h)box was overfull by checking the value of `\badness`; if it is larger then 10000 it definitely means that the box was overfull (`\badness` never exceeds 10000 for underfull boxes). Assuming that `\box0` is the box we want to test, we can say

```
\def\ooops{\hbox to\wd0{\setbox0=\hbox to\wd0{\unhbox0}%
  \unhbox0 \ifnum\badness>10000 \rlap{\sevenrm\quad Ooops!}\fi}}
```

And how to get the box that is the line of a paragraph? By setting the `\interlinepenalty` parameter to a large negative value we can force a page break between every two lines of a paragraph. In the `\output` routine, we can recognize those special penalties via the `\outputpenalty` parameter. The `\output` routine is not necessarily required to `\shipout` the page — it may simply return all its content back to the 'recent contributions'.

```
\interlinepenalty=-50000 % force the break between each two lines
\maxdeadcycles=50        % allow upto 50 \outputs with no \shipout
\newtoks\orioutput \orioutput=\output % wrap the original \output routine
\output
 {\ifnum\outputpenalty>-20000 \the\orioutput
  \else \ifnum\outputpenalty<-\maxdimen \the\orioutput
  \else
    \unvbox255        % flush the entire list back
    \setbox0=\lastbox % strip the very last box
    \nointerlineskip  % avoid doubled interline glue
    \ooops            % make the test and return the box back.
    \advance\outputpenalty by50000
    \penalty\outputpenalty % weak lie that nothing happened...
  \fi\fi}
\hfuzz=\maxdimen % no overfullrule, no messages
\hsize=1.5in     % provoke overfulls
...
```

This completely useless example
shows a not-so-useless trick, which Ooops!
might be used for quite advanced
applications, such as line-numbering, Ooops!
some kind of paragraph decora-
tion, page optimization and pro-
bably many others. Things become Ooops!
much more complicated if math
displays, `\marks`, `\inserts` or `\va-` Ooops!
`djusts` come into play, but they
don't spoil all of the game.

This completely useless example
shows a not-so-useless trick, which▮
might be used for quite advanced
applications, such as line-numbering,▮
some kind of paragraph decora-
tion, page optimization and pro-
bably many others. Things become▮
much more complicated if math
displays, `\marks`, `\inserts` or `\va-`▮
`djusts` come into play, but they
don't spoil all of the game.