OPmac - tipy, triky, návody

Petr Olšák, 2013, 2014, 2015

http://petr.olsak.net/opmac-tricks-cs.html

Obsah

Intro

Tato stránka obsahuje návody na řešení rozličných úkolů při využití OPmac. Popis jednoho návodu by neměl překročit rozsah výšky prohlížeče. Schéma každého návodu je stejné: uživatelký popis následovaný krátkým makrem, které si uživatel může seškrábout myší do svého dokumentu, následované případným vysvětlením, jak makro funguje.

Mají-li další uživatelé OPmac také nějaký takový tip, prosím o jeho zaslání na můj email. Uvítám jej a zařadím na tuto stránku (pochopitelně s uvedením autora).

1 Písmo

0001
P. O.
13. 08. 2013

1.1 Korekce střední výšky písma

Pro některá písma se může stát, že například strojopis `\tt` neladí se základním písmem. Nastává to tehdy, když není stejná střední výška písma, třebaže obě písma jsou zavedana ve stejné velikosti (například `at11pt`). Korekci lze v OPmac udělat jednoduše využitím `\thefontscale`, který škáluje vzhledem k aktuální velikosti písma (nikoli vzhledem k fixní designované velikosti). Takže stačí třeba psát:

```
\def\tt{\tentt\thefontscale[1120]}
```

Uvedená definice zvětší `\tt` font 1,12 krát vzhledem k velikosti okolního fontu. Přepínač `\tt` toto udělá kdekoli při jakékoli aktuální velikosti okolního fontu. Uvedený poměr se hodí například pro kombinaci Bookman s Courier.

0132
P. O.
23. 11. 2015

1.2 Chytřejší \typosize a \typoscale

OPmac definuje makra `\typosize` a `\typoscale` tak, že nastaví odpovídající velikost fontu a resetují variantu fontu na `\rm` bez ohledu na to, jaká byla zvolena před použitím makra. Tedy

```
\bf text \typosize[13/15] velký
```

způsobí, že slovo „velký“ je vytištěno ve variantě `\rm`, nikoli `\bf`. Pokud chceme, aby makra `\typosize` a `\typoscale` respektovala naposledy zvolenou variantu fontu, pak je možné použít tento kód:

```
\let\currvf=\rm
\addto\rm{\let\currvf\rm}   \addprotect\rm
\addto\it{\let\currvf\it}   \addprotect\it
\addto\bf{\let\currvf\bf}   \addprotect\bf
\addto\bi{\let\currvf\bi}   \addprotect\bi

\def\textfontsize[#1]{\if$#1$\else
  \fontdim=#1\ptunit \ifx\fontdimB\undefined \edef\fontdimB{\the\fontdim}\fi
  \let\dgsize=\fontdim
  \edef\sizespec{at\the\fontdim}%
  \resizeall \currvf
  \let\dgsize=\undefined
  \fi
}
```

Makra `\rm`, `\bf`, `\it` a `\bi` nyní kromě nastavení varianty uloží zprávu o aktuálně nastavené variantě do sekvence `\currvf`. Aby nedošlo ke kolizi při expandování ve `\write`, je třeba makra označit pomocí `\addprotect`. Interní makro OPmac `\textfontsize` je předefinováno tak, že místo makra `\rm` původně se vyskytujícího za voláním `\resizeall` je vloženo `\currvf`.

0027
P. O.
05. 09. 2013

1.3 Font dle požadované šířky

Připravíme makro `\scaleto rozměr {text}`, které vytiskne text aktuálním fontem zvětšeným/zmenšeným tak, že šířka textu zaujme specifikovaný rozměr. Takže třeba `\scaleto5cm{ahoj}`

vytvoří box s textem „ahoj“ zvětšeným tak, že šířka textu je `5cm`. Tuto vlastnost asi při běžné sazbě nevyužijeme, ale může se hodit při sazbě plakátů nebo pro umístění nadpisu přesně do `\hsize` atd.

```
\def\scaleto#1#{\bgroup\def\bw{#1}\scaletoA}
\def\scaletoA #1{\expandafter\let \expandafter\thefont \the\font
   \calculatefontdim{#1}\edef\dgsize{\the\fontdim}\letfont\thefont=\thefont at\fontdim
   \calculatefontdim{#1}\letfont\thefont=\thefont at\fontdim
   \hbox to\bw{\thefont#1}%
   \egroup
}
\def\calculatefontdim#1{%
   \setbox0=\hbox{\thefont #1}\tmpdim=\bw \tmpnum=\wd0
   \divide\tmpnum by256 \divide\tmpdim by\tmpnum \multiply\tmpdim by256
   \fontdim=\expandafter\ignorept\the\tmpdim\fontdim
}
```

Makro vypočítává správné zvětšení tak, že spočítá poměr požadované šířky boxu ku skutečné šířce textu a tímto poměrem násobí `\fontdim`. Výpočet dělá nadvakrát. Při prvním výpočtu `\fontdim` nastaví podle něj také `\dgsize`, tj. vyzvedne se také font odpovídající optické velikosti. To ale může změnit skutečnou šířku textu. Takže přepočet je proveden podruhé, a po něm už proběhne jen geometrické zvětšení fontu z předchozího kroku.

Je-li aktivován eTeX, je možné vypočítat poměr dvou velikostí přesněji a pohodlněji. Stačí definovat `\dividedimen` a využít toho, že toto makro pracuje na úrovni expandprocesoru.

```
\def\dividedimen (#1/#2){\expandafter\ignorept\the
   \dimexpr\numexpr\number\dimexpr#1\relax*65536/\number\dimexpr#2\relax\relax sp\relax
}
\def\calculatefontdim#1{%
   \setbox0=\hbox{\thefont #1}%
   \fontdim=\dividedimen(\bw/\wd0)\fontdim
}
```

1.4 Výpis načtených fontů

0029 *P. O.* 24. 09. 2013

Po načtení fontů (např. pomocí `\input lmfonts`) a po jejich zvětšení (např. pomocí `\typosize[11/13]`) je někdy užitečné si připravené fonty vypsat do logu a na terminál, aby si v tom člověk udělal jasno. Stačí napsat `\showfonts`, pokud je tento příkaz definován například takto:

```
\def\showfonts{\bgroup
   \def\wterm##1{\immediate\write16{##1}}
   \def\resizefont##1{\wterm{##1= \fontname##1}}\resizeall
   \tmpnum=0 \loop
      \wterm{\the\tmpnum: \fontname\textfont\tmpnum\space/
                          \fontname\scriptfont\tmpnum\space/
                          \fontname\scriptscriptfont\tmpnum}
      \advance\tmpnum by1 \ifnum\tmpnum<16 \repeat
   \egroup}
```

Příkaz vypíše názvy všech textových fontů registrovaných pomocí `\regfont` a dále názvy všech matematických fontů rodin 0 až 15 ve všech velikostech.

1.5 Méně implicitně zavedených fontů

0111 *P. O.* 17. 06. 2015

Použijeme-li `\showfonts` z předchozího OPmac triku, shledáme, že OPmac implicitně pro matematickou sazbu zavádí kromě standardních CM fontů i AMS fonty a EC fonty. Může se stát, že tyto fonty nepotřebujeme a navíc je nemáme třeba ani instalovány v TeXové distribuci.

Obejít závislost OPmac na těchto fontech lze pomocí `\let\normalmathloading=\relax`, které je třeba vložit před `\input opmac`. Pak musí následovat definice matematické sady `\normalmath` a `\boldmath` následovaná příkazem `\normalmath`. Pokud například chceme používat výhradně CM fonty (v csplainu CSfonty) a žádné jiné, pak je možné psát:

```
\let\normalmathloading=\relax
\input opmac

\def\normalmath{%
  \loadmathfamily 0 cmr  % CM Roman
  \loadmathfamily 1 cmmi % CM Math Italic
  \loadmathfamily 2 cmsy % CM Standard symbols
  \loadmathfamily 3 cmex % CM extra symbols
  \loadmathfamily 8 cmbx % bold
  \setmathfamily  9 \tenbi % bold slanted
  \setmathfamily 10 \tenrm
  \setmathfamily 11 \tenit
  \setmathdimens
}
\def\boldmath{%
  \loadmathfamily 0 cmbx  % CM Roman Bold Extended
  \loadmathfamily 1 cmmib % CM Math Italic Bold
  \loadmathfamily 2 cmbsy % CM Standard symbols Bold
  \loadmathfamily 3 cmex  % CM extra symbols
  \loadmathfamily 8 cmbx  % bold
  \setmathfamily  9 \tenbi % bold slanted
  \setmathfamily 10 \tenrm
  \setmathfamily 11 \tenit
  \setmathdimens
}
\normalmath
```

Pochopitelně, po tomto nastavení nebudou fungovat matematické znaky z ignorovaných fontů, tj. nebude dosažitelné množství AMS symbolů, nebude fungovat `\script`, `\frak`, `\bbchar`, přepínače `\bf` a `\bi` vrátí v matematickém módu běžné serifové znaky (nikoli bezserifové). Na druhé straně OPmac v tuto chvíli není závislý na AMS fontech ani EC fontech a tyto fonty nemusejí být ani instalovány.

Jiným řešením stejného problému může být zkopírování souboru `ams-math.tex` do nového souboru (řekněme `my-math.tex`), pozměnění obsahu tohoto souboru dle soukromých požadavků a načtení tohoto souboru *před příkazem* `\input opmac`. Tedy:

```
\input my-math
\input opmac
```

Zdůvodnění: Jestliže je `\normalmath` definován před `\input opmac`, tak OPmac soubor `ams-math.tex` vůbec nepoužije.

1.6 Stojaté závorky v kurzívě

0064
P. O.
07. 06. 2013

Ukážeme, jak lze jediným řádkem implementovat vlastnost, kterou řeší cca 300 řádků naprosto nečitelného kódu v balíčku `embrac.sty`.

Občas se objeví typografický požadavek ponechat závorky `()` a případně další i v kurzívě stojaté. Takže když napíšeme `{\em kurzíva se (závorkami)}`, měli bychom dostat

```
{\it kurzíva se {\rm(}závorkami{\/\rm)}}
```

Toto se dá řešit následující definicí:

```
\addto\em{\adef({\ifmmode(\else{\rm(}\fi}\adef){\ifmmode)\else{\/\rm)}\fi}}
```

Definice aktivuje závorky v kurzívě a navíc řeší ponechání závorek v původním významu v matematickém módu. takže funguje

```
{\em Text $\bigl((yxy)+z\bigr)$ a něco dalšího (v závorce)}.
```

0075 P. O. 27. 06. 2014

1.7 První písmeno odstavce větší

Navrhneme makro `\Capinsert`, které zvětší první písmeno v odstavci:

```
\Capinsert První písmeno bude větší, tj. P v tomto případě...
nebo
\Capinsert {co vytisknout vlevo} První písmeno bude větší...
```

Je třeba deklarovat, jak moc chceme písmeno zvětšit a jak jej zabudovat do řádků odstavce. Deklarační část bude vypadat takto:

```
\newdimen\ptem     \ptem=.1em        % jednotka závislá na em
\newdimen\Capsize  \Capsize=44\ptem  % požadovaná velikost
\newdimen\Capabove \Capabove=8\ptem  % horní přesah přes účaří
\newdimen\Capafter \Capafter=1\ptem  % mezera za písmenem
\def\Capprefix{\localcolor\Red}      % makro provedené před písmenem

%% Deklarace k jednotlivým písmenům ve tvaru:
% \declCap písmeno {přesah doleva; korekce prvního řádku, druhého, atd.}
\declCap {default} {0;0,0,0}  % výchozí hodnota pro nedeklarovaná písmena
\declCap W {3;0,4,6}
\declCap A {1;6,2,-2}
\declCap L {0;9,0,0}
...
```

Makro, které řeší požadovaný úkol (podobnou věc řeší LaTeXový balíček `lettrine`) může vypadat takto:

```
\def\declCap #1#2{\sxdef{cap:=#1}{#2}}
\def\Capinsert{\def\leftCapmaterial{}\futurelet\next\CapinsertA}
\def\CapinsertA{\ifx\next\bgroup \expandafter\CapinsertB \else
\expandafter\CapinsertC \fi}
\def\CapinsertB #1{\def\leftCapmaterial{#1}\CapinsertC}
\def\CapinsertC #1{\par
  \isdefined{cap:=#1}\iftrue \edef\tmp{\csname cap:=#1\endcsname}%
                     \else   \edef\tmp{\csname cap:=default\endcsname}\fi
  \setbox0=\hbox{{\thefontsize[\expandafter\ignorept\the\Capsize]\Capprefix#1}\kern\Capafter}%
  \expandafter \CapinsertD \tmp,,%
  \noindent\kern-\firstlineindent \rlap{\kern-\protrudeCap\ptem\llap{\leftCapmaterial}%
                                        \vbox to0pt{\kern-\Capabove\box0\vss}}%
  \kern\firstlineindent
}
\def\CapinsertD #1;{\tmpnum=1 \let\firstlineindent=\undefined
   \def\parshapeparams{}\def\protrudeCap{#1}\CapinsertE}
\def\CapinsertE #1,{\ifx,#1,\parshape =\tmpnum \parshapeparams 0pt \hsize
  \else
     \advance\tmpnum by1
     \tmpdim=\wd0 \advance\tmpdim by-#1\ptem \advance\tmpdim by-\protrudeCap\ptem
     \edef\parshapeparams{\parshapeparams\the\tmpdim}%
     \ifx\firstlineindent\undefined \let\firstlineindent\parshapeparams \fi
     \advance\tmpdim by-\hsize \tmpdim=-\tmpdim
     \edef\parshapeparams{\parshapeparams\the\tmpdim}%
     \expandafter \CapinsertE \fi
}
```

Podrobný popis fungování makra je na tex.stackexchange.

1.8 Nepravý bold

0095 P. O. 08. 04. 2015

Někdy není k dizpozici tučný řez fontu, a přitom (zvláště v matematických vzorcích) potřebujeme nějaké písmeno nebo znak zdůraznit tučně. Například nejsou k dispozici matematické symboly ze sady AMS-A a AMS-B v tučné variantě, případně skript. V takovém případě může pomoci podvržený bold, který použijeme například takto:

```
Vzorec: $a+b\fakebold{+}c = \fakebold{\script D}$.
```

Makro `\fakebold` se opírá o PDF kód vložený pomocí `\pdfliteral`:

```
\def\fakebold#1{\pdfliteral{2 Tr .3 w}#1\pdfliteral{0 Tr 0 w}}
```

Kód `2 Tr .3 w` dává PDF rasterizéru pokyn, aby písmo definované okrajem jednotlivých písmen nejen vyplnil barvou, ale také okraj obtáhl čarou tloušťky `.3 bp`. Tím každé písmeno dostane výraznější duktus.

Pro další výtvarné efekty můžete experimentovat i s tím, že písmena vůbec nebudou vyplněna, jen budou obtažena. To zajistí příkaz `1 Tr` a samozřejmě musíte nastavit nenulovou tloušťku obtahu například pomocí `.2 w`.

1.9 Podtržení respektuje dolní přetahy

0112
P. O.
19. 06. 2015

Za pomocí předchozího OPmac triku vytvoříme makro `\underlinee{text}`, které podtrhne text, ale podtrhující čára se nedotýká dolních přetahů. Test: `\underlinee{jumping quickly}` vytvoří:

Test: jumping quickly

Pro písmo Computer Modern `10pt` může makro `\underlinee` vypadat takto:

```
\def\underlinee#1{%
   \leavevmode\vbox to0pt{\vss
      \hrule height.3pt
      \vskip-\baselineskip \kern2.5pt
      \localcolor
      \hbox{\strut\rlap{\White\pdfliteral{2 Tr 1.1 w}#1\pdfliteral{0 Tr 0 w}}#1}
}}
```

Chcete-li podtrhávat jiné písmo, je vhodné individuálně nastavit konstanty `.3pt`, `2.5pt` a `1.1`, abyste dosáhli co nejlepšího vizuálního efektu.

Uvedené makro podtrhne text zapouzdřený v boxu, takže nedovolí zlomit řádek. Pokud byste chtěli vytvořit chytřejší podtrhávání včetně automatického zlomu řádků, můžete se dále inspirovat OPmac trikem 0063. Analogický problém je řešen na tex.sx.com

2 Barvy

2.1 Přepínače barev jako přepínače fontů

0026
P. O.
03. 09. 2013

Tento OPmac trik měl své opodstatnění pro verze OPmac do Nov. 2014. Od verze Dec. 2014 stačí na začátek dokumentu napsat `\localcolor` a je vystaráno. Barvy se pak přepínají lokálně uvnitř TeXových skupin. Je ale potřeba dát pozor na konstrukce typu:

```
\setbox0=\hbox{text \Red text}   % nefunguje, návrat na původní barvu
                                 % se provede za \setbox0
\setbox0=\hbox{{text \Red text}} % funguje, návrat na původní barvu
                                 % se provede uvnitř boxu
```

2.2 Přepínače barev jako v LaTeXu

0034
P. O.
29. 11. 2013

V LaTeXu po zavedení příslušného balíčku je k dispozici makro `\color`, které přepíná lokálně ve skupině barvu podle svého argumentu (např. `{\color{red}červený text}`). Makro `\color` je možné definovat takto:

```
\def\color#1{\localcolor\colorA#1\relax}
\def\colorA#1#2\relax{\uppercase{\csname#1}#2\endcsname\ignorespaces}

Text {\color{red} červený} a taky {\color{blue} modrý} i normální.
```

Toto makro nastaví barvu jako `\localcolor` a spustí příkaz se jménem, jako je parametr příkazu, jen první písmeno je velké.

2.3 Barvy společně s poznámkami mnote

0037 P. O. 22. 03. 2014

Chcete-li přepínat barvy v odstavci a současně používat poznámky na okraji `\mnote`, dočkáte se pravděpodobně barevného zmatení. Poznámky `\mnote` jsou dodatečně vloženy jakoby pod aktuální řádek odstavce, takže přebírají barvu, jakou má řádek na svém konci. Pokud chcete mít barvu poznámek stále stejnou, pište na začátek dokumentu třeba:

```
\def\mnotehook{\noindent\localcolor\Blue}
```

Příkaz `\noindent` je potřebný, protože jinak se značka barvy vloží do vertikálního módu a `\mnote` nebude mít správné vertikální umístění.

Proč nedefinuje OPmac `\mnotehook` jako `\Black` implicitně? Protože implicitní chování předpokládá, že poznámky budou přebírat barvu textu, ve kterém jsou napsány, a tato barva může být neměnná (např. přes několik odstavců).

V parametru `\mnote` je možné mít další vnořené barvy, které jsou ohraničeny skupinami.

2.4 Obarvení pouze jádra bez indexu a exponentu

0066 P. O. 12. 06. 2014

Úkolem je obarvit jinou barvou jen základ vzorečku bez indexu a exponentu. Tedy, když napíšu `$\colormath\Red Y_i^2$`, pak bude červené jen písmeno Y a index i exponent bude stejnou barvou, jako je vnější okolí. Jednoduché `${\localcolor\Red Y}_i^j$` nefunguje správně, protože za konec skupiny se vloží návrat k původní barvě a to znemožní umístit index těsně k písmenu. Byla totiž před tímto návratem k původní barvě už vložena italická korekce. Řešením je neukončovat barvu na konci jádra (základu vzorečku), ale případně obarvit index nebo exponent zapamatovanou vnější barvou a teprve po sazbě indexu a exponentu ukončit skupinu a vrátit se k původní bravě.

```
\def\colormath#1#2{% #1=color #2=colored text
   \bgroup \let\tmpc=\currentcolor % saving current color
   \localcolor#1#2%
   \isnextchar_{\colormathA}{%
      \ifcat\next.\insertmukern{#2}\next\fi
      \ifcat\next x\insertmukern{#2}\next\fi
      \egroup}%
}
\def\colormathA_#1{_{\setcmykcolor\tmpc#1}\isnextchar^{\colormathB}{\egroup}}
\def\colormathB^#1{^{\setcmykcolor\tmpc#1}\egroup}

\def\ignorefracpart#1.#2\relax{#1}
\newmuskip\tmpmudim
\def\insertmukern #1#2{\setbox0=\hbox{$#1#2$}\setbox1=\hbox{$#1\null#2$}%
   \tmpdim=\wd0 \advance\tmpdim by-\wd1
   \tmpmudim=\expandafter\ignorept\the\tmpdim mu \tmpmudim=288\tmpmudim
   \tmpdim=16em \divide\tmpmudim by\expandafter\ignorefracpart\the\tmpdim\relax
   \mkern\tmpmudim
}
```

Makro řeší ještě jeden problém. Mezi znaky „Y“ a čárkou v `cmmi10` je záporný kern, který by nebyl uplatněn, pokud bychom mezi těmi znaky jen ukončili skupinu a vrátli se k původní barvě. Je tedy potřeba kern manuálně spočítat a vložit pomocí `\insertmukern`. Makro `\insertmukern`

zjistí hodnotu kernu porovnámím šířky dvou boxů v `\textstyle`. Tím získá hodnotu v `pt`. Protože se ale sazba může odehrávat v `\scriptstyle` nebo `\scriptscriptstyle`, je v makru hodnota v `pt` přepočítána na hodnotu v mu jednotkách pomocí vzorce `mu = (288 / 16em) pt = 18/\quad pt`. Pronásobení `em` šestnácti sníží zaokrouhlovací chyby při dělení celým číslem.

2.5 Obarvení pouze matematického akcentu

0074
P. O.
24. 06. 2014

Toto je podobný problém, jako v předchozím triku 0066. Můžeme dát akcentu barvu, ale pak chceme předsunout před základ nastavení černé barvy (resp. barvy vnějšího okolí) a toto přepnutí se dělá pomocí `\pdfliteral`. Když vyzkoušíme

```
$\widetilde{E}\ \widetilde{\pdfliteral{}E}\ \coloredaccent\Red\widetilde{E}$
```

dostaneme

$\widetilde{E}\ \widetilde{\pdfliteral{}E}\ \coloredaccent\Red\widetilde{E}$

Je vidět, že prostřední výskyt akcentu dopadl špatně, protože byl do základu vložen `\pdfliteral{}`, což znemožnilo usazení akcentu podle kernigového páru se `\skewchar`. Ani nebylo nutné do toho `\pdfliteral{}` psát přepnutí na barvu. Poslední výskyt je obarvený a správně, protože používá makro `\coloredaccent`, které je definováno takto:

```
\newmuskip\tmpmudim

\def\coloredaccent#1#2#3{% #1=color, #2=accent, #3=base
   {\ifnum\skewchar\textfont1<0 \tmpmudim=0mu \else \calculatemukern {#3}{\char\skewchar\textfont1}\fi
    \let\tmpc=\currentcolor % saving current color
    \localcolor #1\mkern2\tmpmudim #2{\setcmykcolor\tmpc \mkern-2\tmpmudim#3}
   }
}
\def\calculatemukern #1#2{\setbox0=\hbox{\the\textfont1 #1#2}\setbox1=\hbox{\the\textfont1 #1\null#2}%
   \tmpdim=\wd0 \advance\tmpdim by-\wd1
   \tmpmudim=\expandafter\ignorept\the\tmpdim mu \tmpmudim=288\tmpmudim
   \tmpdim=16em \divide\tmpmudim by\expandafter\ignorefracpart\the\tmpdim\relax
}
\def\ignorefracpart#1.#2\relax{#1}
```

Makro `\coloredaccent` při nastaveném `\skewchar` proměří kerningový pár základu se skewchar a převede ho na mu jednotky podobně jako v předchozím triku. Pak usadí akcent posunutý o vypočtenou hodnotu doprava a základ vrátí o stejnou hodnotu doleva. Není mi jasné, proč musím vypočtenou hodnotu vynásobit dvěma. nějak jsem se ve skewchar algoritmu TeXu ztratil. Pokud mi to nějaký čtenář vysvětlí, dostane ode mne pochvalu před nastoupenou jednotkou.

2.6 Barvy zadávané v RGB

0089
P. O.
25. 01. 2015

OPmac nastavuje barvy interně v CMYK. Jak překlopit toto chování celkově do RGB je popsáno v následujícím OPmac triku 0090. Nyní je naším cílem zůstat interně v CMYK, ale umožnit uživateli zadávat barvy v RGB. Třeba

```
\def\Pink{\setRGBcolor{213 30 101}}
nebo
\def\Pink{\setrgbcolor{.835 .118 .396}}
nebo
\def\Pink{\setHEXcolor{D51E65}}
```

Ve všech třech případech tohoto příkladu je definována stejná barva, která je interně realizována jako `\setcmykcolor{0 .859 .526 .165}`. Implementace může vypadat takto:

```
\def\setRGBcolor#1{\setRGBcolorA#1 }
\def\setRGBcolorA#1 #2 #3 {\def\tmpb{}\tmpdim=0pt
   \ifdim#1pt>\tmpdim \tmpdim=#1pt \fi
   \ifdim#2pt>\tmpdim \tmpdim=#2pt \fi
   \ifdim#3pt>\tmpdim \tmpdim=#3pt \fi
   \tmpnum=\expandafter\onedecimaldigit\the\tmpdim\relax \relax
   \ifnum\tmpnum=0 \def\tmpb{0 0 0 1}\else
      \setRGBcolorB{#1}\setRGBcolorB{#2}\setRGBcolorB{#3}%
      \tmpdim=2550pt \advance\tmpdim by-\tmpnum pt \divide\tmpdim by2550
      \edef\tmpb{\tmpb\expandafter\ignorept\the\tmpdim}\fi
   \setcmykcolor\tmpb
}
\def\setRGBcolorB#1{\tmpdim=#1pt
   \tmpdim=-\expandafter\onedecimaldigit\the\tmpdim\relax pt
   \advance\tmpdim by\tmpnum pt \divide\tmpdim by\tmpnum
   \edef\tmpb{\tmpb\expandafter\ignorept\the\tmpdim\space}%
}
\def\onedecimaldigit#1.#2#3\relax{#1#2}
\def\setHEXcolor#1{\setHEXcolorA#1}
\def\setHEXcolorA #1#2#3#4#5#6{\setRGBcolorA "#1#2 "#3#4 "#5#6 }
\def\setrgbcolor#1{\setrgbcolorA#1 }
\def\setrgbcolorA#1 #2 #3 {\def\tmpb{}\dimen0=255pt
   \setrgbcolorB{#1}\setrgbcolorB{#2}\setrgbcolorB{#3}%
   \expandafter\setRGBcolorA\tmpb
}
\def\setrgbcolorB#1{\tmpdim=#1\dimen0
   \edef\tmpb{\tmpb\expandafter\ignorept\the\tmpdim \space}}
\addprotect\setRGBcolor  \addprotect\setrgbcolor  \addprotect\setHEXcolor
```

Makro `\setRGBcolor` provádí konverzi RGB na CMYK dle jednoduchého vzorce (bez použití ICC profilů):

`K'=max(R,G,B),  C=(K'-R)/K', M=(K'-G)/K', Y=(K'-B)/K', K=(255-K')/255`.

Makro umožní pracovat s jedním desetinným místem v údajích `R,G,B`. Proto jsou tyto údaje zpracovány makrem `\onedecimaldigit` a dělíme číslem 2550 a nikoli 255.

Makro `\setrgbcolor` násobí své parametry číslem 255 a převádí výpočet na `\setRGBcolor`. Konečně makro `\setHEXcolor` uvodí své parametry symbolem `"` a převede je rovněž na `\setRGBcolor`.

2.7 Barvy překlopené do RGB

0090
P. O.
25. 01. 2015

OPmac interně pracuje v barevném prostoru CMYK. Konverzi z RGB do tohoto barevného prostoru může obstarat OPmac trik 0089. Někdy se může ale stát, že potřebujeme, aby interně byly barevné povely vkládány přímo v RGB. Například proto, že tento barevný prostor nabízí gamut s jásavějšími barvami a dokument chceme použít pouze na obrazovce počítače, která je RGB zařízením. Nebo proto, že chcete barevně sladit obrázky z Inkscape s barvami v dokumentu, přitom Inkscape exportuje obrázky bohužel výhradně v RGB.

Barvy pak můžete definovat pomocí `\def\Pink{\setrgbcolor{.835 .118 .396}}`.

V OPmac od verze Jul. 2019 je makro `\setrgbcolor` přímo k dispozici. Pokud navíc chcete jednoduchou konverzi z CMYK do RGB, pak lze použít:

```
\def\setcmykcolor#1{\expandafter\setcmykcolorA#1 }
\def\setcmykcolorA #1 #2 #3 #4 {\def\tmpb{}%
   \tmpdim=1pt \advance\tmpdim by-#4pt
   \edef\tmpa{\expandafter\ignorept\the\tmpdim}%
   \setcmykcolorB{#1}\addto\tmpb{ }\setcmykcolorB{#2}\addto\tmpb{ }\setcmykcolorB{#3}%
   \setrgbcolor\tmpb
}
\def\setcmykcolorB#1{\tmpdim=1pt \advance\tmpdim by-#1pt
```

```
\tmpdim=\tmpa\tmpdim
   \edef\tmpb{\tmpb\expandafter\ignorept\the\tmpdim}%
}
\def\setRGBcolor#1{\setRGBcolorA#1 }
\def\setRGBcolorA#1 #2 #3 {\def\tmpb{}\setRGBcolorB{#1}\setRGBcolorB{#2}\setRGBcolorB{#3}%
   \setrgbcolor\tmpb
}
\def\setRGBcolorB#1{\tmpdim=#1pt \divide\tmpdim by255
   \edef\tmpb{\tmpb\expandafter\ignorept\the\tmpdim\space}%
}
```

Toto makro interně vkládá barvy v RGB pomocí `setrgbcolor` a také předefinuje `\setcmykcolor` tak, že toto makro se postará o konverzi a zavolá pak `\setrgbcolor`. Používá přitom vzorce:

`R = (1-C)*(1-K), G = (1-M)*(1-K), B = (1-Y)*(1-K)`.

Dále je zde definováno makro `\setRGBcolor`, které vydělí své tři parametry číslem 255 a zavolá `\setrgbcolor`.

Poznámka: máte-li starší verzi OPmac než Jul. 2019, pak musíte nejprve `\setrgbcolor` definovat:

```
\let\setrgbcolor=\setcmykcolor  \addprotect\setrgbcolor
\def\colorstackpush#1{\pdfcolorstack\colorstackcnt push{#1 rg #1 RG}}
\def\colorstackset#1{\pdfcolorstack\colorstackcnt set{#1 rg #1 RG}}
\ifx\XeTeXversion\undefined \else
   \def\colorstackpush#1{\colorspecialinit \special{color push rgb #1}
   \def\colorstackset#1{\colorspecialinit \special{color pop}\special{color push rgb #1}}
\fi
\def\Black{\setrgbcolor{0 0 0}}
\def\pdfblackcolor{0 0 0} \xdef\currentcolor{\pdfblackcolor}
```

2.8 Míchání barev, jako na paletě malíře

0105
P. O.
24. 04. 2015

Makro `\cmixdef\NovaBarva{lineární kombinace barev}` definuje novou barvu jako lineární kombinaci daných barev. Název `\cmixdef` značí „color mix define“. Příklad:

```
\cmixdef\myCyan {.3\Green + .5\Blue}  % směs 30 % zelené, 50 % modré, zbytek bílá
\cmixdef\myColor {.1\Blue + .4\Brown + .5\Yellow}
\cmixdef\LightBlue {.6\Blue} % Světle modrá: 60 % modré, zbytek bílá
\cmixdef\DarkBlue {\Blue + .4\Black} % Modrá s přimícháním 40 % černé
```

Míchání barev probíhá na úrovni aditivního barevného prostoru CMYK. Pokud má výsledný součet největší složku větší než 1, jsou v závěru všechny složky pronásobeny takovým koeficientem, aby tato složka byla přesně rovna jedné. Chcete-li emulovat činnost malíře, použijte konvexní kombinaci, jako například v druhém řádku ukázky: jeden díl modré, čtyři díly hnědé a pět dílů žluté. Není ale nutné se konvexních kombinací držet, například:

```
\cmixdef\Blue{\Cyan + \Magenta} % vskutku, protože výchozí barvy jsou z CMYK
zatímco:
\cmixdef\myBlue{.5\Cyan + .5\Magenta} je totéž jako \cmixdef\myBlue{.5\Blue}.
```

Místo symbolu plus v lineární kombinaci je možné použít symbol mínus. Pak se barva odečte a v okamžiku odečtení příslušné barvy se záporné složky redukují na nulu. Třeba `\cmixdef\Barva{\Brown-\Black}` odstraní z barvy její „znečištění černou“. Konečně je možné těsně před makro pro barvu napsat symbol `^`, což značí, že se místo této barvy použije barva doplňková.

```
\cmixdef\mycolor{\Grey + .6^\Blue} stejné jako \cmixdef\mycolor{\Grey+.6\Yellow}
```

Implementace je následující:

```
\def\cmixdef#1#2{\bgroup
   \let\setcmykcolor=\relax \edef\tmpb{+#2}%
   \replacestrings{ }{}\replacestrings{+}{\addcolor}\replacestrings{-}{\addcolor-}%
   \replacestrings{^\setcmykcolor}{\setcmykcolor^}%
   \replacestrings{-\setcmykcolor}{-1\setcmykcolor}%
   \def\C{0}\def\M{0}\def\Y{0}\def\K{0}%
   \tmpb \checkcmyk
   \xdef#1{\setcmykcolor{\C\space \M\space \Y\space \K}}%
   \egroup
}
\def\addcolor#1\setcmykcolor#2{\def\tmp{}%
   \tmpdim=\ifx$#1$1\else#1\fi pt
   \ifx^#2\expandafter \addcolorC
   \else \addcolorA#2 \fi
}
\def\addcolorA #1 #2 #3 #4 {%
   \addcolorB\C{#1}n\addcolorB\M{#2}n\addcolorB\Y{#3}n\addcolorB\K{#4}k%
}
\def\addcolorB#1#2#3{\dimen0=#2\tmpdim
   \ifx\tmp\empty\else
      \ifx#3n\advance\dimen0 by\K\tmpdim \dimen0=-\dimen0 \advance\dimen0 by\tmpdim
      \else \dimen0=0pt
   \fi\fi
   \advance\dimen0 by#1pt
   \ifdim\dimen0<0pt \def#1{0}\else \edef#1{\expandafter\ignorept\the\dimen0}\fi
}
\def\addcolorC#1{\def\tmp{^}\addcolorA#1 }

\def\checkcmyk{\tmpdim=\C pt
   \ifdim\M pt>\tmpdim \tmpdim=\M pt \fi
   \ifdim\Y pt>\tmpdim \tmpdim=\Y pt \fi
   \ifdim\K pt>\tmpdim \tmpdim=\K pt \fi
   \ifdim\tmpdim>1pt %\tmpdim=1/\tmpdim:
      \tmpnum=1073741824 \divide\tmpnum by\tmpdim \multiply\tmpnum by4 \tmpdim=\tmpnum sp
      \checkcmykA\C \checkcmykA\M \checkcmykA\Y \checkcmykA\K
   \fi
}
\def\checkcmykA#1{\dimen0=#1\tmpdim
   \ifdim\dimen0>1pt \def#1{1}\else \edef#1{\expandafter\ignorept\the\dimen0}\fi
}
```

2.9 Normalizování barvy CMYK

0106 *P. O.* 24. 04. 2015

Základní barvy CMYK by měly fungovat tak, že smíchání `a*C+a*M+a*Y` dá výsledek stejný jako `a*K`, tj. při plném smíchání `C+M+Y` bychom měli dostat černou. (To platí jen pro ideální tonery, ve skutečnosti dostaneme hodně tmavě hnědou.) Je tedy vidět, že pokud `min(C,M,Y)` není nula, dá se zhruba řečeno odečíst toto minimum od všech tří složek a nahradit složkou černou. Normalizovaná barva popsaná v CMYK tedy má toto minimum nulové, takže obsahuje nejvýše dvě ze tří barevných složek CMY nenulové. Normalizovaná barva CMYK tedy šetří barevnými tonery.

Míchání barev pomocí `\cmixdef` z předchozího OPmac triku může často způsobit, že výsledná barva není normalizovaná. Můžeme ji pak normalizovat makrem `\normalcmyk\Barva`, tedy třeba takto:

```
\cmixdef\myColor{lineární kombinace barev} \normalcmyk\myColor
```

Makro `\normalcmyk` odstraní z každé složky `C,M,Y` jejich společné minimum a přidá to k černé. Toto odstranění a přidání není přímočaré, ale do hry jsou zapojeny vzorečky na konverzi CMYK do RGB (OPmac trik 0090) a zpětně z RGB do CMYK (OPmac trik 0089), tentokrát v normalizovaném tvaru. Implementace je následující:

```
\def\normalcmyk#1{\bgroup \let\setcmykcolor=\relax
   \edef\tmp{#1}\def\tmpa{#1}\expandafter\normalcmykA\tmp
}
\def\normalcmykA#1#2{\normalcmykB #2 }
\def\normalcmykB#1 #2 #3 #4 {\tmpdim=#1pt
   \ifdim#2pt<\tmpdim \tmpdim=#2pt \fi  \ifdim#3pt<\tmpdim \tmpdim=#3pt \fi
   \ifdim\tmpdim=0pt \else
      \ifdim\tmpdim<1pt
         \dimen2=\tmpdim % \dimen2=min(C,M,Y)
         \tmpdim=-\tmpdim \advance\tmpdim by1pt \dimen1=\tmpdim % \dimen1 = 1-min
         \tmpnum=1073741824 \divide\tmpnum by\tmpdim \multiply\tmpnum by4 \tmpdim=\tmpnum sp
         \edef\tmp{\expandafter\ignorept\the\tmpdim}% \tmp = 1 / (1-min)
         \normalcmykC\C{#1}\normalcmykC\M{#2}\normalcmykC\Y{#3}%
         \dimen0=-#4\dimen2 \advance\dimen0 by\dimen2 \advance\dimen0 by#4pt
         \edef\K{\expandafter\ignorept\the\dimen0}% K_new = 1 - (1-min)*(1-K_old)
         \expandafter\xdef\tmpa{\setcmykcolor{\C\space\M\space\Y\space\K}}%
      \else \expandafter\xdef\tmpa{0 0 0 1}%
   \fi\fi \egroup
}
\def\normalcmykC#1#2{\ifdim\dimen2=#2pt \def#1{0}\else
      \dimen0=\dimen1 % C_new = ((1-min) - (1-C_old)) / (1-min)
      \advance\dimen0 by#2pt \advance\dimen0 by-1pt
      \dimen0=\tmp\dimen0 \ifdim\dimen0>1pt \dimen0=1pt \fi
      \edef#1{\expandafter\ignorept\the\dimen0}%
   \fi
}
```

2.10 Deklarace sady barev z X11

0108
P. O.
01. 05. 2015

Soubor `rgb.txt` obsahuje názvy barev a jejich RGB deklarace používané v X Window System. Tato sada názvů je přepsána v LaTeXovém souboru `x11nam.def`. Pokud jej správně načteme, budeme mít k dispozici sadu as tří set přepínačů barev s názvy dle tohoto souboru, tj. třeba `\DeepPink`, `\Azure`, `\CadetBlue` atd. Tyto jednotlivé názvy jsou v souboru navíc vždy ve čtyřech variantách. Jedničkou jsou označeny barvy nejvíc zářivé a další v pořadí 2, 3 a 4 jsou barvy odvozené tak, že je postupně ubrána barva a přidána šedá.

Načteme soubor `x11nam.def` tak, aby barvy označené jedničkou byly přístupné v přímém názvu (tj. místo `\Chocolate1` přímo `\Chocolate`) a barva s označením 2 měla v názvu připojeno písmeno `B`, barva 3 připojuje v názvu `C` a barva 4 připojuje `D`. Například

```
\def\zkus#1{{#1\vrule height10pt depth5pt width20pt}\kern2pt}
\zkus\DeepPink \zkus\DeepPinkB \zkus\DeepPinkC \zkus\DeepPinkD
```

vytvoří sadu čtyř růžových obdélníčků, každý další je poněkud zašpiněnější než ten předchozí. Povšimněte si, že barva `\DeepPinkA` neexistuje.

Načtení souboru `x11nam.def` provedeme takto:

```
\long\def\tmp#1\preparecolorset#2#3#4#5{\tmpa #5;,,,;}
\def\tmpa#1,#2,#3,#4;{\ifx,#1,\else
   \def\tmpb{#1}\replacestrings{1}{}\replacestrings{2}{B}%
   \replacestrings{3}{C}\replacestrings{4}{D}%
   \expandafter\def\csname\tmpb\endcsname{\setrgbcolor{#2 #3 #4}}%
   \expandafter\tmpa\fi
}
\expandafter\tmp\input x11nam.def
```

Dále je potřeba buď překlopit správu barev na RGB (podle OPmac triku 0090) nebo nastavit konverzi z RGB do CMYK (podle OPmac triku 0089).

3 Transformace

0046 P. O. 07. 04. 2014

3.1 Transformovaný box s přepočtenými rozměry

Vytvoříme makro `\transformbox{transformace}{obsah boxu}`, které vloží transformovaný `\hbox` do vnějšího boxu takových rozměrů, že minimálně obklopuje transformovaný box. Počátek transformace zůstává na účaří. Například

```
\picw=5cm \transformbox{\pdfrotate{45}}{\inspic cmelak.jpg }
```

vytiskne čmeláka otočeného o 45 stupňů. Je-li dejme tomu tento obrázek čtvercový, bude mít vnější box výšku i šířku `5cm * sqrt(2)`, hloubka zůstane nulová.

Čáry kolem naznačují hranici vnějšího boxu, ve skutečnosti se nevytisknou. Nebo třeba

```
\transformbox{\pdfscale{2}{1.2}\pdfrotate{-30}}{Box}
```

vytvoří box šířky, výšky a hloubky tak, jak je naznačeno na obrázku:

Je také možné definovat otáčení boxu jako `\rotbox{úhel}{text boxu}` jednoduše takto:

```
\def\rotbox#1#2{\transformbox{\pdfrotate{#1}}{#2}}
```

Makro `\transformbox` je definováno takto:

```
\newdimen\vvalX \newdimen\vvalY
\newdimen\newHt \newdimen\newDp \newdimen\newLt \newdimen\newRt
\let\oripdfsetmatrix=\pdfsetmatrix

\def\multiplyMxV #1 #2 #3 #4 {% matrix * (vvalX, vvalY)
   \tmpdim = #1\vvalX \advance\tmpdim by #3\vvalY
   \vvalY  = #4\vvalY \advance\vvalY  by #2\vvalX
   \vvalX = \tmpdim
}
\def\multiplyMxM #1 #2 #3 #4 {% currmatrix := currmatrix * matrix
   \vvalX=#1pt \vvalY=#2pt \expandafter\multiplyMxV \currmatrix
   \edef\tmpb{\expandafter\ignorept\the\vvalX\space \expandafter\ignorept\the\vvalY}%
   \vvalX=#3pt \vvalY=#4pt \expandafter\multiplyMxV \currmatrix
   \edef\currmatrix{\tmpb\space
      \expandafter\ignorept\the\vvalX\space \expandafter\ignorept\the\vvalY\space}%
}
\def\transformbox#1#2{\hbox{\setbox0=\hbox{#2}\preptransform{#1}%
```

```
   \kern-\newLt \vrule height\newHt depth\newDp width0pt
   \setbox0=\hbox{\box0}\ht0=0pt \dp0=0pt \pdfsave#1\rlap{\box0}\pdfrestore \kern\newRt}%
}
\def\preptransform #1{\def\currmatrix{1 0 0 1 }%
   \def\pdfsetmatrix##1{\edef\tmpb{##1 }\expandafter\multiplyMxM \tmpb\unskip}#1%
   \setnewHtDp 0pt  \ht0  \setnewHtDp 0pt  -\dp0
   \setnewHtDp \wd0 \ht0  \setnewHtDp \wd0 -\dp0
   \let\pdfsetmatrix=\oripdfsetmatrix
}
\def\setnewHtDp #1 #2 {%
   \vvalX=#1\relax \vvalY=#2\relax \expandafter\multiplyMxV \currmatrix
   \ifdim\vvalX<\newLt \newLt=\vvalX \fi \ifdim\vvalX>\newRt \newRt=\vvalX \fi
   \ifdim\vvalY>\newHt \newHt=\vvalY \fi \ifdim-\vvalY>\newDp \newDp=-\vvalY \fi
}
```

Makro implementuje maticovou aritmetiku pro currentmatrix. Dále dočasně (během `\preptranform`) předefinuje `\pdfsetmatrix` tak, že zadané transformace matici nenastaví, ale jen ji spočítají. Pak touto maticí pronásobíme čtyři body, které tvoří vrcholy boxu, a podíváme se na jejich obrazy. Podle jejich obrazů nastavíme nové rozměry vnějšího boxu `\newHt`, `\newDp` a dále prostor před počátkem transformace `\newLt` a za ním `\newRt`.

