verze Dec. 2012

OPmac – rozšiřující makra plainTEXu

Petr Olšák

www.olsak.net/opmac.html

Obsah

1 Úvod

OPmac je balík jednoduchých doplňujících maker k plainTEXu umožňující uživatelům základní LaTEXovou funkcionalitu: změny velikosti písma, automatickou tvorbu obsahu a rejstříku, práci s `bib` databázemi, referencemi, možnost proložení referencí hyperlinkovými odkazy atd.

2 Uživatelská dokumentace

Uživatelská dokumentace je zatím v souboru `opmac-u.tex` a `opmac-u.pdf`. Do tohoto místa ji zahrnu později a prolinkuji ji s technickou dokumentací.

3 Technická dokumentace

Tato část dokumentace je určena pro tvůrce maker, kteří se chtějí zde uvedenými makry inspirovat a případně je přizpůsobit svému požadavku. Předpokládá se znalost TEXu, tj. například aspoň zběžná orientace v TEXbooku naruby. Na tuto knihu je na mnoha místech odkazováno pod zkratkou TBN.

3.1 Základní makra

Na začátku souboru `opmac.tex` zjistíme, zda není soubor čtený podruhé. V takovém případě čtení odmítneme. Ptáme se na to, zda je definováno makro \OPmacversion, které vzápětí definujeme. Je-li někdo překvapen, proč jsem nepoužil `\expandafter\endpinput\fi`, může si prostudovat TBN, stranu 358, heslo `\endinput`.

opmac.tex
```
\ifx\OPmacversion\undefined \else \endinput \fi
\def\OPmacversion{Dec. 2012}
```

Dva pracovní registry:

opmac.tex
```
\newcount\tmpnum % auxiliary count
\newdimen\tmpdim % auxiliary dimen
```

OPmac nebude nikdy hlásit chyby. Často ale bude psát pomocí \opwarning na terminál varování.

opmac.tex
```
\def\opwarning#1{\immediate\write16{l.\the\inputlineno\space OPmac WARNING: #1.}}
```

Makro \addto ⟨*makro*⟩{⟨*tokeny*⟩} přidá na konec ⟨*makra*⟩ dané ⟨*tokeny*⟩.

opmac.tex
```
\def\addto#1#2{\expandafter\def\expandafter#1\expandafter{#1#2}}
```

V OPmac budeme pracovat se seznamem \protectlist, který bude obsahovat makra, jež chceme mít tzv. robustní, tj. chceme, aby se při `\write` v output rutině neexpandovala. Každému makru v seznamu předchází \doprotect, takže seznam \protectlist vypadá takto:

```
\doprotect⟨makro1⟩ \doprotect⟨makro2⟩ ...
```

Seznam budeme spouštět v output rutině s tím, že \doprotect tam bude mít význam makra, které zařídí, aby jeho parametr získal význam `\relax`. Tím bude zabráněno jeho expanzi. Naprogramujeme \addprotect ⟨*makro*⟩, které zařídí vložení ⟨*makra*⟩ do seznamu.

\OPmacversion: 3 \tmpnum: 16, 20, 24–25, 29–30, 32–33, 37, 39 \tmpdim: 6–8, 27, 35–37, 45
\opwarning: 3, 6, 11, 13, 15, 18, 20, 27, 29–32, 34, 36–38, 40, 42–43, 45 \addto: 3–4, 15–16, 18, 23, 34, 41, 44–45 \protectlist: 3–4, 30, 44 \addprotect: 4, 6, 9, 30, 44

opmac.tex
```
\def\protectlist{}
\def\addprotect#1{\addto\protectlist{\doprotect#1}}
\addprotect~
```

Některá makra budou fungovat jen v pdfTEXu při nastaveném `\pdfoutput=1`. Připravíme si tedy test `\ifpdftex`, který pak použijeme při čtení souboru `opmac.tex`. Test nikdy nebudeme vkládat do maker, takže při čtení souboru `opmac.tex` už musí být jasné, zda bude výstup směrován do DVI nebo PDF. Pozdější změna `\pdfoutput` může způsobit potíže.

opmac.tex
```
\newif\ifpdftex  \pdftextrue
\ifx\pdfoutput\undefined \pdftexfalse
\else \ifnum\pdfoutput=0 \pdftexfalse \fi \fi
```

Makra `\sdef` a `\sxdef` umožňují pohodlně definovat kontrolní sekvence ohraničené pomocí `\csname...\endcsname`.

opmac.tex
```
\def\sdef#1{\expandafter\def\csname#1\endcsname}
\def\sxdef#1{\expandafter\xdef\csname#1\endcsname}
```

Makro `\adef` umožní nastavit znak na aktivní a rovnou ho definovat, což normálně uvnitř maker není jednoduché (TBN str. 25 a 26). Využijeme toho, že ~ je aktivní znak a pomocí `\lccode` a `\lowercase` jej přepíšeme na požadovaný znak. Dostaneme tím aktivní token s požadovanou ASCII hodnotou a tento token definujeme. Pomocí `\lccodetiezero` vrátíme po přiřazení (tj. po provedení definice) `\lccode` vlnky na původní hodnotu.

opmac.tex
```
\def\adef#1{\lccode`\~=`#1\catcode`#1=13
   \afterassignment\lccodetiezero
   \lowercase{\def~}%
}
\def\lccodetiezero{\lccode`\~=0 }
```

Makrem `\isdefined {⟨jméno⟩}\iftrue` se ptáme, zda je definovaná `\csname`⟨*jméno*⟩`\endcsname`. To závěrečné připojené `\iftrue` makro sežere, ale uživatel ho píše zejména z toho důvodu, aby mu tato konstrukce fungovala uvnitř vnořených `\if..\fi`

opmac.tex
```
\def\isdefined #1#2{\expandafter\ifx \csname#1\endcsname \relax
     \csname iffalse\expandafter\endcsname
   \else
     \csname iftrue\expandafter\endcsname
   \fi
}
```

Makro `\isinlist` ⟨*list*⟩`{`⟨*tokeny*⟩`}\iftrue` zjistí, zda ⟨*tokeny*⟩ jsou (jako string) obsaženy v makru ⟨*list*⟩. Přitom sežere `\iftrue` ze stejných důvodů, jak je uvedeno před chvílí.

opmac.tex
```
\def\isinlist#1#2#3{\def\tmp##1#2##2\end{\def\tmp{##2}%
   \ifx\tmp\empty \csname iffalse\expandafter\endcsname \else
                  \csname iftrue\expandafter\endcsname \fi}%
   \expandafter\tmp#1#2\end
}
```

Makro `\isnextchar` ⟨*znak*⟩`{`⟨*co-dělat-při-ano*⟩`}{`⟨*co-dělat-při-ne*⟩`}` pracuje poněkud odlišně od předchozích maker. Zjistí, zda následující znak je ⟨*znak*⟩ a pokud ano, vykoná vnitřek první závorky, jinak vykoná vnitřek druhé závorky. Pomocí `\futurelet` uloží zkoumaný znak do `\next` a spustí `\isnextcharA`.

opmac.tex
```
\def\isnextchar#1#2#3{\def\tmpa{#2}\def\tmpb{#3}%
   \let\tmp=#1\futurelet\next\isnextcharA
}
\def\isnextcharA{\ifx\tmp\next\expandafter\tmpa\else\expandafter\tmpb\fi}
```

Předefinujeme makro \uv z CSplainu. Tam je toto makro navrženo tak, aby mohlo mít za svůj parametr verbatim text. Důsledkem toho nefunguje správně kerning. Považuji za lepší mít správně kerning a případné uvozování verbatim textů řešit třeba pomocí \clqq...\crqq.

opmac.tex
```
\def\uv#1{\clqq#1\crqq}
```

Knuth v souboru plain.tex zanechal řídicí sekvenci \\ v provizorním stavu (cvičení: podívejte se v jakém). Domnívám se, že je lepší ji dát jednoznačný význam \undefined. Některým uživatelům totiž může OPmac připomínat LaTeX a není tedy vyloučeno, že je napadne psát \\. Měli by na to dostat jednoznačnou odpověď: undefined control sequence.

opmac.tex
```
\let\\=\undefined
```

Do pracovního souboru určeného k novému načtení budeme chtít vložit komentáře za znakem procento. K tomu potřebujeme mít procento jako obyčejný znak kategorie 12. Na tento znak se v našem kódu překlopí otazník, Takže \percent expanduje na znak procento s kategorií 12.

opmac.tex
```
{\lccode`\?=`\%  \lowercase{\gdef\percent{?}}}
```

Makro plainTeXu \, funguje jen v matematické sazbě. Uživatel bude chtít makro často použít například mezi číslem a jednotkou v textovém módu: 5\,mm, takže makro předefinujeme.

opmac.tex
```
\def\,{\ifmmode \mskip\thinmuskip \else \kern.166em \fi}
```

3.2 Globální parametry

Zakážeme vdovy a sirotky a dále nastavíme registry pro listingy tiskového materiálu na smysluplnější hodnoty, než jsou implicitní.

opmac.tex
```
\widowpenalty=10000
\clubpenalty=10000
\showboxdepth=7
\showboxbreadth=30
```

Následující makra a registry ovlivní chování klíčových maker OPmac způsobem, jak je popsáno v komentářích. Mnohé z těchto maker a registrů byly zmíněny v uživatelské dokumentaci.

opmac.tex
```
\newdimen\iindent  \iindent=\parindent
   % indentation of items, TOC, captions, list of bib. references
\newdimen\ttindent \ttindent=\parindent
   % indentation in \begtt...\endtt and \verbinput

\def\ttskip{\medskip}       % space above and below \begtt, \verbinput
\mathchardef\ttpenalty=100  % penalty between lines in \begtt, \verbinput
\def\tthook{}               % hook in \begtt, \verbinput
\def\intthook{}             % hook in in-text verbatim

\def\iiskip{\medskip}     % space above and below \begitems...\enditems
\def\bibskip{\smallskip}  % space between bibitems

\def\tabstrut{\strut}     % strut in the \table
\def\tabiteml{\enspace}   % left material before each \table item
\def\tabitemr{\enspace}   % right material after each \table item
\def\vvkern{1pt}          % space between vertical lines
\def\hhkern{1pt}          % space between horizontal lines

\def\multiskip{\medskip}      % space above and below \begmulti...\endmulti
\newdimen\colsep \colsep=2em  % space between columns

\newdimen\mnoteindent \mnoteindent=10pt % ditance between mnote and text
\newdimen\mnotesize   \mnotesize=20mm   % the width of the mnote paragraph
```

\uv: 5 \percent: 5, 10, 42 \iindent: 5, 13–15, 19, 41–43 \ttindent: 5, 31–33 \ttskip: 31, 33
\ttpenalty: 31, 33 \tthook: 31–33 \intthook: 31 \iiskip: 14 \bibskip: 41–43
\tabstrut: 33–35 \tabiteml: 34 \tabitemr: 34 \vvkern: 34–36 \hhkern: 35–36
\multiskip: 24 \colsep: 5, 24–25 \mnoteindent: 5, 38 \mnotesize: 5, 38

```
91:
92: \def\picdir{}     % the directory with picture files
93: \def\bibtexhook{} % hook in \usebibtex and \usebbl macros
94: \def\chaphook{}   % hook in \chap
95: \def\sechook{}    % hook in \sec
96: \def\secchook{}   % hook in \secc
97: \def\cnvhook{}    % hook before conversion of outlines
98: \def\pghook{}     % hook in \output rutine
```

3.3 Loga

V logu \OPmac je pomocí \thefontscale zvětšeno písmeno O. Logo \CS je přepsáno beze změny z CSTEXu. Tím snadno vytvoříme i logo \csplain.

opmac.tex

```
103: \def\OPmac{\leavevmode
104:    \lower.2ex\hbox{\thefontscale[1400]O}\kern-.86em P{\em mac}}
105: \def\CS{$\cal C$\kern-.1667em\lower.5ex\hbox{$\cal S$}}
106: \def\csplain{\CS plain}
```

Troufám si tvrdit, že logo \LaTeX (ačkoli je plainTEXisté asi moc nebudou potřebovat) je v následujícím kódu daleko lépe řešeno, než v samotném LaTEXu. Počítá totiž ve spolupráci s makrem \slantcorr i se sklonem písma při usazování zmenšeného A.

opmac.tex

```
108: \def\LaTeX{\tmpdim=.42ex L\kern-.36em \kern\slantcorr % slant correction
109:   \raise\tmpdim\hbox{\thefontscale[710]A}%
110:   \kern-.15em \kern-\slantcorr \TeX}
111: \def\slantcorr{\expandafter\ignorept\the\fontdimen1\the\font\tmpdim}
```

Loga se občas mohou vyskytnout v nadpisech. Zabezpečíme je tedy proti rozboření při zápisu do REF souboru.

opmac.tex

```
113: \addprotect\TeX   \addprotect\OPmac   \addprotect\CS  \addprotect\LaTeX
```

3.4 Velikosti fontů, řádkování

CSplain od verze ⟨*Nov.-2012*⟩ definuje makro \resizefont ⟨*fontselector*⟩, které změní velikost fontu daného svým přepínačem a tento změněný font si ponechá stejný přepínač. Změna velikosti je dána obsahem makra \sizespec. Tam může být například napsáno `at13pt` nebo `scaled800`. Dále CSplain definuje makro \resizeall, které změní velikost fontů s registrovanými přepínači. Registrování se provádí makrem \regfont. Implicitně jsou registrovány přepínače \tenrm, \tenit, \tenbf, \tenbi, a \tentt. Do nových velikostí tedy půjdeme se starými názvy přepínačů \ten⟨*něco*⟩ a to slovo `ten` budeme chápat jen jako historický relikt, který nám ovšem napoví, že kontrolní sekvence je fontovým přepínačem.

