Pearls of TEX Programming

The title of the BachoTEX 2005 conference is "The Art of TEX Programming," TAOTP for short, therefore the idea of a "Pearls of TEX Programming" session arose. Bogusław Jackowski came up with the session motto: "Behold"—Bhaskara (see, e.g., http://www.aurora.edu/mathematics/bhaskara.htm).

The idea was to invite TEXies known to be TEXperts, TEX Masters or perhaps even TEX Grandmasters[1], to contribute.

The call stated what was wanted:

- a short TEX, METAFONT or METAPOST macro/macros (preferably a few lines)
- results should be virtually useful yet not obvious
- easy to explain: 10 minutes at most

Martin Schröder

Colours separation in pdfTEX

```
\newcommand*{\AC@addColor}[5]{%
    \immediate\pdfobj stream
        attr {
            /FunctionType 4
            /Domain [0.0 1.0]
            /Range [0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0]
            }
        { { dup ?2 mul exch
            dup ?3 mul exch
            dup ?4 mul exch
            ?5 mul } }%
    \edef\AC@ColorFunctionObj{\the\pdflastobj}%
    \immediate\pdfobj {[ /Separation /?1
                         /DeviceCMYK
                         \AC@ColorFunctionObj\space 0 R ]}%
    \begingroup
        \toks@\expandafter{\AC@colorhook}%
        \edef\x{%
            \endgroup
            \gdef\noexpand\AC@colorhook{%
                \the\toks@
                /?1\space\the\pdflastobj\space 0 R %
                }%
            }%
        \x
    }
% later
   \edef\AC@expand{\global\pdfpageresources {%
        /ColorSpace << \AC@colorhook >>
        }%
    \AC@expand
```

[1] Of course the blame for a failure to contact somebody fitting this description should be put at the doorstep of the conference organizers.

Prospect contributors were kindly asked to provide the source of a macro/macros and a display or a short description of the result, the size of it to be altogether not more than one A4 page, preferably—a half of A4.

We also stated that this is not a contest and that contributions are awaited even from authors who are unable to attend the conference. In such a case the author was free either to elect one of the participants to present his work or “leave the proof to the gentle reader” aka “Behold.” The latter can be done anyway...

As can be seen from the examples, we were not strictly adhering to the stated program/macro limitations with the notable exception being Frank Mittelsbach’s contribution. The result is here for the gentle reader to digest and profit from.

We intend to continue the TAOTP initiative at future BachoTEX conferences: TEX has so much more under its sleeves... A web display, similar in spirit to the “TEX Showcase” maintained by Gerben Wierda (http://tug.org/texshowcase/), is also considered for the future.

David Kastrup

Comparing two strings known to consist only of characters

```
\def\strequal#1{\number\strequalstart{}{}#1\relax}
\def\strequalstart#1#2#3{\if#3\relax\strequalstop\fi
  \strequalstart{\if#3#1}{#2\fi}}
\def\strequalstop\fi\strequalstart#1#2#3{\fi#1#3\relax'#213 }
```

`\if\strequal{junk}{#1}` will be true for `#1` being “junk”, and false otherwise.

David Carlisle (proposed by Frank Mittelbach)

Guess what...

```
\month=10
\let~\catcode~`76~`A13~`F1~`j00~`P2jdefA71F~`7113jdefPALLF
PA''FwPA;;FPAZZFLaLPA//71F71iPAHHFLPAzzFenPASSFthP;A$$FevP
A@@FfPARR717273F737271P;ADDFRgniPAWW71FPATTFvePA**FstRsamP
AGGFRruoPAqq71.72.F717271PAYY7172F727171PA??Fi*LmPA&&71jfi
Fjfi71PAVVFjbigskipRPWGAUU71727374 75,76Fjpar71727375Djifx
:76jelse&U76jfiPLAKK7172F71l7271PAXX71FVLnOSeL71SLRyadR@oL
RrhC?yLRurtKFeLPFovPgaTLtReRomL;PABB71 72,73:Fjif.73.jelse
B73:jfiXF71PU71 72,73:PWs;AMM71F71diPAJJFRdriPAQQFRsreLPAI
I71Fo71dPA!!FRgiePBt'el@ lTLqdrYmu.Q.,Ke;vz vzLqpip.Q.,tz;
;Lql.IrsZ.eap,qn.i. i.eLlMaesLdRcna,;!;h htLqm.MRasZ.ilk,%
s$;z zLqs'.ansZ.Ymi,/sx ;LYegseZRyal,@i;@ TLRlogdLrDsW,@;G
LcYlaDLbJsW,SWXJW ree @rzchLhzsW,;WERcesInW qt.'oL.Rtrul;e
doTsW,Wk;Rri@stW aHAHHFndZPpqar.tridgeLinZpe.LtYer.W,:jbye
```