3.2 Jednoduché otočení boxu

0101
P. O.
18. 04. 2015

Velice často potřebujeme sazbu otočit jen o 90 nebo -90 stupňů. V takovém případě nemusíme používat složitá makra z předchozího OPmac triku a vystačíme si s podstatně jednodušším makrem

```
\rotsimple{90}\hbox{obsah otočeného boxu}
nebo
\rotsimple{-90}\vbox to\hsize{...}
```

které může být implementováno takto:

```
\def\rotsimple#1{\hbox\bgroup\def\tmpb{#1}\afterassignment\rotsimpleA \setbox0=}%
\def\rotsimpleA{\aftergroup\rotsimpleB}
\def\rotsimpleB{\setbox0=\hbox{\box0}%
   \ifnum\tmpb>0 \kern\ht0 \tmpdim=\dp0 \else \kern\dp0 \tmpdim=\ht0 \fi
   \vbox to\wd0{\ifnum\tmpb>0 \vfill\fi
               \vfil \wd0=0pt \dp0=0pt \ht0=0pt
               \pdfsave\pdfrotate{\tmpb}\box0 \pdfrestore
               \vfil}%
   \kern\tmpdim
   \egroup}
```

Povšimněte si, že toto makro dává celou původní šířku boxu do výšky nového boxu bez ohledu na to, zda otáčíme o 90 nebo -90 stupňů. Tím se toto makro také liší od `\rotbox` z předchozího OPmac triku, ve kterém při záporném úhlu rotace se obsah boxu stěhuje do hloubky nového boxu.

Chcete rotovat box kolem jeho středu? Pište třeba

```
\rotsimple{90}\hbox to0pt{\hss obsah otočeného boxu\hss}
```

Pravda, text výsledného otočeného boxu bude nahoru a dolu přesahovat přes nulové rozměry boxu, ale to v konkrétní aplikaci typicky nevadí.

3.3 Jednoduché zvětšení/zmenšení boxu

0104
P. O.
23. 04. 2015

Vytvoříme makra

```
\shbox to rozměr{text}
\svbox to rozměr{text}
```

která se chovají jako `\hbox` to `rozměr{text}` a `\vbox` to `rozměr{text}`, ale s tím rozdílem, že materiál uvnitř boxu nepruží a sestaví se se základní velikostí. Následně pomocí lineární

transformace se celý box zvětší nebo změnší, aby se dosáhla požadovaná šířka (pro `\hbox`) nebo výška (pro `\vbox`).

Implementace může být následující:

```
\def\shbox to#1#{\sboxA\hbox\wd{#1}}
\def\svbox to#1#{\sboxA\vbox\ht{#1}}
\def\sboxA#1#2#3#4{#1to#3{%
   \setbox0=#1{#4}\tmpdim=#3\relax \tmpnum=#20
   \divide\tmpnum by256 \divide\tmpdim by\tmpnum \multiply\tmpdim by256
   \edef\tmp{\expandafter\ignorept\the\tmpdim}%
   \ifx#1\hbox \vrule height\tmp\ht0 depth\tmp\dp0 width0pt
   \else \hrule width\tmp\wd0 height0pt \tmpdim=\dp0 \fi
   \ht0=0pt \dp0=0pt \wd0=0pt \pdfsave\pdfscale{\tmp}{\tmp}\box0\pdfrestore
   \ifx#1\hbox\hfil
   \else \vfil \hrule height0pt width0pt depth\tmp\tmpdim \fi}%
}
```

Nejprve je setaven box0, pak je vypočítán poměr požadované velikosti ku skutečné velikosti a je uložen do `\tmp`. Pak je pomocí `\pdfscale` tento box umístěn do vnějšího boxu s rozměry nastavenými pomocí neviditelných `\hrule`, `\vrule`.

3.4 Různě zakřivené šipky

0062 *P. O.* 24. 05. 2014

Vytvoříme makro

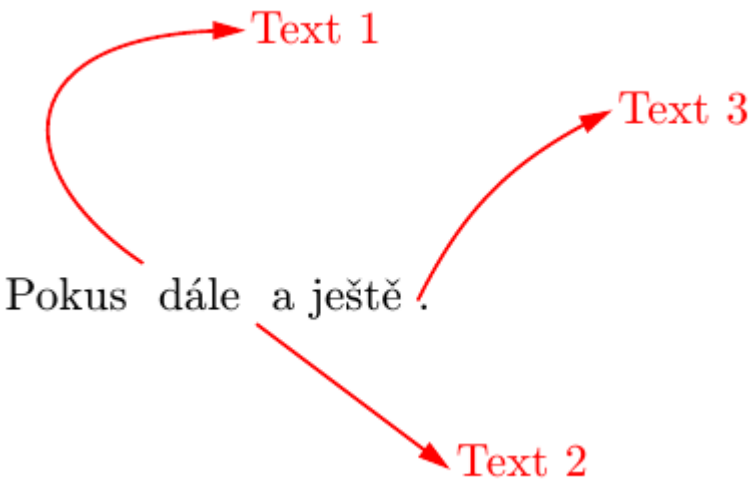
```
\arrowcc x0 y0 {cx0 cy0 cx1 cy1} x1 y1 (dx1 dy1) {Text}
```

které nakreslí čáru od `x0 y0` po `x1 y1` zakončenou šipkou. Část `{cx0 cy0 cx1 cy1}` může být prázdná, tedy `{}`, pak se vykreslí úsečka zakončená šipkou. Jinak čísla `{cx0 cy0 cx1 cy1}` udávají kontrolní body Bézierovy křivky. Na konci šipky se vypíše Text posunutý od konce šipky o `xd1 dy1`. Čísla `cx0 cy0 cx1 cy1 x1 y1` jsou relativní k počátku `x0 y0`. Tento počátek je relativní k pozici aktuálního bodu sazby. Vše je implicitně v jednotkách `bp`. V makru `\arrowccparams` je možné uložit výchozí nastavení (barvu, tloušťku čáry) a v makru `\arrowccspike` je možné deklarovat jinou kresbu hrotu, než je předdefinovaná. Například:

```
\vglue5cm
\def\arrowccparams{1 0 0 rg 1 0 0 RG}  % kresba bude červená

Pokus \arrowcc 0 10 {-30 20 -30 50} 20 50 (3 -3) {Text 1}
dále \arrowcc 0 -3 {} 40 -30 (3 -3) {Text 2}
a ještě \arrowcc 0 2 {10 20 20 30} 40 40 (3 -2) {Text 3}.
```

vytvoří:



```
\def\arrowccspike{2 0 m -5 2 l -5 -2 l h f}
\def\arrowcc #1 #2 #3 #4 #5 (#6)#7{%
   % x0 y0 {cx1 cy1 cx2 cy2} x1 y1 (dx1 dy1) {Text}
   \pdfsave\rlap{\pdfliteral{%
   .7 w \arrowccparams\space 1 0 0 1 #1 #2 cm 0 0 m
   \if ^#3^#4 #5 l \else #3 #4 #5 c \fi S 1 0 0 1 #4 #5 cm}%
   \if^#3^\calculateargofvector(0 0) (#4 #5)\else \preparedirection #3 (#4 #5)\fi
   \pdfsave\pdfrotate{\argofvector}\pdfliteral{\arrowccspike}\pdfrestore
   \if^#7^\else\pdfliteral{1 0 0 1 #6 cm}\hbox{#7}\fi}\pdfrestore
}
\def\preparedirection #1 #2 #3 #4 {\calculateargofvector(#3 #4) }
\def\arrowccparams{}
```

Makro využívá jednoduchou PDF grafiku a výpočet směru podle OPmac triku 0061.

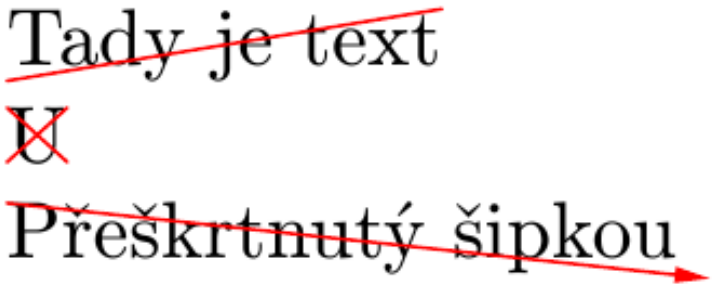
3.5 Přeškrtnutý text

0082
P. O.
19. 12. 2014

Makro `\cancel typ {text}` vytiskne text přeškrtnutý podle typ. Je-li typ `/`, bude text přeškrtnutý zdola nahoru, je-li typ `\\`, bude text přeškrtnutý shora dolů a je-li typ `X` nebo `x`, bude text přeškrtnutý oběma čarami. Tedy například

```
\cancel /{Tady je text}
\cancel X {U}
```

vytvoří první dva řádky ukázky:



K implementaci potřebujeme následující kód:

```
\def\cancel#1#2{\setbox0=\hbox{#2}%
   \dimen0=.9963\wd0  \dimen1=.9963\ht0 \advance\dimen1 by.9963\dp0
   \hbox{\rlap{#2}\vbox to0pt{\kern\dp0 \csname cancel:\string#1\endcsname \vss}\kern\wd0}%
}
\def\cancelB#1{\expandafter\ignorept\the\dimen#1 }
\sdef{cancel:x}{\pdfliteral{q 0.4 w 1 J \cancelcolor\space 0 0 m \cancelB0 \cancelB1 l
                           0 \cancelB1 m \cancelB0 0 l S Q}}
\expandafter\let\csname cancel:X\expandafter\endcsname \csname cancel:x\endcsname
\sdef{cancel:/}{\pdfliteral{q 0.4 w 1 J \cancelcolor\space 0 0 m \cancelB0 \cancelB1 l S Q}}
\sdef{cancel:\string\\}{\pdfliteral{q 0.4 w 1 J \cancelcolor\space 0 \cancelB1 m \cancelB0 0 l S Q}}
\def\cancelcolor{1 0 0 RG}
```

Chceme-li škrtat šipkou (jak ukazujeje poslední řádek v ukázce), je možné použít makro `\cancel` s typem `^` (šipka šikmo nahoru) nebo `_` (šipka šikmo dolů). Ukázka byla vytvořena pomocí `\cancel_{Přeškrtnutý šipkou}`.

V implementaci je využito makro `\arrowcc` z předchozího OPmac triku 0062

```
\sdef{cancel:^}{\arrowcc 0 0 {} \cancelB0 \cancelB1 (0 0) {}}
\sdef{cancel:_}{\arrowcc 0 \cancelB1 {} \cancelB0 -\cancelB1 (0 0) {}}
\def\arrowccparams{1 0 0 rg 1 0 0 RG .4 w 1 J}  % kresba bude červená
\def\arrowccspike{4 0 m -.3 1 l -.3 -1 l h f}
```

3.6 Text podél kružnice

0109
P. O.
01. 06. 2015

Navrhneme makro `\circletext {poloměr}{úhel}{TEXT}{korekce}`, které vypíše TEXT podél kružnice s daným poloměrem, přitom první písmeno začíná v místě výchozího úhlu.

Čtvrtý parametr umožňuje doplnit mezi dvojicemi znaků korekce (kerningové páry) k lepšímu vizuálnímu vyrovnání písmen, protože standardní tabulka kerningových párů je v při sazbě do kružnice vypnuta. Text jde při kladném údaji `poloměr` ve směru hodinových ručiček a střed kružnice je pod písmeny. Při záporném údaji `poloměr` jde text proti směru hodinových ručiček a střed kružnice je nad písmeny. Makro vytvoří bezrozměrnou sazbu, střed kružnice je v místě aktuálního bodu sazby. Například


```
\hbox{%
\circletext {1.7cm} {212}  {{$\bullet$} UNIVERSITAS CAROLINA PRAGENSIS {$\bullet$}}
                           {\spaceskip=.5em \kpcirc TA{-.1}\kpcirc NA{-.05}}
\circletext {-1.7cm} {237} {Facultas MFF}
                           {\kpcirc Fa{-.1}}
}
```

Ve čtvrtém parametru `korekce` je možné nastavit mezislovní mezeru pomocí `\spaceskip=...` a dále korekci mezi písmeny pomocí makra `\kpcirc AB{num}`, které vloží mezi každou dvojici písmen AB `\kern num` v jednotkách `em`. V ukázce je také sazba „složitějšího“ objektu (`$\bullet$`), který musí být v textu uzavřen do svorek.

Jednotlivá písmena ve slovech mohou být prostrkaná, pokud definujete `\def\circletextS{\kern hodnota` Implicitně je nastaveno nulové prostrkání.

Makro `\circletext` může být definováno takto:

```
\newdimen\tmpdimA
\def\circletext#1#2#3#4{\hbox\bgroup
    \tmpdim=14668pt \tmpdimA=#1 \divide\tmpdimA by256 \divide\tmpdim by\tmpdimA
    \edef\tmpC{\expandafter\ignorept\the\tmpdim}% \tmpC=(180/pi)/R
    \setbox0=\hbox{\ifdim#1<0pt X\fi}% lap letters by X height if R<0
    \tmpdimA=0pt \baselineskip=#1 \advance\baselineskip by-\ht0 \lineskiplimit=-\maxdimen
    \tmpdim=#2pt \advance\tmpdim by-\ifdim#1<0pt-\fi 90pt
    \def\tmpb{{}#3}\replacestrings{ }{{ }}#4% spaces => {spaces}
    \pdfsave \pdfrotate{\expandafter\ignorept\the\tmpdim}%
    \expandafter\circletextA\tmpb\relax
}
\def\circletextA#1{\ifx#1\relax\pdfrestore\egroup\ignorespaces\else
    \ifx^#1^\else \setbox0=\hbox{#1\circletextS}%
       \ifdim\tmpdimA=0pt \else
          \advance\tmpdimA by.5\wd0 \dimen0=\tmpC\tmpdimA
          \pdfrotate{-\expandafter\ignorept\the\dimen0}%
       \fi
       \tmpdimA=.5\wd0
       \vbox to0pt{\vss\hbox to0pt{\hss#1\hss}\null}%
    \fi
    \expandafter\circletextA\fi
}
\def\kpcirc#1#2#3{\replacestrings{#1#2}{#1{\kern#3em}#2}}
\def\circletextS{}
```

3.7 Úprava makra \ignorept a návrh makra \nopt

0169
Petr Krajník
23. 06. 2019

Při použití `\pdfliteral` se často vyskytne potřeba použít makro `\ignorept`. Pak stojí za to, aby `\ignorept` generovalo co nejmíň číslic do výsledného kódu. To lze zařídit následujícím předefinováním makra `\ignorept`:

```
\begingroup \lccode`\?=`\p \lccode`\!=`\t \lowercase{\endgroup
   \def\ignorept #1.#2?!{% Print minimized point value
       \ifnum#2=0 #1\else \ifnum#1=0 \expandafter\remzero\fi #1.#2\fi}
}
\def\remzero #10{#1}% Remove zero and leave minus if present
```

Makro vygeneruje číslo v minimální možné formě. Pro PDF kód ušetříme mnoho zbytečných nul a teček. Vše se provádí v expand procesoru.

Dále je užitečné zavést makro `\nopt`, které se postará o zachování mezery za použitím makra a umožní výpočet uvnitř svého parametru, tj. třeba `\nopt[1.65\hoffset + 1in]` vygeneruje v argumentu `\pdfliteral` odpovídající konstantu.

```
\def\nopt[#1]{\expandafter\ignorept\the\dimexpr #1\relax}
```

Například kód v OPmac triku 0021 pak vypadá tato:

```
\def\prepghook{\dimen0=.996264truein
  \pdfliteral{q \bgcolor\space k -0.996264 0 0 0.996264 -\nopt[\dimen0] \nopt[\dimen0] cm
  \nopt[\hoffset] \nopt[\voffset] -\nopt[\pdfpagewidth] -\nopt[\pdfpageheight] re f Q}}
```

4 Stránka

4.1 Plovoucí záhlaví

0020
P. O.
30. 08. 2013

OPmac neřeší záhlaví v duchu myšlenky, že návrh typografie dokumentu patří do jiného souboru maker. Nicméně názvy sekcí vkládá do `\mark`, takže je můžeme rovnou vyzvednout pomocí `\firstmark` nebo něčeho podobného. Následující kód navíc přidává na sudé stránky názvy kapitol. Kapitoly vždy zahajují stránku a na stránce se zahájením kapitoly záhlaví záměrně chybí – je až na dalších stránkách.

```
\edef\oriheadline{\the\headline}
\def\printchap#1{\vfil\break
   \headline={\oriheadline\hfil\global\headline={\oriheadline\printheadline}}
   \xdef\headchap{\ifnonum\else\thechapnum. \fi}\global\addto\headchap{#1}
   {\chapfont \noindent \mtext{chap} \dotocnum{\thetocnum}\par
    \nobreak\smallskip\noindent #1\nbpar}\mark{}%
   \nobreak \remskip\bigskipamount \firstnoindent
}
\def\printheadline{\lower4pt\null\vadjust{\hrule}\tenit\thefontsize[10]
   \ifodd\pageno \hfill\firstmark \else \headchap\hfill \fi}
```

Tento kód zachovává původní `\headline` v `\oriheadline` (OPmac jej používá na tisk vodoznaku DRAFT). Dále je z OPmac opsáno makro `\printchap`, ve kterém jsou navíc přidány druhý a třetí řádek. Tam je vyřešeno odložené záhlaví a uložení názvu kapitoly do `\headchap` (expandované číslo a neexpandovaný zbytek). Tisk samotného záhlaví provádí makro `\printheadline`.

4.2 Podkladový obrázek nebo barva na každé straně

0021
P. O.
30. 08. 2013

Tento trik je možné použít obecně pro jakýkoli dokument, ale nejčastější použití bude asi mít pro slídy, aby přestaly být spartánského vzhledu. V nějakém rozumném editoru pro obrázky si připravíme soubor background.pdf a do dokumentu na každou stránku jej dopravíme takto:

```
\def\prepghook{\vbox to0pt{\kern-\voffset\kern-1in
   \hbox to0pt{\kern-\hoffset\kern-1in\background\hss}\vss}%
   \nointerlineskip
}
```

```
\pdfximage width\pdfpagewidth height\pdfpageheight {background.pdf}
\mathchardef\picbackground=\pdflastximage
\def\background{\pdfrefximage\picbackground}
```

Od verze OPmac May 2015 je k dispozici `\prepghook`, který se spustí před konečným sestavením strany ve výstupní rutině. Je třeba box s obrázkem posunout o `\voffset+1in` nahoru a `\hoffset+1in` doleva.

Pomocí `\pdfximage` je načten do dokumentu obrázek požadovaných rozměrů a odkaz na něj se jmenuje `\picbackground`. Makro `\background` pak vloží obrázek odkazem, tj. nepřidává do PDF nová data s obrázkem a tím minimalizuje velikost výstupního PDF.

Pokud chceme mít podklad pouze v jedné barvě (jiné než bílá), není třeba kvůli tomu vytvářet podkladový obrázek. Stačí barevný podklad vložit pomocí `\pdfliteral`:

```
\def\prepghook{\pdfliteral{q \bgcolor\space k -0.996264 0 0 0.996264 -72 72 cm
  \nopt{\hoffset} \nopt{\voffset} -\nopt{\pdfpagewidth} -\nopt{\pdfpageheight} re f Q}}
\def\nopt#1{\expandafter\ignorept\the#1}

\def\setbasecolor#1{#1\expandafter\setbasecolorA#1\pdfblackcolor}
\def\setbgcolor#1{\expandafter\setbasecolorA#1\bgcolor}
\def\setbasecolorA#1#2#3{\def#3{#2}}

\setbasecolor\Yellow
\setbgcolor\Blue

Ahoj. Tady je žlutý text na modrém podkladu.
```

Výše uvedený kód předpokládá, že nepoužijete v dokumentu změnu velikosti pomocí `\magnification` nebo `\magscale`. Máte-li toto v úmyslu, musíte čísla `-72 72` nahradit „přepočítanou“ velikostí `1 in`:

```
\def\prepghook{\dimen0=.996264truein
  \pdfliteral{q \bgcolor\space k -0.996264 0 0 0.996264 -\nopt{\dimen0 } \nopt{\dimen0 } cm
  \nopt{\hoffset} \nopt{\voffset} -\nopt{\pdfpagewidth} -\nopt{\pdfpageheight} re f Q}}
```

0058 P. O. 12. 05. 2014

4.3 Zlom stránky jen někdy

PlainTeX nabízí pro zlom stránky použít `\vfill\break`. Pokud autor toto začne vpisovat do svého dokumentu, aby „vylepšil“ stránkový zlom, a později připíše dopředu nějaké další informace a na toto své vylepšení zapomene, dostane ke své škodě třeba jen popolrázdnou stránku. OPmac proto nabízí od verze May 2014 makro `\maybebreak rozměr` (například `\maybebreak 2cm`), které zlomí stránku nebo řádek, pokud do konce stránky nebo řádku zbývá zhruba méně než `rozměr`. Když je tam místa více, makro neudělá nic.

Makro v horizontálním módu řeší řádkový zlom a ve vertikálním módu stránkový zlom. Pokud tedy chcete stránkový zlom, pište třeba

```
\par\maybebreak 3.5cm
```

0076 P. O. 28. 07. 2014

4.4 Sazba do sloupců jako v novinách

Sazeč novin může natáhnout jednotlivé články do boxů pomocí:

```
\setbox\clanekA=\vbox{\hsize=sirka sloupce \penalty0 \input clanekA.tex }
\setbox\clanekB=\vbox{\hsize=sirka sloupce \penalty0 \input clanekB.tex }
...
```

a dále může ulamovat z těchto boxů jednotlivé části a vkládat je do struktury stránky, kterou sám navrhne pomocí `\hbox/\vbox/\vtop` aritmetiky. Tato myšlenka je rozebrána v TBN na straně 275 a zde jsou některá makra doplněna, aby bylo přesně dodrženo řádkování. Například třísloupcová sazba, ve které druhému a třetímu sloupci vodorovně překáží obrázek (ukázka je zde) může být naprogramována takto:

```
\ulamuj\clanekA
\hbox{\ulom[18]\kern\colsep
      \vtop{\hbox {\ulom[3]\kern\colsep \ulom[3]}
      \vynech[13] \placepic[0pt,\twocols,\picw=9cm]{ovce1.jpg}
      \hbox {\ulom[2]\kern\colsep \ulom[2]}
      }}
```

Makra `\ulom` a `\vynech` mají parametr v hranaté závorce udávající počet odlomených řádků nebo počet vynechaných řádků. V této ukázce čtenář musí nejprve přečíst vše nad obrázkem (druhý i třetí sloupec) a pak pokračuje pod obrázkem. Pokud naopak chceme, aby čtenář nejprve přečetl kompletní druhý sloupec (bez ohledu na obrázek) a pak třetí, pak program pro strukturu stránky může vypadat takto:

```
\hbox{\ulom[21]\kern\colsep
      \vtop{\ulom[3]%
            \vynech[14] \placepic[0pt,\twocols,\picw=10.5cm]{ovce2.jpg}
            \ulom[4]}\kern\colsep
      \vtop{\ulom[3]\vynech[14]\ulom[4]}%
     }
```

Makro `\ulamuj` připraví článek k odlamování. Makro `\ulom` odlomí řádky pomocí `\vsplit` a vypočte hloubku posledního řádku pomocí triku s `\lastbox` (viz TBN str.˜450) a uloží ji do `\lastdepth`. Je-li `\ulom` v horizontálním módu, tj. např. `\hbox{\ulom[9]\ulom[9]\ulom[9]}`, pak další výpočet využívající pomocnou proměnnou `\maxhdepth` zajistí, že nakonec je v `\lastdepth` největší hloubka ze všech `\ulom`, které byly v `\hbox` použity. Makro `\vynech` pak údaj z `\lastdepth` použije.

```
\tmpdim=\baselineskip  \splittopskip=\tmpdim plus.1pt minus.1pt
\def\ulom[#1]{\setbox0=\vsplit\celybox to #1\baselineskip
   \vtop{\kern-.3\baselineskip \unvbox0
         \nointerlineskip\lastbox \global\lastdepth=\prevdepth}%
   \ifhmode \ifdim\lastdepth<\maxhdepth \global\lastdepth=\maxhdepth \fi
            \maxhdepth=\lastdepth \fi
}
\def\ulamuj#1{\let\celybox=#1\setbox0=\vsplit\celybox to0pt}
\def\vynech[#1]{\vskip-.7\baselineskip\vskip#1\baselineskip \prevdepth=\lastdepth}
```

Kompletně celý kód ukázky včetně deklarace a nastavení rozměrů a včetně definice makra

```
\placepic[vertikální posun, šířka boxu, makra před \inspic] {obrázek.přípona}
```

které vloží obrázek tak, aby nezbíral v sazbě žádné místo, je v samostatnném souboru.

4.5 Ořezové značky

0081
P. O.
16. 12. 2014

Vytvoříme makro `\cropmarks`, které po použití makra `\margins` přidá ke stránce ořezové značky. Tj. například:

```
\sdef{pgs:spec}{(150,170)mm}
\margins/1 spec (7,7,7,7)mm \cropmarks
```

Tento příklad deklaruje stránku rozměru `150x170mm` s okraji `7mm`. Příkaz `\cropmarks` zvětší stránku na všech čtyřech stranách o `10mm` a v tomto přidaném okraji v rozích umístí ořezové značky. Je možné po takové deklaraci použít ještě třeba:

```
\margins/1 a4 (,,,)mm \cropcenter
```

což zachová značky vzhledem k sazbě na stejném místě, ale vše umístí doprostřed stránky A4.

Implementace makra může být následující:

```
\newdimen\cropw \cropw=10mm
\def\ltcrop{\vbox{\hbox to\cropw{\hfil\vrule height\cropw depth-.2\cropw}
                  \hbox to\cropw{\vrule height.4pt width.8\cropw \hfil}}}
\def\rtcrop{\vbox{\hbox to\cropw{\vrule height\cropw depth-.2\cropw\hfil}
                  \hbox to\cropw{\hfil\vrule height.4pt width.8\cropw}}}
\def\lbcrop{\vbox{\hbox to\cropw{\vrule height.4pt width.8\cropw \hfil}
                  \kern.2\cropw \hbox to\cropw{\hfil\vrule height.8\cropw}}}
\def\rbcrop{\vbox{\hbox to\cropw{\hfil\vrule height.4pt width.8\cropw}
                  \kern.2\cropw \hbox to\cropw{\vrule height.8\cropw\hfil}}}

\newdimen\lmar \newdimen\tmar \tmar=1truein \lmar=1truein
\def\cropmarks{%
   \ifx\cropwidth\undefined
      \advance\lmar by\hoffset \advance\tmar by\voffset
      \hoffset=-1truein \voffset=-1truein
      \advance\pdfpagewidth by2\cropw \advance\pdfpageheight by2\cropw
      \dimen0=\pgwidth \advance\dimen0 by2\cropw \edef\cropwidth{\the\dimen0}%
      \edef\cropheight{\the\pgheight}
      \let\shipoutori=\shipout
      \def\shipout##1 {\shipoutori % \opmacoutput uses \shipout\box0
         \vbox{\let\vrule=\orivrule \let\hrule=\orihrule
               \offinterlineskip \kern.2pt
               \hbox to\cropwidth{\kern.2pt\ltcrop\hfil\rtcrop\kern.2pt}%
               \kern-.2pt
               \vbox to\cropheight{\kern\tmar\hbox{\kern\cropw\kern\lmar\box0}\vss}
               \kern-.2pt
               \hbox to\cropwidth{\kern.2pt\lbcrop\hfil\rbcrop\kern2.pt}}}%
   \else\errmessage{\noexpand\cropmarks can't by used twice}\fi
}
\def\cropcenter{\advance\hoffset by-\lmar \advance\hoffset by-\cropw
                \advance\voffset by-\tmar \advance\voffset by-\cropw}
```

4.6 Jednostránkový dokument bez čísla stránky

0133 *Jan Šustek* 19. 12. 2015

Při psaní krátkých dokumentů často dopředu nevíme, jestli budou mít jednu, nebo více stran. Přitom, pokud bude mít dokument jen jednu stranu, nechceme, aby obsahoval číslo stránky. Abychom se nemuseli starat, jestli číslo stránky má být, nebo nemá být, stačí na začátek dokumentu napsat

```
\openref\ifnum\lastpage=1\footline{\hss}\fi
```

Pro správné fungování je nutné TeXovat dokument dvakrát.

Trik využívám například při psaní posudků, které mívají 1–2 stránky.

4.7 PDF stránečky dle velikosti boxu

0147 *P. O.* 16. 05. 2017

Vytvoříme makro `\outbox`, které okamžitě vloží do stránky PDF dokumentu následující box, přitom rozměry stránky (medium size) budou stejné, jako rozměry tohoto boxu (zvětšené o okraje). Okraje kolem boxu se dají nastavit pomocí `\outboxmargins(levý,pravý,horní,dolní)jednotka` a jsou implicitně nastaveny na 1pt z každé strany. Takže například:

```
\outboxmargins(1,1,2,2)pt

\outbox\hbox{text}     % vytvoří maličkou stránku s textem "text"
                       % a se stanovenými okraji
\outbox\vbox{odstavec} % vytvoří stránku o šířce odstavce
                       % a výšce textu v odstavci.
```

Implementace je následující

```
\def\outbox{\afterassignment\outboxA \setbox0=}
\def\outboxA{\aftergroup\outboxC}
\def\outboxC{\setbox0=\outboxB
```

```
   \begingroup
      \hoffset=-1in \voffset=-1in
      \pdfpagewidth=\wd0 \pdfpageheight=\ht0
      \shipout\box0
   \endgroup
}
\def\outboxmargins (#1,#2,#3,#4)#5 {\def\outboxB{%
   \vbox{\kern#3#5\hbox{\kern#1#5\box0 \kern#2#5}\kern#4#5}}%
}
\outboxmargins (1,1,1,1)pt
```

Makro `\outbox` spustí po přečtení následujícího boxu makro `\outboxC`. To je potřeba udělat přechodně pomocí `\outboxA`, důvod je vysvětlen v TBN na straně 338. Dále makro `\outboxmargins` připraví makro `\outboxB`, které přidá boxu 0 požadované okraje. Konečně makro `\outboxC` přeboxuje box0 tak, že jsou přidány okraje a velikosti tohoto nového boxu použije pro velikosti média nastavením `\pdfpagewidth` a `\pdfpageheight`. Nakonec se box 0 vyvrhne do samostatné stránky pomocí `\shipout`, což obchází `\output` rutinu.

5 Verbatim prostředí

5.1 Lokální tthook

0002
P. O.
15. 05. 2013

Napíšeme-li `\def\tthook{kód}`, ovlivní tato definice všechna následující prostředí `\begtt...\endtt`. My bychom chtěli ale před `\begtt` psát kódy, které ovlivní jen jedno následující prostředí. Navíc je problém se zdvojováním křížů při zápisu kódu, který způsobuje, že nelze jednoduše použít například `\addto\tthook{kód}`.

Řešením jsou dva registry typu toks: `\gtthook` (obsahuje globální kód ovlivňující všechny následující `\begtt...\endtt` prostředí) a `\ltthook` (obsahuje kód, který se použije jen v následujícím verbatim prostředí). Uživatel může psát například:

```
\gtthook {\typosize[9/11]}    % všechna prostředí budou v 9pt
\ltthook {\adef!#1.{{\it#1}}} % aktivní ! jen pro jedno následující prostředí
\begtt
... !aha. ...
/endtt
```

Kód makra:

```
\newtoks\gtthook  \newtoks\ltthook
\def\tthook{\the\gtthook \the\ltthook \global\ltthook{}}
```

Pokud navíc definujeme

```
\def\addtoks#1#2{#1\expandafter{\the#1#2}}
```

je možné použít například:

```
\bgroup \addtoks\gtthook{kód} ... \egroup
```

a příslušný kód ovlivní všechna verbatim prostředí v rozmezí `\bgroup ... \egroup`. Kódy nevyžadují zdvojování křížů.

Analogická myšlenka se dá použít pro ostatní `\cosihook` z OPmac.

5.2 Verbatim v nadpisech sekcí

0102
P. O.
22. 04. 2015

Verbatim prostředí deklarované pomocí `\activettchar` (např. `\activettchar"`) funguje jen tehdy, když je text mimo parametr jiného makra. Rozhodně nefunguje v nadpisech (např. nelze psát `\sec Cosi "\uff#$"`). Nyní zavedeme makro `\code{text}`, které vytvoří to samé, jako „text“, ale bude možné je bez obav vkládat do parametrů maker včetně nadpisů, ze kterých se texty správně propagují do obsahu, do záhlaví a do PDF záložek. Za cenu této vymoženosti

bude občas potřeba zapsat text pro `\code` ne zcela doslova, ale se znaky backlash navíc. Pravidla jsou tato:

* Místo `\` pište `\\`, místo `#` pište `\#` a místo `%` pište `\%`. * Vše ostatní se přepisuje doslova. * Před další speciální znaky `{`, `}`, `$`, `&`, `^`, `_`, ■ (mezera), `~` můžete ale nemusíte psát backlash. Tento backlash se nezobrazí. * Parametr `{text}` musí být balancovaný vzhledem k závorkám `{}`. Chcete-li vložit do textu nebalancovanou závorku, pište `\{`. * Dvojice zobáků `^^ab` se promění ve specifikovaný znak. Pokud tomu chcete zabránit, pište `^\^ab`. * Více mezer za sebou se slijí do jediné mezery. Pokud tomu chcete zabránit, pište `\■\■` atd.

Příklady:

```
\code{ah_a uff{} &$^}           % vytvoří: ah_a uff{} &$^
\code{\\def\\cosi\#1{cosi=\#1}} % vytvoří: \def\cosi#1{cosi=#1}
\code{a\%b\ \ \ c\%d}           % vytvoří: a%b   c%d
```

Implementace:

```
\def\code#1{\def\tmpb{#1}\edef\tmpb{\expandafter\stripm\meaning\tmpb\relax}%
   \expandafter\replacestrings\expandafter{\string\\}{\bslash}%
   \expandafter\replacestrings\bslash{}%
   \codeP{\leavevmode\hbox{\tt\tmpb}}}
\def\codeP#1{#1}
\def\stripm#1->#2\relax{#2}
\addprotect\code \addprotect\\ \addprotect\{ \addprotect\}
\addprotect\^ \addprotect\_ \addprotect\~
\setcnvcodesA
\addto\cnvhook{\let\code=\codeP}
```

Makro `\code` detokenizuje svůj parametr pomocí `\meaning` a dále pomocí `\replacestrings` provede záměnu `\\` za jediný `\` a ostatní jediné `\` odstraní. Při `\write` se `\code` ani `\\`, `\{`, `\}`, `\$` atd. neexpandují a zapíší se do REF souboru tak, jak jsou. Při zápisu do záložek se v klasickém módu zapíše text stejně jako při `\write` a PDF prohlížeč automaticky nezobrazí backslashe. Pokud ale používáte záložky s `pdfuni.tex`, je potřeba detokenizovat parametry `\code` před použitím `\pdfunidef`, což udělá makro `\precode`:

```
\ifx\pdfunidef\undefined\else \def\cnvhook #1#2{#2\precode \pdfunidef\tmp\tmp}\fi
\def\precode{\def\codeP##1{}\let\bslash=\Bslash
   \edef\tmp{\expandafter}\expandafter\precodeA\tmp\code{}}
\def\precodeA#1\code#2{\addto\tmp{#1}%
   \ifx\end#2\end \else
      \codeA{#2}%
      \expandafter\addto\expandafter\tmp\expandafter{\tmpb}%
      \expandafter\precodeA \fi
}
\let\codeA=\code
```

0123

5.3 Zákaz zlomu mezi řádky výpisu

P. O.
11. 09. 2015

Balíček fancyvrb z LaTeXu umožňuje vkládat `\clubpenalty` za první řádek výpisu a `\widowpenalty` před poslední řádek, takže dokážete zamezit stránkovému zlomu, kdy na stránce zbyde jen osiřelý řádek výpisu. Implementace ve fancyvrb je poněkud těžkopádná. My to uděláme elegantněji a navíc umožníme vkládat údaje pomocí eTeXových primitivů `\clubpenalties` a `\widowpenalties`, tedy postihneme penaltou i více řádků na začátku a na konci výpisu. To fancyvrb nedokáže.

```
\def\tthook{%
   \clubpenalties   3 10000 10000 25
   \widowpenalties  3 10000 10000 25
   \setbox0=\vbox\bgroup \aftergroup\tthookA}
\def\tthookA{\setbox2=\hbox{}\setbox0=\vbox\bgroup\unvbox0 \tthookB}
\def\tthookB{\unskip\unskip\unpenalty \setbox0=\lastbox
   \ifvoid0 \egroup \noindent\unhbox2 \par \egroup
   \else \global\setbox2=\hbox{\box0\penalty0\unhbox2}%
```

```
      \expandafter\tthookB
   \fi}
```

Následující výpis se zlomí jedině mezi třetím a čtvrtým řádkem:

```
\begtt
  první
  druhý
  třetí
  čtvrtý
  pátý
  šestý
/endtt
```

Pomocí `\tthook` necháme celý verbatim výpis setavit do boxu0, ten pak pomocí `\tthookA` rozebereme na řádky a tyto řádky vložíme vedle sebe do hboxu2. Nakonec spustíme `\noindent\unhbox2\par`, takže vystartujeme běžný TeXový odstavec, který se rozlomí do právě připravených řádků a pod řádky pochopitelně vloží deklarované penalty.

5.4 Automatické zvýraznění/obarvení syntaxe

0124
P. O.
27. 09. 2015

Zvýraznění syntaxe (syntax highlighting) nějakého programovacího jazyka mezi `\begtt` a `\endtt` provedeme pomocí `\hisyntax{jazyk}`, což napíšeme těsně před `\begtt` nebo před `\verbinput`. Například:

```
#include <stdio.h>
int main(int argc, char **argv)
{
    printf("Hello World!\n"); // Prints the message
    return 0;
}
```

```
\hisyntax{C}
\begtt
#include <stdio.h>
int main(int argc, char **argv)
{
   printf("Hello World!\n"); // Prints the message
   return 0;
}
/endtt
```

Makro `\hisyntax` lokálně předefinuje zpracování stringu mezi `\begtt...\endtt` pomocí `\ptthook` (od verze OPmac Oct. 2015). Makro `\ghisyntax` na začátku dokumentu způsobí obarvení všech výpisů v dokumentu. Při `\hisyntax{C}` (resp. `\ghisyntax{C}`) se spustí interní `\hisyntaxC` a analogicky to můžete udělat pro jiné jazyky. Definice `\hisyntaxC` může vypadat takto:

```
\def\hisyntax#1{\bgroup\def\ptthook{\csname hisyntax#1\endcsname}}
\def\ghisyntax#1{\def\ptthook{\bgroup\csname hisyntax#1\endcsname}}

\def\hisyntaxC#1\egroup{\let\n=\relax \let\NLh=\relax \let\U=\relax
   \let\TABchar=\relax % pro OPmac trik 0151
   \adef{ }{\n\ \n}\adef\^^M{\n\NLh\n}\edef\tmpb{#1\egroup}%
   \replacestrings{\n\egroup}{\egroup}%
   \replacestrings{/*}{\tmp}\def\tmp##1*/{\U{commentC}{##1}}\edef\tmpb{\tmpb}%
   \replacestrings{//}{\tmp}\def\tmp##1\NLh{\U{commentCpp}{##1}}\edef\tmpb{\tmpb}%
   \edef\tmp{\noexpand\replacestrings{\string\"}{\n{\string\"}}}\tmp
   \replacestrings{"}{\tmp}\def\tmp##1\tmp{\U{stringC}{##1}}\edef\tmpb{\tmpb}%
   \doreplace{##1}{\n{\chCcolor##1}\n}\charsC{}%
   \doreplace{\n##1\n}{\kwC{##1}}\keywordsC{}%
   \edef\tmp{\noexpand\replacestrings{\NLh\n\string##}{\NLh\noexpand\preprocC}}\tmp
```

```
   \doreplace{\n##1}{\numberC##1}0123456789{}%
   \let\NLh=\par \def\n{}\def\U##1{\csname##1\endcsname}%
   \tentt\localcolor\tmpb\egroup}
\def\doreplace#1#2{\def\do##1{\ifx^##1^\else \replacestrings{#1}{#2}\expandafter\do\fi}%
   \expandafter\do}
\def\printttline{\llap{\sevenrm\Black\the\ttline\kern.9em}} % \Black added

\def\commentC#1{{\Green/*#1*/}}
\def\commentCpp#1{{\Green//#1}\NLh}
\def\stringC#1{{\def\ {\char32 }\Magenta"#1"}}
\def\preprocC#1\n{{\Blue\##1}}
\def\numberC#1\n{{\Cyan#1}}
\edef\keywordsC{{auto}{break}{case}{char}{continue}{default}{do}{double}%
   {else}{entry}{enum}{extern}{float}{for}{goto}{if}{int}{long}{register}%
   {return}{short}{sizeof}{static}{struct}{switch}{typedef}{union}
   {unsigned}{void}{while}} % zde jsou všechna klíčová slova jazyka C
\def\kwC#1{{\Red#1}}
\edef\charsC{()\string{\string}+-*/=[]<>,:;\percent\string&\string^|!?}
\let\chCcolor=\Blue
```

Načtený verbatim string je nejpre expandován do `\tmpb`, kde bude dále postupně měněn pomocí `\replacestrings`. Při vstupní expanzi jsou mezery vyměněny za `\n\ \n` a konce řádků za `\n\NLh\n`. Symbol `\n` je netisknoucí značka, která se tak objeví na okrajích slov, takže půjde vyměnit třeba klíčové slovo `\n if\n` za `\kwC{if}`, a přitom se nevymění úsek slova `if` třeba v proměnné `diff`. Dále jsou provedeny náhrady:

```
/* cosi */              -->   \U{commentC}{ cosi }
// cosi (konec řádku)   -->   \U{commentCpp}{ cosi }
\"                      -->   \n{\"}
"cosi"                  -->   \U{stringC}{cosi}
```

Všimněte si, že text „cosi“ je po náhradě uvězněn mezi `{...}`, takže takový text je imunní vůči pozdějším pokusům o náhrady. Konečně jsou všechny znaky typu `(){}+-...` nahrazeny `\n{\chColor znak}\n`, a dále jsou nahrazena všechna klíčová slova a slova preprocesoru. Co nebylo nahrazeno, zůstává černé (měly by to být jen identifikátory jazyka). Také je provedena náhrada `\n číslo` za `\numC číslo`, aby mohlo makro `\numC` obarvit samotná čísla, která nejsou součástí identifikátorů.

5.5 Zvýraznění HTML kódu

0125
P. O.
03. 10. 2015

```
<div class="navis">
<nav id="site">
<div class="container">
<ul class="row" id="navigation">
<li class="span2"><h6><a href="/fakulta" >Fakulta</a></h6>
<ul><li><a href="/fakulta/ud.htm" >Úřední deska</a></li>
    <li><a href="/fakulta/pracoviste" >Přehled pracovišť</a></li>
    <li><a href="/fakulta/predpisy" >Vnitřní předpisy</a></li>
    <li><a href="/fakulta/budovy" >Budovy a areály</a></li>
    <li><a href="/fakulta/tiskoviny" >Informační tiskoviny</a></li>
</ul></li>
<li class="span2"><h6><a href="/studium/uchazec">Pro uchazeče</a></h6>
<ul><li><a href="/studium/studium-bc.htm">Bakalářské studium</a></li>
    <li><a href="/studium/studium-mgr.htm">Magisterské studium</a></li>
    <li><a href="/studium/prijriz-phd.htm">Doktorské studium</a></li>
    <li><a href="/studium/prihlaska/" >Elektronická přihláška</a></li>
</ul></li></ul>
</div>
</nav>
<nav id="other" class="container">
   <span class="phonav">
      <a href="#navigation" class="sider">Jiná stránka</a>
      <a href="#" class="search">Vyhledat</a>
   </span>
   <a href="/to.en/dekan/2011/smer05.htm" class="lang">English</a>
</nav>
</div>
```

Předvedeme příklad modifikace předchozího OPmac triku na zvýrazňování jazyka C, tentokrát zvýrazníme HTML nebo XML kód. Po:

```
\hisyntax{HTML}
\begtt
<div class="navis">
<nav id="site">
...
</nav>
</div>
/endtt
```

dostaneme výstup, jak je znázorněno vpravo. Implementace:

```
\def\hisyntax#1{\bgroup\def\ptthook{\csname hisyntax#1\endcsname}}
\def\ghisyntax#1{\def\ptthook{\bgroup\csname hisyntax#1\endcsname}}

\def\hisyntaxHTML#1\egroup{\let\n=\relax \let\NLh=\relax \let\U=\relax
   \let\TABchar=\relax % pro OPmac trik 0151
   \adef{ }{\n\ }\adef\^^M{\n\NLh}\edef\tmpb{#1\egroup}%
   \replacestrings{<!--}{\tmp}\def\tmp##1-->{\U{commentH}{##1}}\edef\tmpb{\tmpb}%
   \replacestrings{<}{\tmp}\def\tmp##1>{\U{tagH}{##1}}\edef\tpb{\tmpb}%
   \expandafter\replacestrings\expandafter{\string&}{\entityH}%
   \let\NLh=\par \def\n{}\def\U##1{\csname##1\endcsname}%
   \tentt\localcolor\tmpb\egroup}

\def\commentH#1{{\Green<!--#1-->}}
\def\tagH#1{\tagHa#1\n^}
\def\tagHa#1\n#2^{\def\tmpb{#2}\replacestrings{"}{\stringH}%
   {\Cyan<}{\Red#1}{\Cyan\tmpb>}}
\def\stringH#1\stringH{{\def\ {\char32 }\Magenta"#1"}}
\def\entityH#1;{{\Blue\&#1;}}

\let\hisyntaxXML=\hisyntaxHTML
```

Makro `\hisyntaxHTML` nejprve převede mezery na `\n\ ` a konce řádků na `\n\NLh`. Pak převede komentáře §tt§char60`!--cosi--`§tt§char62 na `\U{commentH}{cosi}` a následně tagy

§tt§char60cosi§tt§char62 na `\U{tagH}{cosi}`. Makro `\tagH` dále separuje jméno tagu od případných parametrů a provede konverzi stringů. Stringy jsou tedy vyznačeny jen uvnitř parametrů tagu.