OPmac si zjistí, zda je definovaný \regfont. Pokud ne, upozorní na starou verzi CSplainu na terminálu a potřebná makra si definuje. Je to kopie kódu ze souboru `csfontsm.tex` z balíčku CSplain.

opmac.tex

```
118: \ifx\regfont\undefined
119:   \opwarning{csplain version <Nov. 2012> or later is recommended}
120:   % macros from csplain, file csfontsm.tex:
121:   \font\tenbi=csbxti10 \def\bi{\tenbi}
122:   \def\sizespec{}
123:   \def\resizefont #1{\expandafter
124:     \font\expandafter#1\expandafter\resizefontskipat\fontname#1 \relax}
125:   \def\regfont#1{\expandafter\def\expandafter\resizeall\expandafter{%
126:     \resizeall \resizefont#1}}
127:   \def\resizeall{}
128:   \regfont\tenrm \regfont\tenit \regfont\tenbf \regfont\tenbi \regfont\tentt
129: \fi
```

Implicitně jsou zavedeny CSfonty, takže k nim přidáme AMS fonty z `ams-math.tex`, které vizuálně odpovídají. Později si může uživatel zavést jiné makro (např. `tx-math.tex`) a zavede si třeba i jiné

\picdir: 36 \bibtexhook: 42 \chaphook: 12 \sechook: 12 \secchook: 12 \cnvhook: 30
\pghook: 44–45 \OPmac: 6 \CS: 6 \csplain: 6 \LaTeX: 6 \slantcorr: 6
\resizefont: 6–8 \sizespec: 6–8 \resizeall: 6–8 \regfont: 6

textové fonty. To nezmění vlastnosti maker v OPmac, pokud nové soubory maker správně předefinují makra `\setmathsizes[⟨text⟩/⟨script⟩/⟨scriptscript⟩]`, `\normalmath` a `\boldmath`.

opmac.tex
```
131: \input ams-math  % ams-math.tex is in csplain package Nov. 2012 or later
```

Po načtení souboru ams-math disponujeme makry `\regtfm` na registraci různých metrik pro různé designované velikosti fontů a `\whichtfm`, které expanduje na svůj parametr nebo na metriku, která je registrována pro velikost `\dgsize`. Registrace metrik CSfontů je rovněž provedena v souboru `ams-math.tex`. Jak bylo řečeno, makro `\resizefont⟨fontselector⟩` CSplainu změní velikost fontu ⟨*fontselector*⟩ podle obsahu makra `\sizespec` (viz soubor `csfontsm.tex`). Toto makro `\resizefont` volá pomocné makro `\resizefontskipat` na odstranění „sizespec“ z názvu metriky. Toto pomocné makro je v OPmac předefinováno s využitím `\whichtfm`, takže když před voláním makra `\resizefont` připravíme správnou `\dgsize`, TEX použije metriku designovanou na požadovanou velikost:

opmac.tex
```
133: \def\resizefontskipat#1 #2\relax{\whichtfm{#1} \sizespec\relax}
```

Makra `\typosize`, `\fontsizex`, `\textfontsize`, `\setbaselineskip` požadují zápis parametru bez jednotky. Jednotkou je `\ptunit`, která je nastavena na 1pt. Uživatel může jednotku změnit (např. `\ptunit=1mm` při návrhu plakátu). Dále `\fontdim` je registr, který udává aktuální velikost písma.

opmac.tex
```
135: \newdimen\ptunit    \ptunit=1pt
136: \newdimen\fontdim   \fontdim=10pt
```

Často budeme potřebovat odstranit jednotku `pt` ve výpisu `\the⟨dimen⟩`. Provedeme to pomocí `\expandafter\ignorept\the⟨dimen⟩`. Protože `\the` vyrábí znaky pt s kategorií 12, je makro `\ignorept` definováno trikem přes `\lowercase`. Z otazníku vznikne p kategorie 12 a z vykřičníku vznikne t.

opmac.tex
```
138: {\lccode`\?=`\p \lccode`\!=`\t  \lowercase{\gdef\ignorept#1?!{#1}}}
```

Makra `\typosize` a `\typoscale` změní velikosti a nastavují výchozí font `\tenrm` a výchozí matematiku `\normalmath`. Nehrajeme si na OFS nebo NFSS, které se snaží ctít naposledy nastavený duktus a variantu. Uživatel si variantu písma a tučný duktus pro matematiku musí nastavit až po zavolání makra na změnu velikosti fontu.

opmac.tex
```
140: \def\typosize[#1/#2]{\fontsizex[#1]\setbaselineskip[#2]\ignorespaces}
141: \def\typoscale[#1/#2]{\fontscalex[#1]\scalebaselineskip[#2]\ignorespaces}
```

Makro `\fontsizex` `[⟨velikost⟩]` předpokládá svůj parametr bez jednotky. Písmeno x v názvu značí, že makro není v uživatelské dokumentaci. Uživatel totiž může použít `\typosize[⟨velikost⟩/]` a třeba ho napadne si nějaké vlastní makro `\fontsize` definovat. Je-li parametr ⟨*velikost*⟩ prázdný, makro `\fontsizex` neudělá nic. Jinak pomocí `\textfontsize` nastaví velikost textových fontů. Dále zavolá `\setmathsizes[\fontsize/.7\fontsize/.5\fontsize]`, ovšem v parametru musí odstranit jednotky a parametr přichystá pro makro `\setmathsizes` expandovaný. Příkazem `\normalmath` nakonec nastaví matematické fonty do nové velikosti.

opmac.tex
```
143: \def\fontsizex[#1]{\if$#1$\else
144:   \textfontsize[#1]%
145:   \tmpdim=0.7\fontdim \edef\tmpa{\expandafter\ignorept\the\tmpdim}%
146:   \tmpdim=0.5\fontdim \edef\tmpb{\expandafter\ignorept\the\tmpdim}%
147:   \edef\tmp{\noexpand\setmathsizes[\expandafter\ignorept\the\fontdim/\tmpa/\tmpb]}%
148:   \tmp \normalmath
149:   \fi
150: }
```

Makro `\textfontsize` `[⟨velikost⟩]` předpokládá svůj parametr bez jednotky. Připojí jednotku `\ptunit`, nastaví `\dgsize` a `\sizespec` a zavolá `\resizeall`, což je makro definované v CSplainu, které postupně volá `\resizefont` na všechny registrované fonty.

opmac.tex
```
151: \def\textfontsize[#1]{\if$#1$\else
152:   \fontdim=#1\ptunit
153:   \let\dgsize=\fontdim
154:   \edef\sizespec{at\the\fontdim}%
```

\regtfm \whichtfm: 7 \dgsize: 7–8 \resizefontskipat: 6–7 \ptunit: 7–8 \fontdim: 7–8
\ignorept: 6–8, 37, 46 \typosize: 7, 9, 27 \typoscale: 7, 9, 11, 37–38 \fontsizex: 7–8
\textfontsize: 7–9

```
155:   \resizeall \rm
156:   \fi
157: }
```

Makro \setbaselineskip [⟨velikost⟩] předpokládá parametr bez jednotky. Připojí jednotku \ptunit a nastaví \baselineskip bez dodatečné pružnosti. Nastaví další registry, které s \baselineskip souvisejí. Záměrně není nastavena \topskip, \splittopskip, \above/belowdisplayskip. Tyto parametry (globální pro celý dokument) by si měl uživatel nastavit sám.

opmac.tex

```
158: \def\setbaselineskip[#1]{\if$#1$\else
159:   \tmpdim=#1\ptunit
160:   \baselineskip=\tmpdim \relax
161:   \bigskipamount=\tmpdim plus.33333\tmpdim minus.33333\tmpdim
162:   \medskipamount=.5\tmpdim plus.16666\tmpdim minus.16666\tmpdim
163:   \smallskipamount=.25\tmpdim plus.08333\tmpdim minus.08333\tmpdim
164:   \normalbaselineskip=\tmpdim
165:   \jot=.25\tmpdim
166:   \maxdepth=.33333\tmpdim
167:   \setbox\strutbox=\hbox{\vrule height.709\tmpdim depth.291\tmpdim width0pt}%
168:   \fi
169: }
```

Makro \withoutunit \makro⟨dimen⟩ odstraní jednotku z ⟨dimen⟩ a takto upravené číslo vloží do parametru \makro, které očekává údaj bez jednotky v hranaté závorce.

opmac.tex

```
170: \def\withoutunit#1#2{\expandafter#1\expandafter[\expandafter\ignorept\the#2]}
```

Makra \fontscalex ⟨factor⟩, \textfontscale ⟨factor⟩ a \scalebaselineskip ⟨factor⟩ přepočítají ⟨factor⟩ podle aktuálního \fontdim resp. \baselineskip na absolutní jednotku a zavolají odpovídající makro definované před chvílí.

opmac.tex

```
172: \def\fontscalex[#1]{\if$#1$\else
173:   \tmpdim=#1pt \divide\tmpdim by1000
174:   \tmpdim=\expandafter\ignorept\the\tmpdim \fontdim
175:   \withoutunit\fontsizex\tmpdim
176:   \fi
177: }
178: \def\textfontscale[#1]{\if$#1$\else
179:   \tmpdim=#1pt \divide\tmpdim by1000
180:   \tmpdim=\expandafter\ignorept\the\tmpdim \fontdim
181:   \withoutunit\textfontsize\tmpdim
182:   \fi
183: }
184: \def\scalebaselineskip[#1]{\if$#1$\else
185:   \tmpdim=#1pt \divide\tmpdim by1000
186:   \tmpdim=\expandafter\ignorept\the\tmpdim \baselineskip
187:   \withoutunit\setbaselineskip\tmpdim
188:   \fi
189: }
```

Makro \thefontsize si alokuje aktuální font do sekvence \thefont a tento nový fontový přepínač podrobí změně velikosti \resizefont. Makro \thefontscale přepočítá parametr na absolutní velikost a zavolá \thefontsize.

opmac.tex

```
190: \def\thefontsize[#1]{%
191:   \expandafter\let \expandafter\thefont \the\font
192:   \def\sizespec{at#1\ptunit}\def\dgsize{#1\ptunit}\resizefont\thefont
193:   \thefont \ignorespaces
194: }
195: \def\thefontscale[#1]{%
196:   \tmpdim=#1pt \divide\tmpdim by1000
197:   \tmpdim=\expandafter\ignorept\the\tmpdim \fontdim
198:   \withoutunit\thefontsize\tmpdim
199: }
```

\setbaselineskip: 7–8 \withoutunit: 8 \fontscalex: 7–8 \textfontscale: 8–9
\scalebaselineskip: 7–8 \thefontsize: 8–9, 44 \thefont: 8, 31, 33 \thefontscale: 6, 8–9, 31, 33

PlainTEXový \magstep má na konci \relax, takže nefunguje jako pouze expandující makro. My ale \magstep očekáváme v parametrech příkazů \typoscale a podobných, proto v \magstep je nahrazeno \relax méně drsným \space. To separuje číselný parametr dostatečně.

opmac.tex
```
\def\magstep#1{\ifcase#1 1000\or1200\or1440\or1728\or2074\or2488\fi\space}
```

Makro \em přepíná kontextově do odpovídající varianty a ve spolupráci s makrem \additcorr přidává italickou korekci. Makro \additcorr si pomocí \lastskip zapamatuje poslední mezeru, pak ji odstraní, vloží italickou korekci a nakonec vrátí tu odstraněnou mezeru.

opmac.tex
```
\def\em {\expandafter\ifx \the\font \tenit \additcorr \rm  \else
         \expandafter\ifx \the\font \tenbf \bi\aftergroup\/\else
         \expandafter\ifx \the\font \tenbi \additcorr \bf  \else
         \it \aftergroup\/\fi\fi\fi}
\def\additcorr{\ifdim\lastskip>0pt \skip0=\lastskip \unskip\/\hskip\skip0 \else\/\fi}
```

Fontová makra zabezpečíme proti rozkladu v parametru \write.

opmac.tex
```
\addprotect\thefontsize   \addprotect\thefontscale
\addprotect\typosize      \addprotect\typoscale
\addprotect\textfontsize  \addprotect\textfontscale
\addprotect\em
```

3.5 Texty ve více jazycích

Makro \mtext ⟨*značka*⟩ je zkratkou za „multilingual text“. Toto makro si podle značky a aktuálního jazyka (dle registru \language) vyhledá, jaký text má vypsat.

opmac.tex
```
\def\mtext#1{\csname mt:#1:\csname lan:\the\language\endcsname\endcsname}
```

Jednotlivé texty definujeme pomocí \sdef{mt:⟨*značka*⟩:⟨*jazyk*⟩} takto:

opmac.tex
```
\sdef{mt:chap:en}{Chapter}  \sdef{mt:chap:cz}{Kapitola}  \sdef{mt:chap:sk}{Kapitola}
\sdef{mt:t:en}{Table}       \sdef{mt:t:cz}{Tabulka}      \sdef{mt:t:sk}{Tabu\v lka}
\sdef{mt:f:en}{Figure}      \sdef{mt:f:cz}{Obr\'azek}    \sdef{mt:f:sk}{Obr\'azok}
```

Některé texty jsou zapsány pomocí \v notace. Je lepší udělat to takto než vytvořit soubor opmac.tex závislý na kódování. Aby byla tato notace správně interpretována, spustíme \csaccents, což je makro CSplainu. Pokud někdo používá OPmac s jiným formátem, než CSplain, neprovede se nic, protože konstrukce \csname␣csaccents\endcsname se v takovém případě přerodí v \relax. Makro \csaccents spustíme jen tehdy, pokud je už uživatel nespustil před \input␣opmac. To poznáme podle toho, zda je definovaná sekvence \r.

opmac.tex
```
\ifx\r\undefined \csname csaccents\endcsname \fi
```

CSplain od verze Nov.␣2012 připravuje následující makra, která konvertují číslo \language na značku jazyka pro všechny jazyky, které mají nataženy vzory dělení slov. Pro jistotu (pokud je použita starší verze CSplainu) tuto koverzi „naučíme“ i makro OPmac:

opmac.tex
```
\sdef{lan:0}{en}  \sdef{lan:100}{en} \sdef{lan:101}{en}
\sdef{lan:5}{cz}  \sdef{lan:15}{cz}  \sdef{lan:115}{cz}
\sdef{lan:6}{sk}  \sdef{lan:16}{sk}  \sdef{lan:116}{sk}
```

3.6 REF soubor

OPmac používá pro všechny potřeby (obsah, reference, citace, rejstřík, poznámky na okraji) jediný soubor \jobname.ref (tzv. REF soubor). Navíc, pokud není potřeba, vůbec tento soubor nezakládá. Často totiž budeme chtít dělat s OPmac jen jednoduché věci a je únavné pořád na disku kvůli tomu uklízet smetí.