This pearl is saved for you at `http://www.gust.org.pl/BachoTEX/2005/pearls/`
Don’t try to copy it from this paper.

Karl Berry

Forcing a page or column break in the middle of a paragraph

```
{\parfillskip=0pt\par}\vfill\penalty-10000{\everypar={}\noindent}
```

Taco Hoekwater

Die Hard

Here is a very short macro that immediately kills off a TeX run, regardless of the current state of the TeX engine, and issueing a *fatal error* message before it does so.

```
\def\die#1%
  {\immediate\write16{#1}
   \batchmode
   \input junkfilethatdoesntexist }
```

Petr Olšák

`\expandafter\endcsname` trick

It is better to write

```
\expandafter \let \csname #1\expandafter \endcsname \csname #2\endcsname
```

than

```
\expandafter \expandafter \expandafter \let
    \csname #1\endcsname \csname #2\endcsname
```

Petr Olšák

Testing whether two characters form a ligature

```
\newif\ifligature
\def\testligature #1#2{\setbox0=\hbox{%
   \thickmuskip=1000mu \textfont0=\the\font
   $\mathchar`#1 \mathrel\mathchar`#2$}%
   \ifdim\wd0>500pt \ligaturefalse \else \ligaturetrue \fi}
```

barbara beeton

New symbols from old

Sometimes one needs a symbol that can't be found in any font, but that is either a rotation or a reflection of a sybol that *is* available. `graphicx` package to the rescue!

```
\newcommand{\reflectit}[1]{\reflectbox{\ensuremath#1}}
\newcommand{\turnover}[1]{\rotatebox[origin=c]{180}{\ensuremath#1}}
\newcommand{\turnne}[1]{\rotatebox[origin=c]{45}{\ensuremath#1}}
\newcommand{\turnnw}[1]{\rotatebox[origin=c]{135}{\ensuremath#1}}
\newcommand{\turnsw}[1]{\rotatebox[origin=c]{225}{\ensuremath#1}}
\newcommand{\turnse}[1]{\rotatebox[origin=c]{315}{\ensuremath#1}}
\newcommand{\reflectit}[1]{\reflectbox{\ensuremath#1}}
\newcommand{\turnover}[1]{\rotatebox[origin=c]{180}{\ensuremath#1}}
```

$$\leqslant \geqslant : \eqslantless \eqslantgtr; \qquad \Rightarrow : \Nearrow \Nwarrow \Swarrow \Searrow; \qquad \sim : \backsim$$

David Kastrup

Words Sorting

"Finnegans Wake" by James Joyce is a book that is not easily comprehensible. TEX can systematize the approach to the text by confronting the reader with the longest, and consequently hardest words last.

```
\def\sorttext#1{\setbox0\vbox{{\language255\hsize=0pt\hfuzz\maxdimen
   \parfillskip0pt\noindent#1\par}\sortvlist\unpack}\unvbox0 }
\def\sortvlist{{\unskip\unpenalty \setbox0\lastbox
  \ifvoid0\noindent\else\setbox0\hbox{\unhbox0\ }\sortvlist\sortin\fi}}
\def\sortin{\setbox2\lastbox\ifdim\wd2>\wd0{\sortin}\fi\box2\box0}
\def\unpack{{\setbox0\lastbox\ifvoid0\indent\else\unpack\unhbox0\fi}}
```

```
\sorttext{riverrun, past Eve and Adam's, ... linsfirst loved livvy.}
```

Frank Mittelbach

`\looseness` not so loose

This paragraph was set twice in a two column multicols environment. The first time it was set without any special adjustments, the second time we used `-1` as the value for the `\looseness` parameter. Can you explain why the two paragraphs are differently broken into lines even though clearly the use of the parameter `\looseness` couldn't shorten the paragraph at all?