5.6 Zvýraznění kódu Pythonu

Petr Krajník
03. 06. 2016

Jako semestrální práce z předmětu BI-TeX bylo odevzdáno makro vycházející z předchozích dvou OPmac triků a implementující zvýraznění kódu Pythonu. Úkol to byl ještě obtížnější než v případě jazyků C a HTML. Pan Krajník se toho zhostil velice pečlivě a dobře a výsledek je dostupný na gitlab.

5.7 Zvýraznění kódu C#

Michal Franc
21. 06. 2019

Jako semestrální práce z předmětu BI-TeX bylo odevzdáno makro vycházející z předchozích dvou OPmac triků a implementující zvýraznění kódu C#. Výsledek je dostupný na gitlab.

5.8 Listingy v rámečku

P. O.
07. 10. 2015

Chceme, aby listingy produkované pomocí `\begtt...\endtt` nebo pomocí `\verbinput` byly zarámované. Rámeček se má zlomit, pokud listing přesahuje na další stránku.

Od verze OPmac Oct. 2015 je možné tento úkol vyřešit bez redefinování celého makra `\begtt` a `\doverbinput`. Je totiž možné vložit do `\tthook` přiřazení `\everypar`, které vloží čáry vlevo a vpravo každého řádku. Zbytek kódu se věnuje předefinování makra `\ttskip`, které vytvoří nejprve zahajovací díl rámečku (`\ttskipA`) a poté ukončovací díl rámečku (`\ttskipB`).

```
\def\srule{\line{\vrule height 3pt \hfil\vrule}}
\def\ttskipA{\bigskip\hrule\srule \global\let\ttskip=\ttskipB}
\def\ttskipB{\nointerlineskip\srule\hrule\bigskip \global\let\ttskip=\ttskipA}
\let\ttskip=\ttskipA
\def\tthook{\offinterlineskip \everypar={\rlap{\kern-\ttindent\line{\vrule\strut\hss\vrule}}}}
```

5.9 Zalamování řádků ve verbatim listingu

P. O.
11. 10. 2015

OPmac implicitně neřeší automatické zalamování řádků v listingu, pokud je řádek příliš dlouhý. Řádek sice ulomí, ale doplní hlášení o overfull box. V následující ukázce si ukážeme nastavení, při kterém jsou řádky zlomitelné v mezerách a při zlomení se v místě zlomu na konci řádku objeví smluvená značka (v ukázce `$\setminus$`) a pokračovací řádek respektuje odsazení výchozího řádku. Takže při:

```
\begtt
if i=0
   tady je velmi dlouhý řádek kódu, který se bohužel nevejde do šířky výstupního řádku
fi
/endtt
```

bude pokračovací řádek začínat o tři mezery posunut, tedy přesně pod slovem „tady“.

```
\def\tthook{\rightskip=0pt plus1fil
   \def\ {\discretionary{\hbox{$\;\setminus$}}{}{\kern.5em}}%
   \everypar={\countspaces}}
\def\countspaces#1\fi{#1\fi\hangindent=\ttindent \countspacesA}
\def\countspacesA{\futurelet\next\countspacesB}
\def\countspacesB{\ifx\next\spacemacro \expandafter\countspacesC\fi}
\def\countspacesC#1{\advance\hangindent by.5em\ \countspacesA}
\def\spacemacro{\ }
```

Makro `\countspaces` spuštěné v `\everypar` nejprve dokončí práci předefinovaného `\par` (až po `\fi`) a pak spočítá mezery ze začátku řádku. Za každou mezeru zvětší `\hangindent` o šířku mezery.

Toto řešení není v souladu s možností orámovat listing dle předchozího OPmac triku. Rámeček se nám v pokračovacích řádcích po stranách začne trhat. Aby se netrhal, je třeba do

definice `\tthook` vložit `\offinterlineskip\everypar={\makeframe}` a makro `\makeframe` definovat takto:

```
{\catcode`\^^M=13
\gdef\makeframe#1^^M{\endgraf\hbox{\vrule\vbox{%
   \normalbaselines\everypar={}\strut\countspaces#1\strut}\vrule}^^M}%
}
```

5.10 Tabulátory ve verbatim výpisech

0150
P. O.
02. 06. 2016

PlainTeX nastavuje znaku `^^I` (tabulátor, ASCII 9) význam jedné mezery. Pokud užíváte ve výpisech zpracovaných pomocí `\begtt...\endtt` nebo `\verbinput` tento znak, vytiskne se místo něj jedna mezera. To může být málo. Chcete-li místo něj vytisknout více mezer, definujte třeba:

```
\def\tthook{\adef\^^I{\space\space\space}} % TAB = 3 mezery
```

Tabulátor v textovém souboru se ale chová obvykle ještě inteligentněji. Vytvoří aspoň jednu mezeru a další text pokračuje za nejbližším násobkem osmé pozice. Například, pokud předchozí text obsahuje 19 znaků, tabulátor vytvoří 5 mezer, takže další text začíná za pozicí 24. Pokud ale předchozí text zabírá 23 znaků, tabulátor vytvoří jednu mezeru a další text začíná také za pozicí 24. V TeXu s užitím OPmac můžeme tuto inteligenci naprogramovat takto:

```
{\obeylines\gdef\everyttline#1^^M{\setbox0=\hbox\bgroup#1\egroup \box0 ^^M}}%
\def\tthook{\everypar={\everyttline}\adef\^^I{\TABchar}}

\def\TABchar{\egroup
   \tmpdim=\TABskip \advance\tmpdim by-.1em
   \loop \ifdim\tmpdim<\wd0 \advance\tmpdim by\TABskip\relax \repeat
   \advance\tmpdim by-\wd0 \advance\tmpdim by.1em
   \box0 \kern\tmpdim
   \setbox0=\hbox\bgroup
}
\def\TABskip{4em} % 4em = 8 pozic, protože každý znak má šířku .5em
```

Uvedené makro pomocí `\everypar` zpracovává každý řádek jako `\setbox0=\hbox{text řádku}\box0`. Pokud ale se v textu řádku vyskytne tabulátor, spustí se `\TABchar`, který čtení řádku předčasně ukončí, změří jeho šířku a podle ní vloží odpovídající `\kern` a znovu zahájí činnost `\setbox0=\hbox\bgroup`.

Pro odsazování řádků zdrojového kódu je vhodnější použít následující makro, kde v prostoru pro odsazení můžete míchat mezery a tabulátory a navíc celkové odsazení můžete nastavit dle potřeby tisku.

5.11 Řízené odsazení řádků kódu ve verbatim listingu

0151
P. O.
02. 06. 2016

Existují konvence, kolik dát mezer pro každou další úroveň odsazení řádků v listingu programu. Například první úroveň je bez odsazení, druhá posunuje řádky o čtyři znaky doprava, další o 8 znaků atd. Tj. odsazení `\verbindentIN` je rovno čtyřem. Při tisku kódu pomocí `\begtt...\endtt` nebo `\verbinput` může být takové velké odsazení méně vhodné. Uděláme makro, které místo vstupního odsazení `\verbindentIN` užitého skutečně ve vstupním kódu vytiskne jiné odsazení `\verbindentOUT`. Například nastavíme `\verbindentOUT` na dvojku a všechna odsazení budou ve výstupu poloviční.

```
\def\tthook{\everypar={\everyttline}\adef\^^I{\TABchar}}
\def\everyttline#1\fi{\fi \def\TABchar{{\ }}%
   \ifx\NLh\undefined \def\spaceGEN{\ }\def\spaceTAB{\TABchar}\else
      \let\spaceGEN=\ \let\spaceTAB=\TABchar\fi
   \tmpnum=0 \def\everyttlineC##1{\everyttlineA}\everyttlineA
}
\def\everyttlineA{\futurelet\next\everyttlineB}
\def\everyttlineB{\ifx\next\n \else
```

```
   \ifx\next\spaceGEN \advance\tmpnum by1 \else
   \ifx\next\spaceTAB \divide\tmpnum by\TABskipIN % TAB position
      \advance\tmpnum by1 \multiply\tmpnum by\TABskipIN \relax \else
   \def\everyttlineC{\everyttlineD\everyttlineE}\fi\fi\fi
   \everyttlineC
}
\def\everyttlineD{% maybe line numbers:
   \ifnum\ttline<0 \else
      \ifx\glob\undefined \global \else \glob\fi \advance\ttline by1 \printttline\fi
}
\def\everyttlineE{% recalculation of indentation:
   \tmpdim=.5em \tmpdim=\the\tmpnum\tmpdim \tmpdim=\verbindentOUT\tmpdim
   \divide\tmpdim by\verbindentIN \kern\tmpdim
}
\def\TABskipIN{8}     % vzdalenost zarazek pro TAB (ASCII 9)
\def\verbindentIN{4}  % velikost odsazeni na vstupu
\def\verbindentOUT{2} % velikost odsazeni na vystupu
```

Makro zjistí pro každý řádek velikost jeho odsazení, tuto vzdálenost vynásobí `\verbindentOUT` a vydělí `\verbindentIN` a takto upravenou vzdálenost skutečně při tisku použije. Při zjišťování velikosti odsazení si makro všímá jednak mezer a jednak tabulátorů, které jsou typicky s těmi mezerami všelijak smíchány. Pro každý tabulátor vypočte následující pozici tabulátorové zarážky dle vzdálenosti zarážek, která je dána v `\TABskipIN`. Při běžném nastavení `\TABskipIN` na 8 dává například `mezera mezera tabulátor tabulátor mezera` celkové vstupní odsazení 17 znaků.

Mezery mezi slovy na řádku nejsou nijak dotčeny. Tabulátor mezi slovy se chová jako jedna mezera. Makro spolupracuje i s makry na automatické obarvení textů dle triku 0124 a 0125.

6 Poznámky

0030 *P. O. 27. 09. 2013*

6.1 Změny kategorií v poznámce pod čarou

Pokud napíšeme například `\activettchar"` a dále `\fnote{tady je příkaz "\cosi"}`, tak to nebude fungovat, neboť `\fnote` čte svůj parametr bez změny kategorií. Problém je vyložen podrobně v TBN na str. 26. Nicméně lze snadno předefinovat `\fnote` tak, aby změny kategorií v parametru fungovaly:

```
\expandafter\def\expandafter\afnote\expandafter{\fnote{\unhbox0}}
\def\fnote{\setbox0=\hbox\bgroup\typobase\typoscale[800/800]\aftergroup\afnote\let\next}
```

Po takovém předefinování `\fnote` nemá parametr, ale načte následující text jako součást `\setbox0=\hbox`. Po ukončení `\hboxu` se provede původní `\fnote` z OPmac tentokrát s parametrem `\unhbox0`. Povšimněte si, že je pořeba nastavit přepínače ovlivňující font už při čtení obsahu `\hboxu`, protože nastavení fontů při `\unhbox0` už není účinné (sazba už byla provedena).

Podobně jako `\fnote` je možné předefinovat `\mnote` a mnohá další podobná makra:

```
\expandafter\def\expandafter\amnote\expandafter{\mnote{\unhbox0}}
\def\mnote{\setbox0=\hbox\bgroup\aftergroup\amnote\let\next}
```

0044 *P. O. 02. 04. 2014*

6.2 Rozdílné formátování značky poznámky pod čarou

Může se stát, že potřebujeme v textu vyznačit číslo poznámky pod čarou jinak, než podruhé před vlastní poznámkou pod čarou. K tomu je možné využít `\fnotehook`, který lokálně předefinuje `\thefnote`.

Použijete-li ukázkový kód uvedený níže, dostanete do textu v místě odkazů šipku dolů následovanou horním indexem s číslem poznámky a před vlastní poznámku bude umístěno číslo poznámky přímo na řádku a bez šipky.

```
\def\thefnote{$\,\downarrow^{\locfnum}$}
\def\fnotehook{\def\thefnote{\locfnum}}
```

6.3 Odkaz s poznámkou spojený hyperlinky

0044
P. O.
19. 03. 2020

Požadavek je propojit odkaz v textu na poznámku hyperlinkovým odkazem a zpět, poznámku propojit s jejím odkazem v hlavním textu. To lze udělat třeba takto:

```
\hyperlinks\Green\Green

\let\fnmarkboth=\fnmarkx

\def\fnmarkx{\link[fnx:\the\fnotenum]{\localcolor\Red} {\fnmarkboth}%
             \dest[fny:\the\fnotenum]}
\def\fnmarky{\link[fny:\the\fnotenum]{\localcolor\Blue}{\fnmarkboth}%
             \dest[fnx:\the\fnotenum]}

\def\fnote{\fnoteG\fnmarky}
\def\fnotemark#1{{\advance\fnotenum by#1\relax \fnmarky}}

\def\fnxborder{1 0 0}
\def\fnyborder{0 0 1}
```

Ukázka zavádí pro dva nové druhy odkazů barvy `\Red` a `\Blue` a definuje pro ně také červený a modrý rámeček `\fnxborder` a `\fnyborder`. Rámečky definovat nemusíte.

6.4 Nastavení tvaru odstavce poznámky pod čarou

0107
P. O.
30. 04. 2015

OPmac formátuje odstavec poznámky pod čarou stejně jako běžný odstavec s odstavcovou zarážkou, do které umístí značku poznámky. Pokud chcete mít druhý a další řádky odsazeny o `\parindent` (takže celý odstavec je zleva zarovnán a odsazen o `\parindent`, jen značka je vysunuta do vzniklého levého okraje), je možné přistoupit k následujícímu triku:

```
\addto\footstrut{\hang}
```

Tento trik rozšiřuje makro plainTeXu `\footstrut` o nastavení `\hangindent=\parindent`. Makro `\footstrut` zřejmě není (dle názvu) k tomu původně určeno, ale v plainTeXu se použije jen v poznámce pod čarou. Přitom přidat takové nastavení do \fnotehook je neúčinné, protože uvnitř makra plainTeXu `\vfootnote` se zahajuje `\insert` a v rámci tohoto zahájení se automaticky zresetuje `\hangindent` (jako při `\par`). Také `\leftskip` a `\rightskip` jsou uvnitř `\vfootnote` přenastaveny, takže jejich změna uvnitř `\fnotehook` rovněž nebude mít vliv. Pokud je ale nastavíte v rámci `\footstrut`, změna se projeví. Při rozsáhlejší změně designu poznámek pod čarou asi bude vhodné předefinovat celé makro plainTeXu `\vfootnote`.

Vzdálenost značky poznámky od prvního znaku textu je v plainTeXu nastavena na `\enspace` v rámci makra `\textindent`. Usazení této značky můžete změnit předefinováním makra `\textindent`, ovšem plainTeX toto makro používá ještě v rámci `\item` a `\itemitem`. OPmac `\texindent` ani `\item` a `\itemitem` nepoužívá (pro prostředí `\begitems...\enditems` má vlastní makra), takže když klasická plainTeXová makra v dokumentu nevyužijete, je možné usazení značky v poznámce řídit novou definicí makra `\textindent`. Například

```
\def\texindent#1{\indent\llap{#1\kern3pt}}
```

6.5 Poznámky pod čarou na více stránkách s barevnými texty

0167
P. O.
12. 02. 2019

V TeXu je konceptuální problém s barevnými texty, které mají přecházet z jedné strany na další. Pro barvy jsou užity barevné přepínače, které PDF rasterizér na každé stránce resetuje. PdfTeX proto zavádí pro práci s barvami tzv. colorstack a OPmac to využívá a funguje to v případě barevných textů, které přecházejí ze strany na stranu v hlavním bloku. Ale v případě takto rozdělených textů v pznámkách pod čarou se z toho stává hodně komplikovaný problém interně v pdfTeXu (zatím) neřešený. Když napíšete `{\localcolor\Red text ... text}` v hlavním textu, tak přechod ze strany na stranu funguje, ale v poznámkách pod čarou to fungovat nemůže, pokud jde o dlouhou poznámku rozdělenou na více stránek a obarvený úsek textu v poznámce přeteče na další stranu.

Pokud přesto chcete, aby Váš text i v dlouhých poznámkách pod čarou (přecházejících na více stran) hýřil všemi barvami, je možné v poznámkách psát `{\fnotecolor\Red text ... text}` a využít následující makro:

```
\def\fnotecolor#1{\expandafter\definecolor#1%
   \expandafter\fnoteRedA\expandafter{\iffalse}\fi}
\def\definecolor#1#2{\def\everyColor{#2 k}}
\long\def\fnoteRedA#1{\everyspacecolor#1 {} \egroup\pdfliteral{0 g}}
\long\def\everyspacecolor#1 {\ifx\end#1\end \unskip \else
   \pdfliteral{\everyColor}#1 \expandafter\everyspacecolor\fi
}
```

Toto makro vkládá před každé slovo obarveného textu přepínač na příslušnou (stále stejnou) barvu. Je to v 99 procentech zbytečné, ale když se takto obarvený text přelomí přes stranu v poznámce pod čarou, přepínač před prvním slovem nové strany zabere. Navíc je použit PDF primitivní přepínač barev, který obchází colorstack, takže implicitní řešení barev pomocí colorstacku bude dále fungovat v hlavním textu.

V tomto případě trochu spoléháme na to, že se v poznámkách pod čarou (zvláště na rozhraní mezi stránkami) nebudou dělit slova a obarvený text nebude obsahovat další složitější makrokonstrukce, které by způsobily konflikt s makrem rozkládajícím tento text na slova.

0033 P. O. 07. 10. 2013

6.6 Lokální poznámky pod tabulkami

Kromě poznámek pod čarou je často potřeba označit jednotlivé položky v tabulce a přidat k nim poznámky přímo pod tabulku. Nikoli tedy poznámky pod čarou, které se vztahují k celé stránce. Uživatel například napíše

```
\table{lll}{
  aha\tnote{projev pochopení}  & bha & ha \cr
  uha\tnote{údiv}              & eha & ha \cr
  xha\tnote{extra pochopení}   & fha & nha \cr
}
\nobreak\medskip\tnotes
```

a u položek aha, uha, xha se zjeví exponenty 1, 2, 3 a pod tabulkou v místě `\tnotes` se vytisknou tytéž exponenty následované slovy projev pochopení, údiv a extra pochopení. K tomu účelu může posloužit následující makro:

```
\newcount\tnotenum  \newbox\tnotebox
\def\tnote#1{\global\advance\tnotenum by1
   \hbox{\thetnote}%
   \global\setbox\tnotebox=\vbox{\unvbox\tnotebox
      \typobase\typoscale[800/800] \noindent\enspace\llap{\thetnote\ }#1\strut}}
\def\tnotes{\global\tnotenum=0 \box\tnotebox}
\def\thetnote{$^{\the\tnotenum}$}
```

Chcete-li místo čísel mít v exponentech písmena, pište třeba

```
\def\thetnote{$^{\rm\athe{\the\tnotenum}}$}
```

Poznámky se postupně kumulují do boxu `\tnotebox` a nakonec je tento box vytištěn. Box s poznámkami má šířku `\hsize`, takže pokud máte krátké poznámky a štíhlou tabulku a chcete obojí centrovat, je třeba například psát:

```
\centerline{\vbox{\table{...}{...}\medskip\rlap{\tnotes}}}
```

0071 P. O. 18. 06. 2014

6.7 Poznámky mnote pootočené

Pokud chcete pootočit poznámky na okraji o 45 nebo 90 stupňů, stačí psát:

```
\fixmnotes\right
\def\mnotehook#1\endgraf{\noindent\pdfsave\pdfrotate{45}\rlap{#1}\pdfrestore}
```

Rotace proběhne kolem výchozího bodu první řádky poznámky. Možná bude na základě toho potřeba upravit rozměry `\mnoteindent` a `\mnotesize`.

6.8 Číslované poznámky mnote

0122
P. O.
25. 08. 2015

Vyřešíme požadavek číslovat poznámky po straně podobným způsobem, jako poznámky pod čarou. Tj. v textu se objeví číslo a stejné číslo uvozuje postranní poznámku. Takové řešení najdeme třeba zde.

Vytvoříme makro `\note{text}`, které způsobí číslovanou poznámku po straně. Makro může být implementováno třeba takto:

```
\margins/1 a4 (2,6,2,2)cm
\mnotesize=4.5cm

\newcount\notenum
\fixmnotes\right
\def\note#1{\global\advance\notenum by 1
   $^{\the\notenum}$%
   \mnote{\typoscale[800/800]$^{\the\notenum}$\kern.3em #1}%
}
```

Zde je pomocí `\margins` vytvořeno více místa v pravém okraji, kde budou poznámky. Čítač `\notenum` se při každém `\note` zvětší o jedničku a zobrazí se v textu i v poznámce.

Protože `\note` je implementována pomocí jednoduchého OPmac makra `\mnote`, neřeší automat případné překrývání poznámek a vystrčení poznámky dole na stránce. Doporučuji při přípravě dokumentu si těchto problémů nevšímat a v závěrečné korektuře, kdy už je jasné definitivní rozmístění textu na stránkách, případně posunout vertikálně postranní poznámky pomocí `\mnoteskip` a doladit tak vzhled jednotlivých stran manuálně.

7 Odrážky

7.1 Odrážka pro vnořené \begitems...\enditems

0003
P. O.
13. 08. 2013

Prostředí `\begitems...\enditems` pracuje s implicitní odrážkou, která se použije, pokud se nepoužije `\style`. Je přitom možné mít tuto implicitní odrážku automaticky nastavenu jinak pro vnořená prostředí `\begitems...\enditems` a ani pro ně tedy autor nemusí používat `\style`. Stačí přidat na začátek dokumentu následující kód:

```
\def\slet#1#2{\expandafter\let\csname#1\expandafter\endcsname\csname#2\endcsname}
\addto\begitems{\slet{normalitem}{item:o}}
```

Implicitně je `\normalitem` nastaven na velký puntík. Po aplikaci uvedeného kódu budou mít odrážky první úrovně velký puntík a odrážky druhé úrovně malý puntík bez nutnosti používat `\style`. Chcete-li mít odlišné i odrážky třetí úrovně, pište místo předchozího druhého řádku toto:

```
\addto\begitems{\slet{normalitem}{item:o}\addto\begitems{\slet{normalitem}{item:-}}}
```

7.2 Jiné řešení vnořeného \begitems...\enditems

0031
Jan Šustek
28. 09. 2013

Abychom nemuseli ve výčtovém prostředí pokaždé psát příkaz `\style`, můžeme si nadefinovat implicitní styl. Tak navíc jednodušeji zajistíme jednotnost odrážek v celém dokumentu. Implicitní styl můžeme nadefinovat i pro vnořené výčty.

```
\addto\begitems{\style a
   \addto\begitems{\style n
      \addto\begitems{\style i }}}
```

Syntaxi `\begitems\style` můžeme používat i nadále, čímž podle očekávání přebijeme implicitní chování.

0004
P. O.
13. 08. 2013

7.3 Vertikální mezerování při použití \begitems...\enditems

Nad a pod každé prostředí `\begitems...\enditems` je přidána vertikální mezera z makra `\iiskip`, kterou OPmac nastavuje na `\medskip` (poloviční řádek). Tyto mezery se při vnořených prostředích pro výčty mohou sčítat a to nevypadá dobře. Zrušit zcela tuto mezeru lze pomocí:

```
\def\iiskip{}
```

Chcete-li mezeru zrušit jen ve vnořených výčtech, pište

```
\addto\begitems{\def\iiskip{}}
```

Chcete-li naopak mít mezeru mezi každou položkou ve výčtu a stejnou kolem prostředí výčtu, je možné to udělat takto:

```
\newdimen\iiskipamount  \iiskipamount=3pt
\def\iiskip{\vskip\iiskipamount}
\addto\begitems{\removelastskip\parskip=\iiskipamount \def\iiskip{}}
```

Toto makro ruší vertikální mezery kolem vnořených výčtů a přidává mezery mezi každým odstavcem ve výčtech pomocí `\parskip`. U výčtu na vnější úrovni potřebujeme odstranit pomocí `\removelastskip` mezeru před první položkou z `\iiskip`. Na konci výčtu na vnější úrovni ale `\iiskip` potřebujeme, za ním už totiž nebudou mezery z parskip.

Pokud nechcete mezery mezi odstavci uvnitř položek (obsahujících více odstavců) ale chcete mezery mezi položkami, je potřeba předefinovat makro `\startitem`, které OPmac používá při startu každé položky:

```
\newdimen\iiskipamount  \iiskipamount=3pt
\def\addbefore#1#2{\toks1=\expandafter{#1}\toks2={#2}\edef#1{\the\toks2 \the\toks1}}

\def\iiskip{\vskip\iiskipamount}
\addto\begitems{\removelastskip \def\iiskip{}}
\addbefore\startitem{\vskip\iiskipamount\relax}
```

0019
P. O.
27. 08. 2013

7.4 Po výčtu neodsazovat odstavec

Existuje typografické pravidlo, které praví, že první odstavec po seznamu opatřeném odrážkami by neměl mít odsazení, protože odsazeny jsou údaje v seznamu a může to působit jako matoucí. Při použití `\begitems...\enditems` v OPmac stačí do definice na začátek dokumentu napsat:

```
\addto\enditems{\afternoindent}
```

a je uvedenému typografickému pravidlu vyhověno.

0134
Jan Šustek
19. 12. 2015

7.5 Průběžné číslování items

Někdy chceme, aby číslování ve výčtu začalo tam, kde skončilo číslování v předcházejícím výčtu. Toho docílíme vytvořením globálního čítače `\gitemnum`, do nějž vložíme poslední hodnotu `\itemnum` z předchozího výčtu, kterou potom načteme na začátku následujícího výčtu.

```
\newcount\gitemnum
\addto\begitems{\itemnum\gitemnum}
\addbefore\enditems{\global\gitemnum\itemnum}
```

V definici jsme využili makro `\addbefore` z OPmac triku 0004. Trik funguje pouze pro jednoúrovňové výčty.

8 Draft

8.1 Tisk lejblíků při \draft

0005
P. O.
13. 08. 2013

V případě, že je aktivní `\draft` (tj. dokument je určen ke korekturám), se mi osvědčilo tisknout do dokumentu lejblíky v místě cílů odkazů `\ref` a `\cite`. Autor nemusí vzpomínat, jaký použil lejblík, vidí to totiž přímo před očima.

```
\addto\draft{\let\destbox=\draftdestbox}
\def\draftdestbox[#1#2:#3]{\vbox to0pt{\kern-\destheight
   \ifx\pdfdest\undefined\special{pdf:dest (#1#2:#3) [@thispage /XYZ @xpos @ypos null]}%
   \else\pdfdest name{#1#2:#3} xyz\relax\fi
   \if#1r\llap{\localcolor\Red\tt\thefontsize[10][\detokenize\expandafter{#3}]}\vss
   \else \if#1c\vss\llap{\localcolor\Red\tt[\detokenize\expandafter{\tmpb}] }\kern-\prevdepth
   \else \vss \fi\fi}}
```

Toto makro předefinovává `\destbox` z OPmac. Hlavním účelem `\destbox` je umístit do výšky `\destheight` cíl odkazu. Navíc v makru přidáváme při odkazech typu `ref` (`#1=r`) `\llap` následovaný `\vss` (tj. `\llap` je taky ve výšce odkazu) a při odkazech typu `cite` (`#1=c`) je přidáno `\vss \llap`, tedy `\llap` je na účaří a vyskytne se v seznamu literatury. Lejblík je tištěn zelenou barvou a v sazbě nepřekáží, tj. sazba bez něj dopadne zcela stejně.

Příkaz `\detokenize` (z eTeXu) je v kódu použit proto, aby bylo možné tisknout lejblíky, ve kterých se vyskytuje podtržítko. Bez použití eTeXu by bylo potřeba tisk provést pomocí kombinace `\string` a `\csname...\endcsname`. Další možnost je vyhnout se použití lejblíků s podtržítkem.

Kromě toho je praktické při `\draft` přidat třeba do patičky datum zpracování:

```
\addto\draft{\def\drafttext{\llap{\the\day. \the\month. \the\year\Black}}}
\def\drafttext{}
\footline={\hss\rm\thefontsize[10]\the\pageno\hss\drafttext}
```

8.2 Interní poznámky

0056
P. O.
11. 05. 2014

Je užitečné si do textu psát interní poznámky typu „k tomu se ještě musím vrátit“, „tady je třeba doplnit obrázek“, které se v dokumentu objeví jen při nastaveném `\draft`. Nabízím řešení pomocí maker `\rfc` a `\makerfc`. První z nich (request for correction) vloží do textu neviditelnou poznámku a druhé z nich sepíše seznam všech poznámek na novou stránku (na konci dokumentu) a u každé poznámky je zpětný odkaz do místa, kde byla poznámka napsaná. To vše ale funguje jen při zapnutém `\draft`. Jakmile je `\draft` vypnuté, žádný seznam poznámek se netiskne. Použití vypadá takto:

```
Tady je normální text\rfc{ověřit, zda netvrdím blbosti}.
Tady pokračuje další text\rfc{zjistit, zda naměřené hodnoty nejsou moc ulítlé}.
...
% na konci dokumentu:
\makerfc
```

Při `\draft` se objeví nyní závěrečná stránka, kde je řečeno:

[rfc-1] ověřit, zda netvrdím blbosti [rfc-2] zjistit, zda naměřené hodnoty nejsou moc ulítlé

a přitom při `\hyperlinks` jsou odkazy [rfc-1] a [rfc-2] klikací a zpětně navedou do místa textu, kterého se poznámka týká.

Implementace může vypadat takto:

```
\newcount\rfcnum

\def\rfclist{}
\def\rfcactive#1{\global\advance\rfcnum by1
   \dest[rfc:rfc-\the\rfcnum]\global\addto\rfclist{\rfcitem #1}}
\def\rfc#1{}
\def\rfcitem{\advance\rfcnum by1
   \medskip\noindent\llap{\link[rfc:rfc-\the\rfcnum]{\localcolor\Red}{[rfc-\the\rfcnum] }}}
```

```
\addto\draft{\let\rfc=\rfcactive}

\def\makerfc{\ifx\rfc\rfcactive
   \vfil\break {\secfont \noindent Requests for correction}\par
   \bgroup\rfcnum=0 \rfclist\egroup\fi}
```

Makro definuje `\rfc` jako prázdné a do `\draft` přidá `\let\rfc=\rfcactive`, tj. při `\draft` se `\rfc` probudí k životu: přidá do `\rfclist` text poznámky uvozený makrem `\rfcitem`, které se stará o zpětné odkazy. Makro `\makerfc` prostě spustí `\rfclist`. Navíc při aktivaci lejblíků, jak je posáno v předchozím triku, se v textu, kterého se poznámka týká, objeví v hranaté závorce [rfc-číslo].

9 Číslování, odkazy

9.1 Přehledná deklarace číslování

0007
P. O.
15. 08. 2013

OPmac používá `\tnum` pro čísla tabulek, `\fnum` pro čísla obrázků a `\dnum` pro čísla rovnic. Dále je `\chapnum` číslo kapitoly, `\secnum` číslo sekce a `\seccnum` číslo podsekce. Čísla tabulek, obrázků i rovnic OPmac resetuje v každé sekci a vypisuje pro tabulku trojici čísel `chap.sec.tab`, pro obrázek `chap.sec.fig` a rovnice vypisuje ve formátu (rce). Pokud bychom toto chování chtěli změnit a mít nad tím přehled, je možné nejprve vykostit originální resetování registrů v kódu maker `\chap`, `\sec`, `\secc` a převzít řízení do svých rukou. Výmaz uvedených kódů obstarají makra `\chaphook`, `\sechook` a `\secchook`, která ve svém parametru `#1` sejmou resetovací kód až po `\relax` (to ukončuje resetovací kód).

```
\def\chaphook#1\relax{\dochaphook}  \def\dochaphook{}
\def\sechook#1\relax{\dosechook}    \def\dosechook{}
\def\secchook#1\relax{\dosecchook}  \def\dosecchook{}

\def\chapreset#1{\addto\dochaphook{\global#1=0 }}
\def\secreset#1{\addto\dosechook{\global#1=0 }\chapreset#1}
\def\seccreset#1{\addto\dosecchook{\global#1=0 }\secreset#1}

\chapreset\secnum   \secreset\seccnum
```

Makrem `\chapreset\counter` dáváme najevo, že chceme `\counter` resetovat jen v kapitole, dále `\secreset\num` zresetuje `\num` v kapitole i sekci a konečně `\seccreset\cosi` zresetuje `\cosi` v kapitole, sekci i podesekci. Dejme tomu, že chceme obrázky, tabulky i rovnice mít resetovány jen v kapitole a průběžně číslovány v sekcích a podsekcích. Chceme je označovat pomocí dvojice `chap.tab` nebo `chap.fig` a v případě rovnice zvolíme formát (`chap.rce`). Pak stačí psát:

```
\chapreset\tnum   \def\thetnum{\the\chapnum.\the\tnum}
\chapreset\fnum   \def\thefnum{\the\chapnum.\the\fnum}
\chapreset\dnum   \def\thednum{(\the\chapnum.\the\dnum)}
```

Nechť dále třeba chceme číslovat věty (propositions) a budeme třeba úplně praštění při návrhu číslování: budou se resetovat v každé podsekci a tisknout ve tvaru čtveřice `chap.sec.secc.prop`. Pak můžeme psát:

```
\newcount\propnum  \seccreset\propnum
\def\thepropnum{\the\chapnum.\the\secnum.\the\seccnum.\the\propnum}

\def\proposition{\global\advance\propnum by 1
   \noindent\wlabel{\thepropnum}{\bf Proposition \thepropnum.}\space}
```

9.2 Seznam obrázků a tabulek

0013
P. O.
17. 08. 2013

OPmac nabízí jen automaticky generovaný obsah kapitol, sekcí a podsekcí. Chceme-li vytvořit automaticky generovaný obsah obrázků a tabulek, je třeba to naprogramovat například takto:

```
\def\totlist{} \def\toflist{}
\def\Xtab#1#2#3{\addto\totlist{\totline{#1}{#2}{#3}}}
\def\Xfig#1#2#3{\addto\toflist{\tofline{#1}{#2}{#3}}}

\input opmac

\def\tofline#1#2#3{{\leftskip=\iindent \rightskip=\iindent plus1em
   \noindent\llap{\bf\ref[#1].\enspace}%
   {#3\unskip}\nobreak\tocdotfill\pgref[#1]\nobreak\hskip-\iindent\null\par}}
\let\totline=\tofline

\def\captionhook#1{\typosize[10/12]%
   \ifx\clabeltext\undefined \else
      \ifx#1t{\protectlist\immediate\wref\Xtab{{\lastlabel}{\thetnum}{\clabeltext}}}%
      \else  {\protectlist\immediate\wref\Xfig{{\lastlabel}{\thefnum}{\clabeltext}}}%
   \fi\fi
   \global\let\clabeltext=\undefined
}
\def\clabel[#1]#2{\gdef\clabeltext{#2}\label[#1]}
```

Je zde využit soubor REF z OPmac a do něj jsou zapisovány údaje tvaru `\Xtab{lejblík}{číslo}{text}` a `\Xfig{leblík}{číslo}{text}`. Tyto údaje jsou zapisovány okamžitě, protože neobsahují číslo strany. Číslo strany v obsahu obrázků a tabulek je nakonec vytištěno pomocí `\pgref[lejblík]`. Soubor REF je čten během `\input opmac`, proto je potřeba makra `\Xtab`, `\Xref` definovat před načtením opmac. Tato makra si data pouze zapamatují do `\totlist` a `\toflist`, kde se kumulují údaje ve tvaru `\totline{lejblík}{číslo}{text}` a podobně pro `\tofline`. Definice maker `\tofline`, `\totline` je inspirovaná makrem `\tocline` z OPmac.

Uživatel musí napsat `\clabel[lejblík]{text}` následovaný tabulkou nebo obrázkem včetně příkazu `\caption/t` nebo `\caption/f`. Makro `\clabel` založí lejblík a makro `\caption` prostřednictvím `\captionhook` pošle potřebné údaje do REF souboru. Použití může vypadat například takto:

```
\nonum\notoc\sec Seznam obrázků

\toflist

\nonum\notoc\sec Seznam tabulek

\totlist

\clabel[tabA]{Tabule A}
... Tabulka A\par\nobreak
\caption/t Toto je velmi zajímavá tabulka A.

\clabel[tabB]{Tabule B}
... Tabulka B\par\nobreak
\caption/t Nyní se můžete pokochat tabulkou B.
```

9.3 Popisky pro výpisy kódů

0035
P. O.
29. 11. 2013

OPmac nabízí dva druhy nezávisle číslovaných popisků: obrázky pomocí `\caption/f` a tabulky pomocí `\caption/t`. Ukážeme si, jak doplnit třetí druh. Například popisky pod výpisy kódů pomocí `\caption/l`.

```
\newcount\lnum
```

```
\def\thelnum{\thechapnum.\the\lnum}
\sdef{mt:l:en}{Listing} \sdef{mt:l:cs}{Výpis} \sdef{mt:l:sk}{Výpis}
\def\chaphook#1\relax{{\globaldefs=1 \chapnums}}
\def\chapnums{\secnum=0 \seccnum=0 \tnum=0 \fnum=0 \dnum=0 \lnum=0 }

\chyph  % Zkouška v češtině:

\chap Pokus

\begtt
int main() { ... }
/endtt
\nobreak \label[main]
\caption/l Toto je výpis funkce main.
```

Uvedená ukázka zavádí čítač `\lnum` a definuje vzhled čísla `\thelnum`. Dále jsou uvedeny tři jazykové varianty slova Výpis a přidáno resetování čísla `\numl` v každé kapitole.

0048 9.4 Změna formátu popisků pod obrázky a tabulkami

P. O. 26. 04. 2014

OPmac sází popisek jako odstavec s centrovaným posledním řádkem. Využívá k tomu trik popsaný v TBN na straně 234. Může se ovšem vyskytnout jiný požadavek na formátování popisků. Třeba takový: je-li popisek jednořádkový, pak má centrovat. Přeteče-li na více řádků, má se chovat jako normální odstavec. Řešení tohoto požadavku přinese následující kód:

```
\def\printcaption#1#2{\leftskip=\iindent \rightskip=\iindent
   \setbox0=\hbox\bgroup \aftergroup\docaption{\bf#1 #2.}\enspace}
\def\docaption{\tmpdim=\hsize \advance\tmpdim by-2\iindent
   \ifdim\wd0>\tmpdim \unhbox0 \else \hfil\hfil\unhbox0 \fi \endgraf \egroup}
```

Makro přeměří text popisku uložený v boxu0 a pokud se vejde na jeden řádek, předřadí před něj `\hfil\hfil`, což je pružinka stejné síly, jako `\parfillskip` (v popiscích dle OPmac) a text centruje. Je-li popisek delší, žádná pružinka se nepředřadí a vysází se obyčejný odstavec.

Nenapadlo mě řešení této úlohy postavené na záporných fill, jako to je v případě centrování posledního řádku. Moje inteligence na to nestačí. Ani se mi nedaří najít důkaz, že to nemá řešení. Má-li někdo jedno nebo druhé, prosím o informaci.

0050 9.5 Zlom v dlouhém URL

P. O. 26. 04. 2014

OPmac nabízí makro `\url` na tisk internetových odkazů. Toto makro dává potenciální místa zlomu za lomítka, tečky, otazníky a rovnítka. Dále od verze May 2014 nabízí OPmac makro `\|`, které se dá použít v argumentu `\url` pro vyznačení dalších míst zlomu. Toto makro se při tisku \url chová jako `\urlspecchar`, přitom `\urlspecchar` je definováno implitině jako `\penalty10`. Protože je ale v argumentu `\url` schována jen malá pružnost, nemusí vhodné místo zlomu přesně vyhovovat. Je možné předefinovat `\urlspecchar` tak, aby v případě, že se v tomto místě zlomí, zůstal řádek nezarovnán na pravý okraj.

```
\def\urlspecchar{\nobreak\hskip0pt plus2em \penalty110 \hskip0pt plus-2em\relax}
```

Makro umožní zlomit v penaltě 110, pak před ní zůstává pružná mezera. Pokud ale ke zlomu nedojde, pak se dvě za sebou jdoucí pružné mezery vyruší, takže v sazbě není žádná mezera.

Další možností je vložit mezi každé dva znaky v argumentu `\url` pružnou mezeru. To se dá udělat takto:

```
\addto\urlfont{\replacestrings{}{\nobreak\hskip0pt plus.05em minus.03em\relax}}
```

Odeberete-li `\nobreak`, bude argument `\url` zlomitelný mezi každými dvěma znaky.

10 Kapitoly, sekce

10.1 Podpodsekce

0055
P. O.
11. 05. 2014

Opmac nabízí jen třístupňovou hiearchii nadpisů: kapitoly, sekce a podsekce. Často se sektávám s požadavkem zařadit i možnost tisku podpodsekcí. Osobně to nepovažuji za příliš účelné a zbytečně to mate čtenáře. Ale je-li k tomu pádný důvod, můžete si vytvořit makro `\seccc` takto:

```
\newcount\secccnum

\eoldef\seccc#1{%
   \ifx\prevseccnum\theseccnum \global\advance\secccnum by1
   \else \global\let\prevseccnum=\theseccnum \global\secccnum=1
   \fi
   \edef\thesecccnum{\theseccnum.\the\secccnum}%
   \printseccc{#1}%
}
\def\printseccc#1{\norempenalty-100 \medskip
   {\bf \noindent \thesecccnum\quad #1\nbpar}%
   \nobreak \smallskip \firstnoindent
}
```

Makro řeší uvedený požadavek co nejjednodušším způsobem: zavádí číslování v registru `\secccnum` a tiskne ve fontu `\bf` (nezvětšená velikost). Neřeší dopravení informací do obsahu (tam je to stejně nevhodné) ani do záhlaví. Byl-li by takový požadavek, je potřeba využít makro `\wcontents` pro obsah a primitiv `\mark` pro záhlaví.

10.2 Zlom nadpisu v textu a v obsahu

0057
P. O.
12. 05. 2014

V textu se doporučuje lámat dlouhé nadpisy kapitol, sekcí atd. pomocí příkazu `\nl`. Ten se v textu projeví jako zlom řádku a v obsahu se projeví jako obyčejná mezera. Někdy se ovšem může stát, že potřebujeme specifikovat i zlom nadpisu v obsahu. Pak je možné použít:

```
\let\NL=\nl  \addprotect\NL
```

Nyní je k dispozici příkaz `\NL`, který způsobí zlom nadpisu v textu i obsahu zároveň. Ovšem je možné, že potřebujeme zlom jen v obsahu a ne v textu. Pak lze připravit makro `\toclonly{text}`, které zpracuje text jen v obsahu, zatímco v běžném textu se neprojeví. Píšeme třeba:

```
\sec Příliš\nl dlouhý\toconly\NL\ nadpis
```

a v obsahu máme „Příliš dlouhý/nadpis“ zatímco v textu je „Příliš/dlouhý nadpis“. K tomu můžeme použít tento kód:

```
\let\NL=\nl  \addprotect\NL
\def\toconly#1{} \addprotect\toconly
\let\maketocori=\maketoc \def\maketoc{{\def\toconly##1{##1}\maketocori}}
```

Makro `\toconly` implicitně nedělá nic, ale v obsahu je lokálně předefinováno.

Pokud nechcete, aby `\toconly` rozbořilo PDF záložky, použijte `\input pdfuni` nebo (xor) definujte `\def\cnvhook{\def\toconly##1{}}`.

10.3 Přidání další úrovně nadpisů včetně zařazení do obsahu

0099
P. O.
15. 04. 2015

Vytvoříme makro `\part` §tt§char60`text`§tt§char62, které bude vytvářet číslované části díla, jež jsou nadřazeny nad kapitoly. Při `\maketoc` se vytisknou do obsahu i informace o dělení knihy na části.

```
\newcount\partnum
\def\part#1\par{%
  \chaphook {\globaldefs=1 \chapnum=0 \secnum=0 \seccnum=0 \tnum=0 \fnum=0 \dnum=0}\relax
  \edef\thepartnum{\the\partnum}\let\thetocnum=\thepartnum \xdef\tocilabel{\thepartnum}%
  \vfil\break\null
  \vskip3cm
  \centerline{\typosize[15/18]\bf Part \uproman\partnum}
  \wtotoc{-1}\bfshape{#1}\dest[toc:part.\thepartnum]
  \tit #1\par
  \vfil\break
}
\def\printpart#1{\vfil\break\null
   \vskip3cm
   \centerline{\typosize[15/18]\bf Part \uproman\partnum}
   \tit #1\par
   \vfil\break
}
\def\uproman#1{\uppercase\expandafter{\romannumeral#1}}
\let\optocline=\tocline % original OPmac \tocline is saved.
\def\tocline#1{\ifnum#1=-1 \let\next=\ptocline \else \def\next{\optocline{#1}}\fi \next}
\def\ptocline#1#2#3#4{\medskip
   \def\tocilabel{#2}%
   \centerline{#1Part \uproman{#2}}\nobreak
   \centerline{#1#3\unskip}\nobreak
}
\addto\maketoc{\def\tocilabel{}}
```

Makro vkládá informace do obsahu pomocí `\wtotoc` (k dispozici od verze OPmac Apr. 2015). Následuje číslo úrovně nadpisu, kapitola má úroveň 0, sekce 1 a podsekce 2. Při použití `\part` tedy vkládáme údaj -1. Dále je zde upraveno makro `\tocline`, které při úrovni 0 a více spustí originální `\tocline`, zatímco při úrovni -1 se spustí `\ptocline`, které vloží dva řádky do obsahu. Makro `\part` rovněž nastaví interní lejblík `\tocilabel`, aby byly odlišeny části obsahu pro jednotlivé části při použití `\hyperlinks`. Samotná čísla kapitol totiž v jednotlivých částech nezaručují unikátnost hyperlinkového lejblíku.