Je potřeba deklarovat souborové deskriptory \reffile a \testin:

opmac.tex
```
\newwrite\reffile
\newread\testin
```

Do souboru zapisujeme makrem \wref \⟨sekvence⟩{⟨data⟩}, které vloží do \reffile řádek obsahující \⟨sekvence⟩⟨data⟩. Implicitně ale není \reffile založeno, takže implicitní hodnota tohoto makra je \wrefrelax, tedy nedělej nic.

opmac.tex

```
\def\wrefrelax#1#2{}
\let\wref=\wrefrelax
```

Makro \inputref spustíme na konci čtení souboru opmac.tex, tedy v situaci, kdy už budeme mít definovány všechny kontrolní sekvence, které se v REF souboru mohou vyskytnout. Nyní si toto makro jen připravíme. Makro ověří existenci souboru \jobname.ref a pokud existuje, provede \input␣\jobname.ref. V takovém případě po načtení REF souboru jej otevře k zápisu a připraví \wref do stavu, kdy toto makro bude ukládat data do souboru.

opmac.tex

```
\def\inputref{
  \openin\testin=\jobname.ref
  \ifeof\testin \else
    \closein\testin
    \input \jobname.ref
    \fnotenum=0 \mnotenum=0
    \immediate\openout\reffile=\jobname.ref
    \def\wref##1##2{\write\reffile{\string##1##2}}
    \immediate\write\reffile {\percent\percent\space OPmac - REF file}
  \fi
```

Makro \openref kdekoli v dokumentu si vynutí založení souboru \jobname.ref. Toto makro neprovede nic, je-li REF soubor už založen. To pozná podle toho, že makro \wref nemá význam \wrefrelax. Jestliže soubor ještě není založen, makro jej založí, předefinuje \wref a vloží první řádek do souboru. Tím je zaručeno, že při příštím TEXování dokumentu je soubor neprázdný, takže jej OPmac rovnou přečte a znovu založí na začátku své činnosti. Nakonec se \openref zasebevraždí, aby se nemuselo při opakovaném volání obtěžovat vykonávat nějakou práci. Práce už je totiž hotova.

opmac.tex

```
\def\openref{%
  \ifx\wref\wrefrelax
     \immediate\openout\reffile=\jobname.ref
     \gdef\wref##1##2{\write\reffile{\string##1##2}}%
     \immediate\write\reffile
       {\percent\percent\space OPmac - REF file (\string\openref)}%
  \fi
  \gdef\openref{}%
}
```

Pro zápisy do REF souboru používáme tuto konvenci: první kontrolní sekvence na řádku je vždy tvaru \X⟨název⟩, takže máme přehled, která kontrolní sekvence pochází z REF souboru.

3.7 Lejblíky a odkazy

K vytvoření zpětného odkazu provedeme tři kroky (v tomto pořadí):

- V místě \label[⟨lejblík⟩] si zapamatujeme ⟨lejblík⟩.
- V době vygenerování čísla (sekce, kapitoly, caption, atd.) propojíme ⟨lejblík⟩ s tímto číslem. Provedeme to pomocí \sxdef{lab:⟨lejblík⟩}{⟨číslo⟩}.
- V místě \ref[⟨lejblík⟩] vytiskneme \csname␣lab:⟨lejblík⟩\endcsname, tedy ⟨číslo⟩.

To je základní idea pro zpětné odkazy. V takovém případě nepotřebujeme REF soubor. Pokud ale chceme dopředné odkazy, je potřeba použít REF soubor zhruba takto:

- V době vygenerování čísla (sekce atd.) navíc uložíme informaci \Xlabel{⟨lejblík⟩}{⟨číslo⟩} do REF souboru.
- V době čtení REF souboru (tedy na začátku dokumentu) provede \Xlabel přiřazení pomocí \sdef{lab:⟨lejblík⟩}{⟨číslo⟩}.
- Nyní může přijít \ref[⟨lejblík⟩] se svým \csname␣lab:⟨lejblík⟩\endcsname kdekoli v dokumentu.

\wref: 10–12, 15, 26, 37–38, 41–42, 44 \wrefrelax: 10 \inputref: 10, 46 \openref: 10–11, 15, 26, 29, 37–38, 40–42

Přejdeme od idejí k implementaci. Makro \label [⟨lejblík⟩] si pouze zapamatuje ⟨lejblík⟩ do makra \lastlabel, aby s touto hodnotou mohlo později pracovat makro, které automaticky generuje nějaké číslo.

opmac.tex
```
\def\label[#1]{\xdef\lastlabel{#1}\ignorespaces}
```

Makro, které automaticky generuje nějaké číslo, má za úkol zavolat \wlabel ⟨číslo⟩. Toto makro propojí \lastlabel a ⟨číslo⟩ tak, že definuje sekvenci \lab:\lastlabel jako makro s hodnotou ⟨číslo⟩. Kromě toho zapíše expandované \lastlabel i ⟨číslo⟩ do REF souboru (jen, je-li otevřen, zpětné reference totiž fungují i bez souboru). Nakonec vrátí makru \lastlabel jeho původní nedefinovanou hodnotu, tj. lejblík už byl použit. Další makro automaticky generující číslo zavolá \wlabel, který nyní neprovede nic (pokud tedy uživatel nenapsal další \label[⟨lejblík⟩]).

opmac.tex
```
\def\wlabel#1{%
  \ifx\lastlabel\undefined \else
     \dest[ref:\lastlabel]{2.1em}%
     \edef\tmp{\wref\Xlabel{{\lastlabel}{#1}}}\tmp
     \isdefined{lab:\lastlabel}\iftrue \else
        \sxdef{lab:\lastlabel}{#1}%
     \fi
     \global\let\lastlabel=\undefined
  \fi
}
```

Makro \ref [⟨lejblík⟩] zkontroluje definovanost \lab:⟨lejblík⟩. Je-li to pravda, vytiskne \lab:⟨lejblík⟩ (krz reflink, aby to bylo případně klikací). Jinak vytiskne dva otazníky a předpokládá, že v tomto případě jde o dopřednou referenci. Vynutí si tedy otevření REF souboru zavoláním \openref.

opmac.tex
```
\def\ref[#1]{\isdefined{lab:#1}%
  \iftrue \reflink[#1]{\csname lab:#1\endcsname}%
  \else ??\opwarning{label [#1] unknown. Try to TeX me again}\openref
  \fi
}
```

Makro \pgref [⟨lejblík⟩] dělá podobnou práci, jako \ref, jen s makrem \pgref:⟨lejblík⟩. Toto makro je definováno až při znovunačtení REF souboru makrem \Xlabel, protože ke správnému určení čísla stránky potřebujeme asynchronní \write.

opmac.tex
```
\def\pgref[#1]{\isdefined{pgref:#1}%
  \iftrue \pglink{\csname pgref:#1\endcsname}%
  \else ??\opwarning{pg-label [#1] unknown. Try to TeX me again}\openref
  \fi
}
\def\Xlabel#1#2{\sxdef{lab:#1}{#2}\sxdef{pgref:#1}{\the\lastpage}}
```

3.8 Kapitoly, sekce, podsekce

Fonty pro titul, kapitoly a sekce \titfont, \chapfont, \secfont a \seccfont jsou definovány jako odpovídající zvětšení a nastavení tučného duktu. Ten je nastaven pomocí \bfshape jako \bf a navíc je ztotožněn \tenit s \tenbi, takže když nyní uživatel napíše \it, dostane tučnou kurzívu.

opmac.tex
```
\def\titfont{\typoscale[\magstep4/\magstep4]\bf}
\def\chapfont{\typoscale[\magstep3/\magstep3]\bfshape}
\def\secfont{\typoscale[\magstep2/\magstep2]\bfshape}
\def\seccfont{\typoscale[\magstep1/\magstep2]\bfshape}
\def\bfshape{\let\tenit=\tenbi \boldmath \bf}
```

Pro číslování kapitol, sekcí a podsekcí potřebujeme čítače:

opmac.tex
```
\newcount\chapnum \newcount\secnum \newcount\seccnum
```

\label: 10–11 \lastlabel: 11 \wlabel: 11–14 \ref: 10–11 \pgref: 11 \Xlabel: 10–11
\titfont: 11–12 \chapfont: 11, 13 \secfont: 11, 13 \seccfont: 11, 13 \bfshape: 11
\chapnum: 12 \secnum: 12 \seccnum: 12

Makro pro titul \tit počítá s tím, že bude titul na více řádcích. Sází ho tedy jako odstavec s pružnými \leftskip a \rightskip. Příkaz \unskip těsně za parametrem #1 odstraní mezeru z konce řádku, která tam obvykle je. Teprve poté je nadpis řádně centrován.

opmac.tex

```
\def\tit#1\par{\vglue4em
  {\leftskip=0pt plus1fill \rightskip=\leftskip
   \titfont \noindent #1\unskip\par}%
  \nobreak\bigskip
}
```

Makra \chap, \sec a \secc nastaví odpovídající čítače, dále vytvoří číslování pro tisk (sestávající z více čísel) v makrech \thechapnum, \thesecnum a \theseccnum a zavolají odpovídající makro \printchap, \printsec a \printsecc. Poté zavolá \wlabel na propojení případného labelu s číslem kapitoly/sekce/podsekce a pomocí \wcontents uloží do \reffile potřebná data pro obsah.

opmac.tex

```
\def\chap#1\par{\advance\chapnum by1
  \chaphook \secnum=0 \seccnum=0 \relax
  \edef\thechapnum{\the\chapnum.}%
  \printchap{#1\unskip}%
  \wlabel\thechapnum \wcontents\Xchap\thechapnum{#1}\nobreak
}
\def\sec#1\par{\par\advance\secnum by1
  \sechook \seccnum=0 \tnum=0 \fnum=0 \dnum=0 \relax
  \edef\thesecnum{\othe\chapnum.\the\secnum}%
  \printsec{#1\unskip}%
  \insertmark\thesecnum{#1\unskip}%
  \wlabel\thesecnum \wcontents\Xsec\thesecnum{#1}\nobreak
}
\def\secc#1\par{\advance\seccnum by1
  \secchook \relax
  \edef\theseccnum{\othe\chapnum.\othe\secnum.\the\seccnum}%
  \printsecc{#1\unskip}%
  \wlabel\theseccnum \wcontents\Xsecc\theseccnum{#1}\nobreak
}
\def\wcontents#1#2#3{%
  \expandafter\wref\expandafter#1\expandafter
     {\expandafter{#2}{#3}{\the\pageno}}%
}
\def\othe#1.{\ifnum#1>0 \the#1.\fi}
\def\thechapnum{} \def\thesecnum{} \def\theseccnum{}
```

Text titulu sekce a její číslo jsou vloženy do \mark, takže tyto údaje můžete použít v plovoucím záhlaví. Tato vlastnost není dokumentována v uživatelské části, protože je poněkud techničtějšího charakteru. Makro \insertmark \thesecnum{⟨*text*⟩} vloží do \mark data ve formátu {⟨*secnum*⟩}␣{⟨*text*⟩}, takže je možno je použít přímo expanzí např. \firstmark, nebo je oddělit a zpracovat zvlášť. Parametru ⟨*text*⟩ je zabráněna expanze pomocí protažení tohoto parametru přes \toks, viz TBN str. 54 dole a strany 55–57. Příkaz \mark se totiž snaží o expanzi.

opmac.tex

```
\def\insertmark#1#2{\toks0={#2}\mark{{#1} {\the\toks0}}}
```

Příklad použití plovoucího záhlaví v \headline:

```
\headline{\expandafter\domark\firstmark\hss}
\def\domark#1#2{\llap{\it\headsize #1. }\rm\headsize #2}
\def\headsize{\thefontsize[10]}
```

Makro \headsize v této ukázce zaručí, že bude mít záhlaví vždy požadovanou velikost. Bez toho ta záruka není, pokud tedy uživatel v sazbě dokumentu střídá velikosti písma. Output rutina totiž může přijít náhle, třeba v okamžiku, kdy je zapnutá jiná velikost písma.

Pokud chcete kombinovat na levých a pravých stránkách plovoucí záhlaví z kapitol a sekcí, inspirujte se v TBN na stranách 259 a 260.