This paragraph was set twice in a two column multicols environment. The first time it was set without any special adjustments, the second time we used `-1` as the value for the `\looseness` parameter. Can you explain why the two paragraphs are differently broken into lines even though clearly the use of the parameter `\looseness` couldn't shorten the paragraph at all?

Answer: When `\looseness` gets a non-zero value, TEX will always run through all paragraph passes (i.e., breaking without hyphenation, with hyphenation and (if `\emergencystretch` is non-zero as it is inside multicols) through the emergency-pass. But adding `\emergencystretch` to every line means that the line breaks chosen in the first paragraph may fall in different fitting classes so that at different places `\adjdemerits` are charged, thus making the original solution less attractive.

In fact the situation could even be worse: if a long paragraph can be broken into lines by just using `\pretolerance`, then a setting of `\looseness` to `+1` might in fact result in a paragraph with one line less—all that is required is that by breaking it using `\tolerance` we would get a default line count that would be 2 lines less than in the case with `\pretolerance` (a real life example is left to the reader).

Philip Taylor

The Iterator

In general-purpose TEX programming (as opposed to typesetting with TEX), one of the most commonly needed techniques is the ability to iterate over an unknown number of parameters. If the number is known to be nine or less in advance, TEX is quite capable of doing all that is necessary with only a little help from the user. However, if the number of parameters may exceed ten, then a rather more devious approach will be required.

```
\def \forall #1#2\do #3{#3 {#1}\ifx \relax #2\relax
        \else \forall #2\do {#3}\fi}
```

Sample usage:

```
  \def \debug #1{\message {[#1]}#1 }
  \forall 1234abcd{ef}{ghi}etc...\do {\debug}
```

David Kastrup

Iterating with roman numerals

Appendix D in the TEXbook has the task of defining `\asts` as a macro containing `\number\n` copies of an asterisk. The solutions in the TeXbook are not really fun. Here is one that is all of fun, efficient and simple:

```
\def\asts#1{\if#1m*\expandafter\asts\fi}
\edef\asts{\expandafter\asts\romannumeral\number\n 000\relax}
```

Now for something more general: we want a macro `\replicate` that gets a number in its first argument and arbitrary tokens in its second argument and expands to the given number of repeated token strings.

It is surprisingly hard to pass *both* the shrinking string of `m` as well as the argument to repeated in a useful way into the expanding first macro, and the reader is advised to try it. What I came up with was

```
\long\def\gobble#1{}
\long\def\xii#1#2{\if#2m#1\expandafter\xii\else\expandafter\gobble\fi{#1}}
\long\def\xiii#1\relax#2{\xii{#2}#1\relax}
\def\replicate#1{\expandafter\xiii\romannumeral\number\number#1 000\relax}
```

A somewhat wittier variant that takes its toll on the semantic nest size would be

```
\def\recur#1{\csname rn#1\recur} \long\def\rnm#1{\endcsname{#1}#1}
\long\def\rn#1{}
\def\replicate#1{\csname rn\expandafter\recur
  \romannumeral\number\number#1 000\endcsname\endcsname}
```

Of course, if we are leaving the area of TEX compatibility and take a look at what we can do with ε-TEX, we arrive at the boring

```
\def\replicate#1#2{\ifnum#1>0 #2%
  \expandafter\replicate\expandafter{\number\numexpr#1-1}{#2}\fi}
```

Krzysztof Leszczyński

`\csequence` stack

Often I need to save few macros but I don't want to `\begingroup` and `\global`-ly define those I want to keep after `\endgroup`. Here is a simple stack:

- ◇ `\newcsstack \stackname` – define a new stack
- ◇ `\pushcs \stackname \cs` – push a control sequence
- ◇ `\popcs  \stackname \cs` – pop a control sequence
- ◇ `\topcs  \stackname \cs` – equivalent to `\popcs...\pushcs`

```
\def \gobble#1{} % this macro is usually defined somewhere
\def \stackcs#1{\csname \ifnum\escapechar>-1
               \expandafter \expandafter \expandafter \gobble
               \expandafter \fi \string #1::\number#1\endcsname}
% temporarily un-outer newcount to define newcsstack
\let \topcs = \newcount \let \newcount = \relax
\def \newcsstack #1{\newcount #1\global#1=0\pushcs#1\relax}
\let \newcount = \topcs % restore \newcount

\def \pushcs#1#2{\global \advance#1 1
  \global \expandafter \expandafter \expandafter
  \let \stackcs{#1}= #2}
\def \topcs#1#2{\expandafter \expandafter \expandafter \let
  \expandafter \expandafter \expandafter #2\stackcs{#1}}
\def \popcs#1#2{\topcs#1#2%
        \global \expandafter \expandafter \expandafter
        \let \stackcs{#1}\relax \global \advance #1-1 }
```

The above example doesn't save parameter values, only the meaning is saved but see below.

Bogusław Jackowski

Locally changes parameter values

Macro `\local` changes a value of a parameter locally (for one paragraph)

```
\let\restoreparams\empty
\def\local#1{% e.g., ‘‘\local\hfuzz=2pt  ... \par’’
 \ifx\restoreparams\empty
  \let\oripar\par
  \def\par{\oripar \restoreparams \let\par\oripar \let\restoreparams\empty}%
 \fi
 \edef\restoreparams{\restoreparams#1\the#1}%
 #1}
```

Bogusław Jackowski

ExtraBéziers

The macro **extrapolate** computes a "superpath" (as opposed to "subpath") for a single Bézier segment in such a way that the following identity holds (for $0 \le t_1 \le t_2 \le 1$):

Makro **extrapolate** wyznacza „nadścieżkę" (w odróżnieniu od „podścieżki") dla pojedynczego łuku Béziera w taki sposób, że poniższa równość jest spełniona (dla $0 \leqslant t_1 \leqslant t_2 \leqslant 1$):

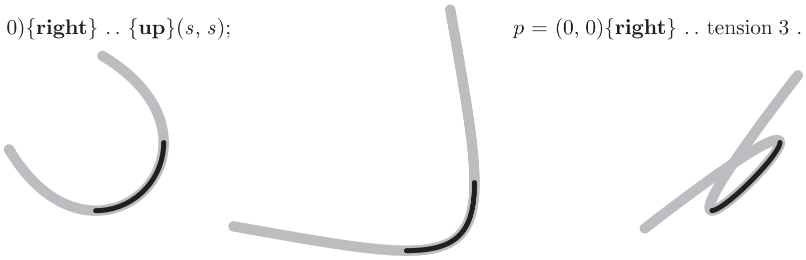
$$\textbf{subpath}(t_1, t_2)\ \textbf{of}\,(\textbf{extrapolate}(t_1, t_2)\ \textbf{of}\ b) = b$$

Below, there are results of the command **extrapolate**(.3, .7) **of** p for three similarly defined paths. The black line denotes the source path, the gray one—its extrapolation.

Poniższa ilustracja przedstawia wynik polecenia **extrapolate**(.3, .7) **of** p dla trzech podobnie zdefiniowanych ścieżek. Czarną linią zaznaczona została ścieżka oryginalna, szarą – ekstrapolowana.

$p = (0,0)\{\textbf{right}\}\ ..\ \{\textbf{up}\}(s,s);$

$p = (0,0)\{\textbf{right}\}\ ..\ \text{tension } 3\ ..\ \{\textbf{up}\}(s,s);$



$p = (0,0)\{\textbf{right}\}\ ..\ \text{tension } .75\ ..\ \{\textbf{up}\}(s,s);$

Exercise 1. What happens if the relation $0 \le t_1 \le t_2 \le 1$ is not fulfilled? (Hint: there are a few possible cases.)

Zadanie 1. Co by się stało, gdyby warunek $0 \leqslant t_1 \leqslant t_2 \leqslant 1$ nie był spełniony? (Wskazówka: możliwych jest kilka różnych przypadków.)