Pokud chceme, aby `\part` fungoval i při `\outlines`, je třeba ještě vyvinout další úsilí:

```
\let\outlinesAA=\outlinesA
\def\outlinesA#1{\ifnum#1=-1 \def\next{}\else \def\next{\outlinesAA{#1}}\fi \next}
\let\outlinesBB=\outlinesB
\def\outlinesB#1{\ifnum#1=-1 \def\next{\outlinesBp}\else \def\next{\outlinesBB{#1}}\fi \next}
\def\outlinesBp#1#2#3#4{\def\tocilabel{#2}%
   \pdfoutline goto name{toc:part.#2} count 0 {PART: #3}\relax
}
```

10.4 Úvodní stránka kapitoly neobsahuje \topinsert

0135 P. O. 21. 12. 2015

Při užití `\topinsert ...\endinsert` bychom chtěli vyloučit umístění obrázku či tabulky na první stránku kapitoly, tedy nad nadpis kapitoly. To lze provést poměrně snadno vložením následujících řádků na začátek dokumentu:

```
\addto\chaphook{\global\dimen\topins=0pt }
\addto\endoutput{\global\dimen\topins=\vsize}
```

10.5 Nastavení \nonum pro všechny kapitoly a sekce

0142 P. O. 10. 04. 2016

Nechcete psát `\nonum` před každou `\chap`, `\sec` a `\secc` a chcete svou vůli deklarovat jedinkrát na začátku dokumentu. V takovém případě lze psát:

```
\nonum \def\nonumfalse{\pagedepth=\pagedepth}
```

Trik spočívá v tom, že je zneplatněno makro `\nonumfalse`, které přestane vracet logickou hodnotu `\ifnonum` na false. Makru `\nonumfalse` předchází v místě jeho užití v makrech

OPmac prefix `\global`, takže potřebujeme tento prefix uspokojit nějakým přiřazením. Zvolil jsem přiřazení registru `\pagedepth`, který je globální sám o sobě, takže výše uvedené přiřazení opravdu neudělá vůbec nic.

11 Pracovní soubor

11.1 Kontrola konzistence REF souboru

0045
P. O.
05. 04. 2014

Výjimečně se může stát, že změny v textech odkazů mohou změnit stránkování a tato změna stránkování změní po dalším průchodu TeXem texty odkazů. Přitom i po opakovaném TeXování se sazba neustálí na konečné podobě. Nelze snadno a univerzálně tento problém vyřešit, ale je možné na něj aspoň uživatele upozornit. Uživatel si pak může uložit dva po sobě jdoucí obsahy REF souborů a porovnat je třeba pomocí diff. Na základě toho zjistí, kde je problém a pokusí se to řešit manuálně.

OPmac neimplementuje upozornění na nekonzistenci REF souborů po dvou za sebou jdoucích průchodech. Je to ovšem možné doprogramovat tímto kódem:

```
\let\Xend=\relax
\long\def\readREFcontents #1\Xend{\def\REFcontents{#1}}
\openin0=\jobname.ref
\ifeof0 \def\REFcontents{}
\else \expandafter \readREFcontents \input \jobname.ref
\fi

\input opmac

\let\endprimitive=\end
\def\end{%
   \ifx\wref\wrefrelax \else
      \vfil\break
      \immediate\write\reffile{\string\Xend}
      \immediate\closeout\reffile
      \let\tmpa=\REFcontents
      \expandafter \readREFcontents \input \jobname.ref
      \ifx\tmpa\REFcontents \message{Congratulations, references are consistent}
      \else                 \opwarning{Inconsistent REF file, TeX me again}
   \fi\fi
   \endprimitive
}
```

V tomto makru na začátku uložíme do `\REFcontents` kompletní obsah REF souboru z předchozího běhu. To je potřeba učinit ještě před `\input opmac`, protože OPmac nám REF soubor přemaže. Dále na konci zpracování dokumentu (předefinováním `\end`) proběhne kontrola, zda obsah REF souboru zůstal stejný, jako na začátku. REF soubor je v obou případech čten makrem `\readREFcontents`, které očekává na konci souboru zarážku `\Xend`. Proto je tato zarážka do souboru na konci zpracování vložena.

11.2 Data QR kódu v REF souboru

0116
P. O.
02. 07. 2015

Pro tvorbu QR kódů je možné použít balíček qrcode.tex pro plainTeX, který je dostupný zde. Tento balíček sice ukládá výsledek výpočtu QR kódu do makra `\qrdata` k rychlému opakovanému použití, ale tyto údaje nejsou uloženy do pracovního souboru. Použít při dalším běhu TeXu už jednou předpočítaná data výrazně urychlí zpracování dokumentu, protože výpočet QR kódu zabere jistý výpočetní čas. Ukážeme, jak k tomu účelu využít REF soubor.

Makro `\qrcode` z `qrcode.tex` obsahuje (implicitně prázdná) „hook“ makra `\qrbeginhook` a `\qrendhook`, která definujeme tak, aby se do REF souboru ukládala informace o QR kódu ve tvaru:

```
\Xqrdata{kódovaný-text}{velikost}{11111110111110...0010111001}
```

kde `velikost` a jedničky a nuly jsou údaje z `\qrdata` vypočteného QR kódu. Dále je třeba před `\input opmac` definovat `\Xqrdata`, které definuje řídicí sekvenci `\qr:kódovaný-text` tak, že obsahuje vypočtená data. Konečně je potřeba modifikovat chování makra `\qrcode` tak, aby v případě, že existuje již definovaná řídicí sekvence `\qr:kódovaný-text`, tak ji využije a nepočítá QR kód znovu. Makra mohou vypadat takto:

```
\def\Xqrdata{\bgroup\qrverbatim\XqrdataA}
\def\XqrdataA#1#2#3{\sxdef{qr:#1}{{#2}{#3}}\egroup}
\input qrcode
\input opmac

\openref
\def\qrendhook{%
   \qrmessage{<Saving calculated QR code...}%
      \sxdef{qr:\qretext\expandafter}{\qrdata}%
      \toks0=\expandafter{\qrtext}%
      \immediate\wref\Xqrdata{{\the\toks0}\qrdata}%
   \qrmessage{done>^^J}%
}
\def\qrbeginhook#1\qrendhook{%
   \expandafter \ifx \csname qr:\qretext\endcsname \relax
      #1%                  %%% do calculation
   \else
      \qrmessage{<Restoring QR code from saved \string\qrdata...}%
         \global\expandafter\let \expandafter\qrdata \csname qr:\qretext\endcsname
         \qrrestore\qrdata %%% printing QR code from saved data
      \qrmessage{done>^^J}%
   \fi
   \qrendhook
}
```

Poznamenejme, že na pořadí záleží. Nejprve je třeba definovat `\Xqrdata`, pak provést `\input qrcode` dřív než `\input opmac` a nakonec definovat “hooks”. Poté může následovat dokument s QR kódy.

12 Vyznačování

12.1 Podtržení, přeškrtnutí, prostrkání

0063 *P. O.* 06. 06. 2014

Implementujeme makro `\ul{tady je text}`, které vytvoří podtržený nebo přeškrtnutý text. Přidáme také makro `\ltsp{tady je text}`, které vytvoří prostrkaný text. Vlastnosti rozsáhlého balíčku `soul.sty` vměstnáme do deseti řádků kódu za cenu toho, že naše makro bude umět rozdělit text jen v mezislovních mezerách a pokud bychom chtěli rozdělit slovo, je potřeba makru napovědět `\ul{ta\-ko\-vým způ\-so\-bem}`. Následující trik 0065 ukazuje, jak vyhledat tato místa pro dělení slov automaticky.

```
\def\ul#1{{\ulRedefine\leavevmode\wordscanA #1 {} }}
\def\wordscanA#1 {\ifx^#1^\unskip\else \wordscanB#1\-\end \expandafter\wordscanA\fi}
\def\wordscanB#1\-#2\end{\ifx^#2^\wordprintA{#1}\else
```

```
   \wordprintB{#1}\def\next{\wordscanB#2\end}\expandafter\next\fi}
\def\wordprintA#1{\setbox0=\hbox{#1}\hbox{\rlap{\copy0}\uline\wd0}\uline\uspace\relax}
\def\wordprintB#1{\setbox0=\hbox{#1}\hbox{\rlap{\copy0}\uline\wd0}\-}
\def\uline{\leaders \vrule height-1.9pt depth2.3pt\hskip}
\def\uspace{\fontdimen2\font plus\fontdimen3\font minus\fontdimen4\font}
\def\ulRedefine{\def~{\egroup\hbox{\rlap{\copy0}\uline\wd0}\nobreak\uline\uspace\relax
        \setbox0=\hbox\bgroup}}
```

Makro spouští opakovaně `\wordscanA` na jednotlivá slova a pomocí `\wordscanB` tato slova rozdělí na části vyznačené `ta\-ko\-vým způ\-so\-bem`. Celé slovo nebo jeho koncovou část pak tiskne pomocí `\wordprintA` zatímco část ukončenou znakem `\-` tiskne pomocí `\printwordB`. Definováním makra `\uline` nastavíme výšku a tloušťku podtrhávací nebo přeškrtávací čáry. V příkladu je čára podtrhávací začínající 1,9pt pod účařím a mající tloušťku 0.4pt. V makru `\uspace` definujeme velikost mezislovní mezery včetně pružnosti. V příkladu přebíráme parametry běžné mezislovní mezery z příslušných `\fontdimen`.

Přítomnost vlnky jako nezlomitelné mezery v textu (např. `\ul{v~textu}`) je ošetřena poněkud trikoidním kódem v `\ulRedefine`. Jinak totiž má tato nezlomitelná mezera Wordoidní vlastnost, tj. nepruží. To by bylo samozřejmě špatně.

Makro `\ltps{prostrkaný text}` pracuje se stejnými makry `\wordscanA` a `\wordscanB`, ale jinak definuje `\printscanA` a `\printscanB`. Tato makra rozeberou slova nebo jejich části na jednotlivá písmena. Makro pracuje s dimen registry `\ltspA`, `\ltspB` a `\ltspC`. jejich význam je uveden níže v komentářích.

```
\def\ltsp#1{\ifhmode\hskip\ltspA \else\leavevmode\fi \wordscanA #1 {} \hskip\ltspA}
\def\wordscanA#1 {\ifx^#1^\unskip\else \wordscanB#1\-\end \expandafter\wordscanA\fi}
\def\wordscanB#1\-#2\end{\ifx^#2^\wordprintA{#1}\else
   \wordprintB{#1}\def\next{\wordscanB#2\end}\expandafter\next\fi}
\def\wordprintA#1{\letterspaceA #1{}\unskip\unpenalty\hskip\ltspB}
\def\wordprintB#1{\letterspaceA #1{}\-}
\def\letterspaceA#1{\if^#1^\else\hbox{#1}\nobreak\hskip\ltspC \expandafter\letterspaceA\fi}

\newskip\ltspA \ltspA=6pt   % mezera vložená na začátek prostrkaného textu
\newskip\ltspB \ltspB=5pt plus2pt minus1pt  % mezislovní mezera
\newskip\ltspC \ltspC=2pt   % mezera mezi písmeny.
```

12.2 Vyhledání míst pro dělení slov ve slově

0065
P. O.
07. 06. 2014

Vytvoříme makro `\hyphenprocess{slovo}`, které vrátí parametr v makru `\listwparts` ve tvaru `slo\-vo` podle aktuálního nastavení dělení slov.

```
\newdimen\hyphdim
\let\hyphentt=\tentt  \hyphdim=\fontdimen6\hyphentt \divide\hyphdim by2

\def\hyphenprocess#1{\def\tmp{#1}\let\listwparts=\undefined
   \setbox0=\vbox\bgroup\hyphenpenalty=-10000 \hsize=0pt \hfuzz=\maxdimen
      \edef\hychar{\hyphenchar\hyphentt=\the\hyphenchar\hyphentt}\hyphenchar\hyphentt=45
      \def~{\nobreak\hskip.5em\relax}\tt\noindent\hskip0pt\relax #1\par \hychar
      \hyphenprocessA
   \expandafter\let\expandafter\listwparts\expandafter\empty
      \expandafter\hyphenprocessB\listwparts
}
\def\hyphenprocessA{\setbox2=\lastbox
   \ifvoid2 \egroup \else \unskip \unpenalty
      \setbox2=\hbox{\unhbox2}%
      \tmpnum=\wd2 \advance\tmpnum by100 \divide\tmpnum by\hyphdim
      \ifx\listwparts\undefined \xdef\listwparts{,}%
      \else \advance\tmpnum by-1 \xdef\listwparts{\the\tmpnum,\listwparts}\fi
              \expandafter\hyphenprocessA \fi
}
\def\hyphenprocessB#1,{\if^#1^\expandafter\hyphenprocessC
   \else \tmpnum=#1 \expandafter\hyphenprocessD\tmp\end
```

```
   \fi
}
\def\hyphenprocessC{\expandafter\addto\expandafter\listwparts\expandafter{\tmp}}
\def\hyphenprocessD#1#2\end{\addto\listwparts{#1}%
   \advance\tmpnum by-1
   \ifnum\tmpnum>0 \def\next{\hyphenprocessD#2\end}%
   \else
   \def\tmp{#2}\def\next{\addto\listwparts{\-}\expandafter\hyphenprocessB}\fi
   \next
}
```

Makro si rozlomí slovo v pracovním `\vboxu` ve fontu `\hyphentta` pak v makru `\hyphenprocessA` posbírá boxy, proměří je a na základě toho rozpozná, z kolika písmen se skládají. Do `\listsparts` uloží počty písmen jednotlivých úseků, takže například pro slovo „rozdělení“ uloží do `\listwparts` čísla `3,2,,`. Povšimněte si, že číslo pro poslední úsek záměrně chybí, protože je nebudeme potřebovat. Makra `\hyphenprocessB`, `C` a `D` s těmito údaji pracují a sbírají z `\tmp` (kde je slovo uloženo) patřičný počet písmenek a přemisťují je do `\listwparts`. Mezi jednotlivé úseky vkládají znak `\-`.

Makro `\ul` nebo `\ltsp` z předchozího příkladu 0063 můžeme nyní snadno vylepšit, aby automaticky našlo dělení slov. Stačí předefinovat interní makro `\wordscanA` následovně:

```
\def\wordscanA#1 {\ifx^#1^\unskip\else
   \hyphenprocess{#1}\expandafter\wordscanB\listwparts\-\end \expandafter\wordscanA\fi}
```

0085

12.3 Podbarvený text

P. O.
05. 01. 2015

Přímo v dokumentaci k OPmac je ukázka, jak vytvořit makro, které vysází text podbarvený specifikovanou barvou. Ovšem toto makro vytvoří text jako nezlomitelný box. V následujícím triku ukážeme makro `\coltext\BarvaA\BarvaB{text}`, které vytvoří uvnitř odstavce text v bavě `\BarvaA` podbarvený barvou `\BarvaB`, přitom tento text podléhá řádkovému zlomu jako běžný text v odstavci. Například:

```
Zde je běžný text odstavce.
\coltext\Yellow\Blue{Tady je podbarvený textík, který se láme do řádků.}
A pokračujeme v běžném textu odstavce.
```

vytvoří:

Zde je běžný text odstavce. Tady je podbarvený textík, který se láme do řádků. A pokračujeme v běžném textu odstavce.

Makro je mírnou modifikací makra `\ul` z OPmac triku 0063, přitom využívá makro `\hyphenprocess` z OPmac triku 0064.

```
\def\coltextstrut{height2ex depth.6ex}
\def\coltext#1#2#3{{\localcolor\let\Tcolor=#1\let\Bcolor=#2\relax
   \setbox1=\hbox{-\kern.05em}%
   \setbox1=\hbox{{\Bcolor\vrule\coltextstrut width\wd1}\kern-.05em\llap{\Tcolor -}}%
   \def\-{\discretionary{\copy1}{}{}}%
   \def\uline##1{\skip0=##1\advance\skip0 by.05em
      \Bcolor\leaders \vrule\coltextstrut\hskip\skip0 \hskip-.05em\relax}%
   \def\uspace{\fontdimen2\font plus\fontdimen3\font minus\fontdimen4\font}%
   \def~{\egroup\hbox{\uline{\wd0}\llap{\Tcolor\copy0}}\nobreak{\uline\uspace}\relax \setbox0=\hbox\bgroup}%
   \leavevmode\coltextA #3 {} }}
\def\coltextA#1 {\ifx^#1^\unskip\unskip\else
   \hyphenprocess{#1}\expandafter\coltextB\listwparts\-\end
\expandafter\coltextA\fi}
```

```
\def\coltextB#1\-#2\end{\ifx^#2^\coltextC{#1}\else
   \coltextD{#1}\def\next{\coltextB#2\end}\expandafter\next\fi}
\def\coltextC#1{\setbox0=\hbox{#1}\hbox{\uline{\wd0}\hbox{\llap{\Tcolor\copy0}}}\uline\uspace\relax}
\def\coltextD#1{\setbox0=\hbox{#1}\hbox{\uline{\wd0}\llap{\Tcolor\copy0}}\-}
```

Na rozdíl od makra `\ul` zde sázíme „přeškrtnutí a text“ v opačném pořadí, tj. nejprve „přeškrtnutí“ a potom „text“. Přitom „přeškrtnutí“ má nyní výšku a hloubku danou makrem `\coltextstrut`, tedy je natolik široké, že vytvoří podbarvení. V úvodu makra `\coltext` je také v odpovídajících barvách připraven `\hyphenchar` jako `\box1`.

12.4 Čára v okraji vyznačující text

0128
P. O.
04. 11. 2015

Uživatel uzavře vyznačovaný text pomocí `\beginspec...\endspec`. Začátek textu může být někde uvnitř odstavce a konec třeba v dalším odstavci. Makro vytvoří v pravém okraji čáru začínající ve výšce prvního slova vyznačeného textu a končící ve výšce posledního slova vyznačeného textu.

Implementace se opírá o pdfTeXový primitivní příkaz `\pdfsavepos` a využívá REF soubor ke zjištění polohy začínajícího a končícího slova. Takže čáry se stabilizují ve správné poloze až po druhém běhu TeXu. Implementace vypadá takto:

```
\def\preparespec#1{\expandafter\ifx \csname spec:#1\endcsname \relax \sdef{spec:#1}{}\fi}
\def\XspecB#1{\preparespec{\the\lastpage}
    \expandafter\addto\csname spec:\the\lastpage\endcsname{,\specdraw{#1}}}
\def\XspecE#1{\preparespec{\the\lastpage}
    \expandafter\addto\csname spec:\the\lastpage\endcsname{{#1}}}

\input opmac % načtení OPmac musí proběhnout až po definici \XspecB a \XspecE

\newif\ifspecenv \specenvfalse
\def\beginspec{\ifspecenv \opwarning{\noexpand\beginspec inside spec environment, ignored}%
   \else \global\specenvtrue \openref
      \ifvmode \pdfsavepos \else \vbox to0pt{\vss\pdfsavepos\kern\ht\strutbox}\fi
      \wref\XspecB{{\the\pdflastypos}}\fi
}
\def\endspec{\ifspecenv \global\specenvfalse
      \ifvmode \pdsavepos \else\vbox to0pt{\kern\dp\strutbox\pdfsavepos\vss}\fi
      \wref\XspecE{{\the\pdflastypos}}%
   \else \opwarning{\noexpand\endspec outside spec environment, ignored}\fi
}
\def\prepghook{\moveright\hsize\vbox to0pt{\preparespec{\the\pageno}
   \expandafter\expandafter\expandafter
   \specdrawB\csname spec:\the\pageno\endcsname \botypos\relax}\nointerlineskip
}
\def\specdraw#1#2#3{\moveright 5pt\vbox to0pt{
   \tmpnum=\topypos \advance\tmpnum by-#1 \kern\tmpnum sp
   \tmpnum=-#2 \advance\tmpnum by#1 \hrule height\tmpnum sp width 2pt \vss}\nointerlineskip
   \ifx#3\relax \global\let\specdrawB=\specdrawBrun \fi
}
\def\specdrawB#1{}
\def\specdrawBrun#1#2{\gdef\specdrawB##1{}\specdraw{\topypos}{#1}{#2}}

\tmpdim=\pdfpageheight \advance\tmpdim by-1in \advance\tmpdim by-\voffset \edef\topypos{\number\tmpdim}
\advance\tmpdim by-\vsize \advance \tmpdim by-\dp\strutbox \edef\botypos{\number\tmpdim}
```

Makra `\XspecB{od}` a `\XspecE{do}` uloží během načtení REF souboru údaje do maker `\spec:pageno` (pageno je číslo příslušné strany) a jsou tam ve tvaru:

```
,\specdraw{od}{do},\specdraw{od}{do},\specdraw{od}{do}
```

V tomto tvaru je použije výstupní rutina v `\prepghook`: Spustí `\specdrawB` následovaný expandovaným `\spec:pageno` následovaný dalšími dvěma tokeny `\botypos\relax`. Přitom

`\specdrawB` typicky pouze ignoruje první čárku a `\specdraw` přečte `{od}` a `{do}` a následující token, který typicky rovněž ignoruje (tj. ignoruje následující čárku nebo token `\botypos`). Dále `\specdraw` nakreslí čáru `{od}--{do}`.

Zajímavá je situace, kdy čára na příslušné straně není uzavřena. Pak chybí údaj `{do}` a místo něj `\specdraw` načte `\botypos`, tj. polohu spodní hrany stránky. Třetí ignorovaný parametr je pak `\relax` a v takovém případě `\specdraw` předefinuje `\specdrawB` pro následující stranu jako `\specdrawBrun`, aby tam čára mohla pokračovat. Za domácí cvičení prostudujte definici `\specdrawBrun` a rozmyslete si, že makro funguje i pro případy, kdy `\spec:pageno` je prázdná a `\specdrawB` pracuje v původním významu (žádná čára na stránce) nebo ve významu `\specdrawBrun` (čára na stránce odshora dolů).

Další domácí úkol: modifikujte makro tak, aby vytvářelo barevné rámečky pod textem (užití `\beginspec` a `\endspec` se v tomto případě předpokládá ve vertikáním módu mezi odstavci).

13 Makro přemety

0060

13.1 Využití \replacestrings

P. O.
15. 05. 2014

OPmac disponuje makrem `\replacestrings`, které používá v makru `\url`. Makro `\replacestrings` v přechodném bufferu `\tmpb` vymění stringy za jiné stringy. Toto makro jsem využil, když jsem potřeboval v šabloně CTUStyle vygenerovat genitiv (tj. druhý pád) fakulty. Fakultu napíše student jen jako značku F1 až F8 do `\toks` stringu `\faculty`. Tedy například napíše `\faculty{F3}`. Pomocí `\mtext` je tato značka konvertovaná na název fakulty dle právě aktuálního jazyka. Když je tímto jazykem čeština, objeví se název fakulty v prvním pádě. Někdy ale potřebujeme jej ohnout do genitivu. Použil jsem následující kód:

```
\def\facultygenitiv{\edef\tmpb{\mtext{\the\faculty} }%
   \replacestrings{ulta }{ulty }%
   \replacestrings{á }{é }%
   \tmpb}
```

Když je například `\faculty{F4}` a jazyk český, pak se `\mtext{\the\faculty}` expanduje na „Fakulta jaderná a fyzikálně inženýrská“. Při použití makra `\facultygenitiv` přitom dostanu „Fakulty jaderné a fyzikálně inženýrské“. Pomocí výše uvedených čtyř řádků jsem schopen převést do genitivu všech osm fakult ČVUT.

0061

13.2 Výpočet směru vektoru

P. O.
24. 05. 2014

Je dán vektor v rovině a je třeba najít úhel mezi tímto vektorem a osou `x` ve stupních. Je to potřebné, když chceme například otočit nějaký grafický prvek (např. hrot šipky) dle požadovaného směru. Vytvoříme makro

```
\calculateargofvector (X0 Y0) (X Y)  % (X0 Y0) je počátek vektoru, (X Y) konec
například
\calculateargofvector (0 0) (2 3)
\calculateargofvector (1.1 2.7) (-4.7 6.3)
```

které vrátí výsledek v makru `\argofvector`. Protože TeX nedisponuje interně implementovanými funkcemi pro odmocninu a `arccos`, je makro poněkud náročnější.

```
\newdimen\dimX  \newdimen\dimY

\def\processatan #1 {\tmpdim=\ifx\relax#1\maxdimen \else.#1pt\fi\relax
   \ifdim\dimY<\tmpdim\expandafter\processatanE
   \else \advance\tmpnum by1 \def\tmp{#1}\expandafter \processatan \fi}
\def\processatanE #1\relax{}

\def\calculateargofvector (#1 #2) (#3 #4){\def\tmp{0}\tmpnum=0
   \dimX=#3pt \advance\dimX by-#1pt \dimY=#4pt \advance\dimY by-#2pt
   \ifdim\dimX=0pt \ifdim\dimY=0pt \dimX=1pt \dimY=0pt \fi \fi
```

```
   \ifdim\dimY<0pt \dimY=-\dimY \dimX=-\dimX \advance\tmpnum by180 \fi
   \ifdim\dimX<0pt \tmpdim=\dimX \dimX=\dimY \dimY=-\tmpdim \advance\tmpnum by90 \fi
   \ifdim\dimY>\dimX \tmpdim=\dimX \advance\tmpdim by\dimY
                   \advance\dimY by-\dimX \dimX=\tmpdim \advance\tmpnum by45 \fi
   \ifdim\dimX<63pt \multiply\dimX by256 \multiply\dimY by256
   \else \ifdim\dimX<1023pt \multiply\dimX by16 \multiply\dimY by16 \fi\fi
   \divide\dimX by256 \divide\dimY by\dimX \multiply\dimY by256
   \processatan 017 035 052 07 087 105 123 14 158 176 194 21 23 25 268 287 306 325 344 364
                384 404 424 445 466 488 51 532 554 577 6 625 649 675 7 727 754 781 81 839
                869 9 932 966 99999 {\relax} \relax
   \edef\argofvector{\the\tmpnum}%
   \ifdim\tmpdim=\maxdimen \tmpnum=0 \else
      \advance\tmpdim by-.\tmp pt \advance\dimY by-.\tmp pt \multiply\dimY by10
      \tmpnum=\dimY \divide\tmpnum by\tmpdim
   \fi
   \ifnum\tmpnum>0 \edef\argofvector{\argofvector.\the\tmpnum}\fi
}
```

V prvé části výpočtu otočíme vektor případně o 180 a 90 stupňů, abychom jej dostali do prvého kvadrantu. Otočení ve stupních přičítáme k výsledku v `\tmpnum`. Dále vektor otočíme do první půle kvadrantu o 45 stupňů, pokud tam již není. Při tomto otočení se změní jeho velikost, což nám nevadí, protože hned dalším krokem je normalizace vektoru tak, že má souřadnici X rovnou jedné. Tj. souřadnice `Y` je rovna tangentě hledaného úhlu. Za makrem `\processatan` jsou napsány tangenty úhlů 1, 2, 3 až 45 a toto makro cyklem najde odpovídající úhlel a zbylá data až po `\relax` přeskočí. V závěru je výpočet desetiny úhlu lineární aproximací v intervalu `tg(n)--tg(n+1)`.

13.3 Definování makra s nepovinným parametrem

0067
P. O.
12. 06. 2014

Budeme chtít definovat `\makro` s dvojím využitím:

```
\makro parametry
nebo
\makro [optional] parametry
```

K tomu účelu definujeme `\optdef`, což se použije třeba takto:

```
\optdef\makro [default] #1 #2 {opt=\opt, 1=#1, 2=#2.}

\makro prvni druhy        ... expanduje na: opt=default, 1=prvni, 2=druhy.
\makro [kuk] treti ctvrty ... expanduje na: opt=kuk, 1=treti, 2=ctvrty.
```

Například je možné předefinovat kapitoly, sekce a podsekce, aby bylo možné zadat lejblík jako nepovinný parametr:

```
\let\chapOri=\chap  \let\secOri=\sec   \let\seccOri=\secc
\optdef\chap [] {\ifx\opt\empty\else\label[\opt]\fi \chapOri}
\optdef\sec  [] {\ifx\opt\empty\else\label[\opt]\fi \secOri}
\optdef\secc [] {\ifx\opt\empty\else\label[\opt]\fi \seccOri}
```

Makro `\optdef` může být definováno takto:

```
\def\optdef#1[#2]{%
   \def#1{\def\opt{#2}\isnextchar[{\csname oA:\string#1\endcsname}{\csname oB:\string#1\endcsname}}%
   \sdef{oA:\string#1}[##1]{\def\opt{##1}\csname oB:\string#1\nospaceafter\endcsname}%
   \sdef{oB:\string#1\nospaceafter}%
}
\def\nospaceafter#1{\expandafter#1\romannumeral-`\.}
```

`\optdef` definuje `\makro` jako `\isnextchar[{\oA:\makro}{\oB:\makro}` a dále definuje `\oA:\makro[text]` jako `\def\opt{text}\oB:\makro` a konečně definuje `\oB:\makro` jako to, co napsal uživatel za `\opdef[text]`. Ignorování případné mezery za zavírací hranatou závorkou obstará příkaz `\rommannumeral-`\.`, který expanduje na úplné nic, ale navíc při expanzi zkonzumuje případnou mezeru, která může následovat.

0068 ## 13.4 Odstranění poslední mezery v parametru

P. O. 13. 06. 2014

Například parametry příkazů `\chap`, `\sec` a `\secc` obvykle obsahují na svém konci mezeru, ale někdy taky ne. Například při `\sec Prokletý \LaTeX` §tt§char60`prázný řádek`§tt§char62 na konci parametru mezera není. OPmac řeší odstranění závěrečné mezery v těchto případech přidáním `\unskip`. Ovšem my bychom chtěli odstranit tuto závěrečnou případnou mezeru na úrovni maker. Přitom v parametru může být mezer více, nás zajímá jen ta na úplném konci.

Stačí si parametr uložit do `\tmpb` například pomocí `\def\tmpb{#1}` a dále provést:

```
\addto\tmpb\end \replacestrings{ \end}{}\replacestrings{\end}{}
```

A nyní už `\tmpb` obsahuje parametr bez případné závěrečné mezery.

0121 ## 13.5 Makro s parametrem do konce řádku

P. O. 14. 08. 2015

Makro `\eoldef` umožní definovat makro s jedním parametrem, který je separován koncem řádku. Například takto:

```
\eoldef\aha#1{\message{param: "#1"}}

\aha tady je parametr
a tu je další text.
```

Vidíme, že pomocí `\eoldef` definujeme jakoby makro s neseparovaným parametrem, ale ono makro si vezme parametr až do konce řádku. V uvedeném příkladu si tedy do `#1` vezme text „tady je parametr“.

Upozornění. Od verze OPmac Apr. 2016 je makro `\eoldef` přímo součástí OPmac a pomocí `\eoldef` jsou definovány `\tit`, `\chap`, `\sec` a `\secc`. Změna není stoprocentně zpětně kompatibilní, ale přesto doufám, že výhody převáží nad nevýhodami. Odpadá třeba problém s případnou mezerou na konci parametru a dále problém uživatelů, napíšou-li `\sec` na poslední řádek souboru. Nevýhody: Chcete-li rozdělit titulek ve zdrojovém textu do více řádků, je třeba na konec pokračovacích řádků dát `%`. Pokud chcete použít `\sec` uvnitř vlastního makra, nelze psát `... \sec#1\par ...`, protože se objeví chybová hláška

```
! Paragraph ended before \eoldefA was complete.
```

Je možné použít `\bracedparam` z OPmac triku 0036 a psát ve svém makru třeba `... \bracedparam\sec{#`

Pokud chcete vypnout separaci koncem řádku a vrátit se k původní separaci parametru maker `\tit`, `\chap`, `\sec` a `\secc` prázdným řádkem, můžete použít:

```
\def\eoldefA{\endgroup\eoldefB}
\def\eoldefB#1#2\par{\csname\string#1:M\endcsname{#2}}
```

0069 ## 13.6 Slovníky tvaru klíč=hodnota

P. O. 16. 06. 2014

Umožníme zadávat údaje ve tvaru `klíčA=hodnotaA`, `klíčB=hodnotaB` atd., tj. čárkami oddělený seznam přiřazení. Makro `\kv{klíč}` pak expanduje na hodnotu nebo na `\kvunknown`, jestliže klíč nemá přidělenu hodnotu. Ke čtení seznamu přiřazení poslouží makro `\kvscan`, za kterým musí následovat seznam přiřazení ukončený čárkou, čárkou, rovnítkem, čárkou. Makra `\kv` a `\kvscan` využije programátor maker například takto

```
\def\mymacrodefault          {color={}, width=0.4pt}
\optdef\mymacro [] {\bgroup
   \expandafter \kvscan\mymacrodefault,,=,%  implicitni hodonty
   \expandafter \kvscan\opt,,=,%  hodnoty dane uzivatelem
   \if^\kv{color}^\else \localcolor \kv{color}\fi % nastaveni barvy
   \let\vruleprimitive=\vrule
   \def\vrule{\vruleprimitive width\kv{width}}% nastaveni sily car
   ...
   \egroup
}
```

Zde je navíc využito makro `\optdef` z triku 0067. Uživatel pak může psát `\mymacro` bez parametrů, nebo třeba `\mymacro[width=.7pt]` nebo `\mymacro[width=.8pt, color=\Red]`.

Makra `\kv` a `\kvscan` lze implementovat takto:

```
\def\kv#1{\expandafter\ifx\csname kv:#1\endcsname \relax \expandafter\kvunknown
   \else \csname kv:#1\expandafter\endcsname\fi
}
\def\kvunknown{???}
\def\kvscan #1#2=#3,{\ifx#1,\else \kvdef{kv:#1#2}{#3}\expandafter\kvscan\fi}
\let\kvdef=\sdef
```

Makro `\kvscan` čte klíč rozložen do dvou parametrů `#1#2`. Tím umožní za čárkami v seznamu přiřazení dávat nepovinné mezery, které jsou parametrem `#1` ignorovány. Pokud umožníte uživateli dávat nepovinné mezery i kolem rovnítka, je možné upravit jeho seznam přiřazení třeba takto:

```
... \let\tmpb=\opt \replacestrings{ =}{=}\replacestrings{= }{=}%
    \expandafter \kvscan\tmpb,,=,%
```

Místo `\let\kvdef=\sdef` je možné použít třeba

```
\def\kvdef#1{\expandafter\edef\csname#1\endcsname}
```

pokud si přejete, aby byly hodnoty expandovány v okamžiku přiřazení. Chcete-li ve svém makru ošetřit, zda má klíč přiřazenu hodnotu, použijte `\isdefined{kv:klíč}\iftrue`.

13.7 Přepínače ve slovnících tvaru klíč=hodnota

0114
P. O.
30. 06. 2014

Do slovníku, který čárkami odděluje `klíč=hodnota`, chceme přimíchat čárkami oddělené „přepínače“, tj. samostatná slova bez následujícího rovnítka a hodnoty. Například:

```
\mymacroset {width=0.8pt, draft, silent}
```

Řešením může být užití `\replacestrings`, jak je uvedeno v následujícím příkladu:

```
\def\mymacroset#1{\def\tmpb{#1,}\replacestrings{ =}{=}\replacestrings{= }{=}%
   \replacestrings{draft,}{my-final=0,}%
   \replacestrings{final,}{my-final=1,}%
   \replacestrings{silent,}{my-message=0,}%
   \replacestrings{verbose,}{my-message=1,}%
   \expandafter\kvscan\tmpb,=,%
   \if1\kv{my-message}\let\mymessage=\message \else \def\mymessage##1{}\fi
   ...
}
\mymacroset {width=0.7pt, final, silent} % default values
```

Přepínače jsme interně převedli na tvar `klíč=hodnota` a použili jsme `\kvscan` z předchozího OPmac triku 0069.

V dalších makrech se můžeme na použitý přepínač dotazovat například pomocí

```
\if1\kv{my-final}Byl použit přepínač final.\else Byl použit přepínač draft.\fi
```

13.8 Vnořené závorky jiného typu než {}

0077
P. O.
09. 08. 2014

TeX si hlídá automaticky párování a vnoření pouze jednoho typu závorek: `{}`. OPmac používá pro parametry občas ještě závorky `[]`, ale ty se nedají vnořovat. Napíšete-li tedy třeba `\label[a[b]c]`, dostanete lejblík `a[b` a dále `c]` se vytiskne. Pokud si chcete pohlídat i párování takových typů závorek (hranatých i jiných), můžete použít makro `\ensurerebalanced` tímto způsobem:

```
\def\makro[#1]{\ensurebalanced[]\makroA{#1}}
\def\makroA#1{zde má parametr "#1" balancovány závorky [].}
například:
\makro[a[b]c] vytiskne: zde má parametr "a[b]c" balancovány závorky [].
```

Můžete si předefinovat třeba makro `\label` tak, aby akceptovalo balancovaný text s hranatými závorkami:

```
\def\tmp{\def\labelA##1}
\expandafter\tmp\expandafter{\label[#1]}
\def\label[#1]{\ensurebalanced[]\labelA{#1}}
```

Makro `\ensurebalanced` je definováno následovně:

```
\def\ensurebalanced#1#2#3#4{%
   \isbalanced#1#2{#4}\iftrue #3{#4}%
   \else
      \def\ensurebalancedA##1##2#2{%
         \isbalanced#1#2{##1#2##2}\iftrue #3{##1#2##2}%
         \else \def\next{\ensurebalancedA{##1#2##2}}\expandafter\next\fi
      }%
      \def\next{\ensurebalancedA{#4}}\expandafter\next\fi
}
\def\isbalanced#1#2#3\iftrue{\tmpnum=0 \isbalancedA#1#2#3\isbalanced}
\def\isbalancedA#1#2#3{%
    \ifx\isbalanced#3\def\next{\csname ifnum\endcsname\tmpnum=0 }%
    \else \def\next{\isbalancedA#1#2}%
        \isonetoken#3\iftrue
           \ifx#3#1\advance\tmpnum by1\fi
           \ifx#3#2\advance\tmpnum by-1\fi
     \fi\fi\next
}
\def\isonetoken#1#2\iftrue{\ifx\isbalanced#2\isbalanced}
```

Makro `\ensurebalanced` pohlídá pomocí `\isbalanced`, zda je přečtený text balancovaný na závorky `[=#1` a `]=#2`. Pokud ano, spustí `\makroA`, tedy `#3` následované přečteným parametrem. Pokud ne, zavolá (případně i rekurzívně) makro `\ensurebalancedA`, které přečte další část textu parametru.

0088

13.9 Čtení parametru token po tokenu

P. O.
20. 01. 2015

Vytvoříme makro `\readtoks{parametr}`, který čte jednotlivé tokeny parametru, může je na základě jejich typu třeba nějak modifikovat, a výsledek čtení a modifikace uloží do `\readtoksO`, což je implicitně definováno jako `\toks1`. Užití tohoto makra najdete v následujícím OPmac triku 0079, kde makro `\readtoks` prochází seznam tokenů a jakmile narazí na kříž (kategorii 6), promění ho ve dva kříže. Jakmile narazí na sekvenci `\internalXpram`, promění ji v jeden kříž. Takže třeba po

```
\readtoks{aha # uff {\internalXparam1 {a}} \line}
budeme mít:
\toks1={aha ## uff {#1 {a}} \line}
```

Makro jsem též v mírných obměnách použil v odpovědích na otázku o proměnlivých separátorech nebo na otázku o alternaitvních svorkách na tex.stackexchange.com

Základní problém makra `\readtoks` je, že nemůže bezhlavě nabírat jednotlivé tokeny do neseparovaného parametru, protože to zničí mezery a svorkaté závorky. Je tedy třeba tyto situace ošetřit zvlášť.

```
\newtoks\readtoksT \newif\ifreadtoksG
\def\readtoks{\begingroup \let\bgroup=\relax \let\egroup=\relax
   \readtoksT={}\readtoksGfalse \afterassignment\readtoksA \let\next= }
\def\readtoksA{\futurelet\tmpc\readtoksB}
\def\readtoksB{\let\next=\readtoksD \csname readtoksX\endcsname
   \ifcat\space\noexpand\tmpc \let\next=\readtoksC \def\nexxt{\readtoksD{ }}\fi
   \ifcat{\noexpand\tmpc      \let\next=\readtoksC \let\nexxt=\readtoksE \fi
   \ifcat}\noexpand\tmpc      \let\next=\readtoksC \let\nexxt=\readtoksF \fi
   \next
```

```
}
\def\readtoksC{\afterassignment\nexxt \let\next= }
\def\readtoksD#1{\readtoksT=\expandafter{\the\readtoksT#1}\readtoksA}
\def\readtoksE{\begingroup \readtoksGtrue \readtoksT={}\readtoksA}
\def\readtoksF{\ifreadtoksG
      \expandafter\endgroup\expandafter\readtoksT\expandafter\expandafter\expandafter
         {\expandafter\the\expandafter\readtoksT\expandafter{\the\readtoksT}}%
      \expandafter\readtoksA
   \else
      \expandafter\endgroup\expandafter\readtoksO\expandafter{\the\readtoksT}%
   \fi
}
\def\readtoksO{\toks1}
```

Makro postupně sbírá tokeny a spouští uživatelsky definované makro `\readtoksX`, ve kterém může uživatel tokeny pozměňovat. Pokud toto makro není definováno, pak se pouze postupným sbíráním tokenů naplní výsledný registr `\readtoksO` stejně, jako při přímém čtení `\readtoksO={parametr}`. Všimněte se, že nejvíce práce dají svorky. Narazíme-li na vstupní svorku, vnoříme se do nové skupiny, kde začínáme plnit `\readtoksT` od začátku, tedy naplníme ho vnitřním obsahem párujících svorek. Narazíme-li pak na ukončovací svorku, využijeme přečtený `\readtoksT`, který obalíme do svorek a před něj dáme původní `\readtoksT`, který obsahuje text načtený před zahajovací svorkou. A v rámci toho ukončíme skupinu.

13.10 Expandující čtení textu token po tokenu

0153
P. O.
06. 06. 2016

Předchozí OPmac trik popisuje možnost čtení jednotlivých tokenů hlavním procesorem. Nyní vytvoříme makro, které čte tokeny jen expand procesorem. TeX bohužel při čtení následujícího tokenu makrem s neseparovaným parametrem přeskočí případnou mezeru. Takže cyklus:

```
\def\apply#1{[#1]}
\def\readtokens#1{\ifx\end#1\else\apply{#1}\expandafter\readtokens\fi}
\readtokens Tady je nějaký text.\end
```

přečte totéž, jako při `\readtokens Tadyjenějakýtext.\end`. Naším úkolem je vytvořit makro, které mezery respektuje a je plně expandující. Další problém při čtení tokenů makrem s neseparovaným parametrem spočívá v tom, že třeba `{abc}` přečte naráz a navíc odstraní kučeravé závorky. Naše makro `\etoks{tokeny}` oba problémy řeší a spustí makro `\eapply` na každý jednotlivý token svého parametru. Respektuje mezery i kučeravé závorky. Takže třeba po

```
\def\eapply#1{[#1]} % co provést s každým jednotlivým tokenem
\message{... \etoks{ab c{ aa bc {bb}}cb}}
```

dostaneme `... [a][b][ ][c]{[ ][a][a][ ][b][c][ ]{[b][b]}}[c][b].`

Implementace makra je následující:

```
\def\etoks#1{\etoksA #1{\end}}
\def\etoksA#1#{\etoksB#1 {\end} }
\def\etoksB#1 #2 {\etoksC#1\end
    \ifx\end#2\empty\expandafter\etoksD\else\eapply{ }\fihere{\etoksB#2 }\fi}
\def\etoksC#1{\ifx\end#1\else\eapply{#1}\expandafter\etoksC\fi}
\def\etoksD#1{\ifx\end#1\empty\else\fihere{{\etoks{#1}}\etoksA}\fi}
\def\fihere#1\fi{\fi#1}
```

Makro se opírá o možnost separace závorkou pomocí konstrukce `\def\x#1#{...}`. Dále pomocí `\etoksB` odlišuje jednotlivá „slova“ oddělená mezerami a `\etoksC` čte teprve jednotlivé tokeny ve slovech. Po ukončení čtení slov následuje závorka, kterou zpracuje `\etoksD` a případně rekurzivně zavolá `\etoks` na vnitřek závorky.