\tit: 12 \chap: 6, 12 \sec: 6, 12 \secc: 6, 12 \thechapnum: 12–13 \thesecnum: 12–13 \theseccnum: 12–13 \wcontents: 12 \insertmark: 12

Makra \printchap, \printsec a \printsecc vytisknou číslo a titul kapitoly, sekce resp. podsekce. Příkazem \firstnoindent dávají najevo, že následující odstavec nebude mít odstavcovou zarážku.

opmac.tex
```
\def\printchap#1{\vfil\break
  {\chapfont \noindent \dest[toc:\thechapnum]{1.3em}%
     \mtext{chap} \thechapnum\par\nobreak\smallskip\noindent #1\nbpar}%
  \nobreak\bigskip
  \firstnoindent
}
\def\printsec#1{\removelastskip \goodbreak \bigskip
  {\secfont \noindent \dest[toc:\thesecnum]{1.3em}\thesecnum\quad #1\nbpar}%
  \nobreak\medskip
  \firstnoindent
}
\def\printsecc#1{\removelastskip \goodbreak \medskip
  {\seccfont \noindent
   \dest[toc:\theseccnum]{1.3em}\theseccnum\quad #1\nbpar}%
  \nobreak\medskip
  \firstnoindent
}
```

Makro \afternoindent potlačí odstavcovou zarážku pomocí přechodného naplnění \everypar kódem, který odstraní box vzniklý z \indent a vyprázdní pomocí \wipeepar registr \everypar. Makro \firstnoindent je ztotožněno s \afternoindent, ale uživatel může psát \let\firstnoindent=\relax. Pak bude \afternoindent pracovat jen za verbatim výpisy.

opmac.tex
```
\def\afternoindent{\global\everypar={\wipeepar\setbox0=\lastbox}}
\def\wipeepar{\global\everypar={}}
\let\firstnoindent=\afternoindent
```

Nechceme, aby se nám odstavce tvořené z titulů kapitol a sekcí rozdělily do více stran. Proto místo příkazu \par je použito \nbpar. Konečně uživatel může odřádkovat například v titulu pomocí \nl. Tomuto makru později v \output rutině změníme význam na mezeru.

opmac.tex
```
\def\nbpar{{\interlinepenalty=10000\par}}
\def\nl{\hfil\break}
```

3.9 Popisky, rovnice

Nejprve deklarujeme potřebné čítače:

opmac.tex
```
\newcount\tnum  \newcount\fnum  \newcount\dnum
```

Výchozí hodnoty maker, které se vypisují v místě generovaného čísla jsou:

opmac.tex
```
\def\thetnum{\thesecnum.\the\tnum}
\def\thefnum{\thesecnum.\the\fnum}
\def\thednum{(\the\dnum)}
```

Makro \caption /⟨*typ*⟩␣ zvedne čítač \⟨*typ*⟩num o jedničku, dále nastaví pružné mezery s odsazením \iindent a s centrováním posledního řádku (viz TBN str. 234), propojí pomocí \wlabel číslo s případným lejblíkem, vytiskne slovo Tabulka/Obrázek v závislosti na jazyku makrem \mtext, vytiskne číslo \the⟨*typ*⟩num a zpracuje odstavec s nastavenými \leftskip, \rightskip. Předefinuje \par tak, že první výskyt prázdného řádku ukončí skupinu a tím se všechna nastavení vrátí do původního stavu.

opmac.tex
```
\def\caption/#1 {\isdefined{#1num}%
   \iftrue \advance \csname #1num\endcsname by1
   \else   \opwarning{Unknown caption /#1}%
   \fi
   \bgroup
      \leftskip=\iindent plus1fil
      \rightskip=\iindent plus-1fil
      \parfillskip=\iindent plus2fil
```

\printchap: 12–13 \printsec: 12–13 \printsecc: 12–13 \afternoindent: 13, 31
\wipeepar: 13, 24, 31, 33 \firstnoindent: 13 \nbpar: 13 \nl: 13, 44 \tnum: 12–13
\fnum: 12–13 \dnum: 12–14 \caption: 13

```
367:       \def\par{\endgraf\egroup}%
368:       \noindent \wlabel{\csname the#1num\endcsname}%
369:       {\bf \mtext{#1} \csname the#1num\endcsname}\enspace
370: }
```

Makro \eqmark zvedne \dnum o jedničku. V display módu pak použije primitiv \eqno, za kterým následuje \thednum. V interním módu (v boxu) vytiskne jen \thetnum. V obou případech propojí případný lejblík s číslem pomocí makra \wlabel.

opmac.tex

```
371: \def\eqmark{\global\advance\dnum by1
372:   \ifinner\else\eqno \fi \wlabel\thednum \thednum
373: }
```

3.10 Odrážky

Odsazení každé další vnořené úrovně odrážek bude o \iindent větší. Jeho hodnota je nastavena na \parindent v době čtení opmac.tex. Jestliže uživatel později změní \parindent, měl by odpovídajícím způsobem změnit \iindent. Kromě toho deklarujeme čítač pro odrážky \itemnum.

opmac.tex

```
377: \newcount\itemnum  \itemnum=0
```

Makro \begitems vloží \iiskip, zahájí novou skupinu, pronuluje \itemnum, zvětší odsazení o \iindent a pomocí \adef definuje hvězdičku jako aktivní makro, které provede \startitem. Makro \enditems ukončí skupinu a vloží \iiskip.

opmac.tex

```
379: \def\begitems{\iiskip\bgroup
380:   \itemnum=0 \adef*{\startitem}
381:   \advance\leftskip by\iindent
382:   \let\printitem=\normalitem
383: }
384: \def\enditems{\par\egroup\iiskip}
```

Makro \startitem ukončí případný předchozí odstavec, posune čítač, zahájí první odstavec jako \noindent a vyšoupne doleva text definovaný v \printitem, který je implicitně nastaven makrem \begitems na \normalitem.

opmac.tex

```
386: \def\startitem{\par \advance\itemnum by1
387:    \noindent\llap{\printitem}\ignorespaces}
388: \def\normalitem{$\bullet$\enspace}
```

Makro \style ⟨*znak*⟩ přečte ⟨*znak*⟩ a rozvine jen na makro \item:⟨*znak*⟩. Tato jednotlivá makra jsou definována pomocí \sdef. Není-li makro \item:⟨*znak*⟩ definováno, použije se \normalitem.

opmac.tex

```
390: \def\style#1{\expandafter\let\expandafter\printitem\csname item:#1\endcsname
391:   \ifx\printitem\relax \let\printitem=\normalitem \fi
392: }
393: \sdef{item:o}{\raise.4ex\hbox{$\scriptscriptstyle\bullet$} }
394: \sdef{item:-}{- }
395: \sdef{item:n}{\the\itemnum. }
396: \sdef{item:N}{\the\itemnum) }
397: \sdef{item:i}{(\romannumeral\itemnum) }
398: \sdef{item:I}{\uppercase\expandafter{\romannumeral\itemnum}\kern.5em}
399: \sdef{item:a}{\athe\itemnum) }
400: \sdef{item:A}{\uppercase\expandafter{\athe\itemnum}) }
401: \sdef{item:x}{\raise.3ex\fullrectangle{.6ex} }
402: \sdef{item:X}{\raise.2ex\fullrectangle{1ex}\kern.5em}
```

Čtvereček kreslíme jako \vrule odpovídajících rozměrů makrem \fullrectangle {⟨*dimen*⟩}.

opmac.tex

```
404: \def\fullrectangle#1{\hbox{\vrule height#1 width#1}}
```

Pro převod mezi numerickou hodnotou čítače a příslušným písmenem a, b, c atd. je vytvořeno makro \athe ⟨*number*⟩.

\eqmark: 14 \itemnum: 14 \begitems: 5, 14 \enditems: 5, 14 \startitem: 14
\printitem: 14 \normalitem: 14 \style: 14 \fullrectangle: 14 \athe: 14–15

opmac.tex
```
406: \def\athe#1{\ifcase#1?\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or k\or l\or
407:    m\or n\or o\or p\or q\or r\or s\or t\or u\or v\or w\or x\or y\or z\else ?\fi
408: }
```

3.11 Tvorba obsahu

Do \toclist budeme ukládat data pro obsah. Pomocí \ifischap se budeme ptát, zda v dokumentu jsou kapitoly.

opmac.tex
```
412: \def\toclist{} \newif\ifischap \ischapfalse
```

V době čtení REF souboru vložíme veškerá data obsahu do makra \toclist tak, že v tomto bufferu budeme mít za sebou sekvence \tocline následované pěti parametry. Makra \Xchap, \Xsec a \Xsecc mají parametry {⟨číslo⟩}{⟨text⟩}{⟨strana⟩} a jsou definovány následovně:

opmac.tex
```
414: \def\Xchap#1#2#3{\ischaptrue\addto\toclist{\tocline{0}{\bf}{#1}{#2}{#3}}}
415: \def\Xsec#1#2#3{\addto\toclist{\tocline{1}{\rm}{#1}{#2}{#3}}}
416: \def\Xsecc#1#2#3{\addto\toclist{\tocline{2}{\rm}{#1}{#2}{#3}}}
```

Makro \tocline {⟨odsazení⟩}{⟨font⟩}{⟨číslo⟩}{⟨text⟩}{⟨strana⟩} vytvoří řádek obsahu. Řádek tiskneme jako odstavec, protože ⟨text⟩ může být třeba delší. Registr \leftskip nastavíme jako součin ⟨odsazení⟩ krát \iindent. Pokud se v dokumentu vyskytují kapitoly, odsadíme ještě o další \iindent. Registr \rightskip nastavíme na 2\iindent, aby delší ⟨text⟩ se zalomil dřív než v místě, kde jsou stránkové číslice. Konec odstavce se stránkovou číslicí pak vytáhneme mimo tento rozsah pomocí \hskip-2\iindent. Odstavec pruží v \tocdotfill, protože tento výplněk má větší pružnost než \parfillskip. Registr \parfillskip má 1fil, zatímco \tocdotfill má pružnost 1fill. Makro \tocdotfill je implementováno pomocí \leaders jako opakované tečky, které budou lícovat pod sebou.

opmac.tex
```
418: \def\tocline#1#2#3#4#5{{\leftskip=#1\iindent \rightskip=2\iindent
419:    \ifischap\advance\leftskip by\iindent\fi
420:    \ifnum#1>1 \advance\leftskip by\iindent\fi
421:    \noindent\llap{#2\toclink{#3}\enspace}%
422:          {#2#4\unskip}\tocdotfill\pglink{#5}\hskip-2\iindent\null\par}}
423: \def\tocdotfill{\leaders\hbox to.8em{\hss.\hss}\hfill}
```

Makro \maketoc jednoduše spustí \toclist, Pokud je \toclist prázdný, upozorní o tom adekvátním způsobem na terminál.

opmac.tex
```
425: \def\maketoc{\par \ifx\toclist\empty
426:       \opwarning{\noexpand\maketoc -- data unavailable, TeX me again}\openref
427:    \else \toclist \fi}
```

3.12 Sestavení rejstříku

Slovo do rejstříku vložíme pomocí \iindex {⟨heslo⟩}. Protože výskyt slova na stránce není v době zpracování znám, je nutné použít REF soubor s asynchronním \write.

opmac.tex
```
432: \def\iindex#1{\openref\wref\Xindex{{#1}{\the\pageno}}}
```

Nyní naprogramujeme čtení parametru makra \ii ⟨slovo⟩,⟨slovo⟩,...⟨slovo⟩␣. Vzhledem k tomu, že za přítomnosti zkratky @ budeme potřebovat projít seznam slov oddělených čárkou v parametru ještě jednou, zapamatujeme si tento seznam do \tmp.

opmac.tex
```
434: \def\ii #1 {\leavevmode\def\tmp{#1}\iiA #1,,}
```

Makro \iiA sejme vždy jedno slovo ze seznamu. Podle prázdného parametru poznáme, že jsme u konce a neděláme nic. Při výskytu ⟨slovo⟩=@ (poznáme to podle shodnosti parametru s \iiatsign), spustíme \iiB, jinak vložíme údaj o slově do rejstříku pomocí \iindex. Nakonec makro \iiA volá rekurzivně samo sebe.

\toclist: 15, 29 \ifischap: 15 \Xchap: 12, 15 \Xsec: 12, 15 \Xsecc: 12, 15
\tocline: 15, 29 \tocdotfill: 15 \maketoc: 15 \iindex: 15–16 \ii: 15–16 \iiA: 15–16
\iiatsign: 16

opmac.tex
```
\def\iiA #1,{\if$#1$\else\def\tmpa{#1}%
   \ifx\tmpa\iiatsign \expandafter\iiB\tmp,,%
   \else\iindex{#1}\fi
   \expandafter\iiA\fi}
\def\iiatsign{@}
```

Makro \iiB rovněž sejme vždy jedno slovo ze seznamu a na konci volá rekurzivně samo sebe. Toto makro ovšem za použití makra \iiC prohodí pořadí prvního podslova před prvním lomítkem se zbytkem. Není-li ve slově lomítko, pozná to makro \iiC podle toho, že parametr #2 je prázdný. V takovém případě neprovede nic, neboť slovo je už zaneseno do rejstříku v makru \iiA.

opmac.tex
```
\def\iiB #1,{\if$#1$\else
   \iiC#1/\relax
   \expandafter\iiB\fi
}
\def\iiC #1/#2\relax{\if$#2$\else\iindex{#2#1}\fi}
```

Makro \iid ⟨*slovo*⟩␣ pošle slovo do rejstříku a současně je zopakuje do sazby. Pomocí \futurelet a \iiD zjistí, zda následuje tečka nebo čárka. Pokud ne, vloží mezeru.

opmac.tex
```
\def\iid #1 {\leavevmode\iindex{#1}#1\futurelet\tmp\iiD}
\def\iiD{\ifx\tmp,\else\ifx\tmp.\else\space\fi\fi}
```

Při čtení REF souboru se vykonávají makra \Xindex {⟨*heslo*⟩}{⟨*strana*⟩}, která postupně vytvářejí makra tvaru \,⟨*heslo*⟩, ve kterých je shromažďován seznam stránek pro dané ⟨*heslo*⟩. Kromě toho každé makro \,⟨*heslo*⟩ je vloženo do seznamu \iilist. Na konci čtení REF souboru tedy máme v \iilist seznam všech hesel jako řídicí sekvence (to zabere nejmíň místa v TEXu). Každé \,⟨*heslo*⟩ na konci čtení REF souboru obsahuje dva údaje ve svorkách, první údaj obsahuje pomocná data a druhý obsahuje seznam stránek. Tedy \,⟨*heslo*⟩ je makro s obsahem {⟨*pomocná-data*⟩}{⟨*seznam-stránek*⟩}.