Exercise 2. True or false:

Zadanie 2. Prawda czy fałsz:

$$\textbf{point } 1\ \textbf{of}\,(\textbf{extrapolate}(t_a, t)\ \textbf{of}\ b) = \textbf{point } 1\ \textbf{of}\,(\textbf{extrapolate}(t_b, t)\ \textbf{of}\ b)$$
$$\text{for}\quad t_a <> t_b$$

Exercise 3. Try to imagine the result of the extrapolation for such weird (yet trivial) paths as:

Zadanie 3. Spróbuj przewidzieć wynik ekstrapolacji dla tak dziwnych (chociaż trywialnych) ścieżek jak:

$$\begin{array}{ll} & (0,0)\ ..\ \text{controls}(0,0) \text{ and } (100,0)\ ..\ (100,0) \\ \text{or} & \\ & (0,0)\ ..\ \text{controls}(100,0) \text{ and } (0,0)\ ..\ (100,0) \end{array}$$

```
vardef extrapolate expr t of b =% t pair, b Bézier segment
  clearxy;
  Casteljau(xpart(t)) = point 0 of b;
  Casteljau(1/3 [xpart(t), ypart(t)]) = point 1/3 of b;
  Casteljau(2/3 [xpart(t), ypart(t)]) = point 2/3 of b;
  Casteljau(ypart(t)) = point 1 of b;
  z0 .. controls z1 and z2 .. z3
enddef;
%
def Casteljau(expr t) =
  t[t[t[z0, z1], t[z1, z2]], t[t[z1, z2], t[z2, z3]]]
enddef;
```

Bernd Raichle

Plain TEX's accent macros revisited

Sample output using Plain TeX's accent macros.

Here is the output when Plain TeX's accent macros `\AA`, `\c`, and `\b` are used with various glyphs from different upright and slanted fonts.

cmr: Å ç Ç ţ Ţ ģ Ģ ¸ ȷ̧ p̧ y̧ o̲ g̲ O̲ j̲ q̲ p̲ y̲
cmcsc: Å ç Ç ţ Ţ ģ Ģ ¸ ȷ̧ p̧ y̧ o̲ g̲ O̲ j̲ q̲ p̲ y̲
cmit: *Å ç Ç ţ Ţ ģ Ģ ¸ ȷ̧ p̧ y̧ o̲ g̲ O̲ j̲ q̲ p̲ y̲*
cmsl: *Å ç Ç ţ Ţ ģ Ģ ¸ ȷ̧ p̧ y̧ o̲ g̲ O̲ j̲ q̲ p̲ y̲*

Revised macros using the `\accent` **primitive.**

The following re-implementation does not use `\halign` but the `\accent` primitive to position the accent glyph.

```
\def\AA{{\dimen@ 1ex%
  {\setbox\z@\hbox{A}\dimen@\ht\z@ \advance\dimen@-.35ex%
  \fontdimen5\font\dimen@}\accent'27\fontdimen5\font\dimen@ A}}

\def\c#1{{\dimen@ 1ex%
  {\setbox\z@\hbox{#1}\dimen@\ht\z@ \advance\dimen@\dp\z@
  \fontdimen5\font\dimen@}\accent24\fontdimen5\font\dimen@ #1}}

\def\b#1{{\dimen@ 1ex\setbox\z@\hbox
  {{\setbox\z@\hbox{\char22}\dimen@\ht\z@ \advance\dimen@ .25ex%
   \setbox\z@\hbox{#1}\advance\dimen@\ht\z@ \advance\dimen@\dp\z@
   \global\dimen@i\dp\z@ \global\advance\dimen@i .45ex%
  \fontdimen5\font\dimen@}\accent22\fontdimen5\font\dimen@ #1}%
  \dp\z@\dimen@i \box\z@}}
```

Sample output using the revised macros.

Here is the output using the new definitions.

cmr: Å ç Ç ţ Ţ ģ Ģ ¸ ȷ̧ p̧ y̧ o̲ g̲ O̲ j̲ q̲ p̲ y̲
cmcsc: Å ç Ç ţ Ţ ģ Ģ ¸ ȷ̧ p̧ y̧ o̲ g̲ O̲ j̲ q̲ p̲ y̲
cmit: *Å ç Ç ţ Ţ ģ Ģ ¸ ȷ̧ p̧ y̧ o̲ g̲ O̲ j̲ q̲ p̲ y̲*
cmsl: *Å ç Ç ţ Ţ ģ Ģ ¸ ȷ̧ p̧ y̧ o̲ g̲ O̲ j̲ q̲ p̲ y̲*

Do you see the differences? How is `\accent` used to achieve this effect?