0079
P. O.
17. 01. 2015

13.11 Vylepšené \addto pro makra s parametry

Makro \addto z OPmac přidává ke zvolenému makru další text. Ovšem zvolené makro musí být bez prametrů. Navrhnul jsem tedy další makra \appendto a \prependto, která dokáží rozšířit stávající makro (vzadu nebo vpředu) třebaže toto makro má parametry. Příklady použití:

```
\def\a#1==#2{#1 is equal #2}
\appendto\a{ and this means that #1=#2.}
% \a je nyní makro: #1==#2 -> #1 is equal #2 and this means that #1=#2.
\prependto\a{We have (#1) and (#2). The }
% \a je nyní makro: #1==#2 -> We have (#1) and (#2).
%                        The #1 is equal #2 and this means that #1=#2.
```

Pomocí \let\appendprefix=\long nebo \let\appendprefix=\global je možné specifikovat, jakého typu rozšířené makro bude. Tip: například pomocí

```
\def\appendtoprefix{\protected\long}\appendto\makro{}
```

je možné přetypovat stávající \makro na jiný typ.

```
\let\appendtoprefix=\relax
\def\appendto#1#2{\appendtoA#1%
   \appendtoprefix\expandafter\def\expandafter#1\the\toks0\expandafter
      {\the\toks1 #2}}
\def\prependto#1#2{\appendtoA#1\toks2={#2}%
   \appendtoprefix\expandafter\def\expandafter#1\the\toks0\expandafter
      {\the\toks2\expandafter\space\the\toks1}}
\def\appendtoA#1{\edef\tmpb{\expandafter\appendtoB\meaning#1\end}%
   \scantokens\expandafter{\expandafter\toks\expandafter0\expandafter{\tmpb}}%
   \expandafter\replacestrings\expandafter{\string##}{\#}%
   {\def\###1{{\noexpand\internalXparam##1}}\xdef\tmpa{\toks1={\tmpb}}}\tmpa
   \scantokens\expandafter{\expandafter\toks\expandafter1\expandafter{\the\toks1}}%
   \toks1=\expandafter\expandafter\expandafter{\expandafter#1\the\toks1}%
   \expandafter\readtoks\expandafter{\the\toks1}%
}
\def\appendtoB#1:#2->#3\end{#2}
\def\readtoksX{%
   \ifcat##\noexpand\tmpc   \let\next=\readtoksC \def\nexxt{\readtoksD{########}}\fi
   \ifx\internalXparam\tmpc \let\next=\readtoksC \def\nexxt{\readtoksD{####}}\fi
}
\def\internalXparam{\internalXparam}
```

Makra \appendto i \prependto nejprve připraví pomocí \appendtoA do \toks0 masku prametrů původního makra a do \toks1 obsah původního makra. Masku parametrů připraví tak, že si ji sejme pomocí \meaning a \appendtoB z výpisu významu makra. Pak ji prožene \scantokens, protože kategorie jsou ve výpisu \meaning nastaveny nevhodně. Dále provede \appendtoA tento trik: každý výskyt formálního parametru v masce parametrů obalí do svorek ve formě \internalXparam číslo-parametru a toto uloží přechodně do \toks1. Je-li třeba maska parametrů ve tvaru #1x#2::#3, pak v \toks1 bude {\internalXparam1}x{\internalXparam2}::{\int To provede pomocí \replacestings #-§tt§char62\# a dále definicí \# jako makra s parametrem, které vytvoří požadovaný výsledek expanzí takové masky. Takto připravený \toks1 předloží původnímu makru, které tedy do #1 nabere \internalXparam1, do #2 nabere \intenralXparam2 atd. Tento výsledek expanze proženeme makrem \readtoks z předchozího OPmac triku 0088 a tím dostaneme v \toks1 seznam tokenů připravený k nové definici původního makra. Poznamenávám, že na rozdíl od podobného makra \apptocmd a \pretocmd z LaTeXového balíčku etoolbox, naše makro zachovává všechny kategorie v těle makra, takže je rozbustnější.

13.12 Makro \patchto na modifikaci makra

0087
P. O.
18. 01. 2015

Pokud chceme implemetovat `\patchto`, které ve stávajícím makru vymění text za jiný text, tedy

```
\patchto\makro {vyhledaný text}{vyměněný za}
```

pak můžeme využít předchozí OPmac trik 0079, ale pro jednoduchost provedeme záměnu textu za text po detokenizaci. Záměnu provedeme pomocí `\replacestrings` a nakonec tělo makra zpětně tokenizujeme. Tedy v tomto případě makro pracuje analogicky jako makro `\patchcmd` z LaTeXového etoolbox.

```
\def\patchto#1#2#3{\appendtoA#1%
   \edef\tmpb{\detokenize\expandafter{\the\toks1}}%
   \edef\tmpa{\noexpand\replacestrings{\detokenize{#2}}{\detokenize{#3}}}\tmpa
   \edef\tmpa{\noexpand\replacestrings{\string##\string##}{\string##}}\tmpa
   \scantokens\expandafter{\expandafter\toks\expandafter1\expandafter{\tmpb}}%
   \appendtoprefix\expandafter\def\expandafter#1\the\toks0\expandafter{\the\toks1 }}
```

13.13 Cyklus typu for

0115
P. O.
01. 07. 2015

Vytvoříme makro `\for \i=X to Y by Z {tělo cyklu}`, které alokuje proměnnou `\i`, nastaví ji na hodnotu `X` a opakuje tělo cyklu tak dlouho, dokud proměnná `\i` nepřekročí `Y`, přitom po každém vykonání těla cyklu zvedne `\i` o `Z`. Například:

```
\for \i=1 to 10 by 1 {\for \j=1 to 10 by 1 {\message{a[\the\i,\the\j]}}}
```

vypíše `a[1,1] a[1,2] ... a[1,10] a[2,1] a[2,2] ... a[10,9] a[10,10]`.

Celý cyklus probíhá uvnitř skupiny a je zahájen automatickou alokací proměnné cyklu, která je uvedena jako první parametr. Není tedy nutné tuto proměnnou alokovat. Ono to ani není žádoucí, protože řídicí sekvence `\i`, `\j` bývají v TeXu vyhrazeny k jiným účelům. Po skončení cyklu je `\i` znovu tím, čím bylo (například v plainTeXu) původně definováno.

Makro `\for` můžeme definovat třeba takto:

```
\def\for#1=#2to#3by#4#{\forA{#1}{#2}{#3}{#4}}
\long\def\forA#1#2#3#4#5{\begingroup
   {\escapechar=‘\\ % allocation of #1 as counter:
      \expandafter \ifx\csname for:\string#1\endcsname \relax
         \csname newcount\expandafter\endcsname \csname for:\string#1\endcsname\fi
    \expandafter}\expandafter\let\expandafter#1\csname for:\string#1\endcsname
   #1=#2%
   \def\forB{#5\advance#1by#4\relax \expandafter\forC}%
   \ifnum#4>0 \def\forC{\ifnum#1>#3\relax\else\forB\fi}%
   \else      \def\forC{\ifnum#1<#3\relax\else\forB\fi}%
   \fi
   \ifnum#4=0 \let\forC=\relax \fi
   \forC \endgroup
}
```

V prvé části makra po zjištění parametrů se globálně alokuje proměnná `\for:\i` (pokud je `#1=\i`) nebo `\for:\j` (pokud je `#1=\j`) atd. Tato alokace probíhá jen tehdy, pokud tato proměná nebyla dříve alokována. Tj. opakované použití `\for\i=...` nealokuje proměnnou `\for:\i` opakovaně. Následně se lokálně přiřadí uživatelem deklarovaná proměná cyklu (tj. `\i`, `\j` atd.) té globálně alokované.

Poté je nastavena hodnota této proměnné cyklu na výchozí hodnotu `X`. Cyklus samotný spočívá v opakovaném volání makra `\forC`, které obsahuje kontrolu, zda nebyla překročena koncová hodnota. Pokud ne, volá se `\forB`. Toto makro obsahuje jednak tělo cyklu `#5`, dále zvětšení proměnné cyklu o hodnotu `Z` a konečně nové volání makra `\forC` rekurzí. Jsou rozlišeny dva případy podmínky konce cyklu v závislosti na tom, zda krok `Z` je kladný nebo záporný. Je-li tento krok nulový (nekonečný cyklus), raději neprovedeme nic.

0120 13.14 Cyklus na úrovni expand procesoru

P. O.
25. 07. 2015

Při zpracování cyklu je třeba měnit proměnnou cyklu a to vyžaduje přiřazení. Přiřazení ale nelze provést v expand procesoru. Ukážeme dvě řešení, jak to obejít. První je hodně trikoidní využívající `\romannumeral` a pracuje i v klasickém TeXu. Druhé řešení využívá toho, že eTeXový příkaz `\numexpr` pracuje v expand procesoru.

Vytvoříme makro `\rep{kolik}{co}`, které zopakuje `co` `kolik`rát, tedy třeba `\rep{5}{uf}` expanduje na `ufufufufuf`.

V klasickém TeXu by implementace makra `\rep` vypadala takto:

```
\def\rep#1#2{\expandafter\repA\romannumeral#1000.{#2}}
\def\repA#1.#2{\repB{#2}#1.}
\def\repB#1#2{\ifx.#2\expandafter\repC\else#1\expandafter\repB\fi{#1}}
\def\repC#1{}
```

Nejprve je `\rep{5}{uf}` expandováno na `\repA mmmmm.{uf}` (tedy 5krát `m`) a pořadí parametrů je pomocí `\repA` převráceno na `\repB {uf}mmmmm`. Makro `\repB` provede cyklus, při němž odebírá jednotlivá `m` a končí při tečce.

S využitím eTeXu může implementace makra `\rep` vypadat takto:

```
\def\rep#1#2{\repA 0{#1}{#2}}
\def\repA#1#2#3{\ifnum#2>#1 #3\expandafter\repB\else\expandafter\repC\fi{#1}{#2}{#3}}
\def\repB#1{\expandafter\repA\expandafter{\the\numexpr#1+1}}
\def\repC#1#2#3{}
```

V tomto případě se opakuje `\repA` s parametry `{kolik-bylo}{kolik}{co}`. Toto makro vloží do výstupu `co` za předpokladu, že `kolik-bylo` §tt§char60 `kolik`. V takovém případě pomocí `\repB` a `\numexpr` zvětší hodnotu kolik-bylo o jedničku a zopakuje `\repA`.

Druhé řešení umožní za parametr `kolik` vkládat i registry nebo konstanty, tj. třeba `\chardef\x=10 \rep\x` což první řešení neumožní a vyžaduje takový parametr prefixovat pomocí `\the`.

0086 13.15 LaTeXové newcommand

P. O.
09. 01. 2015

Výjimečně se může stát, že potřebujeme číst LaTeXově napsaný kus kódu, ve kterém je použito `\newcommand`. Toto makro má poměrně obskurní syntaxi: za `\newcommand` následuje definovaná kontrolní sekvence. Pak může v hranaté závorce být uveden počet parametrů (není-li závorka uvedena, je tento počet roven nule). Pak může následovat druhá hranatá závorka, která obsahuje výchozí hodnotu prvního parametru, který se tímto stává nepovinným. Pak teprve následuje obvyklé tělo makra uvnitř svorek. Při deklaraci nepovinného parametru je pak možné definované makro volat dvěma způsoby, buď jako `\makro parametry` nebo jako `\makro [parametr]parametry`. V prvém případě má `#1` defaultní hodnotu a ve druhém hodnotu parametr. Ostatní (povinné) parametry se v tomto případě čtou do `#2`, `#3` atd. Uff.

Pokud se obejdeme bez kontroly, zda je makro už definováno, a bez hvězdičkové verze, je možno `\newcommand` definovat takto:

```
\def\newcommand#1{\isnextchar[{\newcommandA#1}{\newcommandA#1[0]}}
\def\newcommandA#1[#2]{\edef\tmpp{\ifcase#2%
   \or1\or12\or123\or1234\or12345\or123456\or1234567\or12345678\or123456789\fi}%
   \edef\tmpp{\expandafter\addhashs\tmpp.}%
   \isnextchar[{\newcommandB#1}{\long\expandafter\def\expandafter#1\tmpp}%
}
\def\newcommandB#1[#2]{%
   \def#1{\isnextchar[{\runcommand#1}{\runcommand#1[#2]}}%
   \long\expandafter\def\csname\string#1X\expandafter\endcsname\tmpp
}
\def\addhashs#1{\ifx.#1\else #####1\expandafter\addhashs\fi}
\long\def\runcommand#1[#2]{\csname\string#1X\endcsname{#2}}
```

Předpokládejme `\newcommand\makro[4]{cosi}`. Makro `\newcommandA` má v `#2` počet parametrů a tento počet pomocí `\ifcase` a následného `\addhashs` převede v `\tmpp` na sekvenci

např. `#1#2#3#4` (pro případ čtyř parametrů). Když není deklarován nepovinný parametr, je definována přímo sekvence `\makro`. Je-li deklarován nepoviný parametr, je definována sekvence `\\makroX`. Kromě toho je definována sekvence `\makro` jako test, zda následuje hranatá závorka.

13.16 Testovací text Lorem ipsum dolor sit

0100
P. O.
15. 04. 2015

Občas se hodí pro testovací účely vyplnit sazbu nic neříkajícím textem. Takové texty bude generovat makro `\lipsum[číslo]` nebo `\lipsum[od-do]`. Například:

```
\lipsum[13]
\lipsum[3-27]
```

Uvedené číslo (nebo rozsah od do) určuje číslo odstavce (čísla odstavců), jež chceme vytisknout. V souboru `lipsum.sty` je připraveno 150 osdtavců s nic neříkajícím textem typu Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Je tedy třeba vybírat z rozsahu 1 až 150.

```
{\long\def\lipsumskip#1\newcommand\lipsum@i{\newcommand\lipsum@i}
 \def\lips@par{\lipsumpar}\let\lipsumpar=\relax
 \def\newcommand#1#{\advance\tmpnum by1 \sxdef{lips:\the\tmpnum}}
 \tmpnum=0
 \expandafter\lipsumskip\input lipsum.sty }
\def\lipsum[#1]{\lipsumA #1\empty-\empty\end}
\def\lipsumA #1-#2\empty#3\end{\tmpnum=#1 \edef\tmp{\ifx^#2^#1\else#2\fi}%
   \loop \csname lips:\the\tmpnum\endcsname
         \ifnum\tmpnum<\tmp \advance\tmpnum by1 \repeat
}
\let\lorem=\lipsum
\let\lipsumpar=\par
```

Makro využívá existující soubor `lipsum.sty` z LaTeXové distribuce, který obsahuje uvedené texty. Nejprve se přeskočí zbytečná makra v tomto souboru a pak se zahájí čtení vlastních textů za použití předefinovaného `\newcommand`.

13.17 Přeskakování textu podle čtenářů

0117
P. O.
22. 07. 2015

Chceme tisknout text podle toho, zda je určen pro konkrétní užití. Pomocí `\showallow{kdo}` umožníme čtenáři `kdo` tisknout text. Opakovaným `\showallow` umožníme tisknout text případně dalším čtenářům. Poté příkaz

```
\showif {admin,students} {text}
```

vytiskne `text` pouze tehdy, pokud aspoň jeden z vyjmenovaných čtenářů (v čárkami odděleném seznamu) byl deklarován pomocí `\showallow`. Makro `\showif` pracuje na úrovni expand procesoru: za dané podmínky expanduje na `{text}` jinak expanduje na prázdný výstup.

Druhá možnost je psát

```
\showifbegin {admins,students}
text
\showifend
```

To expanduje na text tentokrát bez uzavření do skupiny a také pouze tehdy, je-li aspoň jeden z vyjmenovaných čtenářů povolen. Přechody typu verbatim uvnitř textu jsou možné.

Implementace této vlastnosti mě napadla při poslechu příspěvku Borise Veytsmana „TeX and controlled access to information“ na TUG 2015 a vypadá takto:

```
\def\showif#1{\showifA#1,s:!,}
\def\showifbegin#1{\showifA#1,l:!,}
\def\showifA#1#2,{\expandafter\ifx\csname s:#1#2\endcsname\relax
   \expandafter\showifA \else \csname s:#1#2\expandafter\endcsname \fi}
\long\def\showifT#1:!,{\romannumeral-‘\.}
\long\expandafter\def\csname s:s:!\endcsname#1{}
\long\expandafter\def\csname s:l:!\endcsname#1\showifend{}
\def\showifend{}
```

```
\def\showallow#1{\expandafter\let\csname s:#1\endcsname=\showifT}
\def\showdeny#1{\expandafter\let\csname s:#1\endcsname=\relax}
```

0118 ## 13.18 Spojové seznamy

P. O.
22. 07. 2015

Jako cvičení vytvoříme v TeXu strukturu spojových senznamů. Makro `\addtolist{název}{data}` přidá na konec spojového seznamu `název` další uzel obsahující `data`, nebo (pokud je seznam prázdný) založí v seznamu první uzel. Každý uzel odkazuje na předchozí a následující uzel v seznamu. Makro `\printlist{název}` expanduje na seznam dat odělených `\seplist` v pořadí od prvního do posledního a makro `\printlistrev{název}` expanduje seznam v opačném pořadí. Tato makra proběhnou na úrovni expand procesoru.

```
\newcount\listnum

\def\addtolist#1#2{\advance\listnum by1
   \expandafter\ifx\csname l:#1:1\endcsname \relax
      \setlistnode \relax \relax {#2}{\the\listnum}%
      \expandafter\edef\csname l:#1:0\endcsname{\the\listnum}%
   \else
      \edef\lastnodenum{\csname l:#1:1\endcsname}%
      \expandafter\resetlistnode \csname l:\the\listnum\endcsname 2\lastnodenum
      \expandafter\setlistnode \csname l:\lastnodenum\endcsname \relax {#2}{\the\listnum}%
   \fi
   \expandafter \edef\csname l:#1:1\endcsname{\the\listnum}%
}
\def\setlistnode #1#2#3#4{\expandafter\def\csname l:#4\endcsname{#1#2{#3}}}
\def\resetlistnode #1#2#3{\def\tmp{\expandafter\def\csname l:#3\endcsname}%
   \expandafter\expandafter\expandafter\resetlistnumA \csname l:#3\endcsname #2#1}
\def\resetlistnumA #1#2#3#4#5{\ifcase #4\or\tmp{#5#2{#3}}\or\tmp{#1#5{#3}}\or\tmp{#1#2{#5}}\fi}

\def\printlist#1{\expandafter\ifx\csname l:#1:0\endcsname \relax \else
   \expandafter\expandafter\expandafter\printlistA
      \csname l:\csname l:#1:0\endcsname\expandafter\endcsname \fi}
\def\printlistA#1#2#3{#3\ifx#2\relax \else \listsep
   \expandafter\expandafter\expandafter\printlistA\expandafter#2\fi}

\def\printlistrev#1{\expandafter\ifx\csname l:#1:1\endcsname \relax \else
   \expandafter\expandafter\expandafter\printlistAr
      \csname l:\csname l:#1:1\endcsname\expandafter\endcsname \fi}
\def\printlistAr#1#2#3{#3\ifx#1\relax \else \listsep
   \expandafter\expandafter\expandafter\printlistAr\expandafter#1\fi}

\def\listsep{;}
```

Seznam `název` je reprezentován číslem prvního uzlu v makru `\l:název:0` a číslem posledního uzlu v `\l:název:1`. Každý uzel je makro s názvem `\l:číslo` a s obsahem `{\předchozí \další {data}}`, přitom `\předchozí` a `\další` jsou zase makra typu `\l:číslo` nebo to je `\relax`, pokud předchozí nebo další uzel neexistuje. Číslo je pro každý uzel unikátní. Pomocné makro `\setlistnode \předchozí\další{data}{číslo}` založí uzel `\l:číslo` s daným obsahem. Pomocné makro `\resetlistnode{co}{jak}{číslo}` pozmění uzel `číslo` tak, že `co` vloží jako první, druhý, nebo třetí údaj uzlu v závislosti na `jak`. Ostatní údaje uzlu zůstávají nezměněny. Takže třeba `\resetlistnode{data}3{číslo}` aktualízuje v `\l:číslo` jen data a ponechá pointery `\předchozí` a `\další` nezměněny.

0119 ## 13.19 Expanze seznamu v obráceném pořadí

P. O.
22. 07. 2015

Na expandování seznamu údajů v obráceném pořadí není nutné vytvářet spojové seznamy. Ty byly v předchozím OPmac triku uvedeny spíše jako řešení akademického problému. Napíšeme-li třeba

```
\revlist{aa,bb,ccc,dddd}
```

pak na výstupu dostaneme `dddd,ccc,bb,aa`. Makro `\revlist` pracuje jen na úrovni expand procesoru, pokud je implementováno třeba takto:

```
\def\revlist#1{\revlistA{}#1,,}
\def\revlistA#1#2,{\ifx,#2,\expandafter\revlistB\else \expandafter\revlistA\fi {#2,#1}}
\def\revlistB#1{\revlistC #1,}
\def\revlistC,#1,,{#1}
```

Je-li `n` počet údajů, které je třeba uvést v opačném pořadí, pak makro `\revlist` má kvadratickou složitost `n^2` zatímco projití spojového seznamu z předchozího OPmac triku pozpátku má složitost lineární. Na druhé straně k vytvoření spojového seznamu potřebujeme alokovat `n` řídicích sekvencí, zatímco při použití makra `\revlist` si vystačíme se čtyřmi sekvencemi. To je obvyklé dilema mezi rychlostí a paměťovými nároky.

13.20 Podtržítko se chová mimo matematický mód normálně

0156
P. O.
09. 06. 2016

Kategorie znaku `_` (podtržítko) je v plainTeXu nastavena na 8, aby jej bylo možno použít jako konstrutor indexů v matematické sazbě. Pak ale jeho užití mimo matematickou sazbu ohlásí chybu. Málokdo asi ví, že není nutno podtržítku dávat takovou katogorii. Když mu dáte třeba kategorii 12 (normální znak) pak se mimo matematickou sazbu bude chovat „normálně“, a v matematické sazbě zůstane jeho funkce konstruktoru indexu zachována. Vyzkoušejte:

```
\catcode`_=13 \let_=\sb \catcode`_=12

% test:
Tady je {\tt cosi_kdesi}. A v matematice stále funguje: $a_i^2$.
```

Pointa je v tom, že plainTeX nastavuje mathkód podtržítka na `"8000`, ale protože je defaultně nastavena jeho kategorie na 8, není tento mathhkód využit. Ten se uplatní jen u znaků kategorie 11 nebo 12. A tento speciální kód `"8000` říká: chovej se v matematickém módu, jakobys byl aktivní. V době krátkého aktivního života tohoto znaku jsme mu přidělili význam `\sb`, což je alternativa znaku s kategorií 8, tedy je to konstruktor indexů. Podtržítko bude v matematice tvořit indexy, jako obvykle.

Sice jsme tímto jediným řádkem osvobodili podtržítko od přítěže býti pouze konstruktorem indexů, ale některé obskurity zůstávají. Knuth s tímto řešením asi nepočítal a dal do svého neobvyklého kódování CM fontů na pozici podtržítka akcent tečky. Takže nebude přímo fungovat toto_zde, dokud nevyměníte Knuthovsky kódovaný font za normálně kódovaný font. Všimněte si, že v ukázce výše jsme pro tisk podtržítka použili font `\tt`, který jako jediný ze sady Computer Modern fontů respektuje ASCII, takže s tímto fontem podtržítko funguje.

Opustit Knutovsky kódované fonty můžete například přechodem na XeTeX, ve kterém zavedete OTF fonty v Unicode. Nebo v pdfTeXu lze přejít na kódování T1 (Cork), které také respektuje ASCII. Používáte-li csplain s UTF8 vstupem, stačí na začátek dokumentu psát:

```
v pdfTeXu:  \input t1code  % přechod na T1 kódování
            \input lmfonts % LM fonty (náhrada CM fontů) v T1 kódování

v XeTeXu:   \input ucode   % přechod na Unicode kódování
            \input lmfonts % LM fonty (náhrada CM fontů) v Unicode
```

13.21 Odstranění koncové mezery z parametru

0164
P. O.
29. 10. 2017

Někdy se stane, že přečteme parametr (například až po `\par`) a ten může a nemusí obsahovat koncovou mezeru. Vytvoříme expandující makro `\stripspace`, které případnou koncovou mezeru odstraní. Vnitřní mezery v parametru zůstávají zachovány. Příklad použití:

```
\def\neco#1\par{% nevíme, zda #1 obsahuje koncovou mezeru nebo ne
    \stripspace\necoA{#1}}
\def\necoA#1{% Nyní #1 určitě koncovou mezeru neobsahuje
    ... \message{"#1"}}
```

Makro `\stripspace` vypadá takto:

```
\def\stripspace#1#2{\stripspaceA#2\end/ \end/!#1{#2}}
\def\stripspaceA#1 \end/#2!#3#4{%
   \ifx!#2!\stripspaceB{#3{#4}}\else\stripspaceB{#3{#1}}\fi
}
\def\stripspaceB#1#2\fi{\fi#1}
```

Na konec parametru přidáme `\end/`■`\end/` a necháme jej přečíst až po ■`\end/`. Zbyde-li po čtení nějaký zbytek, pak parametr mezeru obsahoval a nezbyde-li nic, parametr mezeru neobsahoval.

0165 13.22 Variantní separátory parametrů

P. O.
29. 10. 2017

Může se stát, že jednotlivé parametry, které chceme postupně po jednom zpracovat, nejsou odděleny jediným separátorem, ale uživatel při psaní parametrů užívá více rozličných separátorů. Například kromě čárky ještě středník a dvojznak `--`. Napíše-li uživatel:

```
\makro {AAA,BBB;CCC--DDD,EEE}
```

pak toto chceme zpracovat jako

```
\makroX{AAA}{,}\makroX{BBB}{;}\makroX{CCC}{--}\makroX{DDD}{,}\makroX{EEE}{}
```

`\makroX` se tedy dozví v `#1` jednak oddělený parametr a v `#2` se dozví, jakým oddělovačem byl parametr oddělen.

Pro řešení tohoto problému použijeme makro `\replacestrings`, které každému oddělovači předsune znak `&`. Pak parametry přečteme po znak `&`, a za ním si přečteme případný oddělovač. Definice makra `\makro` z výše uvedeného příkladu může vypadat takto:

```
\def\makro#1{\def\tmpb{#1}%
   \replacestrings {,}  {&,}%
   \replacestrings {;}  {&;}%
   \replacestrings {--} {&{--}}%
   \expandafter\makroA\tmpb&{}%
}
\def\makroA#1&#2{\makroX{#1}{#2}\ifx&#2&\else\expandafter\makroA\fi}
```

Povšimněte si, že víceznakový separátor musí být zapouzdřen do svorek a oddělovač posledního parametru je prázdný.

0166 13.23 Čtení parametru po variantní separátor

P. O.
07. 04. 2018

Předchozí OPmac trik 0165 předpokládal, že nejprve skupinu parametrů celou načteme a pak je chceme oddělit od sebe podle separátorů. Jiný přístup je třeba zvolit, pokud chceme parametr načíst po proměnlivý separátor v textu, kde „nevíme co nás čeká“ a čtení chceme okamžitě zastavit, jakmile zjistíme přítomnost separátoru.

Napřílad `\memakro` má přečíst text až do konce odstavce nebo do výskytu `\enditems` podle toho, co nastane dříve. Pak můžeme použít `\vardelim{seznam separátorů}` těsně před `\def\memakro`, které definujeme jakoby s jedním neseparovaným parametrem:

```
\vardelim{\par\enditems}\def\memakro#1{\message{param:"#1"}}

užití: \memakro text text
       \enditems
```

V tomto příkladě bude v `#1 text text` a čtení je ukončeno výskytem `\enditems`. Pozor! Na rozdíl od klasického makra se separovaným parametrem ten výskyt separátoru `\enditems` není po provedení makra zničen, ale zůstává ve vstupní frontě a je pak znovu načten. Chceme-li jej zničit, můžeme definovat `\memakro` se dvěma parametry, tedy

```
\vardelim{\par\enditems}\def\memakro#1#2{\message{param:"#1", sep: "\noexpand#2"}}
```

Pak je v `#2` použitý separátor. Proměnlivý separátor v našem řešení musí být jediným tokenem. Chcete-li obecnější řešení, kdy proměnlivý separátor může sestávat z libovolného počtu tokenů, pak můžete použít mé řešení uvedené zde na TeX stackexchange

Implementace makra `\vardelim` vypadá takto:

```
\long\def\vardelim#1#2#3{%
   \def#3{\expandafter\vardelimA\csname var\string#3\endcsname{#1}}%
   \long\expandafter#2\csname var\string#3\endcsname
}
\long\def\vardelimA#1#2{\bgroup \let\bgroupT=\bgroup \let\bgroup=\relax \def\tmp{}%
   \def\vardelimF##1\vardelimN{\fi\expandafter\egroup\expandafter#1\expandafter{\tmp}}%
   \def\vardelimP{}%
   \vardelimB#2\vardelimB
}
\long\def\vardelimB#1{%
   \ifx\vardelimB#1\expandafter\vardelimC\else
      \addto\vardelimP{\vardelimH#1}\expandafter\vardelimB \fi
}
\def\vardelimC{\futurelet\next\vardelimD}
\def\vardelimD{%
   \ifx\next\bgroupT \vardelimG \fi
   \expandafter\ifx\space\next \vardelimS \fi
   \vardelimP
   \vardelimN
}
\long\def\vardelimG#1\vardelimN#2{\fi\addto\tmp{{#2}}\vardelimC}
\def\vardelimS#1\vardelimN{\fi\addto\tmp{ }\afterassignment\vardelimC\let\next= }
\long\def\vardelimN#1{\addto\tmp#1\vardelimC}
\long\def\vardelimH#1{\ifx\next#1\vardelimF\fi}
```

Makro čte parametr token po tokenu {s výjimkou výskytu skupiny v parametru, kterou přečte celou} a parametr kumuluje v `\tmp`. Nakonec spustí makro s argumentem `{expandované \tmp}`. Makro pracuje vždy tak, jakoby bylo s prefixem `\long`.

14 Struktura

14.1 LaTeXové značkování kapitol a sekcí

0036
P. O.
29. 01. 2014

Některé editory nabízejí inteligentní chování, rozpoznají-li v textu LaTeXové značkování. Například podle `\chapter`, `\section`, `\subsection` interpretují stromovou strukturu textu a umožňují jednotlivé části textu dle potřeby kliknutím otevírat a zavírat. Bohužel OPmac přišel 30 let po LaTeXu a editory si na něj ještě nezvykly. Nechcete-li zrovna řešit překonfigurování editoru, můžete si jednoduše základní LaTeXové značkování doplnit a v dokumentu ty značky používat. Například:

```
\def\bracedparam#1{\csname\string#1:M\endcsname}
\def\chapter{\bracedparam\chap}
\def\section{\bracedparam\sec}
\def\subsection{\bracedparam\secc}
```

Pokud někdo naopak upraví konfiguraci svého editoru tak, aby spolupracoval se značkováním podle OPmac, uvítám o tom informaci a rád na stránkách OPmac dám na takové řešení odkaz.

14.2 Řešení konfliktu jména \sec

0129
P. O.
05. 11. 2015

OPmac předefinovává plainTeXové makro `\sec`, kterému Knuth přidělil úkol vytisknout matematickou funkci secans. Makro je v OPmac vyhrazeno pro vyznačení nadpisu sekce. Tato likvidace původního významu makra byla autorem OPmac provedena záměrně, protože užití `\sec` ve významu secans je velmi ojedinělé, ne-li žádné. Pokud ale uživatel chce používat `\sec`

mimo matematický mód jako sekci a uvnitř matematického módu jako secans, může to zařídit třeba takto:

```
\let\section=\sec
\def\secans{\mathop{\rm sec}\nolimits}
\def\sec{\relax\ifmmode\secans \else \expandafter\section\fi}
\addprotect\sec
```

Povšimněme si, že je nutné deklarovat `\sec` pomocí `\addprotect`, aby i v obsahu správně fungovalo třeba toto:

```
\sec Význam funkce $\sec$
```

15 Jazyky

15.1 Texty ve více jazycích

0014 P. O. 17. 08. 2013

OPmac pracuje pouze s automaticky generovanými slovy Chapter/Kapitola/Kapitola, Table/Tabulka/Tabu a Figure/Obrázek/Obrázok. V dokumentaci je popsáno, jak se dají ta slova modifikovat nebo přidat další pomocí `\sdef`. Při tvorbě šablon CUstyle a CTUstyle jsem potřeboval připravit mnoho dalších slov a umožnit psát v angličtině, češtině nebo slovenštině. Používání `\sdef` při deklaraci slov by bylo moc zlouhavé, vytvořil jsem tedy zkratku `\mtdef`:

```
\def\slet#1#2{\expandafter\let\csname#1\expandafter\endcsname\csname#2\endcsname}
\def\mtdef#1#2#3#4{\sdef{mt:#1:en}{#2} \sdef{mt:#1:cz}{#3}
  \if$#4$\slet{mt:#1:sk}{mt:#1:cz}\else \sdef{mt:#1:sk}{#4}\fi}

\mtdef {abstract}     {Abstract}          {Abstrakt}       {}
\mtdef {author}       {Author}            {Autor}          {}
\mtdef {thanks}       {Acknowledgement}   {Poděkování}     {Poďakovanie}
\mtdef {declaration}  {Declaration}       {Prohlášení}     {Prehlásenie}
\mtdef {keywords}     {Keywords}          {Klíčová slova}  {Kľúčové slová}
\mtdef {title}        {Title}             {Název práce}    {Názov práce}
\mtdef {contents}     {Contents}          {Obsah}          {}
\mtdef {tables}       {Tables}            {Tabulky}        {Tabuľky}
\mtdef {figures}      {Figures}           {Obrázky}        {}
\mtdef {supervisor}   {Supervisor}        {Vedoucí práce}  {Vedúci práce}
\mtdef {supervisorD}  {Supervisor}        {Školitel}       {Školiteľ}
\mtdef {bibliography} {References}        {Literatura}     {Literatúra}
\mtdef {appendix}     {Appendix}          {Příloha}        {Príloha}
\mtdef {specifi}      {Specification}     {Zadání}         {Zadanie}

\mtdef {B} {Bachelor's thesis} {Bakalářská práce}  {Bakalárska práca}
\mtdef {M} {Master's thesis}   {Diplomová práce}   {Diplomová práca}
\mtdef {D} {Ph.D. thesis}      {Dizertační práce}  {Dizertačná práca}
```

Pokud údaj pro slovenštinu chybí, není to proto, že bych si nemohl vzpomenout, ale je to tím, že je použito stejné slovo jako v češtině.

V makru pak v místě, kde se má potřebné slovo objevit, píšeme `\mtext{abstract}`, `\mtext{author}`, `\mtext{B}` atd. Vypíše se odpovídající slovo podle předchozího nastavení vzorů dělení slov `\ehyph`, `\chyph` nebo `\shyph`.

15.2 Přidání dalšího jazyka

0049 P. O. 26. 04. 2014

Jakým způsobem se přidá další jazyk do csplainu je popsáno v souboru `hyphen.lan`. Předpokládejme, že je přidána němčina `\delang` a polština `\pllang`. Je pak možné vytvořit analogický příkaz k výše popsanému příkazu `\mtdef` a pomocí něj zavést stejné fráze do dalších jazyků:

```
\sdef{lan:21}{de}  \sdef{lan:121}{de}
\sdef{lan:23}{pl}  \sdef{lan:123}{pl}
\def\mtdefx#1#2#3{\sdef{mt:#1:de}{#2}\sdef{mt:#1:pl}{#3}}

             % German              % Polish
\mtdefx {D}  {Ph.D. Dissertation}  {Praca doktorska}
...
```

Když nyní zapneme vzory dělení třeba do polštiny (`\pllang`), vytiskne příkaz `\mtext{D}` frázi Praca doktorska.

15.3 Uppercase německého ß

0083
P. O.
20. 12. 2014

UTF-8 znak ß je v csplainu kódován jako `\ss` za pomocí encTeXu. Chceme-li tisknout německé texty konvertované automaticky na velká písmena, má se tento znak proměnit v SS. Toho lze dosáhnout třeba takto:

```
\def\Uppercase#1{\begingroup
   \def\ss{SS}\uppercase{\edef\tmp{#1}}%
   \expandafter\endgroup\tmp
}

\Uppercase{Mainstraße}
```

15.4 Konverze frází z \mtext na velká písmena

0091
P. O.
25. 01. 2015

Makro `\mtext{id}` expanduje na skutečnou frázi ve třech krocích:

```
\mtext{id} -> \csname mt:id:\csname lan:\the\language\endcsname\endcsname
\csname mt:id:\csname lan:\the\language\endcsname\endcsname -> \mt:id:cs
\mt:id:cs -> skutečný text
```

Chceme-li konvertovat takový text na velká písmena, musíme nejprve provést všechny tři úrovně expanze, na což potřebujeme sedm příkazů `\expandafter`:

```
\def\exseven{\expandafter\expandafter\expandafter
             \expandafter\expandafter\expandafter\expandafter}
...\uppercase\exseven{\mtext{id}}...
```

15.5 Hebrejština – sazba zprava doleva

0092
P. O.
26. 01. 2015

Sazbu zprava doleva zvládá i pdfTeX vybavený eTeXem (toto rozšíření se aktivuje při inicializaci formátu a pdfcsplain je s ním obvykle inicializován), protože eTeX obsahuje modul TeXXeT. Tento modul začne pracovat po zadání `\TeXXeTstate=1` na začátku dokumentu a pak akceptuje příkazy `\beginR...\endR`, mezi kterými probíhá sazba (v horizontálním módu) zprava doleva.

Připravíme makro `\hebrew{...text v hejštině...}` a `\hebrewpar{... několik odstavců v hebrejš` které vytvoří sazbu krátkých hebrejských textů. V pdfTeXu použijeme 8bitový font `rcjhbltx.tfm` (běžně dostupný v TeXových distribucích) a překódování z UTF-8 vstupu na tento font provedeme encTeXem.

Poznámka: v textovém editoru, který umí správně pracovat s UNICODE znaky hebrejštiny, píšete text v pořadí znaků, jak se čte, nicméně v okně editoru se automaticky sekvence znaků zobrazuje „pozpátku“, protože editor zná úseky znaků UNICODE tabulky, které má zobrazovat tímto způsobem. Do souboru je ale sekvence znaků uložena v pořadí, jak se čte. Proto je nutné toto pořadí znovu převrátit v sazbě příkazy `\beginR` a `\endR`.

```
\TeXXeTstate=1
\font\hebrewfont=rcjhbltx
\def\texthebrew#1{\leavevmode\beginR{\hebrewfont#1}\endR}
\long\def\hebrewpar#1{\par\hbox{\beginR\vbox{\hebrewfont#1}\endR}}

%      sekvence            UTF-8 kód                                  kód ve fontu
```

```
\mubyte\HEBalef              ^^d7^^90\endmubyte      \chardef\HEBalef              39
\mubyte\HEBbet               ^^d7^^91\endmubyte      \cahrdef\HEBbet               98
...
\mubyte\HEBshin              ^^d7^^a9\endmubyte      \chardef\HEBshin             152
\mubyte\HEBshinshindotdages ^^ef^^ac^^ab\endmubyte  \chardef\HEBshinshindotdages 153
...
```

Kód znaku ve fontu zjistíte pohledem do souboru `cjhebltx.enc`, který je součástí TeXových distribucí. Pokud se vám nechce zjišťovat UTF-8 kódy všech znaků a jste schopni v editoru přímo tyto znaky napsat, můžete kódování fontu deklarovat i přímo pomocí těchto znaků (v ukázce je místo skutečného znaku jen zkratka heb-znak, protože skutečný znak na této WWW stránce zobrazit nelze).

```
\mubytein=0
%      sekvence          skutečný znak                              kód ve fontu
\mubyte\HEBalef             heb-znak\endmubyte  \chardef\HEBalef              39
\mubyte\HEBbet              heb-znak\endmubyte  \cahrdef\HEBbet               98
...
\mubyte\HEBshin             heb-znak\endmubyte  \chardef\HEBshin             152
\mubyte\HEBshinshindotdages heb-znak\endmubyte  \chardef\HEBshinshindotdages 153
...
\mubytein=1
```

Je-li někdo ochoten zaslat mi hotovou tabulku s hebrejskými znaky a soubor s ukázkou sazby hebrejského textu, rád sem zařadím odkaz.

Podobně (ale bez pravolevé sazby) je v souboru `cyrchars.tex` řešena sazba promocí encTeXu azbukou. Soubor `cyrchars.tex` je součástí csplainu a přímo v něm je dokumentace a ukázky.

16 Matematická sazba

16.1 České texty v matematice

0025 *P. O.* 02. 09. 2013

V matematice nefungují akcentované znaky (č, á, ř atd.), protože nemají třídu 7 a nejsou tedy zařazeny do matematické abecedy. Ve zlomku `$cena \over výkon$` chceme, aby byly texty zmenšeny (nelze je dát jednoduše do `\hbox`u) a navíc chceme, aby byly vytištěny správně. Rychlé řešení `$\rm cena\over výkon$` funguje při použití CSfontů, ale po přepnutí na jinou rodinu fontů nastávají potíže. Ty lze vyřešit zařazením české abecedy do znaků třídy 7 (matematická abeceda) takto:

```
\def\setmathalphabetcode#1{\ifx\XeTeXmathcode\undefined
     \tmpnum=\itfam \multiply\tmpnum by256 \advance\tmpnum by‘#1
     \advance\tmpnum by"7000 \mathcode‘#1 = \tmpnum \relax
   \else \XeTeXmathcode‘#1 = 7 \itfam ‘#1 \fi
}
\def\mathalphabetchars#1{\ifˆ#1ˆ\else
   \setmathalphabetcode#1\expandafter\mathalphabetchars\fi}

\mathalphabetchars ÁáÄäČčĎdťÉéĚěÍíĹĺĽľŇňÓóÖöŐőŔŕŘřŠšŤťÚúŮůÜüÝýŽž{}
```

Nyní mají znaky české abecedy stejnou vlastnost, jako jiné znaky matematické abecedy a jsou implicitní v kurzívě a dají se přepínat. Je ovšem nutné, aby v dané matematické rodině byl zaveden font, který tyto znaky obsahuje (což nemusí být vždy splněno).

Makro `\setmathalphabetcode` přidělí znaku třídu 7 s výchozí rodinou `\itfam`. Není-li přitomna 16bitová mašina, použije se k tomu primitiv `\mathcode`, jinak se použije primitiv `\XeTeXmathcode`.

16.2 Odkaz na předchozí rovnici

0028
P. O.
06. 09. 2013

V matematickém textu s číslovanými rovnicemi často odkazujeme na tu poslední nebo předposlední rovnici. Je pak možná zbytečné pro ně vymýšlet lejblíky a odkazovat na ně pomocí `\ref[lejblík]`. Stačí napsat `\eqmark` bez lejblíku a odkazovat pomocí `\lasteq` (poslední rovnice), `\preveq2` (předpolední rovnice), `\preveq3` (před-předposlední rovnice) atd. Takže třeba:

```
$$a^2 + b^2 = c^2  \eqmark$$
Předchozí rovnice \lasteq\ je tvrzením Pythagorovy věty.
```

Makra `\lasteq` a `\preveq` je možné definovat takto:

```
\newcount\eqlabnum
\addto\eqmark{{\global\advance\eqlabnum by1
   \label[.eqlab:\the\eqlabnum]\wlabel\thednum}}
\def\preveq#1{\tmpnum=\eqlabnum\advance\tmpnum by-#1\advance\tmpnum by1
   \ref[.eqlab:\the\tmpnum]}
\def\lasteq{\preveq1}
```

Je zde zaveden globální registr `\eqlabnum`, pomocí kterého každé rovnici označené makrem `\eqmark` se přidělí interní lejblík `.eqlab:\the\eqlabnum`, přičemž číslo `\eqlabnum` se vždy zvětší o jedničku.

16.3 Natahovací tilde nad vzorcem jakékoli velikosti

0070
P. O.
17. 06. 2014

PlainTeX používá makro `\widetilde`, které odkazuje do fontu se třemi postupně se zvětšujícími velikostmi vlnky a dál není nic. To naprosto neuspokojí pokrývače, kteří chtějí pokrýt vzorec libovolné šířky vlnkou. Navrhneme tedy makro `\overtilde`, které změří pokrývaný vzorec a pomocí `\pdfscale` zvětší či zmenší největší element ve fontu pro `\widetilde` tak, že výsledek je vždy celý pokrytý. Například:

```
$$\displaylines{
  \overtilde{b} + \overtilde{ab} + \overtilde{a+bc}+{}\cr
  {}+\overtilde{b+c+d}+ \overtilde{a+b+c+df}
}$$
```

dá tento výsledek:

$$\widetilde{b} + \widetilde{ab} + \widetilde{a + bc} + \\ + \widetilde{b + c + d} + \widetilde{a + b + c + df}$$

Makro `\overtilde` je definováno takto:

```
\mathchardef\widetildemax="0367

\def\widetildeto #1{\bgroup\tmpdim=#1\setbox0=\hbox{$\widetildemax$}%
   \tmpdim=16\tmpdim \tmpnum=\tmpdim \tmpdim=\wd0 \divide\tmpdim by16
   \divide\tmpnum by\tmpdim
   \hbox to#1{\pdfsave\rlap{\pdfscale{\the\tmpnum}{\ifnum\tmpnum<588 1\else\the\tmpnum\fi}%
                            \pdfscale{.00390625}{\ifnum\tmpnum<588 1\else.0017\fi}%
                            \vbox to0pt{\hbox{$\widetildemax$}\vss}}\pdfrestore\hss}%
   \egroup
}
\def\overtilde#1{\setbox1=\hbox{$#1$}%
  \vbox{\offinterlineskip \halign{\hfil##\hfil\cr
        \widetildeto{\wd1}\cr\noalign{\kern.5ex\kern.02\wd1}\box1\cr}}%
}
```

Makro `\widetildeto` zvětší `\widetildemax` na požadovanou velikost `#1`. Čitatel pro výpočet poměru velikostí vynásobí `16`, jmenovatel vydělí `16`, takže výsledený poměr uložený v `\tmpnum` je `256` krát skutečný poměr. Proto jej do `\pdfscale` vložíme a následně vložíme `\pdfscale .00390625`, což je `1/256`. Výšku znaku necháme nezměněnu až do poměru `588/256`. Je-li poměr větší, je mírně zvětšena i výška znaku poměrem `0.4352` krát původní výška. Makro `\overtilde` změří pokrývaný vzorec v `\boxu1`, zavolá `\widetildeto` a sestaví vlnku a vzorec pod sebe pomocí `\halign`.