Seznam stránek není jen tupý seznam všech stránek, na kterých se objevil záznam \ii pro dané slovo. Některé stránky se totiž mohou opakovat a my je chceme mít jen jednou. Pokud stránky jsou souvisle za sebou: 13, 14, 15, 16, chceme navíc takový seznam nahradit zápisem 13–16. Makro \Xindex{⟨*heslo*⟩}{⟨*strana*⟩} je z tohoto důvodu poněkud sofistikovanější.

opmac.tex
```
\def\Xindex#1#2{\bgroup \def~{ }%
   \isdefined{,#1}\iftrue
     \expandafter\firstdata \csname,#1\endcsname \XindexA
     \ifnum#2=\tmpa % \ii on the same page
     \else
        \tmpnum=#2 \advance\tmpnum by-1
        \expandafter\seconddata \csname,#1\endcsname \XindexB
        \if\tmpb+% state: the pagelist ends by a pagenumber
           \ifnum\tmpnum=\tmpa  % the consecutive page
              \sxdef{,#1}{{#2/-}{\tmp\iiendash}}
           \else                % the pages drop
              \sxdef{,#1}{{#2/+}{\tmp, #2}}
           \fi
        \else    % state: the pagelist ends by --
           \ifnum\tmpnum=\tmpa  % the consecutive page
              \sxdef{,#1}{{#2/-}{\tmp}}
           \else                % the pages drop
              \sxdef{,#1}{{#2/+}{\tmp\tmpa, #2}}
           \fi
        \fi
     \fi
   \else % first occurrence of the index item #1
     \sxdef{,#1}{{#2/+}{#2}}
     \global \expandafter\addto \expandafter\iilist \csname,#1\endcsname
   \fi
   \egroup
}
\def\iilist{} \def\iiendash{--}
```

\iiB: 15–16 \iiC: 16 \iid: 16 \iiD: 16 \Xindex: 15–17 \iilist: 16, 18, 22–23

Činnost makra \Xindex si vysvětlíme za chvíli podrobněji. Nejprve ovšem definujeme pomocné makro \firstdata \,⟨heslo⟩ \⟨cs⟩, které expanduje na \⟨cs⟩ ⟨první-datový-údaj-hesla⟩&. Je-li třeba \,aa definováno jako {prvni}{druhy}, pak \firstdata \,aa \cosi expanduje na \cosi prvni&. Tím máme možnost vyzískat data z makra. Podobně makro \seconddata \,⟨heslo⟩ \⟨cs⟩ expanduje na \⟨cs⟩ ⟨druhý-datový-údaj-hesla⟩&.

opmac.tex
```
480: \def\firstdata#1#2{\expandafter\expandafter\expandafter #2\expandafter\firstdataA#1}
481: \def\firstdataA#1#2{#1&}
482: \def\seconddata#1#2{\expandafter\expandafter\expandafter #2\expandafter\seconddataA#1}
483: \def\seconddataA#1#2{#2&}
```

Než se pustíme do výkladu makra \Xindex, vysvětlím, proč pro hesla rejstříku tvořím jednu řídicí sekvenci, která je makrem se dvěma datovými údaji. Mohl bych jednodušeji pracovat se dvěma různými řídicími sekvencemi, např. \,⟨heslo⟩ a \:⟨heslo⟩. Důvod je prostý: šetřím paměť TEXu. Dá se totiž očekávat, že počet hesel v rejstříku můžeme počítat na tisíce a je rozdíl alokovat kvůli tomu tisíce kontrolních sekvencí nebo dvojnásobné množství takových sekvencí.

V makru \Xindex čteme \firstdata na řádku 453 a \seconddata na řádku 457. Čtení je provedeno makry \XindexA a \XindexB. První úsek dat je tvaru ⟨poslední-strana⟩/⟨stav⟩ a druhý úsek dat obsahuje rozpracovaný seznam stránek. Podíváme-li se na definice \XindexA a \XindexB, shledáme, že seznam stránek bude uložen v \tmp, dále ⟨poslední-strana⟩ bude v \tmpa a ⟨stav⟩ je v \tmpb.

opmac.tex
```
485: \def\XindexA#1/#2&{\def\tmpa{#1}\let\tmpb=#2}
486: \def\XindexB#1&{\def\tmp{#1}}
```

Rozlišujeme dva stavy: ⟨stav⟩=+, pokud je seznam stránek zakončen konkrétní stránkou. Tato konkrétní stránka je uložena v ⟨poslední-strana⟩. Druhým stavem je ⟨stav⟩=-, když je seznam stránek ukončen -- (přesněji obsahem makra \iiendash, které můžete snadno předefinovat) a v tomto případě ⟨poslední-strana⟩ obsahuje poslední stránku, na které byl zjištěn výskyt hesla. Tato strana nemusí být v seznamu stránek explicitně uvedena.

Makro \Xindex{⟨heslo⟩}{⟨strana⟩} tedy postupně vytváří seznam stran zhruba takto:

```
if (první výskyt \,⟨heslo⟩) {
  založ \,⟨heslo⟩ do iilist;
  ⟨seznam-stran⟩ = "⟨strana⟩"; ⟨stav⟩ = +; ⟨posledni-strana⟩ = ⟨strana⟩;
  return;
}
if (⟨strana⟩ == ⟨posledni-strana⟩) return;
if (⟨stav⟩ == +) {
  if (⟨strana⟩ == ⟨posledni-strana⟩+1) {
    ⟨seznam-stran⟩ += "--";
    ⟨stav⟩ = - ;
  }
else {
    ⟨seznam-stran⟩ += ", ⟨strana⟩";
    ⟨stav⟩ = + ;
  }
  else {
    if (⟨strana⟩ > ⟨posledni-strana⟩+1) {
      ⟨seznam-stran⟩ += "⟨posledni-strana⟩, ⟨strana⟩";
      ⟨stav⟩ = + ;
    }
  }
}
⟨poslední-strana⟩ = ⟨strana⟩;
```

Makro \makeindex nejprve definuje přechodný význam rekurzivního \act tak, aby byly uzavřeny seznamy stránek (tj. aby seznam nekončil znakem --) a do první datové oblasti každého makra

\firstdata: 16–18, 22 \seconddata: 16–18 \XindexA: 16–18 \XindexB: 16–18 \iiendash: 16
\makeindex: 18–19, 21

typu `\,⟨heslo⟩` vloží konverzi textu ⟨*heslo*⟩ do tvaru vhodném pro abecední řazení českých slov. Pomocí `\expandafter␣\act␣\iilist␣\relax` se požadovaná činnost vykoná pro každý prvek v `\iilist`. Dále makro `\makeindex` provede seřazení `\iilist` podle abecedy makrem `\dosorting` a nakonec provede tisk jednotlivých hesel. K tomu účelu znovu přechodně předefinuje `\act` a předloží mu `\iilist`.

opmac.tex

```
\def\makeindex{\par
  \ifx\iilist\empty \opwarning{index data-buffer is empty. TeX me again}
    \else
    \bgroup
      \setprimarysorting
      \def\act##1{\ifx##1\relax \else
         \firstdata##1\XindexA \seconddata##1\XindexB
         \if\tmpb+%
            \preparesorting##1% converted item by sorting data in \tmpb
            \xdef##1{{\tmpb}{\tmp}}
          \else
            \preparesorting##1% converted item by sorting data in \tmpb
            \xdef##1{{\tmpb}{\tmp\tmpa}}
          \fi
          \expandafter\act\fi}
      \expandafter \act \iilist \relax
    \egroup
    \dosorting  % sorting is in progress
    \iiparparams
    \gdef\act##1{\ifx##1\relax \else \prepii##1%
             \seconddata##1\printiipages \expandafter\act \fi}
      \expandafter \act \iilist \relax
    \orippx \global\let\act=\undefined \global\let\orippx=\undefined
  \fi
}
```

Makro `\printiipages` sebere z ⟨*druhého-datového-údaje*⟩ seznam stránek a jednoduše je vytiskne.

opmac.tex

```
\def\printiipages#1&{#1\par}
```

Makro „prepare index item“ `\prepii \,⟨heslo⟩` odstraní prostřednictvím `\prepiiA` z názvu kontrolní sekvence backslash a čárku a zbytek tiskne pomocí `\printii`. Pokud ale je `\,⟨heslo⟩` uloženo v seznamu `\iispeclist`, pak se expanduje sekvenci s názvem `\\,⟨heslo⟩`, ve které je uloženo, co se má místo hesla vytisknout. Data těchto výjimek jsou připravena makrem `\iis`

opmac.tex

```
\def\prepii #1{\isinlist \iispeclist #1\iftrue
   \expandafter\expandafter\expandafter \printii \csname\string#1\endcsname&
   \else \expandafter\prepiiA\string #1&
   \fi
}
\def\prepiiA #1#2#3&{\printii#3&}
```

Kontrolní otázka: proč se nedotazujeme jednoduše na to, zda je `\\,⟨heslo⟩` definovaná řídicí sekvence? Odpověď: museli bychom ji sestavit pomocí `\csname...\endcsname`, ale to založí do TEXové paměti novou řídicí sekvenci pro každé heslo v rejstříku. My se snažíme počet těchto řídicích sekvencí redukovat na minimum. Počítáme s tím, že obyčejných hesel bude tisíce a výjimek jen pár desítek.

Makro `\iis ⟨heslo⟩␣{⟨text⟩}` vloží další údaj do slovníku výjimek pro hesla v rejstříku. Přesněji: vloží `\,⟨heslo⟩` do `\iispeclist` a definuje sekvenci `\\,⟨heslo⟩` jako ⟨*text*⟩.

opmac.tex

```
\def\iis #1 #2{\bgroup \def~{ }%
    \global\expandafter\addto\expandafter\iispeclist\csname,#1\endcsname
    \global\sdef{\expandafter\string\csname,#1\endcsname}{#2}%
  \egroup
}
\def\iispeclist{}
```

Makro „print index item“ `\printii ⟨heslo⟩&` vytiskne jeden údaj do rejstříku. Makro projde prostřednictvím `\printiiA` jednotlivá podslova oddělená lomítkem a přepíše je do rejstříku odděleny

---

`\printiipages`: 18 `\prepii`: 18 `\prepiiA`: 18 `\iis`: 18 `\iispeclist`: 18 `\printii`: 18–19 `\printiiA`: 19

mezerou. Přitom kontroluje, zda se podslova rovnají odpovídajícím podslovům z předchozího hesla, které je uloženo v \previi. Pokud ano, místo podslova vloží \iiemdash, což je pomlka. Na konci činnosti se spustí makro \printiiB, které nastaví \previi na \currii (nové slovo se pro další zpracování stává předchozím) a vytiskne seznam stránek. Makrem \everyii (implicitně je prázdné) dovolíme uživateli vstoupit do procesu tisku hesla. Může například psát \def\everyii{\indent}, pokud chce.

opmac.tex

```
529: \def\printii #1&{\gdef\currii{#1}\noindent\everyii
530:    \hskip-\iindent \ignorespaces\printiiA#1//}
531: \def\printiiA #1/{\if$#1$\expandafter\printiiB\else
532:    \expandafter\scanprevii\previi/&\gdef\tmpb{#1}%
533:    \ifx\tmpa\tmpb \iiemdash \else#1 \gdef\previi{}\fi
534:    \expandafter\printiiA\fi
535: }
536: \def\printiiB{\let\previi=\currii}
537: \def\iiemdash{\kern.1em---\space}
538: \def\everyii{}
```

Makro \iiparparams nastavuje parametry sazby odstavce v rejstříku. Vlevo budeme mít \leftsip rovný \iindent, ale první řádek posuneme o -\iindent (viz řádek kódu 530) takže první řádek je vystrčen doleva. Vpravo máme pružnou mezeru, aby se seznam čísel stran mohl rozumně lámat, když je moc dlouhý. Makro \iiparparams si musí poznačit do makra \orippx původní údaje měněných hodnot. Není možné se totiž schovat do skupiny, protože rejstřík je obvykle tištěn pomocí \begmulti...\endmulti a toto makro občas ukončuje plnění boxu a spouští \flushcolumns. Kdybychom měli \makeindex ve skupině, pak by při \flushcolumns došlo ke křížení skupin. Na konci práce \makeindex na řádku 510 je makro \orippx zavoláno a tím jsou parametry odstavce vráceny do původní podoby.

opmac.tex

```
540: \def\iiparparams{%
541:    \xdef\orippx{\global\rightskip=\the\rightskip
542:                 \global\leftskip=\the\leftskip
543:                 \global\exhyphenpenalty=\the\exhyphenpenalty}
544:    \global\rightskip=0pt plus1fil
545:    \global\exhyphenpenalty=10000
546:    \global\leftskip=\iindent
547: }
```

Pomocné makro \scanprevii ⟨*expanded-previi*⟩& se podívá do \previi, odloupne z něj úsek před prvním lomítkem a tento úsek definuje jako \tmpa.

opmac.tex

```
549: \def\scanprevii#1/#2&{\def\previi{#2}\def\tmpa{#1}}
```

Výchozí hodnota \previi před zpracováním prvního slova v rejstříku je prázdná.

opmac.tex

```
550: \def\previi{} % previous index item
```

3.13 Abecední řazení rejstříku

Nejprve se zaměříme na vytvoření makra \isAleB \,⟨*heslo1*⟩ \,⟨*heslo2*⟩, které rozhodne, zda je ⟨*heslo1*⟩ řazeno za ⟨*heslem2*⟩ nebo ne. Výsledek zkoumání můžeme prověřit pomocí \ifAleB.

Pro porovnání dvou údajů vyžaduje norma dva průchody. V prvním (primární řazení) se rozlišuje jen mezi písmeny A B C Č D E F G H Ch I J K L M N O P Q R Ř S Š T U V W X Y Z Ž. Pokud jsou hesla z tohoto pohledu stejná, pak se provede druhý průchod (sekundární řazení), ve kterém jsou řazena neakcentovaná písmena před přehlasovaná před čárkovaná před háčkovaná před stříškovaná před kroužkovaná a dále s nejnižší prioritou malá písmena před velká. Připravíme si tedy dvě sady \lccode dvojic: pro první průchod a pro druhý. Porovnávaná hesla zkonvertujeme pomocí \lowercase při nastavení \lccode odpovídajícího průchodu. Pak takto zkonvertovaná hesla teprve začneme porovnávat.

Makro \setprimarysorting připraví \lccode znaků české a slovenské abecedy pro první průchod a \setsecondarysorting pro druhý průchod. Makro \setprimarysorting expanduje \sortingdata a předhodí před takto expandovaná data \act. Povšimneme si, že pro první průchod dostanou stejný

\previi: 19 \iiemdash: 19 \printiiB: 19 \currii: 19 \everyii: 19 \iiparparams: 18–19 \orippx: 18–19 \scanprevii: 19 \setprimarysorting: 18–20, 22 \setsecondarysorting: 20, 22 \sortingdata: 20

\lccode všechny znaky na společném řádku makra \sortingdata, zatímco v druhém průchodu budou mít všechny znaky z tohoto makra rozdílný \lccode, ve vzestupném pořadí. Je to tím, že v makru \setprimarysorting se zvedá \tmpnum jen v místě čárky, zatímco v \setsecondarysorting se \tmpnum zvedá pro každý znak. Nejnižší hodnotu má mezera vyznačená v \sortingdata pomocí {␣}. Tím je zaručeno, že kratší slovo je řazeno dřív než delší slovo se stejným začátkem, obsahující celé kratší slovo (ten tučňák≺tento). Je sice pravda, že ASCII hodnota mezery je ještě menší, ale my musíme mezeru někam šoupnout na jiný kód než 32, jinak by nám ji nepřečetlo makro s neseparovaným parametrem. Ovšem my budeme chtít mezeru přečíst.

opmac.tex

```
\def\sortingdata{%
  /,{ },-,&,@,%
  aA\"a\"A\'a\'A,%
  bB,%
  cC,%
  \v c\v C,%
  dD\v d\v D,%
  eE\'e\'E\v e\v E,%
  fF,%
  gG,%
  h^^HH,%
  ^^T^^U^^V,%
  iI\'i\'I,%
  jJ,%
  kK,%
  lL\'l\'L\v l\v L,%
  mM,%
  nN\v n\v N,%
  oO\"o\"O\'o\'O\^o\^O,%
  pP,%
  qQ,%
  rR\'r\'R,%
  \v r\v R,%
  sS,%
  \v s\v S,%
  tT\v t\v T,%
  uU\"u\"U\'u\'U\r u\r U,%
  vV,%
  wW,%
  xX,%
  yY,\'y\'Y,%
  zZ,%
  \v z\v Z,%
  0,1,2,3,4,5,6,7,8,9,'.%
}
\def\setprimarysorting {\csname sort:\csname lan:\the\language\endcsname \endcsname
  \def\act##1{\ifx##1.\else
     \ifx##1,\advance\tmpnum by1
     \else \lccode`##1=\tmpnum \fi
     \expandafter \act \fi}%
  \ifx\r\undefined
     \opwarning{\noexpand\csaccents is unused, falling back to ASCII sorting}%
     \gdef\sortingdata{.}\global\let\chsorting=n%
  \else
     \xdef\sortingdata{\sortingdata}% expand sorting data now
  \fi
  \tmpnum=133 \expandafter \act\sortingdata \setignoredchars
}
\sdef{sort:en}{\global\let\chsorting=n} % skipping ch processing in English language

\def\setsecondarysorting {\def\act##1{\ifx##1.\else
     \ifx##1,\else \advance\tmpnum by1 \lccode`##1=\tmpnum \fi
     \expandafter \act \fi}%
  \tmpnum=133 \expandafter \act\sortingdata \setignoredchars
}
```

Jedním z problémů českého řazení je dvojhláska ch. Tu potřebujeme proměnit v jediný znak. Pro potřeby řazení proměníme ch v ^^T, Ch v ^^U a CH v ^^V. Uděláme to následujícím okultním kódem,

který definuje makro \preparesorting \,⟨heslo⟩. Toto makro připraví pomocí do \tmpb ⟨heslo⟩ zkonvertované krz \lowercase, ovšem nejprve je dvojhláska ch nahrazena jedním znakem. Makro \chsorting je implicitně nedefinované, což znamená, že pracujeme s dvojhláskou ch. Na řádku 603, 590 a 597 je ovšem nastaveno \chsorting jako n, což je vzkaz, že dvojhlásku ch nechceme interpretovat.

opmac.tex

```
610: \bgroup
611: \lccode‘4=‘c \lccode‘5=‘h \lccode‘6=‘C \lccode‘7=‘H
612: \lowercase{
613: \gdef\iiscanch #145#2\relax{#1\if$#2$\else^^T\iiscanch #2\relax\fi}
614: \gdef\iiscanCh #165#2\relax{#1\if$#2$\else^^U\iiscanCh #2\relax\fi}
615: \gdef\iiscanCH #167#2\relax{#1\if$#2$\else^^V\iiscanCH #2\relax\fi}
616: \gdef\preparesorting#1{\expandafter\preparesortingA\string#1&}
617: \gdef\preparesortingA#1#2#3&{\xdef\tmpb{#3}%
618:   \ifx\chsorting\undefined
619:     \xdef\tmpb{\expandafter\iiscanch\tmpb 45\relax}%
620:     \xdef\tmpb{\expandafter\iiscanCh\tmpb 65\relax}%
621:     \xdef\tmpb{\expandafter\iiscanCH\tmpb 67\relax}\fi
622:   \lowercase\expandafter{\expandafter\gdef\expandafter\tmpb\expandafter{\tmpb}}%
623:   \xdef\tmpb{\expandafter\removedot\tmpb.\relax}%
624: }}
625: \egroup
```

Tento kód je bohužel obtížněji čitelný, protože makra \iiscanch a další potřebují mít separátor ch ve stavu, kdy jednotlivá písmena mají \catcode 12. Pracujeme totiž s výstupem primitivu \string, který bohužel vše (až na mezeru) balí do tokenů s kategorií 12. Proto je celý kód obalen do \bgroup, \egroup a \lowercase. Tam jsou znaky 4, 5, 6, 7, které mají kategorii 12, šoupnuty na c, h, C, H. Po tomto dešifrování tedy vidíme, že makro \iiscanch je definováno takto:

```
\gdef\iiscanch #1ch#2\relax{#1\if$#2$\else^^T\iiscanch #2\relax\fi}
\iiscanch Schází hrách, který bych házel na stěnu.ch\relax
```

Uvedený příklad expanduje postupně na

```
#1<-S
#2<-ází hrách, který bych házel na stěnu.ch
=> s^^T\iiscanch #2\relax

#1<-ází hrá
#2<-, který bych házel na stěnu.ch
=> s^^Tází hrá^^T\iiscanch #2\relax

#1<-, který by
#2<- házel na stěnu.ch
=> s^^Tází hrá^^T, který by^^T\iiscan #2\relax

#1<- házel na stěnu.
#2<-
=> s^^Tází hrá^^T, který by^^T házel na stěnu.
```

a to nahradí všechny výskyty dvojhlásky ch znakem ^^T. Analogicky pracují makra \iiscanCh a \iiscanCH. Makro \preparesortingA nakonec zavolá všechna tři makra, takže máme nahrazeny všechny dvojhlásky ch, Ch i CH.

V rámci optimalizace rychlosti jsou před algoritmem na setřídění seznamu všechna hesla jednorázově zkonvertovaná podle pravidel prvního průchodu řazení a tato data jsou uložena v prvním datovém údaji hesla (ve druhém máme seznam stránek). Není tedy nutné dělat konverzi při každém porovnávání dvou hesel. Ovšem, pokud porovnání hesel vyjde bez rozdílu, je potřeba provést druhý průchod řazení (sekundární řazení). Ten nastavujeme jednotlivě jen pro takové dvojice hesel, kde to je potřeba. Pravděpodobnost, že to je vůbec někdy potřeba, je mizivá. Data pro primární řazení jsou tedy už připravena na řádcích 494 až 502 v makru \makeindex.

\preparesorting: 18, 21–22 \chsorting: 20–21 \iiscanch: 21 \iiscanCh: 21 \iiscanCH: 21 \preparesortingA: 21

V českém řazení se nemá přihlížet na interpunkční znaky (tečka, středník, otazník, atd.). Hesla máme řadit tak, jako kdyby tam tyto znaky nebyly. Kdyby se dvě hesla podle tohoto pravidla nelišila, norma předepisuje nasadit cca čtvrtý průchod, ve kterém se tyto znaky rozliší. Čtvrtý průchod implementován není: hesla lišící se jen interpunkčními znaky, jsou v OPmac při řazení nerozlišitelná, tj. jsou řazena v pořadí, v jakém vstupují do rejstříku. Ignorování interpunkčních znaků je provedeno tak, že všem těmto znakům je přidělen makrem `\setignoredchars` `\lccode` tečky a tečka je při zpracování v `\setprimarysorting` a `\setsecondarysorting` odstraněna makrem `\removedot`.

opmac.tex
```
\def\removedot #1.#2\relax{#1\if$#2$\else\removedot #2\relax\fi}
\def\setignoredchars{\setlccodes ,.;.?.!.:.’."|.(.).[.].<.>.=.+.{}{}}
```

Připravíme si `\newif`, kterým ohlásíme výsledek porovnání dvou hesel:

opmac.tex
```
\newif \ifAleB
```

Makro `\isAleB` `\,`⟨*heslo1*⟩ `\,`⟨*heslo2*⟩ spustí `\testAleB` ⟨*zkonvertované-heslo1*⟩`&\relax` ⟨*zkonvertované-heslo2*⟩`&\relax \,`⟨*heslo1*⟩ `\,`⟨*heslo2*⟩.

opmac.tex
```
\def\isAleB #1#2{%
  \edef\tmp{\firstdata#1\empty\relax\firstdata#2\empty\relax%
            \noexpand#1\noexpand#2}%
  \expandafter \testAleB \tmp
}
```

Idea makra `\testAleB` lexikograficky porovnávající dvě slova je v tom, že ze dvou stringů v parametru oddělených `\relax` postupně odlupuje vždy první znak `#1` a `#3` z každého stringu a ten porovnává a samozřejmě při rovnosti rekurzivně zavolá samo sebe. Pokud jsme se dostali na konec bez rozhodnutí, co je menší, narazíme na znak `&`. V takovém případě přestoupíme do sekundárního průchodu.

opmac.tex
```
\def\testAleB #1#2\relax #3#4\relax #5#6{%
  \if #1#3\if #1&\testAleBsecondary #5#6%
          \else \testAleB #2\relax #4\relax #5#6%
          \fi
  \else \ifnum ‘#1<‘#3 \AleBtrue \else \AleBfalse \fi
  \fi
}
```

Makro `\testAleBsecondary` `\,`⟨*heslo1*⟩ `\,`⟨*heslo2*⟩ založí skupinu, v ní nastaví `\lccode` dle sekundárního řazení a pomocí `\preparesorting` připraví zkonvertovaná data do `\tmpa` a `\tmpb`. Na chvosty těchto dat přidám nulu a jedničku, aby porovnání vždy nějak dopadlo, a spustím `\testAleBsecondaryX`, což pracuje obdobně, jako `\testAleB`.

opmac.tex
```
\def\testAleBsecondary#1#2{%
  \bgroup
     \setsecondarysorting
     \preparesorting#1\let\tmpa=\tmpb \preparesorting#2%
     \edef\tmp{\tmpa0\relax\tmpb1\relax}%
     \expandafter\testAleBsecondaryX \tmp
  \egroup
}
\def\testAleBsecondaryX #1#2\relax #3#4\relax {%
  \if #1#3\testAleBsecondaryX #2\relax #4\relax
  \else \ifnum ‘#1<‘#3 \global\AleBtrue \else \global \AleBfalse \fi
  \fi
}
```

Nyní můžeme pomocí `\isAleB\,`⟨*heslo1*⟩`\,`⟨*heslo2*⟩`\ifAleB` rozhodnout, který ze dvou daných parametrů má být řazen dříve. Stačí tedy už jen naprogramovat celkové řazení seznamu. Toto makro vycházející z algoritmu mergesort vytvořil můj syn Miroslav. Makro bylo poprvé použito v DocByTEXu, což je nástroj, kterým je například pořízena i tato dokumentace.

Makro `\dosorting` pomocí pomocného makra `\act` doplní za každý údaj v `\iilist` čárku a dále předloží makru `\mergesort` jako parametr obsah `\iilist` ukončený `\end,\end`, vyprázdní `\iilist` a spustí `\mergesort`.