16.4 Box s textem ve velikosti podle kontextu (index, indexindex)

0078 P. O. 04. 09. 2014

V LaTeXu je k dispozici makro `\text{cosi}`, které v matematickém módu vysází `cosi` stejně jako `\hbox{cosi}`, ale velikost textu je přizpůsobena tomu, zda se příkaz vyskytuje v základu, indexu nebo indexindexu. Navíc si příkaz pamatuje vnější matematický kontext a když uvnitř textu napíšeme `$...$`, bude sazba ve stejném kontextu (displaystyle, textstyle).

Vytvoříme analogické makro `\mathbox{text}`, které se chová výše popsaným způsobem. Při použití OPmac k tomu potřebujeme tři řádky kódu:

```
\def\mathbox#1{{\mathchoice{\mathboxA\displaystyle[]{#1}}{\mathboxA\textstyle[]{#1}}
   {\mathboxA\textstyle[700]{#1}}{\mathboxA\textstyle[500]{#1}}}}
\def\mathboxA#1[#2]#3{\hbox{\everymath={#1}\if^#2^\else\typoscale[#2/]\relax\fi #3}}
```

16.5 Opakování symbolu na zlomu řádku

0148 P. O. 18. 05. 2016

V tradiční české sazbě (i jinde) se vyskytuje požadavek na opakování znaménka pro operaci či realaci, pokud je v tomto místě zlomen řádek. Připravíme makro `\mrepeatchar` znak, který předeklaruje znak tak, aby byl (pouze) v matematické sazbě aktivní a způsobil své opakování na zlomu řádku. Také připravíme makro `\mrepeatcs \sekvenceA \sekvenceB \sekvenceC ... \relax`, které předeklaruje význam uvedených sekvencí tak, že jako operace či relace se opakují na zlomu řádku. Například:

```
\mrepeatchar+ \mrepeatchar- \mrepeatchar= \mrepeatchar< \mrepeatchar>
\mrepeatcs \approx \asymp \bot \cap \cdot \circ \cup \diamond
   \div \equiv \geq \gg \in \leq \ll \odot \oplus \oslash \otimes \paralell
   \perp \pm \prec \peceq \sim \simeq \subset \subseteq \supset \supseteq
   \top \triangle \triangleleft \triangleright \uplus \vdash \vee \wedge \relax

\hsize=5cm
Test $a\sim b\sim c\sim d\sim e\sim f\sim g\sim i\sim k\sim l\sim m$.

A taky $a+b+c+d+e+f+g+h+y+f<e<w<e$.
```

Po deklarování znaku (např. `\mrepeatchar+`) je možné použít `\NR+`, což se chová v matematické sazbě jako původní `+`. V textové sazbě se znak `+` nemění. Po deklarování sekvence (např. `\mrepeatcs \sim \relax`) je k dispozici sekvence `\NRsim`, která nese původní význam sekvence `\sim`.

Implementace může vypadat takto:

```
\def\NR#1{\csname NR:\string#1\endcsname}
\def\mrepset#1{\begingroup \lccode`\~=`#1\lowercase{\endgroup\edef~}}
\def\mrepeatchar#1{%
   \isNRno\NR#1{\mathchardef\NR#1=\mathcode`#1 }%
   \mrepset#1{\NR#1\nobreak \discretionary{}{\hbox{$\NR#1$}}{}}
   \mathcode`#1="8000
}
\def\NRs#1{\csname NR\expandafter\NRx\string#1\endcsname} \def\NRx#1{}
\def\mrepeatcs#1{\ifx#1\relax \else
   \isNRno\NRs#1{\let\NRs#1=#1}%
   \edef#1{\NRs#1\noexpand\nobreak \discretionary{}{\hbox{$\NRs#1$}}{}}%
   \expandafter\mrepeatcs\fi
}
```

```
\def\isNRno#1#2#3{\expandafter\expandafter\expandafter\ifx#1#2\relax
   \expandafter\expandafter\expandafter#3\else
   \message{\string#2\space declared already}\fi}
```

Řešení je robustní, chytré znaky můžete bez obav použít v nadpisech kapitol a sekcí.

16.6 Inteligentní \dots jako v AMSTeXu

0084
P. O.
14. 12. 2014

AMSTeX nabízí některá zajímavá makra. Můžete si natáhnout `\input amstex` před `\input opmac` a tím tato makra využít. Je však třeba těsně za `\input amstex` napsat `\catcode‘\@=12`, protože AMSTeX nastavuje tento znak jako aktivní a to nedělá dobrotu.

Nebo si zajímavá makra naprogramujeme sami. Například makro `\dots` pracuje v AMSTeXu podle kontextu. V textovém módu se chová jako klasické `\dots`, v matematickém módu se chová jako `\cdots` nebo `\ldots` v závislosti na tom, čím je obklopeno. Tedy:

```
$$
  a_1,\dots a_n, \quad           % chová se jako \ldots
  a_1 + \dots + a_n, \quad       % chová se jako \cdots
  A_1 \subset \dots \subset A_n  % chová se jako \cdots
$$
```

Takovou inteligenci je možné dát makru `\dots` pomocí následujícího kódu:

```
\let\textdots=\dots
\def\dots{\ifmmode \expandafter\mathdots \else \expandafter\textdots \fi}
\def\mathdots{\futurelet\next\mathdotsA}
\def\mathdotsA{\mathdotsB +-=<>()[]\{\}\langle\rangle \end
   \edef\tmpb{!\meaning\next}%
   \expandafter\isinlist\expandafter\tmpb\expandafter{\expandafter!\string\mathchar"}%
   \iftrue \expandafter\mathdotsC\tmpb \end \else \ldots \fi
   \relax
}
\def\mathdotsB#1{\ifx\end#1\else
      \ifx\next#1\cdots \expandafter\expandafter\expandafter\skiptorelax
      \else \expandafter\expandafter\expandafter\mathdotsB
   \fi\fi
}
\def\mathdotsC#1"#2#3\end{\let\next=\ldots
   \ifnum#2=1 \let\next=\cdots \fi % Big OP
   \ifnum#2=2 \let\next=\cdots \fi % Bin
   \ifnum#2=3 \let\next=\cdots \fi % Rel
   \ifnum#2=4 \let\next=\cdots \fi % Open
   \ifnum#2=5 \let\next=\cdots \fi % Close
   \next
}
\addprotect\dots
```

Makro `\mathdots` spuštěné v matematickém módu vloží do `\next` následující token a za pomocí `\mathdotsA` a `\mathdotsB` zjišťuje, zda to je některý ze znaků `+-=§tt§char60§tt§char62()[]{}`. Pokud ano, vloží `\cdots` a pomocí `\skiptorelax` končí. Jinak ještě prozkoumá, zda `\next` je `\mathchar` (např. `\le`). Pokud ne, vloží `\ldots`. Pokud ano, zjistí, pomocí `\mathdotsC`, zda tento `\mathchar` je třídy 1, 2, 3, 4 nebo 5. Pokud ano, vloží `\cdots`, pokud ne, vloží `\ldots`.

16.7 \dddot pro třetí derivaci

0145
P. O.
27. 04. 2016

Plain TeX nabízí nejvýše `\ddot x` pro druhou derivaci `x` dle času a toto makro je implementováno pomocí `\mathaccent`. Akcent pro trojtečku v matematických fontech nemáme, takže makro `\dddot x` pro třetí derivaci bude trochu složitější:

```
\def\dddot#1{{\mathpalette\dddotA{#1}}}
\def\dddotA#1#2{\setbox0=\hbox{$#1#2$}\tmpdim=\ht0 \mathop{#2\kern0pt}\limits
   ^{\vbox to0pt{\kern-.04em\hbox to0pt{\hss\it$#1.\mkern-1.5mu.\mkern-1.5mu.$%
     \kern-\slantcorr\hss}\vss}}}
```

Makro díky `\mathpalette` pracuje správně i ve skriptoidní či skriptskriptoidní velikosti. Také díky makru `\slantcorr` z OPmac je umístění nad znakem s ohledem na skloněnou osu základního znaku. Konstanty v makru jsou voleny tak, aby to vizuálně co nejlépe navazovalo na `\ddot` z CM fontů. Můžete porovnat výstup `\dddot` z LaTeXu při použití `amstex.sty` s tímto řešením.

0144 P. O. 11. 04. 2016

16.8 Řecká písmena jako \bbchar

OPmac používá makro soubor ams-math, který pro písmena zdvojených tahů zavádí font `msbm*.tfm` z rodiny fontů AMS. Taková písmena vypadají daleko lépe než písmena z fontu `bbold*.tfm`. Fonty msbm mají ale jednu nevýhodu: neobsahují zdvojená řecká písmena. Pokud je chcete, naučíte se v tomto OPmac triku vyměnit fonty jedné rodiny za jinou. Pak můžete psát:

```
$\alpha = {\bbchar\alpha}$
```

a objeví se „alfa je rovno dvojitému alfa“. Pohledem do souboru `ams-math.tex` zjistíte, že rodina pro `\bbchar` má číslo 5 a označení `msbm`, takže bude stačit udělat toto:

```
\regtfm msbm 0 bbold5 5.5 bbold6 6.5 bbold7 7.5 bbold8 8.5 bbold9
             9.5 bbold10 11.1 bbold12 15 bbold17 * % using bbchar from bbold*.tfm

\mathchardef\bbalpha="50B   \mathchardef\bbbeta="50C    \mathchardef\bbgamma="50D
\mathchardef\bbdelta="50E   \mathchardef\bbepsilon="50F \mathchardef\bbzeta="510
\mathchardef\bbeta="511     \mathchardef\bbtheta="512   \mathchardef\bbiota="513
\mathchardef\bbkappa="514   \mathchardef\bblambda="515  \mathchardef\bbmu="516
\mathchardef\bbnu="517      \mathchardef\bbxi="518      \mathchardef\bbpi="519
\mathchardef\bbrho="51A     \mathchardef\bbsigma="51B   \mathchardef\bbtau="51C
\mathchardef\bbupsilon="51D \mathchardef\bbphi="51E     \mathchardef\bbchi="50F
\mathchardef\bbpsi="520     \mathchardef\bbomega="57F

\addto\bbchar{\let\alpha\bbalpha \let\beta\bbbeta \let\gamma\bbgamma
   \let\delta\bbdelda \let\epsilon\bbepsilon \let\zeta\bbzeta \let\theta\bbtheta
   \let\iota\bbiota \let\kappa\bbkappa \let\lambda\bblambda \let\mu\bbmu
   \let\nu\bbnu \let\xi\bbxi \let\pi\bbpi \let\rho\bbrho \let\sigma\bbsigma
   \let\tau\bbtau \let\upsilon\bbupsilon \let\phi\bbphi \let\chi\bbchi
   \let\psi\bbpsi \let\omega\bbomega
}
```

0130 P. O. 11. 11. 2015

16.9 Nekurzivní řecká písmena

V matematické sazbě je `$\beta$` v kurzívě bez ohledu na to, zda napíšeme třeba `$\rm\beta$`. Toto chování je možné změnit po deklaraci `\smartgreek`. Pak se přizpůsobí fontu `\rm` i malá řecká písmena. Makro `\smartgreek` vypadá takto:

```
\def\smartgreek{%
   \expandafter\ifx\csname updelta\endcsname\relax \eurmgreek \fi
   \escapechar=-1 \setsmartgreekA
      \alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\pi\rho\sigma
      \tau\upsilon\phi\chi\omega\varepsilon\vartheta\varpi\varrho\varsigma\varphi\relax
   \escapechar=`\\
}
\def\setsmartgreekA#1{\ifx#1\relax \else
   \expandafter \let \csname ori\string#1\endcsname = #1%
   \edef#1{\relax \noexpand\ifnum\fam=0 \thecsname up\string#1\endcsname
          \noexpand\else \noexpand\ifnum\fam=\rmfam \thecsname up\string#1\endcsname
          \noexpand\else \thecsname ori\string#1\endcsname
          \noexpand\fi \noexpand\fi}%
   \expandafter\setsmartgreekA \fi
}
\def\thecsname{\expandafter\noexpand\csname}
\def\eurmgreek{\regtfm eurm 0 eurm5 6 eurm7 8.5 eurm10 *
```

```
   \csname newfam\endcsname \eurmfam \tmpnum=\eurmfam \advance\tmpnum by-10
   \edef\eurmh{\ifcase\tmpnum A\or B\or C\or D\or E\or F\fi}%
   \addto\normalmath{\loadmathfamily {\eurmfam} eurm }\normalmath
   \addto\boldmath{\loadmathfamily {\eurmfam} eurm }%
   \mathchardef \upalpha "0\eurmh 0B    \mathchardef \upbeta  "0\eurmh 0C
   \mathchardef \upgamma "0\eurmh 0D    \mathchardef \updelta "0\eurmh 0E
   \mathchardef \upepsilon "0\eurmh 0F  \mathchardef \upzeta  "0\eurmh 10
   \mathchardef \upeta   "0\eurmh 11    \mathchardef \uptheta "0\eurmh 12
   \mathchardef \upiota  "0\eurmh 13    \mathchardef \upkappa "0\eurmh 14
   \mathchardef \uplambda "0\eurmh 15   \mathchardef \upmu    "0\eurmh 16
   \mathchardef \upnu   "0\eurmh 17     \mathchardef \upxi   "0\eurmh 18
   \mathchardef \uppi   "0\eurmh 19     \mathchardef \uprho   "0\eurmh 1A
   \mathchardef \upsigma   "0\eurmh 1B  \mathchardef \uptau   "0\eurmh 1C
   \mathchardef \upupsilon "0\eurmh 1D  \mathchardef \upphi   "0\eurmh 1E
   \mathchardef \upchi   "0\eurmh 1F    \mathchardef \uppsi   "0\eurmh 20
   \mathchardef \upomega   "0\eurmh 21  \mathchardef \upvarepsilon "0\eurmh 22
   \mathchardef \upvartheta "0\eurmh 23 \mathchardef \upvarpi   "0\eurmh 24
   \let \upvarrho=\varrho \let \upvarsigma=\varsigma
   \mathchardef \upvarphi   "0\eurmh 27
}
```

Je-li natažen tx-math, využijí se znaky `\upalpha`, `\upbeta` atd. odtud, jinak je potřeba znaky deklarovat z fontu eurm, což provede makro `\eurmgreek`. Makro `\setsmartgreek` předefinuje znaky `\alpha`, `\beta`, atd. takto:

```
\let\orialpha=\alpha
\def\alpha{\relax \ifnum\fam=0 \upalpha \else \ifnum\fam=\rmfam \upalpha \else \orialpha \fi\fi}
```

16.10 Normální \bf a \bi v matematickém módu

0146
P. O.
02. 05. 2016

OPmac volá `ams-math.tex` pro zavedení matematických fontů. Tento soubor (z CSplainu) nastavuje pro `\bf` a `\bi` v matematickém módu *bezserifové* varianty písma, protože to je pro značení matic a vektorů v české sazbě obvyklejší. Ovšem někdo může chtít i matematické sazbě použít normální serifové varinaty písma pro `\bf` a `\bi`. Pak může psát:

```
\addto\normalmath{%
   \setmathfamily {\bffam} \tenbf  \setmathfamily {\bifam} \tenbi
}
\addto\boldmath{%
   \setmathfamily {\bffam} \tenbf  \setmathfamily {\bifam} \tenbi
}
\normalmath
```

Typicky nejsou pro tučnou verzi matematické sazby k dispozici supertučné varianty písma, takže jsme se zde museli spokojit v `\boldmath` se stejným řešením jako v `\normalmath`. Používáte-li písmovou rodinu, která disponuje supertučnými písmy, pak můžete napsat něco podobného tomuto:

```
\addto\boldmath{%
   \loadmathfamily {\bffam} heavy-bf \setmathfamily {\bifam} heavy-bi
}
```

16.11 Vektory včetně řeckých znaků tučně bez serifů

0170
P. O.
20. 10. 2019

Jak bylo řečeno v předchozím OPmac triku, OPmac používá pro `\bf` a `\bi` v matematické sazbě tučný bezserifový font, protože je to v české sazbě vektorů a matic obvyklejší. Někdy bychom chtěli matice mít stojatě a vektory skloněné, přitom obojí označovat např. pomocí `\_A\_x = \_b`. Dále bychom chtěli po napsání `\_\gamma` dostat tučný bezseriový znak gamma, což implicitně `\bf` ani `\bi` neumějí. Postará se o to následující makro:

```
\addto\normalmath{\loadmathfamily 12 cmbrmb10 \loadmathfamily 13 cmssbx10 }
\addto\boldmath{\loadmathfamily 12 cmbrmb10 \loadmathfamily 13 cmssbx10 }
```

```
\normalmath

\def\_#1{{\expandafter\vecboldify\string#1\end}}
\def\vecboldify#1#2\end{%
   \ifx\relax#2\relax
      \ifnum\lccode`#1=`#1\bi#1\else\bf#1\fi % skolněná malá, stojatá velká
   \else
      \expandafter\expandafter\expandafter\vecboldifyG % tučný sans pro řečtinu
      \expandafter\meaning \csname #2\endcsname\end
   \fi
}
\def\vecboldifyG #1"#2#3\end{\ifx7#2\fam13\mathchar"7#3 \else \mathchar"C#3 \fi}
```

Nejprve je zaveden font cmbrmb10 do matematické rodiny 12. Ten obsahuje bezserifovou tučnou řečtinu, bohužel i velká písmena této řečtiny jsou skolněná. Takže je zaveden ještě font cmssbx10 do rodiny 13 obsahující bezserifové tučné verzálky řeckých písmen (ale zase bohužel nebsahuje řecké mínusky).

Makro pomocí `\string` ověří, zda následuje písmeno nebo řídicí sekvence. V prvém případě pak pomocí `\lccode` rozhodne, zda použije `\bf` nebo `\bi`. Ve druhém případě pomocí `\meaning` zjistí mathcode daného znaku. Začíná-li sedmičkou, jedná se o velká řecká písmena a použije se rodina 13. Jinak se použije rodina C=12.

0131

16.12 Matematické rodiny dynamicky přidělené

P. O.
22. 11. 2015

Klasický TeX i pdfTeX mají omezení na nejvýše 16 matematických rodin fontů v jednom vzorečku. OPmac pomocí `ams-math.tex` nebo `tx-math.tex` alokuje staticky (pro celý dokument) 12 resp. 14 rodin. Můžete jich staticky alokovat méně a dynamicky doplnit jen potřebné rodiny v rámci každého jednotlivého vzorečku. Pak můžete mít v dokumentu libovolné množství matematických rodin, jen omezení 16 rodin na vzoreček pochopitelně zůstává.

Vytvoříme makro `\dfam{jméno-rodiny}`, které se uvnitř vzorečku chová jako `\fam=číslo`, ale rodinu alokuje dynamicky. Dále vytvoříme makro

```
\dmathchardef\sekvence třída{jméno-rodiny}kód
```

které deklaruje `\sekvenci` podobně jako `\matchardef`, ale odpovídající rodina se automaticky alokuje jen, pokud je `\sekvence` ve vzorečku použita. Pro příklad předpokládejme, že není staticky zavedena rodina eufm pro frakturu a rsfs pro skript. Pak je možné na začátku dokumentu deklarovat:

```
% obecně: \sdef{dfam:jméno-rodiny}{jméno-fontu-jako-v-\loadmathfamily}
\sdef{dfam:fractur}{eufm}   \def\fractur{\dfam{fractur}}
\sdef{dfam:script}{rsfs}    \def\script{\dfam{script}}
% ...
\dmathchardef\cosi 0{fractur}00  % znak třídy 0 kódu 00 (hex) z rodiny fractur
% ...
```

a následně to použít:

```
$a \times {\script B} = {\fractur C}$, a taky $ \beta = \cosi^2 $.
```

V ukázce jsou dva vzorečky. Vedle staticky zavedených rodin jsou v prvním vzorečku navíc dynamicky alokovány rodiny rsfs a eufm a ve druhém jen rodina eufm.

Implementace je následující

```
\chardef\numfamilies=\count18  % počet staticky linkovaných rodin
\everymath={\dfamstart} \everydisplay{\dfamstart} \def\dfamlist{}
\def\dfamstart{\aftergroup\dfamreset \let\dfamstart=\relax}
\def\dfamreset{\global\count18=\numfamilies \gdef\dfamlist{}}

\def\dfam#1{\relax
   \expandafter\ifx\csname dfam:#1\endcsname \relax
```

```
      \opwarning{dynamic math family "#1" is not declared}%
   \else
      \isinlist\dfamlist{,#1,}\iftrue \else
         \begingroup \def\wlog##1{}%
            \csname newfam\expandafter\endcsname \csname dmn:#1\endcsname
            \globaldefs=1
            \loadmathfamily{\csname dmn:#1\endcsname} {\csname dfam:#1\endcsname}
         \endgroup
         \global\addto\dfamlist{,#1,}%
         \tmpnum=\csname dmn:#1\endcsname
         \ifnum \tmpnum<10 \sxdef{dmh:#1}{\the\tmpnum}\else
            \advance\tmpnum by-10
            \sxdef{dmh:#1}{\ifcase\tmpnum A\or B\or C\or D\or E\or F\fi}%
      \fi\fi
      \fam=\csname dmn:#1\endcsname\relax
   \fi
}
\def\dmathchar#1#2#3#4{\relax
   \ifnum#1=0{\fi % aby fungovalo 2^\cosi bez nutnosti psát 2^{\cosi}
   \isinlist\dfamlist{,#2,}\iftrue\else\begingroup\dfam{#2}\endgroup\fi
   \mathchar"#1\csname dmh:#2\endcsname#3#4
   \ifnum#1=0}\fi
}
\def\dmathchardef #1#2#3#4#5{\def#1{\dmathchar#2{#3}#4#5}}
\addprotect\dfam \addprotect\dmathchar
```

Jednotlivé rodiny se zavádějí pomocí `\loadmathfamily` až podle potřeby a globálně. Jména zavedených rodin se ukládají do `\dfamlist`. Sekvence `\dmn:jméno` obsahuje číslo rodiny a `\dmh:jméno` obsahuje totéž číslo jako hexa číslici. Na konci každého vzorečku se hodnoty týkající se dynamicky zavedených rodin resetují pomocí `\dfamreset`. Vnořené matematické vzorečky (např. `\hbox{...$vzorec$...}` uvnitř vzorce) tyto hodnoty neresetují, protože je přechodně nastaveno `\dfamstart` na `\relax`.

Při použití `\dfam` se doporučuje snížit počet staticky zavedených rodin jen na často používané předefinováním maker `\normalmath` a `\boldmath` a odpovídajícím snížením čísla `\count18`. Rodiny 0, 1, 2 a 3 musejí nutně zůstat staticky zavedené, protože TeX potřebuje jejich metriky v každém vzorečku.

Má-li rodina bold variantu, je možné ji deklarovat následujícím způsobem:

```
\newif \ifboldmath   \boldmathfalse
\addto\normalmath{\boldmathfalse}  \addto\boldmath{\boldmathtrue}
% příklad eufm a eufb:
\sdef{dfam:fractur}{\ifboldmath eufb\else eufm\fi}
```

16.13 Matice s vysunutým sloupcem

0137
P. O.
15. 01. 2016

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix} \begin{matrix} \text{první řádek} \\ \text{druhý} \\ \\ \text{poslední} \end{matrix}$$

Vytvoříme makro `\matrixR`, které vytvoří matici s typickými natahovacími závorkami, ale poslední sloupec matice nebude umístěný „uvnitř matice“, ale vně obklopujících závorek. Například

```
$$
  \matrixR(){ccc}{1&2&3&první řádek\cr 4&5&6&druhý\cr 7&8&9\cr 10&11&12&poslední}
$$
```

vytvoří matici, jak je ukázáno na obrázku. Syntaktické pravidlo pro `\matrixR` je

```
\matrixR levá-závorka pravá-závorka {deklarace} {data}
```

přitom `deklarace` obsahuje jen deklaraci sloupců unitř matice, zatímco `data` obsahují i jeden sloupec navíc, který se objeví mimo matici a je vytištěn v textovém módu.

Implementace:

```
\def\matrixR#1#2#3#4{%
   \bgroup \def\tabiteml{$\ }\def\tabitemr{\ $}\dimen0=0pt
   \def\tabdeclareR{\setbox0=\hbox{\,####\unsskip}\ifdim\dimen0<\wd0 \global\dimen0=\wd0 \fi}%
   \setbox1=\vbox{\table{#3R}{#4}}%
   \setbox2=\hbox{$\left#1\vcenter{\copy1}\right#2$}%
   \dimen1=\wd2 \advance\dimen1 by-\wd1 \divide\dimen1 by2
   \def\tabdeclareR{\rlap{\kern\dimen1 \,####\unsskip}}%
   \left#1\vcenter{\table{#3R}{#4}}\right#2\kern\dimen0
   \egroup
}
```

V prvém průchodu je použita `\table` s přidanou deklarací R, která poslední sloupec pouze změří, ale nevytiskne. Nejširší položka z tohoto sloupce má šířku `\dimen0`. Výsledek prvního průchodu je uložen do boxu 1. Dále je sestaven `\box2`, který navíc obsahuje okolní závorky. Z rozdílů šířek těchto dvou boxů je vypočtena šířka natahovací závorky a uložena do `\dimen1`. Ve druhém průchodu je poslední sloupec R tištěn jako `\rlap` s odpovídajícím `\kern` o šířce natahovací závorky. Tento průchod už je skutečně tištěn do výstupu a je připojen `\kern\dimen0`, aby následující text za maticí pokračoval až za posledním sloupcem.

Analogicky můžete vytvořit `\matrixL` nebo `\matrixLR`.

0138

16.14 Matice s vysunutým řádkem

P. O.
16. 01. 2016

Vytvoříme makro `\matrixA`, které se chová stejně jako makro `\matrixR` z předchozího OPmac triku 0137, ale navíc je první řádek vysunut nad matici. Uživatel může ale nemusí použít poslední nedeklarovaný sloupec, tj. může vysunutí řádku kombinovat s vysunutím sloupce. Kromě toho vytvoříme makro `\matrixB`, které na rozdíl od `\matrixA` vysouvá pod matici poslední řádek.

Syntaktické pravidlo pro použití `\matrixA` a `\matrixB` je stejné, jako u `\matrixR` tedy:

```
\matrixA levá-závorka pravá-závorka {deklarace} {data}
```

Implementace:

```
\def\matrixA#1#2#3#4{%
   {\mathop{\matrixI#1#2{#3}{\noalign{\kern-\normalbaselineskip}#4}}%
    \limits^{\textstyle\mathstrut}}}
\def\matrixB#1#2#3#4{%
   {\mathop{\matrixI#1#2{#3}{#4\crcr\noalign{\kern-\normalbaselineskip}}}%
    \limits_{\textstyle\mathstrut}}}
\let\matrixI=\matrixR  % use \matrixR from OPmac trick 0137
```

Matice má vysunutý řádek díky `\noalign{\kern-\normalbaselineskip}`. Celá tato matice je schována do `\mathop` a je přidán horní (resp. dolní) neviditelný index, který vkládá strut, aby i vysunutý řádek měl svou běžnou výšku. Uživatel může napsat `\let\matrixI=\matrixL` nebo `\let\matrixI=\matrixLR`, pokud chce vysouvat i první sloupec a pokud si tato makra vytvořil.

0149

16.15 Názorné zkratky v matematické sazbě

P. O.
26. 05. 2016

$$a \leq b \leq c \Rightarrow a \leq c$$

Je daleko názornější ve zdrojáku matematické sazby psát §tt§char60= místo `\leq`, ==§tt§char62 místo `\Rightarrow`, `+-` místo `\pm` atd. Takže zápis pak může vypadat takto:

```
$$ a <= b <= c ==> a <= c $$
```

Tuto inteligenci matematické sazby lze deklarovat pomocí makra `\msecdef` třeba takto:

```
\mspecdef <<   \ll
\mspecdef <>   \neq
\mspecdef <=   \leq
\mspecdef <=>  \Leftrightarrow
\mspecdef >>   \gg
\mspecdef >=   \geq
\mspecdef -+   \mp
\mspecdef +-   \pm
\mspecdef ==   \equiv
\mspecdef =.   \doteq
\mspecdef ==>  \Rightarrow
```

Konečně ukážeme, jak vypadá kód chytrého makra `\msecdef`:

```
\def\skipnext#1#2{#1}
\def\trynext#1{\trynextA#1.\relax\relax}
\def\trynextA#1#2\relax#3\relax#4#5{%
   \ifx\relax#2\relax \def\next{#4}\else
      \def\next{\isnextchar#1{\skipnext{\trynextA#2\relax#3#1\relax{#4}{#5}}}{#5#3}}\fi
   \next
}
\def\mspecdefA#1#2#3 #4{\ifx#2\undefined
   \def#2{\trynext{#3}{#4}{#1}}\else
   \toks0={\trynext{#3}{#4}}\toks1=\expandafter{#2}%
   \edef#2{\the\toks0{\the\toks1}}\fi
}
\def\mspecdef#1{%
   \ifcat_#1 \expandafter\let\csname m:#1\endcsname=X \catcode`#1=12 \fi
   \expandafter\ifx\csname m:#1\endcsname\relax
      \expandafter\mathchardef\csname m:#1\endcsname=\mathcode`#1 \fi
   \mathcode`#1="8000 \begingroup \lccode`~=`#1
   \lowercase{\endgroup\expandafter\mspecdefA\csname m:#1\endcsname~}%
}
\def\tmp{\adef_##1{_{##1}}\adef^##1{^{##1}}}\tmp
\catcode`_=12 \catcode`^=12 \mathcode`_="8000 \mathcode`^="8000
```

Popis myšlenky: po `\mspecdef ax \U  \mspecdef axy \V  \mspecdef abcd \W` se deklaruje znak `a` jako math-aktivní (má kód `"8000`) a je definován jako:

```
\def a{\trynext{bcd}\W{\trynext{xy}\V{\trynext{x}\U{normal a}}}}
```

Makro nejprve testuje, zda následuje string `bcd` (pomocí opakovaného `\isnextchar`). Pokud to je splněno, vynechá se zbytek makra a spustí se `\W`. Jinak se spustí zbytek makra. To znamená, že se testuje následující string `xy` a když není přítomen, testuje se `x` a když také není přítomen, vytiskne se normální `a`. Všimněte si, že delší stringy (začínající stejně) musejí být pomocí `\mspecdef` deklarovány později než stringy kratší.

Poslední dva řádky kódu zajistí, že `$a_+$` se automaticky převede na `$a_{+}$`, protože přímá syntax `$a_+$` přestává fungovat například po `\mspecdef +- \pm`.

16.16 Sazba závislá na mathstylu

0157
P. O.
03. 05. 2017

K sazbě v matematickém módu, která je závislá na matematickém stylu (display, text, script, scriptscript) se používá primitivní příkaz `\mathchoice` a plainTeX definuje makro `\mathpalette`. Obojí je nešikovné. Vlastnost `\mathchoice` by mohla být implemenována v TeXu pomocí pozdrženého argumentu (jako při `\write`) bez nutnosti v TeXu sestavovat všechny čtyři varianty sazby. To ale nespravíme bez zásahu do samotného TeXu. Na druhé straně makro `\mathpalette`, které podává uživatelem definovanému makru přepínač stylu jako parametr, můžeme opustit a udělat si makro `\varstyle` s názornějším použitím a s daleko širšími možnostmi. Makro `\varstyle` má jeden parametr, který vytvoří matematický seznam. V rámci tohoto parametru je možné používat `\usestyle` pro přepnutí do matematického stylu aktivního

v okamžiku užití `\varstyle` a dále je možné použít `\mathaxis` pro vzdálenost účaří od matematické osy a případně si definovat další na stylu závislá makra pomocí `\ifcase\mstylenum D\or T\or S\or SS\`
Například makro na dvojitou sumu (dva sumační znaky přes sebe) může vypadat takto:

```
\def\dbsumR{\ifcase\mstylenum .35\or.25\or.21\or.15\fi em}
\def\dbsumU{\ifcase\mstylenum .15\or.12\or.09\or.07\fi em}
\def\doublesum{\mathop{\varstyle{\raise\dbsumU\rlap{$\usestyle\sum$}{\kern\dbsumR\sum}}}}

$\displaystyle \doublesum_{ij}^{DN} \textstyle \doublesum_{ij}^{DN}$
```

Makro `\varstyle` a příbuzná makra jsou definována pomocí `\mathchoice`:

```
\newcount\mstylenum
\def\varstyle#1{\mathchoice{\mstylenum=0 #1}{\mstylenum=1 #1}{\mstylenum=2 #1}{\mstylenum=3 #1}}
\def\usestyle{\ifcase\mstylenum \displaystyle\or\textstyle\or\scriptstyle\or\scriptscriptstyle\fi}
\def\mathaxis{\fontdimen22\ifcase\mstylenum \textfont\or\textfont\or\scriptfont\or\scriptscriptfont\fi2 }
```

17 Tabulky

0158 *P. O.* 17. 06. 2017

17.1 Ukázka tabulky s přesahy položek do více řádků a sloupců

OPmac verze Jun 2017 přidává některé nové vlastnosti pro makro `\table`: možnost opakování v deklaraci, deklarátor p, makro `\crlp` na částečné vodorovné čáry a konečně makro `\mspan` na položky přesahující více sloupců.

Ukážeme, jak sestavit tabulku

<table>
<tr><td colspan="2"></td><td colspan="3">Singular</td><td colspan="3">Plural</td></tr>
<tr><td colspan="2"></td><td>Neuter</td><td>Masculine</td><td>Feminine</td><td>Masculine</td><td>Feminine</td><td>Neuter</td></tr>
<tr><td rowspan="2">I</td><td>Inclusive</td><td colspan="3" rowspan="2">O</td><td colspan="3">X</td></tr>
<tr><td>Exclusive</td><td colspan="3">X</td></tr>
<tr><td rowspan="2">II</td><td>Informal</td><td colspan="3">X</td><td colspan="3">X</td></tr>
<tr><td>Formal</td><td colspan="6">X</td></tr>
<tr><td rowspan="2">III</td><td>Informal</td><td rowspan="2">O</td><td>X</td><td>X</td><td colspan="2">X</td><td rowspan="2">O</td></tr>
<tr><td>Formal</td><td colspan="4">X</td></tr>
</table>

pomocí makra `\table`. Tabulka je inspirovaná dotazem na StackExchange. Můžete porovnat přehlednost či nepřehlednost LaTeXového kódu diskutovaného na StackExchange s tímto příkladem, kde jsme vytvořili tutéž tabulku, ale pomocí maker z OPmac.

```
\def\low#1{\vbox to 0pt{\kern1.5ex\hbox{#1}\vss}}
\def\tabstrut{\vrule height 20pt depth10pt width0pt}

\table{|8{c|}}{\crlp{3-8}
   \mspan2[c|]{} &\mspan3[c|]{Singular} &\mspan3[c|]{Plural} \crlp{3-8}
   \mspan2[c|]{} & Neuter & Masculine & Feminine & Masculine & Feminine & Neuter \crl
   \low I    & Inclusive &\mspan3[c|]{\low O} &\mspan3[c|] X \crlp{2,6-8}
             & Exclusive &\mspan3[c|]{} &\mspan3[c|] X \crl
   \low{II}  & Informal  &\mspan3[c|] X &\mspan3[c|] X \crlp{2-8}
             & Formal    &\mspan6[c|] X \crl
   \low{III} & Informal  & \low O & X & X &\mspan2[c|] X & \low O \crlp{2,4-7}
             & Formal    & &\mspan4[c|] X & \crl
}
```

Předefinovali jsme `\tabstrut`, aby tabulka byla vzdušnější. Na StackExchange je totiž typografický paskvil. Definovali jsme makro `\low` na snížení textu mírně pod řádek, což je použito v položkách, které vypadají jako dvouřádkové (obecnější řešení najdete v následujícím triku 0159). Konečně jsme použili `\mspan` na přesah přes více sloupců a `\crlp` na vodorovné čáry jen někde. Čáry v deklaraci `\mspan` jsou v souladu s dokumentací výhradně vpravo deklarátoru `c`.

17.2 Umístění textu vertikálně centrovaného ve více řádcích tabulky

0159
P. O.
17. 06. 2017

Vytoříme makro `\crow číslo {text}`, které sníží v daném řádku `text` tak, aby to vypadalo, že je vertikálně centrovaný přes `číslo` řádků. Chceme tedy místo ad hoc vytvořeného makra `\low{text}` z předchozího příkladu 0158 používat `\crow2{text}`.

```
\def\crow{\afterassignment\crowA \tmpnum=}
\def\crowA#1{\setbox0=\hbox{\tabstrut}\tmpdim=\ht0 \advance\tmpdim by\dp0
   \tmpdim=\the\tmpnum\tmpdim
   \vbox to0pt{\kern-\ht0 \vbox to\tmpdim{\vss\hbox{#1}\kern-\prevdepth\vss}\vss}%
}
```

Makro uloží do `\tmpdim` výšku plus hloubku tabstrutu násobenou `číslem` a vytvoří `\vbox to\tmpdim`, který je schován ve `\vbox to0pt`, aby to v sazbě nepřekáželo.

Poznamenejme, že makro funguje za předpokladu, že všechny řádky tabulky jsou „jednořádkové“, tj. v položkách tabulky se nevyskytují celé víceřádkové odstavce. Potřebujeme totiž dopředu znát výšku všech řádků tabulky. Pokud se ale v řádcích vyskytují víceřádkové odstavce a dopředu nevíme, jak budou tyto položky tabulky vysoké, pak je lepší ručně vyladit posazení textu dolů podobně, jako v makru `\low` v předchozím OPmac triku. Nebo, trváte-li na automatickém výpočtu umístění textu, je někdy možné použít primitivní příkaz `\valign` místo `\halign`. Pokud jsou dopředu známy šířky všech sloupců v tabulce, je užití \halign (o které se opírá i makro `\table`) totiž nadbytečné, protože nepotřebujeme zarovnávat podle nejširší položky v každém sloupci. Potřebujeme ale zarovnávat podle nejvyšší položky v každém řádku, a k tomu se hodí `\valign`, což je transponované `\halign`. Ukázku implementace této myšlenky naleznete zde.

17.3 Modifikace formátu odstavce při použití deklarátoru p

0160
P. O.
17. 06. 2017

V manuálu k OPmac je zmíněno, že odstavec v tabulce deklarovaný pomocí `p{rozměr}` je implicitně formátován jako obvyklý odstavec mimo tabulku, tedy typicky do bloku. To může v úzkých sloupcích dělat potíže. Proto je uvedena možnost použití `p{rozměr\raggedright}` pro vytváření odstavců s pravým roztřepením. Můžete si definovat další možnosti:

```
\let\fL=\raggedright
\def\fR{\leftskip=0pt plus 1fill}
\def\fC{\leftskip=0pt plus1fill \rightskip=0pt plus 1fill}
\def\fX{\leftskip=\iindent plus1fil
        \rightskip=\iindent plus-1fil
        \parfillskip=0pt plus2fil}
```

a potom je odstavec v tabulce:

```
p{42mm}     ... zarovnán do bloku,
p{42mm\fL}  ... zarovnán vlevo,
p{42mm\fR}  ... zarovnán vpravo,
p{42mm\fC}  ... s centrovanými řádky,
p{42mm\fX}  ... zarovnán do bloku s centrovaným posledním řádkem.
```

Všimněte si, že při zarovnání vlevo je možné dělení slov ale při zarovnání vpravo nikoli, protože dělení slov v takové úpravě odstavce není vhodné. Nechcete-li dělit slova ani při `\fL`, definujte jednoduše:

```
\def\fL{\rightskip=0pt plus 1fil}
```

0161 ## 17.4 Odstavce v položkách jinak vertikálně usazené

P. O. 17. 06. 2017

OPmac nabízí jen deklarátor `p{rozměr}`, který vytvoří v položce tabulky odstavec. Je-li takových odstavců v řádku tabulky více vedle sebe, jsou vertikálně umístěny podle prvního řádku (jako `\vtop`). V tomto OPmac triku zavedeme LaTeXovský deklarátor `m{rozměr}` chovající se jako `\vcenter` (centrování na střed) a konečně `b{rozměr}` chovající se jako `\vbox` (usazení podle spodního řádku). Návod, jak to udělat, je také v technické dokumentaci k OPmac.

```
\def\paramtabdeclarem#1{\tabiteml{$\vcenter{\hsize=#1\relax
   \baselineskip=\normalbaselineskip \lineskiplimit=0pt
   \noindent\vbox{\hbox{\tabstrutA}\kern-\prevdepth}##\unsskip
   \vbox to0pt{\vss\hbox{\tabstrutA}}}$}\tabitemr
}
\def\paramtabdeclareb#1{\tabiteml\vbox{\hsize=#1\relax
   \baselineskip=\normalbaselineskip \lineskiplimit=0pt
   \noindent\vbox{\hbox{\tabstrutA}\kern-\prevdepth}##\unsskip}\tabitemr
}
```

Vimněte si, že na začátek textu přidáváme strut, který má výšku jako `\tabstrut`, ale nulovou hloubku a na konec textu přidáváme strut, který má hloubku jako `\tabstrut`, ale nulovou výšku.

0163 ## 17.5 \crlp a dvojité svislé linky

P. O. 18. 06. 2017

Makro `\crlp{seznam}`, které ukončí řádek tabulky a vloží pod něj vodorovné linky jen ve sloupcích podle seznamu, se chová jako `\crli`, tedy linky nezasahují do prostoru mezi dvojitou svislou čarou. Přitom k `\crli` máme alternativu `\crl`, která protne dvojité svislé čáry. Jakou alternativu máme, chceme-li nakreslit linku jen někde?

Touto alternativou je ukončit řádek pomocí `\cr` a nakreslit jakoby další řádek obsahující výhradně `\multispan` buď s prázdným obsahem nebo s `\hrulefill`. Například

```
\cr \multispan2 & \multispan3\hrulefill & \multipan1 & \mutispan4\hrulefill \cr
```

se chová jako `\crlp { 3-5, 7-10 }`, ale přesahuje případné svislé dvojité linky. Vyzkoušejte si:

```
\def\spankern{\kern-\vvkern\kern-\drulewidth}

\table{||c||c||l}{\crl
  a & b & c \crli
  d & e & f \crlp{1-2}
  g & h & i \cr \multispan2\hrulefill & \multispan1 \cr
  j & k & l \cr \multispan1\hrulefill & \multispan2 \cr
  m & n & o \crlp{1}
  p & q & r \cr \multispan1 & \multispan1\spankern\hrulefill & \multispan1 \cr
  s & t & u \crlp{2}
  v & w & x \crl
}
```

Když si tuto ukázku vyzkoušíte a podrobně se podíváte na výsledek, shledáte, že je zde další problém: vodorovná linka vytvořená pomocí `\multispan` v sazbě překáží a v důsledku ní není pravá svislá linka souvislá. Je tedy třeba ještě zapustit vodorovnou linku do sazby pomocí `\noalign{\kern-\drulewidth}`. Tento příkaz vložte na konec řádků `j&k&l` a `p&q&r` (vždy za druhé `\cr`). Analogický záporný kern vkládá i makro `\crlp`, takže při užití `\crlp` není nutné tento problém řešit.

Konečně si všimněte definice a užití makra `\spankern`, které přidává záporný kern, aby čára pod řádem p&q&r protla i dvojitou čáru z předchozího sloupce.