`\setignoredchars`: 20, 22 `\removedot`: 21–22 `\isAleB`: 19, 22–23 `\testAleB`: 22
`\testAleBsecondary`: 22 `\testAleBsecondaryX`: 22 `\dosorting`: 18, 23

opmac.tex

```
\def\dosorting{%
   \message{Opmac: Sorting index...}
   \def\act##1{\ifx##1\relax\else \global\addto\iilist{##1,}%
             \expandafter\act\fi}
   \expandafter\removeiilist \expandafter\act \iilist\relax
   \expandafter\removeiilist \expandafter\mergesort \iilist \end,\end
}
\def\removeiilist{\gdef\iilist{}}
```

Makro \mergesort pracuje tak, že bere ze vstupní fronty vždy dvojici skupin položek, každá skupina je zatříděná. Skupiny jsou od sebe odděleny čárkami. Tyto dvě skupiny spojí do jedné a zatřídí. Pak přejde na následující dvojici skupin položek. Jedno zatřídění tedy vypadá například takto: dvě skupiny: `eimn,bdkz,` promění v jedinou skupinu `bdeikmnz,`. V tomto příkladě jsou položky jednotlivá písmena, ve skutečnosti jsou to kontrolní sekvence, které obsahují celá slova.

Na počátku jsou skupiny jednoprvkové (\iilist odděluje každou položku čárkou). Makro \mergesort v tomto případě projde seznam a vytvoří seznam zatříděných dvoupoložkových skupin, uložený zpětně v \iilist. V dalším průchodu znovu vyvrhne \iilist do vstupní fronty, vyprázdní ho a startuje znovu. Nyní vznikají čtyřpoložkové zatříděné skupiny. Pak osmipoložkové atd. V závěru (na řádku 676) je první skupina celá setříděná a druhá obsahuje `\end`, tj. všechny položky jsou už setříděné v první skupině, takže stačí ji uložit do \iilist a ukončit činnost. Pomocí \gobbletoend odstraníme druhé `\end` ze vstupního proudu.

opmac.tex

```
\def\mergesort #1#2,#3{% by Miroslav Olsak
   \ifx,#1                      % prazdna-skupina,neco,  (#2=neco #3=pokracovani)
      \addto\iilist{#2,}        % dvojice skupin vyresena
      \return{\fif\mergesort#3}%    % \mergesort pokracovani
   \fi
   \ifx,#3                      % neco,prazna-skupina,  (#1#2=neco #3=,)
      \addto\iilist{#1#2,}%     % dvojice skupin vyresena
      \return{\fif\mergesort}%      % \mergesort dalsi
   \fi
   \ifx\end#3                   % neco,konec (#1#2=neco)
      \ifx\empty\iilist             % neco=kompletni setrideny seznam
         \def\iilist{#1#2}%
         \return{\fif\fif\gobbletoend}%   % koncim
      \else                     % neco=posledni skupina nebo \end
         \return{\fif\fif \expandafter\removeiilist    % spojim \indexbuffer+necoa cele znova
               \expandafter\mergesort\iilist#1#2,#3}%
   \fi\fi                       % zatriduji: p1+neco1,p2+neco2, (#1#2=p1+neco1 #3=p2)
   \isAleB #1#3\ifAleB          % p1<p2
      \addto\iilist{#1}%        % p1 do bufferu
      \return{\fif\mergesort#2,#3}%         % \mergesort neco1,p2+neco2,
   \else                        % p1>p2
      \addto\iilist{#3}%        % p2 do bufferu
      \return{\fif\mergesort#1#2,}%         % \mergesort p1+neco1,neco2,
   \fi
   \relax % zarazka, na ktere se zastavi \return
}
```

Jádro \mergesort vidíme na řádcích 683 až 688. Makro \mergesort sejme ze vstupního proudu do `#1` první položku první skupiny, do `#2` zbytek první skupiny a do `#3` první položku druhé skupiny. Je-li `#1<#3`, je do výstupního zatříděného seznamu `\indexbuffer` vložen `#1`, ze vstupního proudu je `#1` odebrán a \mergesort je zavolán znovu. V případě `#3<#1` je do `\indexbuffer` vložen `#3`, ze vstupního proudu je `#3` odebrán a \mergesort je zavolán znovu. Řádky 667 až 673 řeší případy, kdy je jedna ze skupin prázdná: je potřeba vložit do `\indexbuffer` zbytek neprázdné skupiny a přejít na další dvojici skupin. Ostatní řádky makra se vyrovnávají se skutečností, že zpracování narazilo na zarážku `\end,\end` a je tedy potřeba vystartovat další průchod.

3.14 Více sloupců

Makro pro sazbu do více sloupců je převzato z TBN, kde je podrobně vysvětleno na stranách 224 až 245. Základní myšlenka makra spočívá v tom, že se naplní jeden velký `\vbox` (box6) jedním sloupcem a

\mergesort: 22–23 \gobbletoend: 23

\endmulti jej rozlomí do sloupců požadované výšky a strčí do sazby. Není k tomu nutno měnit výstupní rutinu. Makro z TBN je zde v OPmac ve dvou věcech přepracováno:

- Důslednější balancování sloupců vylučující možnost ztráty sazby a umožňující mít sazbu s nezlomitelnými mezerami mezi řádky.
- Makro měří kumulovanou sazbu a umožňuje při rozsáhlém množství tiskového materiálu přechodně přejít do režimu „vyprazdňování“.

Makra \begmulti, \endmulti, \corrsize, \makecolumns a \splitpart pracují zhruba tak, jak je popsáno v TBN.

opmac.tex

```
\def\corrsize #1{%% #1 := #1 + \splittopskip - \topskip
   \advance #1 by \splittopskip \advance #1 by-\topskip}

\def\begmulti #1 {\par\wipeepar\multiskip\penalty0 \def\Ncols{#1}
   \splittopskip=\baselineskip
   \setbox6=\vbox\bgroup\penalty0
   %% \hsize := Sirka sloupce = (\hsize+\colsep) / n - \colsep
   \advance\hsize by\colsep
   \divide\hsize by\Ncols  \advance\hsize by-\colsep
   \dimen0=0pt
   \def\par{\endgraf\advance\dimen0 by\the\prevgraf\baselineskip
            \ifdim\dimen0>.9\maxdimen \message{flushcolumns:}%
               \global\let\balancecolumns=\flushcolumns \expandafter \endmulti
            \fi}%
}
\def\endmulti{\vskip-\prevdepth\vfil\egroup \setbox1=\vsplit6 to0pt
   %% \dimen1 := the free space on the page
   \ifdim\pagegoal=\maxdimen \dimen1=\vsize \corrsize{\dimen1}
   \else \dimen1=\pagegoal \advance\dimen1 by-\pagetotal \fi
   \ifdim \dimen1<2\baselineskip
     \vfil\break \dimen1=\vsize \corrsize{\dimen1} \fi
   \dimen0=\ht6 \divide\dimen0 by\Ncols \relax
   %% split the material to more pages?
   \ifdim \dimen0>\dimen1 \splitpart
   \else \balancecolumns \fi  % only balancing
   \multiskip\relax}
\def\makecolumns{\bgroup % full page, destination height: \dimen1
   \vbadness=20000 \setbox1=\hbox{}\tmpnum=0
   \loop \ifnum\Ncols>\tmpnum
      \advance\tmpnum by1
      \setbox1=\hbox{\unhbox1 \vsplit6 to\dimen1 \hss}
   \repeat
   \hbox{}\nobreak\vskip-\splittopskip \nointerlineskip
   \line{\unhbox1\unskip}
   \egroup}
\def\splitpart{%
   \makecolumns % full page
   \vskip 0pt plus 1fil minus\baselineskip \break
   \dimen0=\ht6 \divide\dimen0 by\Ncols \relax
   \dimen1=\vsize \corrsize{\dimen1}\dimen2=\dimen1
   \advance\dimen2 by-\Ncols\baselineskip
   %% split the material to more pages?
   \ifvoid6 \else
      \ifdim \dimen0>\dimen2 \expandafter\expandafter\expandafter \splitpart
      \else \balancecolumns % last balancing
   \fi \fi
}
```

Výstup rozlomené sazby do sloupců probíhá ve dvou režimech: když je třeba sloupci zaplnit celou stránku, použijeme \makecolumns. Toto makro neřeší otázku, že může v kumulovaném boxu 6 zbýt nějaká sazba, protože se předpokládá, že lámání bude pokračovat na další straně. Pokud ale je na aktuální straně vícesloupcová sazba ukončena, použijeme propracovanější \balancecolumns. Toto makro si zazálohuje

\begmulti: 5, 19, 24–25 \endmulti: 5, 19, 24–25 \corrsize: 24 \makecolumns: 24
\splitpart: 24–25 \balancecolumns: 24–25

materiál z boxu 6 do boxu 7 a jme se zkoušeti rozlomit box 6 na sloupce s výškou \dimen0. Pokud ale po rozlomení není výchozí box 6 zcela prázdný, makro zvětší krapánek (o 0,2\baselineskip) požadovanou výšku, vrátí se k zálohované sazbě v boxu 7 a zkusí rozlomit znovu. To opakuje tak dlouho, dokud je box 6 prázdný.

opmac.tex

```
\def\balancecolumns{\bgroup \setbox7=\copy6 % destination height: \dimen0
   \ifdim\dimen0>\baselineskip \else \dimen0=\baselineskip \fi
   \vbadness=20000
   \def\tmp{%
      \setbox1=\hbox{}\tmpnum=0
      \loop \ifnum\Ncols>\tmpnum
         \advance\tmpnum by1
         \setbox1=\hbox{\unhbox1
              \ifvoid6 \hbox to\wd6{\hss}\else \vsplit6 to\dimen0 \fi\hss}
      \repeat
   \ifvoid6 \else
      \advance \dimen0 by.2\baselineskip
      \setbox6=\copy7
      \expandafter \tmp \fi}\tmp
   \hbox{}\nobreak\vskip-\splittopskip \nointerlineskip
   \hbox to\hsize{\unhbox1\unskip}%
   \egroup
}
```

Když je sazba plněna do boxu 6, může ji být tak moc, že tento box překročí maximální výšku boxu, která je v TEXu bohužel omezena na cca pět metrů (16383 pt). Proto na řádku 708 je předefinován \par, který přičítá do \dimen0 celkovou výšku postupně kumulované sazby. Jakmile tato výška dosáhne 0.9\maxdimen, předefinujeme makro \balancecolumns na \flushcolumns a spustíme předčasně \endmulti. Toto makro vyprázdní box pomocí opakovaného \splitpart, ovšem nevyprázdní ho celý. Jen tu část, která zaplní celé stránky. Jakmile bude chtít \splitpart přejít k vybalancování sazby pomocí \balancecolumns na řádku 742, spustí se místo běžného \balancecolumns makro \flushcolumns. Toto makro ignoruje zbytek činnosti \endmulti až po \relax na řádku 723, takže vyskočí z trojitě zanořeného \if. Musí tedy vrátit příslušné množství \fi a dále vystartuje nový \setbox6=\vbox. Uvnitř tohoto boxu nejprve vysype zbytek boxu 6, pomocí \unskip\unskip odstraní \vfil a \vskip-\prevdepth, který tam vložil \endmulti na řádku 713, vrátí se k zálohovanému významu makra \ibalancecolumns a znovu definuje \par obdobným způsobem jako v \begmulti. Pak pokračuje ve čtení sazby.

opmac.tex

```
\def\flushcolumns#1\relax{\fi\fi\fi
   \setbox6=\vbox\bgroup\penalty0
   \global\let\balancecolumns=\ibalancecolumns
   \dimen0=\ht6 \unvbox6 \unskip\unskip
   \advance\hsize by\colsep
   \divide\hsize by\Ncols  \advance\hsize by-\colsep
   \def\par{\endgraf\advance\dimen0 by\the\prevgraf\baselineskip
            \ifdim\dimen0>.9\maxdimen \message{flush-columns:}%
               \global\let\balancecolumns=\flushcolumns \expandafter \endmulti
            \fi}%
}
\let\ibalancecolumns=\balancecolumns
```

3.15 Barvy

Deklarujeme \ifpgcolor, což budeme potřebovat v makru \locpgcolor. Dále deklarujeme \lastpage. Tento registr budeme průběžně zvětšovat při čtení REF souboru v makru \Xpage a na konci čtení REF souboru tedy bude obsahovat číslo poslední strany.

opmac.tex

```
\newif\ifpgcolor \pgcolorfalse
\newcount\lastpage  \lastpage=0  % the last page of the document
```

Od verze pdfTEXu 1.40 nabízí tento program primitivy \pdfcolorstackinit a \pdfcolorstack. Makra v OPmac tyto primitivy nepoužívají, protože:

\flushcolumns: 19, 24–25 \ibalancecolumns: 25 \ifpgcolor: 25–26 \lastpage: 11, 25, 27, 39

- Řešení v OPmac jsem vytvořil a použil podstatně dřív, než si programátoři pdfTEXu vůbec všimli, že existuje problém s přecházením barev na nové stránky.
- Primitivy `\pdfcolorstackinit` a `\pdfcolorstack` stále nejsou dokumentované.
- Barevné řešení v `pdftex.def` pro LaTEX nepovažuji za koncepčně správné, protože se snaží pomocí `\aftergroup` vytvořit zdání, že barvy fungují lokálně uvnitř skupiny. To je zbytečná komplikace navíc, která jde proti přirozenosti nastavování barev v PDF souboru.