0162 ## 17.6 Verbatim text v tabulkách

P. O. 17. 06. 2017

Protože `\table{deklarace}{data}` čte celý druhý parametr `data` před jeho vykonáním, není možné uvnitř tohoto parametru přecházet do verbatim (např. takto: `$#`) nebo dělat jiné

změny kategorií znaků s požadavkem okamžitého efektu. A přitom stačí málo, číst parametr `data` a okamžitě ho vykonávat. To se dá udělat takto:

```
\activettchar"

\tableverb{deklarace}
data včetně "#&$" verbatim
\etable
```

Všimněte si, že na rozdíl od makra `\table` nejsou `data` uzavřena do svorek a jsou ukončena makrem `\etable`. Kód pro tato makra vypadá takto:

```
\def\tableverb{\vbox\bgroup \catcode`\|=12 \tableB}
\def\tableB#1{\offinterlineskip \colnum=0 \def\tmpa{}\tabdata={}\scantabdata#1\
   \halign\expandafter\bgroup\the\tabdata\cr}
\def\etable{\crcr\egroup\egroup}
```

17.7 Tabulky jako v balíčku booktabs

0009
P. O.
15. 08. 2013

Ortodoxní zastánci LaTeXového balíčku booktabs by ohrnuli nos nad předchozí ukázkou a zdůraznili by, že jsem lama, která neví, že svislé čáry do tabulky nepatří a vodorovné jsou možné v různých tloušťkách, ale v žádném případě ne dvojité. A hlavně s čarami v tabulce je třeba zacházet umírněně a decentně. V dokumentaci mají tuto ukázku:

Item		
Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

kterou pomocí `\table` z OPmac vytvoříme třeba takto:

```
\table{llr}{                              \crtop
  \multispan2\hfil Item\hfil&             \cr
  \multispan2\tablinefil\kern.5em         \cr
   Animal    & Description & Price (\$) \crmid
   Gnat      & per gram    &      13.65 \cr
             & each        &       0.01 \cr
   Gnu       & stuffed     &      92.50 \cr
   Emu       & stuffed     &      33.33 \cr
   Armadillo & frozen      &       8.99 \crbot}
```

Je ovšem potřeba definovat `\crtop`, `\crmid`, `\crbot` a pozměnit tloušťku čáry v `\tablinefil`:

```
\def\crtop{\crcr \noalign{\hrule height.6pt \kern2.5pt}}
\def\crbot{\crcr \noalign{\kern2.5pt\hrule height.6pt}}
\def\crmid{\crcr \noalign{\kern1pt\hrule\kern1pt}}
\def\tablinefil{\leaders\hrule height.2pt\hfil\vrule height1.7pt depth1.5pt width0pt }
```

Konečně je třeba odstranit mezery vlevo v první položce a vpravo v poslední, jinak by ty čáry poněkud přečuhovaly. Přidal jsem `\kern-.5em` do deklarace řádku tabulky zleva a zprava,

protože `\tabiteml` a `\tabitemr` jsou definovány jako `\enspace`. K tomu účelu jsem mírně předefinoval interní makro OPmac `\tableA`:

```
\def\tableA#1#2{\offinterlineskip \def\tmpa{}\tabdata={\kern-.5em}\scantabdata#1\relax
   \halign\expandafter{\the\tabdata\kern-.5em\tabstrutA\cr#2\crcr}\egroup}
```

0010 ## 17.8 Střídavě podbarvené řádky v tabulce

P. O.
15. 08. 2013

V poslední době se rozmohla móda podbarvovat řádky v tabulce.

aa	bb	cc	dd	ee	ff
gg	hh	ii	jj	kk	ll
mm	nn	oo	pp	qq	rr
ss	tt	uu	vv	ww	xx
ab	cd	ef	gh	ij	kl
mn	op	qr	st	uv	wx

Pomocí `\table` vytvoříme výše uvedenou tabulku třeba takto:

```
\frame{\table{llllll}{\crx
  aa & bb & cc & dd & ee & ff \crx
  gg & hh & ii & jj & kk & ll \crx
  mm & nn & oo & pp & qq & rr \crx
  ss & tt & uu & vv & ww & xx \crx
  ab & cd & ef & gh & ij & kl \crx
  mn & op & qr & st & uv & wx}}
```

K tomu stačí dodefinovat příkaz `\crx`, který střídavě vkládá/nevkládá do vertikálního seznamu mezi řádky šedé pruhy:

```
\newcount\tabline  \tabline=1
\def\crx{\crcr \ifodd\tabline \colortabline \fi
         \global\advance\tabline by1 }
\def\colortabline{\noalign{\localcolor\LightGrey
   \hrule height\ht\strutbox depth\dp\strutbox \kern-\ht\strutbox \kern-\dp\strutbox}}
\def\tabiteml{\quad}\def\tabitemr{\quad}
```

0094 ## 17.9 Jednotlivě podbarvené položky v tabulce

P. O.
02. 04. 2015

Vytvoříme deklarátory sloupce tabulky C, R, L, které fungují obdobně, jako c, r, l, ale navíc umožní každou položku podbarvit zvolenou barvou. Je-li první objekt v položce přepínač barvy, tato barva se použije pro podklad položky. Například:

```
... & \Blue Text &        ... % modře podbarvený černý text
... & \Green \Red  Text & ... % zeleně podbarený červený text
... & \relax \Blue Text & ... % modrý text, defaultní pozadí
```

Podbarvení položek vypruží jako obvyklé mezery ve sloupci tabulky, tj. při C pruží z obou stran, při L pruží zprava a při R pruží zleva. Rovněž mezery `\tabiteml` a `\tabitemr` jsou podbarveny.

Je-li nastaveno `\cellcolor` pomocí `\let` jako barva (např. `\let\cellcolor=\Yellow`), pak tato barva se použije jako defualtní pozadí. Není-li `\cellcolor` nastaveno, je defaultní pozadí průhledné.

Mezi jednotlivými sloupci můžete vložit bílou mezeru pomocí nenulové hodnoty `\tabskip` a mezi jednotlivými řádky vznikne bílá mezera pomocí `\noalign{\kern...}`. Příklad:

```
\def\tabstrut{\lower4pt\vbox to15pt{}}
\tabskip=2pt \def\cskip{\noalign{\kern2pt}} \let\cellcolor=\Yellow
\table{CLR}{first item      & second & \Blue \White third item \cr\cskip
           next long item  & \Red X & \Green       YES }
```

vytvoří tabulku:

first item	second	third item
next long item	X	YES

Implementace:

```
\def\tabdeclareC{\futurelet\next\setcellcolor##\end\hfil\hfil}
\def\tabdeclareL{\futurelet\next\setcellcolor##\end\relax\hfil}
\def\tabdeclareR{\futurelet\next\setcellcolor##\end\hfil\relax}
\def\setcellcolor{\ifx\next\global \expandafter\setcellcolorC\else \expandafter\setcellcolorD\fi}
\def\setcellcolorC#1\fi#2\end#3#4{%
   \setbox0=\hbox{{\tabiteml\localcolor#2\unskip\tabitemr}}%
   {\localcolor#1\fi\tabstrut\leaders\vrule\hskip\wd0 \ifx#3\hfil plus1fil\fi}%
   \kern-\wd0 \box0
   \ifx#4\hfil {\kern-.2pt \localcolor#1\fi\leaders\vrule\hskip.2pt plus1fil}\fi
}
\def\setcellcolorD{\ifx\cellcolor\undefined \let\next=\setcellcolorN
   \else \def\next{\expandafter\expandafter\expandafter\setcellcolorC\cellcolor}%
   \fi \next
}
\def\setcellcolorN#1\end#2#3{#2\tabiteml{\localcolor#1\unskip}\tabitemr#3}
```

Makro `\setcellcolor` sebere první token z položky. Makro pracuje v deklarační části tabulky, takže první položku si sejme až po expanzi na úroveň prvního neexpandovatelného tokenu. Tímto tokenem v případě makra pro barvu je `\global` (viz makro `\setcmylcolor`). Je-li toto splněno, spustí se makro `\setcellcolorC`, které si do `#1` uloží rozexpandované tělo makra `\setcmykcolor` (až po poslední `\fi`), do `#2` si uloží zbylý text položky a do `#3` a `#4` uloží vzkaz o způsobu centrování položky. Dále toto makro změří šířku položky v boxu0 a podbarví box0 pomocí `\leaders`. Není-li první token položky `\global`, pak makro `\setcellcolorD` ještě rozhodne, zda je či není definováno makro `\cellcolor` a pokud je, podstrčí makru `\setcellcolorC` správně rozexpandovaný `\cellcolor`, jinak spustí `\setcellcolorN`, což nevytvoří žádný podklad.

17.10 Podbarvené položky přes více sloupců

0103
P. O.
23. 04. 2015

K předchozímu OPmac triku přidáme možnost podvarvovat položky přes více sloupců. Přidáme makro `\multispanc{počet} \Barva {Text}`, které vytvoří položku přes `počet` sloupců podbarvené barvou `\Barva`. Chcete-li mít položku podbarvanou implicitní barvou, pište místo `\Barva \relax`.

Příklad:

```
\def\tabstrut{\lower4pt\vbox to15pt{}}
\tabskip=2pt \def\cskip{\noalign{\kern2pt}} \let\cellcolor=\Yellow
\table{CLR}{\Black\White\bf Table &\multispanc2 \Black {\White\bf Title} \cr\cskip
            first item      & second & \Blue \White third item \cr\cskip
            next long item  & \Red X & \Green       YES }
```

vytvoří:

Table	Title	
first item	second	third item
next long item	X	YES

Makro `\multispanc` může být definováno takto:

```
\def\multispanc#1#2#3{\multispan{#1}%
   \expandafter\expandafter\expandafter\cellcolexp#2#3\end\hfil\hfil\ignorespaces}
\def\cellcolexp{\futurelet\next\setcellcolor}
```

0011

17.11 Tabulky se stanovenou šířkou

P. O.
16. 08. 2013

Vytvoříme makro `\tableto{šířka}{deklarace}{data}`, které vytvoří tabulku se stanovenou šířkou. Mezery mezi sloupci se tomu přizpůsobí. K tomu stačí inspirovat se makrem `\table` z OPmac a zapracovat do něj práci s registrem `\tabskip`.

```
\def\tableto{\vbox\bgroup \catcode`\|=12 \tableAto}
\def\tableAto#1#2#3{\offinterlineskip \def\tmpa{}%
   \tabdata={\tabskip=0pt plus1fil minus1fil}\scantabdata#2\relax
   \halign to#1\expandafter{\the\tabdata\tabskip=0pt\tabstrutA\cr#3\crcr}\egroup}
```

Mezery `\tabiteml` a `\tabitemr` se nyní projeví vlevo od prvního sloupce a vpravo od posledního sloupce. Jinde mezi sloupci bude mezera upravena tak, aby celková šířka tabulky vyhověla stanovenému údaji.

Toto makro funguje na tabulky například z předchozích dvou příkladů. Nefunguje to správně na tabulky se svislými linkami mezi sloupci (na okrajích tabulky svislé linky nevadí). Pokud tedy nejste (stejně jako já) ortodoxní vyznavači booktabs a tedy výjimečně snesete i kultivovaně vedenou svislou linku, je třeba pro natahovací tabulky se svilými linkami použít jiné makro:

```
\newdimen\tabw
\def\countcols#1{\ifx#1\relax\else
   \ifx#1|\else\advance\tmpnum by2 \fi \expandafter\countcols \fi}
\def\tableto#1#2#3{{\def\tabiteml{}\def\tabitemr{}\setbox0=\table{#2}{#3}%
   \tmpnum=0 \countcols#2\relax \tabw=#1\advance\tabw by-\wd0 \divide\tabw by\tmpnum
   \def\tabiteml{\kern\tabw}\def\tabitemr{\kern\tabw}\table{#2}{#3}}}
```

Toto makro nejprve nanečisto do boxu0 vysází tabulku s prázdnými `\tabiteml` a `\tabitemr`. Pak odečte od požadované šířky šířku boxu0 a vydělí to dvojnásobkem počtu sloupců. Tento rozměr pak nacpe do `\tabiteml` a `\tabitemr` a znovu vytiskne. Nyní už bude mít tabulka požadovanou šířku.

Výhoda tohoto řešení (proti postupu např. v TBN, str. 141) je, že není potřeba měnit nic v datech tabulky. Počet sloupců zůstává zachován.

0012

17.12 Rozšiřitelný sloupec v tabulce typu „odstavec“

P. O.
16. 08. 2013

Vytvoříme makro `\tableto{šířkatab}{deklarace}{data}` podobně jako v předchozím příkladě. Předpokládáme, že v deklaraci bude použit jeden deklarátor `P`, který označuje sloupec typu odstavec a jehož šířka bude taková, aby celková šířka tabulky odpovídala spcifikovanému údaji `{šířkatab}`.

```
\newdimen\Pwidth  \newdimen\tabw
\def\tabdeclareP {\tabiteml\vtop{\hsize=\Pwidth \rightskip=0pt plus1fil
   \baselineskip=1.2em \lineskiplimit=0pt
   \noindent ##\unsskip\vbox to0pt{\vss\hbox{\tabstrutA}}}\hss\tabitemr}
\def\tableto#1#2#3{{\Pwidth=.5\hsize \setbox0=\table{#2}{#3}
   \tabw=#1\relax \advance\tabw by-\wd0 \advance\Pwidth by\tabw \table{#2}{#3}}}
```

Toto makro nejprve nanečisto vytvoří požadovanou tabulku s `\Pwidth=\hsize` do boxu0 a pak změří box0 a upraví podle toho konečnou hodnotu registru `\Pwidth`,

0015

17.13 Deklarátory v tabulce z více písmen

P. O.
17. 08. 2013

Ačkoli se to v dokumentaci OPmac explicitně netvrdí, můžete vytvořit deklarátory z více než jednoho písmene. Takový deklarátor musí být v deklaraci obehnán svorkami, což udržuje názornost a přehlednost deklarace. Jako příklad vytvoříme sadu deklarátorů `ic`,`ir`,`il`, které se chovají jako `c`,`r`,`l`, ale provedou sazbu v kurzívě.

```
\def\tabdeclareic{\tabiteml\it\hfil##\unsskip\hfil\tabitemr}
\def\tabdeclareil{\tabiteml\it##\unsskip\hfil\tabitemr}
\def\tabdeclareir{\tabiteml\it\hfil##\unsskip\tabitemr}
```

Příklad použití:

```
\table{cc{ic}c}{první & druhý & třetí & čtvrtý \cr
                a &     b &     c &      d }
```

Třetí sloupec je v kurzívě, zatímco ostatní sloupce jsou v aktuálním fontu nastaveném před použitím `\table`.

17.14 Desetinná čísla v tabulkách

0016
P. O.
17. 08. 2013

Vytvoříme deklarátor `D`, který umožní sazbu desetinných čísel s desetinnou čárkou pod sebou.

```
\def\tabdeclareD{\tabiteml\hfil\scandecnumber##&##\hfil\tabitemr}
\def\scandecnumber#1,{#1\ifdim\lastskip>0pt \unskip\phantom,\else,\fi&}
```

Příklad použití:

```
\table{lDr}{číslo &   3,24   & první \cr
            jiné  & 112,1    & druhé \cr
            celé  &  13     ,& čárka vzadu \cr
            další &   0,123  & malé}
```

Je vidět, že pokud chceme do sloupce s desetinnými čísly vložit číslo celé bez desetinné čárky, musíme to ošetřit jako výjimku. Tj. vložit čárku na úplný konec položky. Tato čárka se pak nevytiskne.

Chtít po makru, aby výjimku pro celé číslo vyřešilo samo, je sice taky možné, ale makro by bylo podstatně komplikovanější. Ctíme přitom zásadu: „v jednoduchosti je síla“ a nevytváříme zbytečné komplikovanosti.

Sloupec D je implementován jako dva sloupce tabulky. Pokud jej tedy chceme přeskakovat pomocí `\multispan`, musíme toto vědět a započítat ho za dva sloupce.

17.15 Desetinná čísla v tabulkách podruhé

0171
P. O.
07. 12. 2019

OPmac trik 0016 pro desetiná čísla v tabulách zakládá „dvojsloupec“, což působí komplikace např. při použití `\tskip`. Zde je tedy jiné řešení, které navíc umožňuje mít ve sloupci i nenumerické texty, např. pro záhlaví tabulky. Řešení zakládá jediný sloupec, tj. nemělo by způsobovat potíže.

Je vytvořen nový deklarátor `N`§tt§char60`číslo`§tt§char62, např. `N3`, který udává, že ve sloupci budou čísla (numbers) s desetinnou tečkou, přitom nejvíce cifer za tečkou se očekává §tt§char60`číslo`§tt§char62. Každé číslo se pak doplní o vhodný počet neviditelných desetinných cifer na stanovený počet a takto jsou čísla zarovnána vpravo. Tím je dosaženo lícování pod sebou desetinné tečky (která na výstupu může být čárkou, předefinujete-li `\decimalpoint`).

Čísla jsou vysázena v matematickém módu, tj. znak „mínus“ vypadá, jak má.

Je-li v položce text bez tečky, je vysázen v textovém módu a je centrován (když je menší než celková šířka čísel ve sloupci) nebo je zarovnán vpravo (když je větší než celková šířka čísel). Je možné přidat další neexistující cifry, tj. §tt§char60`číslo`§tt§char62 může být větší než skutečný počet cifer, pokud se tím vizuálně zlepší např. umístění širšího nenumerického textu. Příklad:

```
\table{c|N3}{
  Ingredients      & w/w (\%) \crl \tskip.2em
  Water            & -97.5    \cr
  Agar powder      &  2.0     \cr
  Solidum chloride &  0.43
}
```

Je-li cifer za tečkou více než devět, musíte použít složené závorky: `N{12}`. Také můžete stanovit necelý počet cifer, např. `N{2.5}`, je-li to k něčemu dobré. Implementace deklarátoru `N` je následující:

```
\def\decimalpoint{.}

\def\paramtabdeclareN#1{\tabiteml\hfil\aligndecdigit#1##.\relax\tabitemr}
\def\aligndecdigit#1#2.#3{%
   \ifx\relax#3\hfil#2\unsskip\hfil \else
   \dimen0=.5em \dimen0=#1\dimen0 $#2$\decimalpoint
   \expandafter\aligndecdigitA\expandafter#3\fi
}
\def\aligndecdigitA#1.\relax{\hbox to\dimen0{$#1$\hss}}

\def\tableA#1#2{\offinterlineskip \colnum=0 \def\tmpa{}\tabdata={}\scantabdata#1\relax
   \def\tmpb{#2}\replacestrings{\crl}{\crcr\crl}%
      \replacestrings{\crli}{\crcr\crli}\replacestrings{\crll}{\crcr\crll}%
      \replacestrings{\crlli}{\crcr\crlli}\replacestrings{\crlp}{\crcr\crlp}%
   \halign\expandafter{\the\tabdata\cr\tmpb\crcr}\egroup}
```

Deklarátor je definován jako `\paramtabdeclareN`. Pomocí `\aligndecdigit` se přečte text až po tečku a vysází se v matematickém módu a pomocí `\aligndecdigitA` se vysází zbytek.

Položky s užitím deklarátoru `N` musejí bezpodmínečně končit znakem `&` nebo `\cr`, nikoli makrem, které symbol konce položky obsahuje. Protože v manuálu OPmac jsou zmíněna marka `\crl` (apod.), která obsahují `\cr`, je předefinována přípravná čast tabulky, kde se pomocí `\replacestrings` nahradí `\crl` za `\crcr\crl`, což udělá nakonec v tabulce stejnou práci. Druhá možnost je nepoužívat předefinování `\tableA` a ukáznit uživatele, ať před každé `\crl` (a jim podobné) předřadí `\cr`, pokud má sloupec `N` v tabule jako poslední.

0023

17.16 Tabulka přes více stránek

P. O.
31. 08. 2013

Primitiv `\halign` vytvoří řádky tabulky a ty je možné rozlámat do stránek. Stačí ho neschovávat dovnitř `\vboxu`. Tuto vlastnost si ukážeme na jednoduché dlouhé tabulce, kde chceme mít linky mezi každým řádkem a pro jednoduchost nebudeme chtít opakovat nadpis tabulky na následujících stránkách (to je vyřešeno až v následující tipu). Uživatel napíše třeba

```
\longtable{|c|c|}{data & data \cr
                  data & data \cr
                  ... moc dat ... & \cr}
```

a vytvoří mu to centrovanou tabulku, která je ochotna se lámat do stránek. Bude asi potřeba deklarovat `\raggedbottom`, protože řádky tabulky neobsahují žádnou pružnost. Celý zázrak se ukrývá v následujících řádcích:

```
\def\longtable{\goodbreak \bgroup \catcode`\|=12 \tableL}
\def\tableL#1#2{\setbox0=\table{#1}{#2}\setbox0=\hbox to\wd0{} % tabulka nanečisto
   \tmpdim=\hsize \advance\tmpdim by-\wd0 \divide\tmpdim by2   % výpočet odsazení
   \def\tabstrut{\vrule height1.1em depth.5em width0pt }       % volnější řádkování
   \everycr={\longtablecr}\offinterlineskip                    % \everycr
   \def\tmpa{}\tabdata={\kern\tmpdim}\scantabdata#1\relax      % \tabdata
   \halign\expandafter{\the\tabdata\tabstrutA\cr#2\crcr}\egroup\goodbreak}
\def\longtablecr{\noalign{\nobreak\ltrule\penalty0\kern-.4pt\ltrule\nobreak}}
\def\ltrule{\moveright\tmpdim\hbox to\wd0{\hrulefill}}
```

Hlavní idea makra spočívá v tom, že pomocí `\everycr` vložíme za každé `\cr` zhruba řečeno: `\noalign{\hrule\penalty0\kern-.4pt\hrule\nobreak}`. Takže kreslíme dvě stejné čáry (tloušťky `.4pt`) přes sebe a mezi nima povolíme pomocí `\penalty0` stránkový zlom. Když se rozlomí, první čára zůstane dole a druhá se objeví nahoře na další stránce. Ostatní čáry jsou kresleny přes sebe, takže jejich zdvojení není vidět.

Dále tento kód řeší centrování dlouhé tabulky. Kvůli tomu ji nejprve vytvoří nanečisto do \box0 a do \tmpdim odměří velikost odsazení každého řádku. Toto odsazení pak je vloženo do deklarace tabulky v \tabdata. Místo \hrule je pak použita posunutá \hrule definovaná jako \ltrule.

17.17 Dlouhá tabulka s opakujícím titulkem

0024
P. O.
31. 08. 2013

Chce-li uživatel vytvořit dlouhou tabulku, která se bude lámat do stránek, na každé stránce zopakuje svůj titulek, a přitom bude mít čáru nad titulkem tabulky, pod titulkem tabulky, na konci stránek a na konci tabulky a jinde ne, může použít \longtableT takto:

```
\def\strutT {\vrule height1.1em depth.8em width0pt} % strut pro titulek
\longtableT {|c|c|c|}{ hlava1 & hlava2 & hlava22 \cr \tskip3pt  % titulek
                       data   & data   & data    \cr
                       data   & ... mraky & dat ...\cr}
```

Je třeba definovat \strutT, který vypodloží titulek. Kolem titulku budou čáry, které by při normálním \strut působily stísněně. Dále je povinnost za titulkem dát \tskip, který se použije nejen pod čárou titulku, ale taky na konci každé stránky před uzavírací čárou. V tabulce samotné nesmí být požadavek na vodorovnou čáru. Uvedené makro \longtableT je implementováno takto:

```
\def\longtableT#1#2{\goodbreak \bgroup \setbox0=\table{#1}{\strutT\relax#2}
   \setbox1=\vbox{\unvbox0 \setbox2=\vbox{}\revertbox}
   \setbox2=\vbox{\unvbox2 \global\setbox4=\lastbox\unskip \global\setbox5=\lastbox\unskip}
   \whatfree4 \advance\tmpdim by-.8pt \ifdim\tmpdim<\baselineskip \vfil\break \fi
   \offinterlineskip \crule\center4\crule\center5 \printboxes
   \center5\crule \egroup \goodbreak
}
\def\whatfree#1{\tmpdim=\vsize \advance\tmpdim by-\pagetotal
   \advance\tmpdim by-\prevdepth \advance\tmpdim by-.4pt
   \advance\tmpdim by-\ht#1 \advance\tmpdim by-\dp#1 \advance\tmpdim by-\ht5 }
\def\revertbox {\setbox0=\lastbox\unskip
   \ifvoid0 \else \global\setbox2=\vbox{\unvbox2\box0} \revertbox\fi }
\def\printboxes{\setbox2=\vbox{\unvbox2 \global\setbox0=\lastbox\unskip}
   \ifvoid0 \else \whatfree0
      \ifdim\tmpdim<0pt \center5\crule\vfil\break \crule\center4\crule\center5 \fi
      \center0 \nobreak \printboxes \fi }
\def\crule{\centerline{\hbox to\wd4{\hrulefill}}\nobreak}
\def\center#1{\centerline{\copy#1}\nobreak}
```

Makro nejprve sestaví tabulku do \box0, ten rozebere a boxy v opačném pořadí pomocí \revertbox vloží do boxu2. Titulek si z boxu2 odebere jako box4 a prostor pod ním z \tskip jako box5. Dále makrem \printboxes odebírá boxy zezadu z boxu2 a klade je do stránky pomocí \center, aby byly centrovány. Pomocí \whatfree vždy nejprve změří zbylý prostor na stránce a pokud se tam už další box nevejde, přidá \box5\crule, odstránkuje a na začátek další stránky přidá \crule\box4\crule\box5.

18 Obrázky

18.1 Mezera separující parametr makra \inspic

0141
P. O.
09. 04. 2016

OPmac předpokládá, že název souboru, tedy parametr makra \inspic, bude ukončen mezerou. Je to z toho důvodu, že stejně jako primitivní příkaz \input nevyžaduje kučeravé závorky, rozhodl jsem se i u makra \inspic nic takového nevyžadovat. Pokud chcete načíst makrem \inspic soubor s mezerou, pište

```
\inspic {soubor s mezerou.pdf}
```

a nezapomeňte za zavírací závorku dát další povinnou mezeru.

Bohužel, makro `\inspic` se chová poněkud jinak než primitivní příkaz `\input`. V případě `\inspic` je mezeru vždy nutné psát. Třeba na rozdíl od zápisu `\line{\input soubor}` je tedy nutné použít `\line{\inspic soubor.pdf }`. Koho to irituje, může pravidlo mezery z makra `\inspic` odstranit a důsledně pak psát parametr makra do kučeravých závorek. K tomu slouží tento kód:

```
\expandafter\def\expandafter\inspic\expandafter#\expandafter1\expandafter{\inspic{#1} }
```

0032
P. O.
02. 10. 2013

18.2 Popisky k obrázkům z programu Inkscape

Program Inkscape umožňuje při ukládání do PDF rozdělit textové popisky od ostatních čar: stačí v dialogovém okně pro ukládání zaškrtnout PDF+LaTeX. Ostatní čáry vloží do souboru `file.pdf` a vedle tohoto souboru vytvoří ještě soubor `file.pdf_tex` s popisky zapsanými LaTeXovou synaxí. V TeXovém dokumentu stačí v místě obrázku psát

```
\inkinspic file.pdf
```

Toto makro načte soubor `file.pdf` s čarami i soubor `file.pdf_tex` s popisky. Šířka obrázku je dána registrem `\picw` jako při použití makra `\inspic`.

Popisky jsou vysázeny ve velikosti a rodině fontu jako okolní sazba, takže Inkscape fonty vůbec neřeší. V náhledu Inkscape můžete použít libovolný font v libovolné velikosti a nebude to mít žádný vliv na výsledou sazbu. Do popisků v náhledu můžete psát TeXové příkazy (např. matematické vzorce vložené mezi dolary). Umístění popisku je možné buď nalevo, napravo nebo na centr pomocí nabídky v Inkscape. To je pak jediný bod, který se zaručeně v sazbě bude shodovat s tím, co vidíte v náhledu. Text můžete také obarvit nebo otáčet.

Abychom mohli v OPmac využít tento rys programu Inkscape připravený pro LaTeX, je potřeba emulovat některé LaTeXové příkazy, které se v exportovaném souboru vyskytují: `\begin{picture}`, `\put`, `\makebox` a další. K tomu stačí do definic dokumentu přidat následující kód:

```
\def\inkinspic #1 {\bgroup
  \ifdim\picw=0pt
    \setbox0=\hbox{\inspic#1 }%
    \picw=\wd0
  \fi
  \the\inkdefs \input \picdir#1_tex \egroup
}
\newtoks\inkdefs  \inkdefs={%
  %\ifdim\picw=0pt \message{inkinspic: \picw set to \hsize}\picw=\hsize \fi
  \def\makeatletter#1\makeatother{}%
  \def\includegraphics[#1]#2{\inkscanpage#1,page=,\end \inspic #2 \hss}%
  \def\inkscanpage#1page=#2,#3\end{\ifx,#2,\else\def\inspicpage{page#2}\fi}%
  \def\put(#1,#2)#3{\nointerlineskip\vbox to0pt{\vss\hbox to0pt{\kern#1\picw
      \pdfsave\hbox to0pt{#3}\pdfrestore\hss}\kern#2\picw}}%
  \def\begin#1{\csname begin#1\endcsname}%
  \def\beginpicture(#1,#2){\vbox\bgroup\hbox to\picw{}\kern#2\picw \def\end##1{\egroup}}%
  \def\begintabular[#1]#2#3\end#4{\vtop{\def\\{\cr}\def\tabiteml{}\def\tabitemr{}\table{#2}{#3}}}%
  \def\color[#1]#2{\scancolor #2,}%
  \def\scancolor#1,#2,#3,{\pdfliteral{#1 #2 #3 rg}}%
  \def\makebox(#1)[#2]#3{\hbox to0pt{\csname mbx:#2\endcsname{#3}}}%
  \sdef{mbx:lb}#1{#1\hss}\sdef{mbx:rb}#1{\hss#1}\sdef{mbx:b}#1{\hss#1\hss}%
  \sdef{mbx:lt}#1{#1\hss}\sdef{mbx:rt}#1{\hss#1}\sdef{mbx:t}#1{\hss#1\hss}%
  \def\rotatebox#1#2{\pdfrotate{#1}#2}%
  \def\lineheight#1{}%
  \def\setlength#1#2{}%
}
```

Pozor na mezeru: makro `\inkinspic` je (stejně jako makro `\inspic`) naprogramováno tak, že má svůj parametr ukončený mezerou. Chová se tedy analogicky, jako příkaz `\input`. Chcete-li například obrázek centrovat, pište:

```
\centerline{\picw=7cm \inkinspic obrazek.pdf }
```

18.3 Jinak formátované \caption

0059
P. O.
13. 05. 2014

OPmac formátuje popisky pod obrázky a tabulkami implicitně tak, že centruje poslední řádek odstavce. Je možné, že chcete jiné formátování. Například takové, aby popisky centrovaly, pokud jsou na samostatném řádku, ale jinak aby se chovaly jako obyčejný odstavec, pokud přetečou na více řádků. To by mohl vyřešit následující kód:

```
\def\printcaption#1#2{\leftskip=\iindent \rightskip=\iindent
   \setbox0=\hbox\bgroup \aftergroup\docaption{\bf#1 #2.}\enspace}
\def\docaption{\tmpdim=\hsize \advance\tmpdim by-2\iindent
   \ifdim\wd0>\tmpdim \unhbox0 \else \hfil\hfil\unhbox0 \fi \endgraf \egroup}
```

Makro přeměří text popisku uložený v boxu0 a pokud se vejde na jeden řádek, předřadí před něj `\hfil\hfil`, což je pružinka stejné síly, jako `\parfillskip` (v popiscích dle OPmac) a text centruje. Je-li popisek delší, žádná pružinka se nepředřadí a vysází se obyčejný odstavec.

18.4 Popisek vedle obrázku

0136
P. O.
06. 01. 2016

Figure 1.1 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur massa turpis, semper quis fringilla ut, viverra nec risus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec nunc lorem, sollicitudin vel sodales eget, vehicula nec mi. Proin ullamcorper rutrum nibh, at porttitor nunc euismod et. Donec faucibus nisi faucibus ipsum porttitor pharetra. Sed elementum, lectus nec congue imperdiet, ipsum leo viverra nisi, sit amet commodo odio odio id nisl. Fusce sagittis lobortis nisi sed consectetur.

Je-li obrázek spíše vysoký než široký, může se hodit umístit popisek k němu vedle obrázku a ne pod ním nebo nad ním. To lze provést třeba takto

```
\hbox to\hsize{\picw=.4\hsize \inspic mapa.jpg
  \hfil\vbox{\hsize=.55\hsize \iindent=0pt \emergencystretch=2em
    \caption/f Lorem ipsum dolor sit amet, consectetur adipiscing elit.
       ...
       Fusce sagittis lobortis nisi sed consectetur.
   \par}}
```

Poznamenejme, že je potřeba vnitřní `\vbox` ukončit `\par`, protože OPmac pro popisky lokálně předefinovává `\par` a jeho první použití vrací věci do normálu. Dále je zde nastaveno `\iindent=0pt`, protože nechceme mít zúžené okraje.

18.5 \inspic natáhne do PDF stejný obrázek jen jednou

0110
P. O.
04. 06. 2015

Víme, že pomocí pimitivního příkazu pdfTeXu `\pdfximage` lze zařídit jedno vložení obrázku do výstupního PDF a dále tento obrázek opakovaně zobrazovat pomocí `\pdfrefximage`. Makro `\inspic` z OPmac je ale koncipováno jen pro jednorázové vložení obrázku, takže když použijete `\inspic` opakovaně se stejným obrázkem, vloží se do PDF dokumentu vícekrát a zbytečně plýtváte místem v PDF souboru.

Pro potřebu postupně odhalovaných slíd (viz OPmac trik 0022 a viz také CTUslides v CTUstyle) jsem potřeboval zvýšit inteligenci makra \inspic tak, aby opakovaně volaný obrázek nevkládal do PDF výstupu data obrázku opakovaně. Řešení je snadné:

```
\let\oriinspic=\inspic
\def\picdim{\the\picwidth:\the\picheight}
```

```
\def\inspic#1 {%
   \expandafter\ifx\csname pic:#1:\picdim\endcsname \relax
      \oriinspic#1 % prvé vložení obrázku
      \global\expandafter\mathchardef\csname pic:#1:\picdim\endcsname=\pdflastximage
   \else \expandafter\pdfrefximage \csname pic:#1:\picdim\endcsname % opakované vložení
   \fi
}
```

Je vidět, že refenenční číslo obrázku je řídicí sekvence obsahující jednak název obrázku a také stav parametrů `\picwidth` a `\picheight`. Při jiném nastavení těchto parametrů je potřeba obrázek natáhnout znovu, nebo je potřeba využít už natažený obrázek a podrobit jej vhodné lineární transformaci. To by sice šlo, ale pro jednoduchost jsem se tím nezabýval. Opakované vložení různě velkých ale jinak stejných obrázků by si musel makroprogramátor ošetřit sám.

Řešení se bohužel opírá o vlastnosti pdfTeXu (dostupné i v LuaTeXu), které nejsou použitelné v XeTeXu. Tam by se to muslelo dělat jinak.

19 Odstavec

19.1 Pohodlné zadávání \parshape

0154 *P. O.* 07. 06. 2016

Obecný tvar odstavce lze specifikovat pomocí primitivu `\parshape`, ale příprava parametrů tohoto primitivu není pro uživatele příliš komfortní. V TBN strana 236 je ukázka marka `\oblom`, které se specializuje na obdélníkové výřezy odstavce. Zde navrhneme makro `\setparshape`, které umožní specifikovat levý nebo pravý okraj odstavce obecněji a poměrně pohodlně. Příklady:

```
\setparshape\left (3*0, 7*35)mm  % po třech řádcích obdélníkový výřez v sedmi řádcích
% nebo
\setparshape\right (4*0, 5,10,15,20,25,20,15,10,5)mm % trojúhelníkový výřez vpravo
```

Makro `\setparshape` nejprve přečte `\left` nebo `\right`, což specifikuje okraj, který bude ovlivněn. Pak v kulaté závorce je čárkami oddělený seznam údajů ve tvaru `počet*rozměr`, kde `počet` odpovídá počtu řádků, které budou zkráceny o `rozměr`. Není-li v údaji hvězdička, je tam `rozměr` týkající se jediného řádku. Za uzavírací kulatou závorkou následuje jednotka pro `rozměry` ukončená mezerou. Pokud chcete rozměry přímo specifikovat v datech, použijte `(...){}` tedy například:

```
\setparshape\left (3*0pt, 7*.4\hsize){}
```

což vytvoří obdélníkový výřez v sedmi řádcích vlevo a dá to stejný výsledek jako:

```
\setparshape\left (3*0, 7*.4){\hsize}
```

Nastavení `\setparshape` se implicitně týká jediného odstavce, který bezprostředně následuje. Pokud chcete ovlivnit několik odstavců pod sebou, je třeba předřadit `\globalshape`, tedy například psát:

```
\globalshape\setparshape\left (3*0, 12*55)mm
```

Implementace je následující:

```
\newcount\parshapenum \newcount\globpar

\def\setparshape#1(#2)#3 {\def\parshapedata{}\parshapenum=1
   \let\parshapeside=#1\def\parshapeunit{#3}\setparshapeA#2,,%
}
\def\setparshapeA#1,{\ifx,#1,\parshape=\parshapenum\parshapedata0pt\hsize
   \else\fihere\setparshapeB#1\empty*\empty\end\fi
}
\def\setparshapeB#1*#2\empty#3\end{%
```

```
   \ifx,#3,\tmpnum=1 \tmpdim=#1\parshapeunit \relax
   \else \tmpnum=#1\tmpdim=#2\parshapeunit \relax \fi
   \loop \ifnum\tmpnum>0
      \advance\tmpnum by-1 \advance\parshapenum by1
      \dimen0=\hsize \advance\dimen0 by-\tmpdim
      \ifx\parshapeside\left
          \edef\parshapedata{\parshapedata\the\tmpdim\space\the\dimen0}%
      \else\edef\parshapedata{\parshapedata0pt\the\dimen0}\fi
   \repeat
   \setparshapeA
}
\def\fihere#1\fi{\fi#1}
\def\globalshape{% TBN page 237
   \global\globpar=0 \gdef\par{\ifhmode\shapepar\fi}%
}
\def\shapepar{\prevgraf=\globpar \parshape=\parshapenum\parshapedata0pt\hsize
   \endgraf \global\globpar=\prevgraf
   \ifnum\prevgraf<\parshapenum \else \global\let\par=\endgraf\fi
}
```

19.2 Obrázek v obdélníkovém výřezu odstavce

0155
P. O.
08. 06. 2016

Vytvoříme makro `\putshape strana číslo{objekt}`, které vloží `objekt` do následujícího odstavce (nebo více odstavců) s obdélníkovým vykrojením textu pro `objekt` vlevo (`strana` je `\left`) nebo vpravo (`strana` je `\right`) tak, že prvních `číslo` řádků odstavce je beze změny, pak následuje vykrojení textu s `objektem`. Velikost odsazení a počet odsazených řádků si makro spočítá samo na základě informace o tom, jak velký je `objekt`. Typické užití vypadá třeba takto:

```
\putshape\right 4 {\picw=3cm \inspic obrazek.png }
```

Makro z ukázky vloží obrázek vpravo pod čtvrtý řádek následuhícího odstavce s obdélníkovým vykrojením závislým na velikosti obrázku. Mezi textem a objektem (obrázkem) je vložena vertikální mezera `\putvmargin` (nad a pod) a `\puthmargin` (vedle mezi obrázkem a textem). Tyto registry jsou implicitně nastaveny na `5pt`.

Implementace se opírá o makro `\setparshape` z předchozího OPmac triku 0154:

```
\newdimen\putvmargin \newdimen\puthmargin
\putvmargin=5pt \puthmargin=5pt
```

```
\def\putshape#1#2#{\let\tmpa=#1\def\tmpb{#2}\putshapeA}
\def\putshapeA#1{\vskip\parskip
   \setbox0=\vbox{\kern\putvmargin\hbox{#1}\kern\putvmargin}\tmpnum=\ht0
   \advance\tmpnum by-10 \divide\tmpnum by\baselineskip \advance\tmpnum by1
   \dimen0=\wd0 \advance\dimen0 by\puthmargin
   \hbox to\hsize{\ifx\tmpa\right \hfill\fi
      \vbox to0pt{\kern\tmpb\baselineskip\kern-.8\baselineskip
         \vbox to\the\tmpnum\baselineskip{\vss\box0\vss}\vss}\hss}
   \nobreak\vskip-\parskip\vskip-\baselineskip
   \edef\tmpb{\tmpa(\tmpb*0pt,\the\tmpnum*\the\dimen0)}
   \globalshape\expandafter\setparshape\tmpb{}
}
```

20 Slídy

20.1 Slídy ve spartánské úpravě

0017 *P. O.* 18. 08. 2013

Pokud chceme připravit text pro projekci, je potřeba především mít landscape formát a dostatečně velké písmo. Hodí se použít písmo bezserifové. Pokud nepotřebujete další specialitky, je možné si vystačit s málem:

```
\input opmac

\margins/1 a5l (1,1,1,1.2)cm        % landscape formát
\input chelvet  \typosize[17/22]    % dostatečně velké písmo

\def\sec#1\par{\centerline
     {\secfont#1}\bigskip}          % sekce uprostřed a nebudou číslovány
\def\pg{\vfil\break}                % makro pro odstránkování
\begitems                           % celý dokument bude jako výčet
```

Vlastní text členíme pomocí hvězdiček na jednotlivé výkřiky, dále můžeme použít `\sec` pro vyznačení nadpisů a makro `\pg` na odstránkování.

```
\sec Nějaká nosná myšlenka

* výkřik první
* výkřik druhý
* výkřik třetí

\pg
\sec Další myšlenka

* výkřik čtvrtý
* výkřik pátý

\enditems\end
```

Pokud potřebujete mít firemní loga, zajímavou grafiku atd, je potřeba si s tím samozřejmě více pohrát a vytvořit si speciální styl. Doporučuji též použít trik o podkladovém obrázku.

20.2 Slideshow – postupné odhalování

0022 *Mirek + P. O.* 30. 08. 2013

Někteří promítači svých myšlenek rádi odhalují skutečnosti postupně, tj. nejprve jen část stránky, pak další část a nakonec celou stránku. Takoví promítači do místa, kde se má myšlenka poodhalit, napíší `\kuk`. To vytvoří stránku až do tohoto místa. Na následující stránce bude vše opsáno a přidán případný další text po další `\kuk`. Definitivní konec stránky musí být označen pomocí `\pg\kuk` (před `\pg` se `\kuk` nepíše) a dokument končí sekvencí `\enditems\end\kuk`. Na začátku dokumentu (po přečtení všech maker) musí být napsáno `\kukstart`. Pokud je tento příkaz schován za procentem, probíhá normální stránkování a příkazy `\kuk` jsou ignorovány.

Implementace `\kuk` navazuje na předchozí ukázku o slídech spartánského vzhledu a přidává (za první `\begitems`) tato makra:

```
\let\kuk=\relax % je-li \kukstart zakomentovano, tiskne normalne dokument
\def\kukdata{}
\long\def\kukstart#1\kuk{\addto\kukdata{#1}%
   \tmpnum=0 \def\endkukdata{}\expandafter\sumkuk \kukdata\sumkuk
   \kukdata\endkukdata \vfil\break \kukstart
}
\long\def\sumkuk#1{\ifx\sumkuk#1% jsem na konci seznamu
   \loop \ifnum\tmpnum>0 \addto\endkukdata{\enditems}\advance\tmpnum by-1 \repeat
   \else
     \ifx\begitems#1\global\advance\tmpnum by1 \fi
     \ifx\enditems#1\global\advance\tmpnum by-1 \fi
     \expandafter\sumkuk \fi
}
\count1=1
\def\advancepageno{\ifx\kukdata\empty \global\advance\pageno by1 \global\count1=1
                   \else \global\advance\count1 by1 \fi}
\def\pg{\def\kukdata{}\vfil\break}
\kukstart
```

Makro `\kukstart` se točí dokola a nabírá text až po `\kuk`, přidává ho k už nabranému textu z minula do `\kukdata` a tiskne stránky. Makro `\pg` nabraný text promaže. Nejvíce péče je vněnováno případu, kdy je `\kuk` napsáno uvnitř další úrovně odrážek (tedy ve skupině). Makro `\sumkuk` si spočítá úroveň vnoření a na základě toho doplní do `\endkukdata` příslušný počet `\enditems`.

Poznámka: důkladnější řešení postupného odhalování slíd najdete v CTUslides, které je součástí CTUstyle.

21 Bibliografické údaje

21.1 Odkazy s vloženou poznámkou

0038
P. O.
01. 04. 2014

Někdy potřebujeme napsat odkaz s vloženou poznámkou, například s upřesněním strany v citovaném dokumentu. Třeba takto:

Obtékání obrázku textem je řešeno v [27, s. 236].

To můžeme zařídit pomocí `\rcite` třeba takto:

```
Obtékání obrázku textem je řešeno v~[\rcite[tbn], s.~236].
```

Možná by se ale hodilo rozšířit funkčnost makra `\cite` tak, že když najde za `[lejblíkem]` lomítko:

```
Obtékání obrázku textem je řešeno v~\cite[tbn]/{s.~236}.
```

vytvoří požadovaný výsledek [27, s. 236]. K tomu poslouží následující kód:

```
\def\cite[#1]{\isnextchar/{\pcite[#1]}{[\rcite[#1]]}}
\def\pcite[#1]/#2{[\rcite[#1],~#2]}
```

Makro `\cite` nyní testuje přítomnost lomítka. Pokud není přítomno, zavolá `\rcite` s obklopenými závorkami. Pokud je přítomno, zavolá `\pcite`, které vykoná požadovanou práci.

21.2 Komprimované odkazy typu [jméno, rok]

0039
P. O.
01. 04. 2014

Při `\nonumcitations` a při značkách typu `[jméno, rok]` se hodí neopakovat stejné jméno v jednom balíku citací. Například při:

```
Česky se píše o \TeX{}u například v~\cite[tst,tbn,tpp,lpp].
```

dostaneme [Olšák, 1995, Olšák, 1997, Olšák, 2013, Satrapa, 2011], ale chtěli bychom spíše odkaz tvaru [Olšák, 1995, 1997, 2013, Satrapa, 2011]. Toto můžeme řešit pomocí `\ecite`, např.

```
[\rcite[tst], \ecite[tbn]{1997}, \ecite[tpp]{2013}, \rcite[lpp]]
```

ale lepší by bylo, kdyby zkracování probíhalo automaticky. K tomu může posloužit následující kód.

```
\nonumcitations
\def\printcite#1{\citesep
   \prepcitelink{#1}\citelink{#1}{\the\bibmark}\def\citesep{,\hskip.2em\relax}%
}
\def\prepcitelink#1{%
   \isdefined{bim:#1}\iftrue
      \expandafter\expandafter\expandafter \ppclink \csname bim:#1\endcsname!\relax
   \else \opwarning{comprimed cites: empty bibmark}\bibmark={#1}%
   \fi
}
\def\ppclink #1, #2!\relax{\def\tmpa{#1}%
   \ifx \lastcitedname\tmpa \bibmark={#2}%
   \else \let\lastcitedname=\tmpa \if^#2^\bibmark={#1}\else\bibmark={#1, #2}\fi
   \fi
}
```

Makro předefinovává makro z OPmac `\printcite` tak, že volá `\citelink` s prarametrem `\the\bibmark`. Přitom `\bibmark` je nejprve připraven pomocí `\prepcitelink`. Makro `\prepcitelink` rozloží značku ve spolupráci s `\ppclink` na část před čárkou s mezerou a zbytek (na jméno a rok). Makro `\lastcitedname` je na začátku implicitně nedefinované a přijímá naposledy citované jméno. Je-li v zápětí toto jméno stejné, je do `\bibmark` vložen jen rok. Jinak je tam vloženo jméno i rok. Protože je celý `\cite` proveden uvnitř skupiny, na konci `\cite` se `\lastcitedname` vrátí do nedefinovaného stavu, takže první údaj nového `\cite` bude vždy jméno obsahovat.

Upozorňuji, že kód funguje jen tehdy, když jsou všechny značky připraveny ve tvaru `jméno, rok`, kde uvedené dva údaje jsou odděleny čárkou následovanou mezerou. Chybí-li údaj `rok`, je třeba, aby značka byla ukončena čárkou a mezerou. BibTeXový styl `apalike` tomuto požadavku vyhovuje.

0043 P. O. 02. 04. 2014

21.3 Modifikace odkazů typu (jméno rok)

Někdy se požaduje, aby odkazy typu „jméno rok“ byly bez čárek mezi autorem a rokem, např. [Olšák 2013]. V takovém případě stačí použít kód z předchozího triku a upravit v něm makro `\ppclink` takto:

```
\def\ppclink #1, #2!\relax{\def\tmpa{#1}%
   \ifx \lastcitedname\tmpa \bibmark={#2}%
   \else \let\lastcitedname=\tmpa \if^#2^\bibmark={#1}\else\bibmark={#1 #2}\fi
   \fi
}
```

BibTeXový styl `apalike` sice čárky pro značky generuje, ale takto modifikované makro `\ppclink` je nakonec odstraní.

Taky se často požaduje, aby byly odkazy v kulatých závorkách, např. takto (Olšák 2013). Není nic jednoduššího, než definovat:

```
\def\cite[#1]{(\rcite[#1])}
```

0096 P. O. 12. 04. 2015

21.4 Značky v seznamu literatury při \nonumcitations

OPmac při `\nonumcitations` píše místo čísel do místa `\cite` značky, ale tyto značky implicitně neopakuje v seznamu literatury. Předpokládá se totiž, že seznam je řazený podle jmen prvního z autorů každé citace a tato jména se ve značkách vyskytují. Zopakování značky v seznamu literatury je pak nadbytečné zejména při použití dlouhých značek typu (Olšák, 2001). Ale při použití krátkých značek [Ol01] může být žádoucí takovou značku v seznamu literatury zopakovat.

K zopakování značek při `\nonumcitations` v seznamu literatury je třeba předefinovat makro `\printbib`, které se stará o zahájení tisku jednoho záznamu v seznamu literatury. Třeba takto:

```
\def\printbib{\hangindent=2\iindent
   \noindent\hskip2\iindent \llap{[\the\bibmark] }%
}
```

21.5 Zkrácené značky při použití opmac-bib

0097
P. O.
12. 04. 2015

Příkaz `\cite` vypisuje značky místo čísel, pokud je nastaveno `\nonumcitations`. Makro pro přímé čtení `.bib` souborů `opmac-bib.tex` nabízí dva stylové soubory (`simple` a `iso690`), které generují tyto značky v dlouhém formátu `Jméno prvního autora, rok`. Některá nakladatelství preferují styl, ve kterém jsou tyto značky zkráceny. Třeba na první dvě písmena jména prvního autora následovaná dvěma ciframi určujícími rok, tedy například [Kn84]. Jak takové zkrácené značky dostat i do seznamu literatury popisuje OPmac trik 0096. Nyní si ukážeme, jak předefinovat generování těchto značek při použití `opmac-bib`, abychom dosáhli zkrácený formát bez nutnosti doplňovat pro každý záznam výjimku do `.bib` souboru pomocí bibmark.

Stylové soubory `opmac-bib-styl.tex` (od verze Apr. 2015) používají k vygenerování značek makro `\setbibmark`. Je tedy potřeba si definovat vlastní makro `\mysetbibmark` a dále pomocí `\bibtexhook` toto makro nastavit:

```
\def\mysetbibmark{%
   \ifx\dobibmark\undefined \def\dobibmark{}\fi
   \RetrieveFieldIn{bibmark}\tmp
   \ifx\tmp\empty
      \RetrieveFieldIn{year}\tmp
      \edef\tmp{\expandafter\bibmarkA\dobibmark{}{}\relax \expandafter\bibmarkB\tmp{}{}{}{}\relax}%
   \fi
   \bibmark=\expandafter{\tmp}%
}
\def\bibmarkA#1#2#3\relax{#1#2}     % first two letters from the author name
\def\bibmarkB#1#2#3#4#5\relax{#3#4} % lats two digits from the year

\def\bibtexhook{\let\setbibmark=\mysetbibmark}
```

Makro využije jméno autora uložené stylovým souborem v \dobibmark a přidá k němu rok. Z prvého extrahuje první dva tokeny použitím \bibmarkA a z druhého extrahuje cifry roku použitím `\bibmarkB`.

Pokud máte v seznamu dvě stejné značky (např. citujete plodného autora, který vytvořil více literárních výstupů za jeden rok), můžete deklarovat výjimku pomocí pole bibmark v `.bib` souboru. Nebo je možné zařadit do automatického generování přidávání písmen a, b, c atd. To ponechám jako cvičení pro šikovného makroprogramátora.

21.6 Kontrola unikátnosti značek bibmark

0098
P. O.
12. 04. 2015

Pokud nemáte k dispozici šikovného makroprogramátora, o kterém se hovoří v předchozím OPmac triku, možná uvítáte aspoň kontrolu, zda všechny značky tištěné v `\cite` při `\nonumcitations` jsou zadány jednoznačně. Implicitně tato kontrola neprobíhá, takže se může stát, že se v dokumentu pracuje se dvěma nebo více stejnými značkami, které příslušejí různým bibliografickým záznamům.

Kontrola je provedena v makru `\bibmarkcheck`, jehož definici i volání je možné vložit například na začátek dokumentu (někam za `\input opmac`).

```
\def\bibmarkcheck{\ifx\lastbibnum\undefined \else
   \bgroup
      \bibnum=0
      \loop
         \advance\bibnum by1
```

```
         \isdefined{bim:\the\bibnum}\iftrue
            \isdefined{\csname bim:\the\bibnum\endcsname}\iftrue
               \opwarning{Duplicated bibmark: "\csname bim:\the\bibnum\endcsname"}%
            \fi
            \sdef{\csname bim:\the\bibnum\endcsname}{}%
         \fi
         \ifnum\bibnum<\lastbibnum \relax \repeat
   \egroup
   \fi
}
\bibmarkcheck
```

0040 21.7 Odsazení seznamu literatury dle největšího čísla

P. O. 01. 04. 2014

OPmac ignoruje údaj generovaný BibTeXem o počtu cifer největšího čísla v seznamu literatury, protože jednak umožňuje seznam také vytvořit pomocí příkazů `\bib` mimo BibTeX, a dále při využití jednou generované databáze pro více účelů nemusí být údaj generovaný BibTeXem pravdivý. Místo toho OPmac zavádí makro `\lastbibnum`, které po přečtení REF souboru obsahuje největší číslo v seznamu literatury. Implicitně toto makro není nijak využito, ale můžete ho při formátování seznamu literatury použít. Například takto:

```
\def\printbib{%
   \ifx\lastbibnum\undefined \tmpdim=\iindent
   \else \setbox0=\hbox{[\lastbibnum] }%
      \ifdim \parindent=0pt \tmpdim=\wd0
      \else \tmpdim=0pt \loop \advance\tmpdim by\parindent
                              \ifdim\tmpdim<\wd0 \repeat
   \fi\fi
   \hangindent=\tmpdim \noindent \hskip\tmpdim \llap{[\the\bibnum] }%
}
```

Zde je předefinováno formátovací makro `\printbib`, které si změří box obsahující `[\lastbibnum]~`. Při nulovém `\parindent` nastaví `\tmpdim` na šířku tohoto boxu a při nenulovém `\parindent` nastaví `\tmpdim` na násobek `\parindent` tak, aby se box do vzniklé mezery vešel. Pak odsadí bibloigrafický záznam podle `\tmpdim`.

0041 21.8 Odsazení seznamu literatury dle nejširší značky

P. O. 01. 04. 2014

Seznam literatury nemusí být vždy číslovaný; může obsahovat například značky. Přitom poslední značka (na rozdíl od posledního čísla) nemusí být ta nejširší. Přesto chceme podle nejširší značky formátovat celý seznam literatury.

Je sice možné k tomu účelu použít `\halign`, ale seznamy literatury generované BibTeXem jsou celé ukryté ve skupině a to by činilo problémy. Ukážeme si zde jiné řešení přes REF soubor. Toto řešení je použitelné i v obdobných případech, kdy zrovna nemusí jít o seznam literatury.

Požádáme uživatele, aby na konec seznamu literatury přidal pokyn `\setbibindent`. Toto makro zapíše údaj o nejširší značce do REF souboru a při příštím čtení tohoto souboru jej využije formátovací makro `\printbib`. Makro `\printbib` mimo jiné měří šířky značek a údaj o nejširší z nich dává do `\bibindent`, aby jej mohlo makro `\setbibindent` uložit. Kód obou maker může vypadat takto:

```
\nonumcitations
\def\printbib{%
   \setbox0=\hbox{[\the\bibmark]\quad}%
   \ifx\bibindent\undefined \def\bibindent{0pt}\fi
   \ifdim\wd0>\bibindent \edef\bibindent{\the\wd0}\fi
   \ifx \Xbibindent\undefined
      \hangindent=\wd0 \noindent [\the\bibmark]\quad
   \else
      \hangindent=\Xbibindent \noindent \hskip\Xbibindent \llap{[\the\bibmark]\quad}%
   \fi
```

```
}
\def\setbibindent{\ifx\bibindent\undefined \else
   \openref\immediate\write\reffile{\def\noexpand\Xbibindent{\bibindent}}\fi}
```

21.9 Více nezávislých seznamů literatury v dokumentu

0042
P. O.
01. 04. 2014

Předpokládejme, že tvoříme třeba sborník, takže se náš dokument skládá z několika článků, přitom každý článek má svůj seznam literatury. Záznamy se mohou v různých seznamech překrývat, ale nebudou pravděpodobně stejně očíslovány. V každém seznamu probíhá číslování od jedné. Lejblíky se mohou rovněž pro různé seznamy překrývat.

Dejme tomu, že v článku prvním je seznam literatury, kde je TBN uvedeno pod číslem 3 a v článku druhém je seznam literatury, kde je TBN pod číslem 21. Když se v článku prvním objeví `\cite[tbn]`, musí se vytisknout odkaz [3] a správně prolinkovat hyperlink. Když se v článku druhém objeví `\cite[tbn]`, musí se vytisknout [21] a rovněž se odpovídajícím způsobem nastaví hyperlink na seznam literatury ve druhém článku.

Stačí před každým článkem pronulovat `\bibnum` a nastavit jednoznačný identifikátor `\bibpart`:

```
\bibnum=0 \def\citelist{} \def\bibpart{A} %%% A je identifikátor prvního článku
... první článek
...
\bibnum=0 \def\citelist{} \def\bibpart{B} %%% B je identifikátor druhého článku
... druhý článek
...
```

Aby toto fungovalo, je třeba mírně modifikovat makra OPmac následujícím kódem:

```
\def\wbib#1#2#3{\dest[cite:\bibpart-\the\bibnum]%
   \ifx\wref\wrefrelax\else \immediate\wref\Xbib{{\bibpart-#1}{\bibpart-#2}{#3}}\fi}

\let\citeAA=\citeA
\def\citeA #1#2,{\if#1,\def\citeAA##1,##2,{}\fi\citeAA \bibpart-#1#2,}

\def\printcite#1{\citesep \prepcite#1\relax
   \citelink{#1}{\tmpa}\def\citesep{,\hskip.2em\relax}}
\def\prepcite #1-#2{\def\tmpa{\citelinkA{#2}}}
\let\writeXcite=\addcitelist

\def\nonumcitations{\lastcitenum=0 \def\sortcitesA{}\def\etalchar##1{$^{##1}$}%
   \def\citelinkA##1{%  \citelinkA needs the \bibpart info:
      \isdefined{bim:\bibpart-##1}\iftrue \csname bim:\bibpart-##1\endcsname
      \else ##1%
      \opwarning{\noexpand\nonumcitations + empty bibmark. Maybe bad BibTeX style}\fi}
}
```

Makra přidávají před každý lejblík `\bibpart-` a stejný prefix přidávají před každé číslo. V REF souboru tedy máme z jednotlivých článků lejblíky i čísla prefixovány odpovídajícím způsobem. V rámci zpracování v `\cite` je pak tento prefix před lejblík přidán a z čísla odstraněn.

21.10 OPmac-bib: konec problémů s BibTeXem

0047
P. O.
19. 04. 2014

BibTeX z roku 1985 umí pracovat jen v ASCII kódování. Máte-li `.bib` databáze v tomto kódování (tj. případné akcenty máte přepsány TeXovsky), nemáte problém s užitím BibTeXu. Jakmile ale do těchto databází přidáte přímý akcentovaný znak nebo něco podobného, mohou nastat problémy. Především podle těchto znaků BibTeX neumí řadit. To částečně řeší varianta programu bibtex8, která pracuje v osmibitovém kódování a řadí podle pravidel deklarovaných v externím souboru `.csf`. Ovšem dnes se používá výhradně vícebytové kódování UTF-8, se kterým si tyto varianty BibTeXu neporadí. Navíc často mohou zmršit výstup, protože interpretují byte=znak. Takže když mají za úkol napsat např. jméno jen s iniciálním písmenem, může vzniknout po stránce kódování vadný soubor. Převádět `.bib` databáze do jednobytového

kódování a pak číst .bbl v tomto jednobytovém kódování by se dalo, ale jistě uznáte, že to není řešení.

Mezi různými záplatami BibTeXu existuje i varianta nazvaná `bibtexu`, která by měla umět UTF-8 vstup/výstup a uvnitř by měla řadit v Unicode s využitím ICU knihovny. Bohužel projekt zůstal opuštěn v ne zcela funkčním stavu (program se dosti často interně zacyklí).

Vedle BibTeXu existuje nový projekt Biber, který je ale pro uživatele plainTeXu *naprosto nepoužitelný*: opustil princip „v jednoduchosti je síla“, předpokládá konfigurační soubory i data v XML a spolupracuje s makrem biblatex. Toto makro produkuje nadbytek elektronického smetí v podobě XML souborů, jež následně čte Biber. Ten na oplátku vyrobí lidsky skoro nečitelný .bbl, kterému rozumí jen biblatex. Toto není cesta, kterou by měl následovat OPmac.

OPmac je schopen přímo číst `.bib` soubory (pchopitelně v UTF-8 kódování) a nahradit kompletně bibTeX. Slouží k tomu příkaz `\usebib`:

```
... \cite[cosi] a \cite[jiny].

Seznam:
\usebib/s (simple) mybase
\end
```

Formátování položek seznamu je deklarováno ve stylovém souboru `opmac-bib-simple.tex`. Podrobnosti jsou dokumentovány v souboru `opmac-bib.tex` a ve druhém stylovém souboru `opmac-bib-iso690.tex`.

22 Slovníček

22.1 Slovníček pojmů na konci dokumentu

0051 *P. O.* 11. 05. 2014

V textu dokumentu se mohou vyskytovat například zkratky, které je potřeba čtenáři vysvětlit, nejlépe na konci dokumentu ve speciální sekci. V místě použití zkratky napíšeme `\glos{zkratka}{vysvětlení}`, přitom v tomto místě se v sazbě nic neobjeví. Například:

```
Pracuji na ČVUT.\glos{ČVUT}{České vysoké učení technické v Praze}
```

Makro `\glos` si ukládá postupně informace do paměti a pak je na konci dokumentu vyvrhne v místě, kam napíšeme `\makeglos`. Makra `\glos` a `\makeglos` mohou vypadat takto:

```
\def\gloslist{}
\def\glos #1#2{%
   \expandafter\isinlist\expandafter\gloslist\csname;#1\endcsname
   \iftrue \opwarning{Duplicated glossary item ‘#1’}%
   \else
      \global\sdef{;#1}{{#1}{#2}}%
      \global\expandafter\addto\expandafter\gloslist\csname;#1\endcsname
   \fi
}
\def\makeglos{%
   \ifx\gloslist\empty \opwarning{Glossary data unavailable}%
   \else
      \bgroup
        \let\iilist=\gloslist
        \dosorting
        \def\act##1{\ifx##1\relax \else \expandafter\printglos##1\expandafter\act\fi}
        \expandafter\act\iilist\relax
      \egroup
   \fi
}
\newdimen\glosindent  \glosindent=2\parindent
\def\printglos #1#2{\noindent \hangindent=\glosindent \hbox to\glosindent{#1\hss-- }#2\par}
```

Slovníček bude abecedně seřazen podle zkratek. Chcete-li jej mít seřazen podle pořadí výskytu v textu, zakomentujte příkaz `\dosorting`.

Makro `\printglos` si můžete samozřejmě naprogramovat dle svého přání. Má dva parametry `{zkratka}{vysvětlení}` a jeho úkolem je vytisknout jeden údaj ve slovníčku. Výše je `\printglos` uděláno tak, že nastavuje pevné odsazení `2\parindent` pro všechny zkratky. To nemusí vyjít dobře. Je tedy možné údaj `\glosindent` nastavit jinak nebo použít myšlenku z OPmac triku 0041.

Jiný jednodušší příklad makra `\printglos`:

```
\def\printglos #1#2{\noindent #1 -- #2\par}
```

Vysvětlení makra. Makro `\glos` si ukládá do `\gloslist` postupně `\;zkratka \;zkratka` atd. a navíc definuje `\;zkratka` jako `{zkratka}{vysvětlení}`. Makro `\makeglos` lokálně převede `\gloslist` na `\iilist` a nechá jej seřadit jako rejstřík makrem z OPmac \dosorting. Nakonec jej postupným opakováním `\act` vytiskne.

22.2 Slovníček pojmů na začátku dokumentu

0052
P. O.
11. 05. 2014

Pokud chcete umístit `\makeglos` (z předchozího triku) na začátek dokumentu, máte dvě možnosti: nashromáždit všechny potřebné příkazy `\glos` před `\makeglos` na začátek dokumentu (protože na umístění příkazů `\glos` v dokumentu nezáleží) nebo použít REF soubor. Druhou možnost rozvedeme podrobněji.

Při prvním TeXování se data pro slovníček do REF souboru uloží a při opakovaném TeXování se použijí.

```
\def\gloslist{}
\def\Xglos #1#2{%
   \expandafter\isinlist\expandafter\gloslist\csname;#1\endcsname
   \iftrue \opwarning{Duplicated glossary item ‘#1’}%
   \else
      \sdef{;#1}{{#1}{#2}}%
      \expandafter\addto\expandafter\gloslist\csname;#1\endcsname
   \fi
}

\input opmac

\def\glos #1#2{\openref\toks0={#2}\immediate\wref\Xglos{{#1}{\the\toks0}}}
```

Povšimněme si, že definici pomocného makra `\Xglos` a výchozí nastavení `\gloslist` musíme umístit před `\input opmac`, protože opmac čte REF soubor z předchozího běhu, takže v té době už musí makro `\Xglos` znát.

Makro `\makeglos` zůstává beze změny, jako v předchozím OPmac triku.

22.3 Slovníček pojmů setříděný dle českých pravidel

0053
P. O.
11. 05. 2014

Slovníček z OPmac triku 0051 je abecedně seřazen podle ASCII hodnot zkratek. Někdy je to postačující (například v anglicky psaném dokumentu), ale někdy bychom rádi řadili podle českých pravidel řazení stejně, jak to umí OPmac v případě rejstříku. V takovém případě přidejte do makra `\makeglos` před `\dosorting` ještě příkaz `\preparespecialsorting` (ostatní zůstává nezměněno) a tento příkaz definujte následovně:

```
\def\makeglos{%
   ...
   \preparespecialsorting \dosorting
   ...
}
\def\preparespecialsorting{%
   \setprimarysorting
   \def\act##1{\ifx##1\relax\else \preparesorting##1%
      \expandafter\edef\csname\string##1\endcsname{\tmpb}%
      \expandafter\act\fi}%
```

```
   \expandafter\act\iilist\relax
   \def\firstdata##1{\csname\string##1\endcsname&}%
}
```

Nyní při `\chyph` bude řazení české a při `\ehyph` bude anglické.

Vysvětlení makra. OPmac řadí `\iilist` obsahující `\?cosia \?cosib \?cosic`, kde `?` je jakýkoli znak. V rejstříku to je `\,cosia \,cosib \,cosic` zatímco ve slovníčku to je `\;cosia \;cosib \;cosic` Tato makra musí osahovat `{první údaj}{druhý údaj}`, přitom OPmac řadí v prvním průchodu podle prvního údaje za použití makra `\firstdata`. Makro `\preparesorting \?cosi` připraví do `\tmpb` slovo makra (tj. např. cosi) zkonvertované přes `\lccode` do stavu vhodného pro české řazení. My si tento výsledek uložíme do `\\?cosi` a přesměrujeme tam i makro `\firstdata`, takže řazení v prvním průchodu probíhá podle `\\?cosi` a nikoli dle prvního údaje. Případný druhý průchod spustí dodatečnou konverzi slova (cosi) dle pravidel druhého průchodu a už to nikam neukládá, takže i druhý průchod funguje.

0054 P. O. 11. 05. 2014

22.4 Slovníček pojmů s hyperlinky

Pokud chcete, aby v textu v místě použití zkratky byla zkratka obarvena a fungovala jako hyperlink, je možné psát do textu:

```
...text... \glosref{zkratka}{význam} ...text... \glosref{jiná zkratka}{význam}
...text... \glosref{zkratka}{}
...
\makeglos
```

V textu budou pak vytištěny zkratky obarvené barvou a fungující jako hyperlinky odkazující do místa, kde je seznam zkratek vytištěný pomocí `\makeglos`. Povšimněte si, že pro každou zkratku je třeba zapsat význam jen jednou a při opakování stejné zkratky je třeba nechat význam prázdný. Kdyby byla zkratka významově určena vícekrát, makro napíše varování a opakovaný význam ignoruje.

Řešení může vypadat třeba takto:

```
\hyperlinks\Blue\Blue

\def\glosref #1#2{\if^#2^\else \glos{#1}{#2}\fi
   \expandafter\isinlist\expandafter\gloslist\csname;#1\endcsname
   \iftrue \makegloslink{#1}\link[glos:\tmp]{\localcolor\Blue}{#1}%
   \else #1%
   \fi
}
\def\printglos#1#2{\noindent \makegloslink{#1}\dest[glos:\tmp]#1 .. #2\par}

\def\makegloslink#1{\def\tmp{}\expandafter\makegloslinkA#1\relax}
\def\makegloslinkA#1{\ifx#1\relax\else
   \edef\tmp{\tmp\number`#1.}\expandafter\makegloslinkA\fi}
```

Řešení předpokládá některou z definic maker `\glos` a `\makeglos`, jak je uvedeno výše. Makro `\glosref` kontroluje, zda je druhý parametr prázdný a když není, použije `\glos`. Tam se zkontroluje, zda není význam deklarován vícekrát. Dále makro `\glosref` zavede hyperlink pomocí `\link` jen tehdy, když je údaj zanesen do `\gloslist` (což udělalo makro `\glos`, nebo se tak stane až při opakovaném TeXování). Důvod tohoto opatření: je třeba zabezpečit, aby link měl svůj cíl, což při prvním průchodu TeXem nemusí být pravda.

Makro `\printglos` *musí* obsahovat deklaraci cíle pomocí `\dest`.

Interní link je vytvořen ze zkratky pomocí `\makegloslink`, ale není to přímo zkratka, protože ta může obsahovat háčky a čárky a link pak nefunguje. Takže je každé píseno převedeno na číslo (svůj kód) a seznam takových kódů oddělených tečkami je použit jako interní link.

Pomocí `\def\glosborder{R G B}`, například `\def\glosborder{1 0 0}` je možno definovat barvu rámečků pro aktivní hyperlinky týkající se zkratek.

Poznámka: Hyperlinky v tomto řešení fungují od prvního výskytu hesla s neprázdným druhým parametrem. Předchází-li totéž heslo s prázdným parametrem, není pro něj sestaven hyperlink. Co s tím? Je možné například na začátek dokumentu uvést celý soubor dat pomocí `\glos{zkratka}{význam} \glos{jiná zkratka}{jiný význam}` atd. a poté do textu důsledně psát `\glosref` s prázdným druhým parametrem.

22.5 Akronymy

0113
P. O.
25. 06. 2015

Pohrajeme si s makrem `\ac[lejblík]`, které při prvním výskytu v dokumentu vytiskne `dlouhý název (zkratka)` a při každém dalším výskytu vytiskne jen `zkratka`. K tomu účelu je potřeba nejprve akronym na začátku dokumentu definovat pomocí

```
\acrodef [lejblík] {zkratka} {dlouhý název}
```

Explicitně zkratku vytiskneme pomocí `\acs[lejblík]` a dlouhý název pomocí `\acl[lejblík]`. Seznam všech zkratek (ve stejném pořadí, jak byly definovány na začátku dokumentu) vytiskneme pomocí `\acrolist`, což můžeme umístit kamkoli do dokumentu, ovšem až za sadu definic `\acrodef`. Na stránku, kde je poprvé akronym použit, je možné se odkazovat pomocí `\pgref[acro:lejblík]`.

Chceme-li vytisknout akronym s velkým písmenem na začátku, je možné použít `\Ac[lejblík]`. Chceme-li akronym s nějakou koncovkou, musíme kromě `\acrodef[lejblík]` definovat obdobně `\acrodefx[lejblík-koncovka]` a v textu použít `\ac[lejblík-koncovka]`. Příklad:

```
\acrodef  [CVUT]    {ČVUT} {České vysoké učení technické v Praze}
\acrodefx [CVUT-em] {ČVUT} {Českým vysokým učením technickým v Praze}
\acrodef  [UK]      {UK}   {Univerzita Karlova}
\acrodef  [voda]    {$\rm H_2O$} {voda}

Tady se seznámíme s \ac[CVUT-em], vedle kterého je \ac[UK]. Ještě jednou \ac[CVUT] a taky \ac[UK].
\Ac[voda] je základ života. Takže \ac[voda] se dá použit opakovaně.

\acrolist
```

Sada maker na akronymy může vypadat třeba takto:

```
\def\acrolist{}
\def\acrodef[#1]#2#3{\sdef{as:#1}{#2}\sdef{al:#1}{#3}\addto\acrolist{\acroitem{#1}}}
\def\acrodefx[#1-#2]#3#4{\sdef{as:#1-#2}{#3}\sdef{al:#1-#2}{#4}}
\def\acs[#1]{\isdefined{as:#1}\iftrue \acroprint{as:#1}\else\acno{#1}\fi}
\def\acl[#1]{\isdefined{al:#1}\iftrue \acroprint{al:#1}\else\acno{#1}\fi}
\def\acno#1{\opwarning{acro [#1] undefined}ac?#1}
\def\ac[#1]{\acX#1-\end
   \isdefined{ad:\acA}\iftrue \acs[\acA\acB]\else
   \label[acro:\acA]\openref\wlabel{}\global\sdef{ad:\acA}{}%
   \acl[\acA\acB] \let\acrocap=\undefined(\acs[\acA\acB])\fi}
\def\acX#1-#2\end{\def\acA{#1}\ifx\end#2\end \def\acB{}\else\acY#2\fi}
\def\acY#1-{\def\acB{-#1}}
\def\acroprint#1{{\expandafter\expandafter\expandafter\acroprintA\csname#1\endcsname\end}}
\def\acroprintA#1#2\end{\ifx\acrocap\undefined\expandafter\acroprintB\else\uppercase\fi{#1}#2}
\def\acroprintB#1{#1}
\def\Ac[#1]{{\let\acrocap=\active\ac[#1]}}
\def\acroitem#1{\par\noindent{\bf\boldmath\acs[#1]} -- \acl[#1] \dotfill \pgref[acro:#1]\par}
\addprotect\acs \addprotect\acl
```

23 Rejstřík

23.1 Využití řadicího algoritmu rejstříku pro jiné účely

0080
P. O.
10. 12. 2014

OPmac má řadicí algoritmus zabudován do maker pro setavení rejsříku a není zcela snadné jej od rejstříku oddělit. V tomto triku to uděláme. Vytvoříme makro `\sort`, které přečte své agumenty, abecedně je zatřídí a výsledek uloží do pomocného makra `\tmpb`. Tedy:

```
\sort{uu}{tt}{zz}{aa}\relax
nyní makro \tmpb obsahuje: {aa}{tt}{uu}{zz}
```

Sadu parametrů makra `\sort` je nutné ukončit příkazem `\relax`. V parametrech nesmí být obsažen příkaz, který je neexpandovatelný (analogicky jako v parametru makra `\iindex`). Implementace může vypadat takto:

```
\def\sort{\begingroup\setprimarysorting\def\iilist{}\sortA}
\def\sortA#1{\ifx\relax#1\sortB\else
  \expandafter\addto\expandafter\iilist\csname,#1\endcsname
  \expandafter\preparesorting\csname,#1\endcsname
  \expandafter\edef\csname,#1\endcsname{{\tmpb}{}}%
  \expandafter\sortA\fi
}
\def\sortB{\def\message##1{}\dosorting
  \def\act##1{\ifx##1\relax\else \expandafter\sortC\string##1\relax \expandafter\act\fi}%
  \gdef\tmpb{}\expandafter\act\iilist\relax
  \endgroup
}
\def\sortC#1#2#3\relax{\global\addto\tmpb{{#3}}}
```

Makro `\sort` pracuje ve skupině `\begingroup...\endgroup`, aby neovlivnilo data, která se používají v rámci rejsříku. Parametry připraví lokálně do `\iilist` a spustí třídicí algoritmus `\dosorting`. Výsledek řazení ukládá globálně do `\tmpb`.

Povšimněte si, že makro definuje druhé datové pole sekvencí `\,slovo` jako prázdné (na předposledním řádku `\sortA`). Toto pole můžete využít pro ukládání vlastních dat, která je pak možné zpětně vytěžit po zatřídění pomocí makra `\seconddata` z OPmac.

0006

23.2 Klikací stránky v rejstříku

P. O.
13. 08. 2013

Při vytvoření rejstříku pomocí makra `\makeindex` nejsou vedle hesel čísla stránek aktivní, ačkoli je použita deklarace `\hyperlinks`. Aktivní (připravené ke klikání) jsou jen čísla stránek v obsahu.

Důvod, proč OPmac neimplementuje aktivní čísla stránek v rejstříku jsou dva: drží se kréda „v jednoduchosti je síla“ a šetří pamětí TeXu. Za každý odkaz na stránku by totiž do makroprostoru musel přidat tři tokeny. Při tisíci heslech a průměrně třech odkazech u hesla dostáváme 9 tisíc tokenů navíc. Tomu se chce OPmac vyhnout.

Pokud přesto chcete mít čísla stránek v rejstříku aktivní, stačí použít tento kód:

```
\def\printiipages#1&{\usepglinks#1\relax\par}
\def\usepglinks{\afterassignment\usepglinksA \tmpnum=}
\def\usepglinksA{\pglink{\the\tmpnum}\futurelet\next\usepglinksB}
\def\usepglinksB{\ifx\next\relax \def\next{}\else
   \ifx\next,\def\next,{, \afterassignment\usepglinksA \tmpnum=}\else
   \ifx\next-\def\next--{--\afterassignment\usepglinksA\tmpnum=}%
   \fi\fi\fi\next}
```

Makro `\usepglinks` má parametr např. `5, 15--18, 24` ukončený `\relax`. Makro očichá čísla ve svém parametru a každé z nich promění v `\pglink{číslo}`.

0072

23.3 Alternativní seznamy stránek v rejstříku

P. O.
22. 06. 2014

Někdy je potřeba mít vedle rejstříkových hesel více seznamů stránek. Jeden seznam základní (většinou tištěn antikvou) a potom třeba zvýrazněné seznamy tištěné kurzívou nebo tučně. Najdeme to například v rejstříku TeXbooku, kde kurzívou nebo podtrženě jsou vytištěny výskyty hesla, které jsou nějakým způsobem významnější.

Od verze OPmac Jun. 2014 je k dispozici makro `\Xindexg{prefix}{heslo}{strana}`, které lze použít v REF souborech podobně, jako `\Xindex{heslo}{strana}`. Toto makro vytváří alternativní seznamy stránek uložené v sekvencích `\prefixheslo` a seznamy těchto sekvencí ukládá do `\iilist:prefix`. Seznam stránek je zpracován stejně jako standardní `\,heslo`, tj.

data jsou ve dvou částech, v první je příznak zpracování a ve druhé je komprimovaný seznam stránek (např. `3,4,4,5` se redukuje na `3--5`).

Jako příklad vytvoříme makro `\iindexb{heslo}`, které uloží výskyt hesla do alternativního seznamu stránek prefixovaný pomocí `b:`. Pokud heslo bude mít tento seznam, budeme jej tisknout v rejstříku před obecný seznam stránek tučně a červeně.

```
\def\iindexb#1{\openref\wref\Xindexg{{b:}{#1}{\the\pageno}}}
\def\printiipages#1&{ % second space between word and pagelist
   \expandafter \isinlist \csname iilist:b:\expandafter\endcsname \csname b:\currii\endcsname
   \iftrue
      \expandafter\seconddata \csname b:\currii\endcsname \XindexB
      {\localcolor\Red \bf \tmp}, \fi
   #1\par
}
```

V prvním řádku je definováno `\iindexb` tak, že výskyt hesla pošle do alternativního seznamu prefixovaného `b:`. Dále je předefinováno makro `\printiipages`, které dostane v parametru `#1` obecný seznam stránek a má je vytisknout. Pomocí `\isinlist \iilist:b: \b:heslo \iftrue` se zeptáme, zda heslo má alternativní seznam stránek. Pokud ano, vložíme `{\localcolor\Red \bf \tmp},` přitom v `\tmp` je uložen druhý údaj z `\b:heslo` pomocí `\seconddata \b:heslo \Xindex`.

Uvedný příklad bude fungovat jen v případě, že heslo má i svůj normální výskyt, protože do rejstříku jsou zařazena a abecedně setříděna jen hesla s normálním výskytem. Dále se zde předpokládá, že seznam stránek prefixovaný pomocí `b:` bude uzavřen, nebo obsahuje jen izolované stránky. Není-li to pravda, je potřeba seznam nejprve uzavřít. Oba tyto problémy řeší následující modifikace makra.

```
\def\iindexb#1{\openref\wref\Xindexg{{b:}{#1}{\the\pageno}}\wref\Xindex{{#1}{}}}
\def\printiipages#1&{ % second space between word and pagelist
   \expandafter \isinlist \csname iilist:b:\expandafter\endcsname \csname b:\currii\endcsname
   \iftrue
      \expandafter\firstdata  \csname b:\currii\endcsname \XindexA
      \expandafter\seconddata \csname b:\currii\endcsname \XindexB
      {\localcolor\Red \bf \tmp\ifx\tmpb-\pgfolioA{\tmpa}\fi}%
      \if^#1^\else, \fi % comma only if the next list isn't empty
   \fi
   #1\par
}
```

23.4 Heslo pro rejstřík dané rozsahem od–do

0073
P. O.
24. 06. 2014

Někdy je účelné říci, že dané heslo se vyskytuje na všech stránkách určité kapitoly nebo na všech stránkách od tohoto výskytu po tento výskyt. Nemusí to být vždy úplně pravda, ale to nebudeme řešit. Podstatné je, že uvnitř tohoto rozsahu nemusíme jednotlivě označovat výskyty hesla pro rejstřík. Navrhneme tedy dvojici maker `\iindexbeg{heslo} ... \iindexend{heslo},` která vyznačují výchozí a konečnou stránku výskytu hesla, přitom se předpokládá, že se heslo vyskytuje na všech stránkách mezi nimi. V seznamu stránek s výskytem hesla bude tento interval stránek zahrnut a sloučen s ostatními stránkami vyznačujícími jednotlivé další výskyty hesla.

```
\def\Xindexgend#1#2#3{\bgroup \def~{ }% #1=prefix, #2=index-item, #3=pageno
   \expandafter\firstdata \csname#1#2\endcsname \XindexA
   \expandafter\seconddata \csname#1#2\endcsname \XindexB
   \ifnum#3=\tmpa \else
      \if\tmpb+%
         \sxdef{#1#2}{{#3/-}{\tmp\iiendash}}
      \else
         \sxdef{#1#2}{{#3/-}{\tmp}}
   \fi\fi \egroup
}
\input opmac
```

```
\def\iindexbeg#1{\iindex{#1}\expandafter\iimute\csname,#1\endcsname}
\def\iindexend#1{\expandafter\isinlist\expandafter\iimutelist\csname,#1\endcsname\iftrue
   \wref\Xindexgend{,{#1}{\the\pageno}}\expandafter\iiunmute\csname,#1\endcsname \fi
}
\def\iindex#1{\expandafter\isinlist\expandafter\iimutelist\csname,#1\endcsname\iftrue
   \else \openref\wref\Xindex{{#1}{\the\pageno}}\fi
}
\def\iimute#1{\global\addto\iimutelist#1}
\def\iiunmute#1{\def\tmp##1#1##2\end{\gdef\iimutelist{##1##2}}\expandafter\tmp\iimutelist\end}
\def\iimutelist{}
```

Pomocné makro `\Xindexgend` budeme zapisovat do REF souboru, a proto je potřebujeme definovat před `\input opmac`. Makro `\iindexbeg` pouze zapíše výskyt hesla pomocí `\iindex` a dále umlčí makrem `\iimute` všechny pokusy typu `\iindex{heslo}` vyskytující se uvnitř rozsahu `\iindexbeg{heslo}` až `\iindexend{heslo}`. Tyto pokusy by nám totiž mohly rozbít souvislý sled stránek. Makro `\iindexend{heslo}` vloží do REF souboru `\Xindexgend,{heslo}{strana}` a toto makro při zpracování REF souboru otevře uzavřený seznam stránek od výchozí strany po aktuální nebo protáhne otevřený seznam stránek až po aktuální stranu.

Makro `\iindex` je předefinováno tak, aby do REF souboru zapisovalo jen hesla, která nejsou na seznamu `\iimutelist`. A konečně makra `\iimute` a `\iiunmute` přidávají a odebírají heslo ze seznamu `\iimutelist`.

0018

23.5 Experiment s lexikografickým řazením

P. O.
26. 08. 2013

PdfTeX implementuje primitiv `\pdfstrcmp{string1}{string2}`, který vrátí nulu, když jsou stringy stejné, vrátí `-1`, když `string1` je lexikograficky menší než `string2`, a vrátí `1` jinak. OPmac řeší abecední řazení rejstříku v makru `\isAleB` na úrovni klasických nástrojů TeXu, tj. bez využití tohoto primitivu. Kvůli tomu postupně expanduje až do čtyř různých rekurzivních maker, odlupuje písmenka a porovnává je. Stálo za experiment vyměnit tato makra v OPmac za jednodušší makro, které využije `\pdfstrcmp`:

```
\def\isAleB #1#2{%
   \edef\tmp{\noexpand\pdfstrcmp{\firstdata#1\empty}{\firstdata#2\empty}}
   \ifnum\tmp<0 \AleBtrue
   \else \ifnum\tmp>0 \AleBfalse
   \else \bgroup \setsecondarysorting
            \preparesorting#1\let\tmpa=\tmpb \preparesorting#2%
            \edef\tmp{\noexpand\pdfstrcmp{\tmpa}{\tmpb}}%
            \ifnum\tmp<0 \global\AleBtrue \else \global\AleBfalse \fi
         \egroup
   \fi\fi
}
```

Ukazuje se, že jsme si překvapivě ve strojovém čase moc nepomohli. Řazení ukázky `kuk8.tex` z přednášky z prosince 2012 obsahující přes šest tisíc hesel trvalo s klasickými makry 4,13 sekundy a s využitím `\pdfstrcmp` 4,04 sekundy. Je vidět, že je v TeXu expanze maker implementována velmi efektivně a ani vestavěná funkce na lexikografické třídění ji o mnoho nepřekoná.

0140

23.6 Rozdělení rejstříku do oddílů nadepsaných písmeny

P. O.
08. 06. 2016

Slova v restříku chceme mít oddělena podle jejich prvního písmene do oddílů, každý oddíl má být uvozen „nadpisem“, který obsahuje zvětšené písmeno, kterým všechna slova daného oddílu začínají.

Řešení se opírá o makro `\everyii`, které je implicitně prázdné a OPmac ho vkládá před tisk každého hesla v rejstříku.

```
\def\lastchar{}
\def\everyii{\expandafter\makecurrchar\currii\end
```

```
   \ifx\currchar\lastchar \else
      \bigskip{\typosize[20/24]\bf\currchar}\par\nobreak\medskip\noindent
      \let\lastchar=\currchar
   \fi
}
\def\makecurrchar#1#2\end{\uppercase{\def\currchar{#1}}}
```

Na začátku je `\lastchar` prázdné. Před vytištěním každého slova se zjistí `\currchar` jako první písmeno slova zvětšené pomocí `\uppercase`. Je-li `\currchar` různé od `\lastchar`, vytiskneme záhlaví dalšího oddlílu a nastavíme nově `\lastchar` podle `\currchar`.

24 Meta

24.1 Makra OPmac vložená do formátu

0139
P. O.
12. 03. 2016

Aby uživatel neusel na začátku dokumentu neustále psát `\input opmac`, může si vygenerovat formát. Například `.ini` soubor pro csplain + OPmac s výstupem do PDF může vypadat takto:

```
\input csfonts    % implicitní fonty budou CSfonty
\input plain      % Knuthův plain.tex
\restorefont      % obnova původního významu primitivu font
\input csfontsm   % základní makra pro zvětšování písma
\input il2code    % výchozí kódování IL2
\input etex-mac   % eTeXové rozšíření pro alokace registrů
\input hyphen.lan % vzory dělení slov
\input plaina4    % implicitní strana A4
\input csenc-u    % UTF-8 kódování vstupu
\pdfoutput=1      % implicitní výstup do PDF
\input opmac      % makra OPmac
\everyjob=\expandafter{\the\everyjob
   \message{The format: csplain + OPmac, PDF output}%
   \inputref}
\dump
generovat pomocí:  pdftex -ini -etex -enc tentosoubor.ini
```

Protože OPmac během `\input opmac` načítá .ref soubor (pokud existuje) pomocí makra `\inputref`, je třeba toto makro přidat do `\everyjob`. Pak nebude muset uživatel formátu psát na začátek souboru zhola nic.

Chcete-li kombinovat OPmac s `etex.src` formátem, pak příslušný `.ini` soubor může vypadat takto:

```
\pdfoutput=1
\input etex.src
\input opmac
\everyjob=\expandafter{\the\everyjob
   \message{The format: etex.src + OPmac, PDF output}%
   \inputref}
\dump
generovat pomocí:  pdftex -ini -etex tentosoubor.ini
```

24.2 Zobrazení seřazených názvů triků z této stránky

0143
Radek Matouše
11. 12. 2015

Třeba uvítáte možnost si pomocí příkazového řádku zobrazit v terminálu seznam názvů všech triků seřazených podle jejich čísla, tedy relativně od nejstarších k nejnovějším. Můžete k tou použít tento příkaz:

```
curl http://petr.olsak.net/opmac-tricks.html | tac \
 | sed -n '/CLASS.*datum/{:a;N;/<h2><a/!ba;s/\n.*\n//; s/<a[^>]*>/ : /;s/<[^>]*>//g;p}' | sort
```

Tuto věc zveřejnil autor na své stránce v článku s názvem Opmac-triky – sort.