Barvy se v PDF přepínají pomocí PDF speciálů ⟨*num*⟩␣⟨*num*⟩␣⟨*num*⟩␣⟨*num*⟩␣`k` (pro text a plochy) a ⟨*num*⟩␣⟨*num*⟩␣⟨*num*⟩␣⟨*num*⟩␣`K` pro linky. Pro oba typy barev připravíme barevná makra `\Blue`, `\Red`, `\Brown`, `\Green`, `\Yellow`, `\Cyan`, `\Magenta`, `\White`, `\Grey`, `\LightGrey`, `\Black`. Uživatel si může definovat další.

opmac.tex
```
\def\Blue{\setcmykcolor{1.0 1.0 0.0 0}}
\def\Red{\setcmykcolor{0.0 1.0 1.0 0}}
\def\Brown{\setcmykcolor{0 0.67 0.67 0.5}}
\def\Green{\setcmykcolor{1.0 0.0 1.0 0.0}}
\def\Yellow{\setcmykcolor{0.0 0.0 1 0}}
\def\Cyan{\setcmykcolor{1.0 0.0 0 0}}
\def\Magenta{\setcmykcolor{0.0 1.0 0 0}}
\def\White{\setcmykcolor{0 0 0 0}}
\def\Grey{\setcmykcolor{0 0 0 0.5}}
\def\LightGrey{\setcmykcolor{0 0 0 0.2}}
\def\Black{\setcmykcolor{0 0 0 1}\global\pgcolorfalse}
```

Problém barev v PDF je, že od výskytu speciálu pro barvu jsou změněné barvy až po jiný výskyt takového speciálu nebo po konec strany. Na každé nové straně začíná sazba v barvě černé. Nám ovšem někdy může obarvený text přetéci na další stranu. Pak ale musíme v `\output` rutině (v makru `\prepage`) nastavit barvu, která přetekla. Ovšem jak poznáme, že něco přeteklo do další strany? Jedině pomocí REF souboru a asynchronního `\write`. Proto makro `\setcmykcolor` nevkládá do PDF jen požadovaný speciál, ale taky ukládá o sobě informaci do REF souboru. Jakmile se při čtení REF souboru objeví `\Xpage`, zkontrolujeme, zda naposledy nastavená barva byla černá. Pokud ne, dáme pokyn makru `\prepage` prostřednictvím makra `\pg:`⟨*číslo-strany*⟩, aby aktivovalo hned po `\headline` barvu naposledy nastavenou. To je tedy základní idea, nyní implementace:

opmac.tex
```
\ifpdftex
  \def\setcmykcolor#1{\special{PDF:#1 \pdfK}%
    \ifpgcolor\else
       \openref\expandafter\wref\expandafter\Xpdfcolor\expandafter{\pdfK{#1}}%
    \fi
  }
\else
  \def\setcmykcolor#1{}
\fi

\def\linecolor#1{{\def\pdfK{K}#1}}
\def\pdfK{k}
\def\locpgcolor{\global\pgcolortrue}
```

Makro `\pdfK` má implicitně hodnotu `k` (barva pro texty a plochy) a přechodně při použití `\linecolor` má hodnotu `K`. Právě nastavenou barvu pak makro `\setcmykcolor` uloží do REF souboru. Jen tehdy, když je explicitně známo, že barva nepřeteče do další strany, je možné použít před použitím přepínače barvy prefix `\locpgcolor`, který potlačí v takovém případě tisk informací do REF souboru. Následně budou potlačeny zápisy informací do REF souboru všech barevných přepínačů až po `\Black` včetně. Další barevné přepínače (po `\Black`) se chovají už normálně. Toto opatření je zde jen proto, aby byl REF soubor poněkud menší a přehlednější i v případě, kdy navrhujeme bohatě barevný dokument.

Makra `\Xpdfcolork` a `\XpdfcolorK` udržují při čtení REF souboru informaci o naposledy vložené barvě v makrech `\pdflastcolork` a `\pdflastcolorK` (název koresponduje typu barvy).

\Blue: 26 \Red: 26 \Brown: 26 \Green: 26 \Yellow: 26 \Cyan: 26 \Magenta: 26
\White: 26 \Grey: 26 \LightGrey: 26–27 \Black: 26–28, 44 \setcmykcolor: 26
\pdfK: 26 \linecolor: 26 \locpgcolor: 25–28, 44 \Xpdfcolork: 27 \XpdfcolorK: 27
\pdflastcolork: 27 \pdflastcolorK: 27

opmac.tex
```
\def\Xpdfcolork#1{\def\pdflastcolork{#1}}
\def\XpdfcolorK#1{\def\pdflastcolorK{#1}}
\def\pdfblackcolor{0 0 0 1}
\let\pdflastcolork=\pdfblackcolor \let\pdflastcolorK=\pdfblackcolor
```

Output rutina vkládá na každé stránce do REF souboru `\Xpage {⟨číslo-strany⟩}`. Toto makro při čtení REF souboru kontroluje, zda byla naposledy použita na předcházející straně barva černá. Pokud ne, přidá do makra `\pg:⟨číslo-strany⟩` pokyn pro nastavení barvy ve tvaru `\setpgcolor ⟨k-nebo-K⟩{⟨čísla-barvy⟩}`. Toto makro se spustí v output rutině na straně ⟨číslo-strany⟩ za `\headline`, ale před sestavením boxu s textem.

opmac.tex
```
\def\Xpage#1{\lastpage=#1 \fnotenumlocal=0
   \ifx\pdflastcolork\pdfblackcolor\else
      \isdefined{pg:#1}\iftrue \else \sxdef{pg:#1}{}\fi
      {\let\setpgcolor=\relax \sxdef{pg:#1}%
            {\csname pg:#1\endcsname\setpgcolor k{\pdflastcolork}}}\fi
   \ifx\pdflastcolorK\pdfblackcolor\else
      \isdefined{pg:#1}\iftrue \else \sxdef{pg:#1}{}\fi
      {\let\setpgcolor=\relax \sxdef{pg:#1}%
            {\csname pg:#1\endcsname\setpgcolor K{\pdflastcolorK}}}\fi
}
\def\setpgcolor#1#2{\special{PDF:#2 #1}}
```

Makro `\draft` vloží do `\headline` box nulové výšky a šířky `\draftbox`, který vystrčí svou šedou sazbu ven ze svého rozměru a je tištěn dřív, než jakýkoli jiný materiál na stránce.

opmac.tex
```
\def\draft{\edef\tmp{\headline={\noexpand\draftbox{\tenbf DRAFT}\the\headline}}
   \ifpdftex \tmp \else
      \opwarning{\string\draft: Grey color is possible in pdfTeX only}\fi}
```

V makru `\draftbox {⟨text⟩}` je ⟨text⟩ otočen o 55 stupňů, zvětšen desetkrát a vytištěn v barvě `\LightGrey`. K tomu jsou využity PDF transformace souřadnic.

opmac.tex
```
\def\draftbox#1{\vbox to0pt{\setbox0=\hbox{\typosize[10/]#1}%
   \kern.5\vsize \kern4\wd0 \hbox to0pt{\kern.5\hsize \kern-2.5\wd0
   \pdfsave \pdfrotate{55}\pdfscale{10}{10}%
   \hbox to0pt{\locpgcolor\LightGrey \box0\hss\Black}%
   \pdfrestore
\hss}\vss}\hss}
```

3.16 Klikací odkazy

Makro `\destactive [⟨typ⟩:⟨lejblík⟩]{⟨výška⟩}` založí cíl odkazu jen tehdy, když je ⟨lejblík⟩ neprázdný. Ve vertikálním módu se nalepí na předchozí box díky `\prevdepth=-1000pt` a po vložení boxu s cílem vrátí hodnotu `\prevdepth` do původního stavu, aby následující box byl správně řádkován. V horizontálním módu postě vloží `\destbox`. Makro `\destbox [⟨typ⟩:⟨lejblík⟩]{⟨výška⟩}` vytvoří box nulové výšky a z něj vystrčí nahoru cíl klikacího odkazu vzdálený od účaří o ⟨výška⟩. Interně použije pdfTEXový primitiv `\pdfdest` s parametrem `xyz`, což charakterizuje obvyklou možnost chování PDF prohlížeče při odskoku na cíl. Podrobněji viz manuál k pdfTEXu.

opmac.tex
```
\def\destactive[#1:#2]#3{\if$#2$\else\ifvmode
      \tmpdim=\prevdepth \prevdepth=-1000pt
      \destbox[#1:#2]{#3}\prevdepth=\tmpdim
   \else \destbox[#1:#2]{#3}%
   \fi\fi
}
\def\destbox[#1]#2{\vbox to0pt{\kern-#2 \pdfdest name{#1} xyz\vss}}
\def\dest[#1]#2{}
```

V uživatelské dokumentaci je zmíněno místo `\destactive` makro `\dest` se stejnými parametry. Toto makro je implicitně prázdné a tedy nečiné. Teprve `\hyperlinks` je přinutí k činnosti.

\Xpage: 25–27, 38–39, 44 \setpgcolor: 27 \draft: 27 \draftbox: 27 \destactive: 27–28
\destbox: 27 \dest: 11, 13, 27–29, 41–42, 44

Klikací text vytvoří makro \link [⟨typ⟩:⟨lejblík⟩]{⟨barva⟩}{⟨text⟩}. Makro používá pdfTEXový primitiv \pdfstartlink, ve kterém je vymezena výška a hloubka aktivní plochy. Nakonec přepne na požadovanou ⟨barvu⟩ (pokud není černá), vytiskne aktivní ⟨text⟩ a přepne zpět na černou barvu. PdfTEXový primitiv \pdfendlink ukončí sazbu aktivního textu.

opmac.tex

```
\def\link[#1:#2]#3#4{\leavevmode\pdfstartlink height.9em depth.3em
      \pdfborder{#1} goto name{#1:#2}\relax
      \ifx\Black#3#4\else#3#4\Black\fi \pdfendlink
}
```

Makro \urllink [⟨typ⟩:⟨lejblík⟩]{⟨text⟩} pracuje analogicky jako \link. Jen navíc přidává některé atributy do PDF výstupu a pracuje s barvou \urlcolor.

opmac.tex

```
\def\urllink[#1:#2]#3{{\leavevmode\pdfstartlink height.9em depth.3em
   \pdfborder{#1}user{/Subtype/Link/A << /Type/Action/S/URI/URI(#2)>>}\relax
   \urlcolor{#3}\Black\pdfendlink}%
}
```

Makro \url {⟨odkaz⟩} vysází ⟨odkaz⟩ strojopisem, přičemž znak / je na přechodnou dobu aktivní a definován jako \\. Toto makro při sestavování externího odkazu zpětně expanduje na normální /, ale při sazbě expanduje na \discretionary, aby bylo možné v něm rozdělit sazbu do více řádků.

opmac.tex

```
\def\url{\leavevmode\bgroup \adef{/}{\\}%
   \catcode`\_=12 \catcode`\#=12 \catcode`\~=12 \dourl}
\def\dourl#1{\def\\{/}\ulink[url:#1]{\tt\def\\{\discretionary{/}{}{/}}#1}\egroup}
```

Makra \toclink, \pglink, \citelink, \reflink, \ulink, která se specializují na určitý typ linku, implicitně nedělají nic:

opmac.tex

```
\def\toclink#1{#1}
\def\pglink#1{#1}
\def\citelink#1{#1}
\def\reflink[#1]#2{#2}
\def\ulink[#1]#2{#2}
```

Ovšem po použití makra \hyperlinks {⟨barva-lok⟩}{⟨barva-url⟩} se uvedená makra \toclink, \pglink, \citelink a \reflink probouzejí k životu:

opmac.tex

```
\def\hyperlinks#1#2{%
   \let\dest=\destactive
   \def\toclink##1{\locpgcolor\link[toc:##1]{#1}{##1}}%
   \def\pglink##1{\locpgcolor\link[pg:##1]{#1}{##1}}%
   \def\citelink##1{\locpgcolor\link[cite:##1]{#1}{##1}}%
   \def\reflink[##1]##2{\locpgcolor\link[ref:##1]{#1}{##2}}%
   \def\ulink[##1]##2{\urllink[##1]{##2}}%
   \def\urlcolor{#2}%
}
```

PdfTEXové primitivy pro klikací odkazy dovolují dopravit do PDF další atributy odkazu za slovem attr. Tam je možné dát najevo, že chceme vidět aktivní plochy ve formě rámečků. To zařídí makro \pdfborder {⟨typ⟩}, které expanduje na nic, pokud není kontrolní sekvence \⟨typ⟩border definována. Jinak expandují na arrt /C s obsahem podle \⟨typ⟩border.

opmac.tex

```
\def\pdfborder#1{\if$#1$\else \isdefined{#1border}\iftrue
   \if$\csname#1border\endcsname$\else
      attr{/C[\csname#1border\endcsname] /Border[0 0 1]}%
   \fi\fi\fi
}
```

Pokud je dokument zpracován do DVI výstupu, je vhodné výše zmíněná makra deaktivovat:

\link: 28–29 \urllink: 28–29 \url: 28, 42 \toclink: 15, 28 \pglink: 11, 15, 28
\citelink: 28, 40 \reflink: 11, 28 \ulink: 28 \hyperlinks: 27–29 \pdfborder: 28

opmac.tex
```
\ifpdftex \else
  \def\dest[#1]#2{}
  \def\link[#1]#2#3{#3}
  \def\urllink[#1]#2{#2}
  \def\hyperlinks#1#2{}
\fi
```

3.17 Outlines – obsah v záložce PDF dokumentu

Hlavní problém implementace strukturovaného obsahu do záložky PDF dokumentu spočívá v tom, že při vkládání jednotlivých položek obsahu je nutno znát počet přímých potomků každé položky (v rámci stromové struktury položek), ovšem tito přímí potomci budou zařazeni později. OPmac tento problém řeší dvěma průchody nad daty, které jsou vytvořeny pro tisk obsahu, tj. v makru `\toclist`. V prvním průchodu spočítá potřebné potomky a ve druhém průchodu zařadí všechny položky postupně jako „outlines“ do záložky. Připomeneme si, že v `\toclist` se nachází seznam maker tvaru `\tocline{`⟨*odsazení*⟩`}{`⟨*font*⟩`}{`⟨*číslo*⟩`}{`⟨*text*⟩`}{`⟨*strana*⟩`}`. Makro `\outlines {`⟨*úroveň*⟩`}` nejprve nastaví `\tocline` na hodnotu `\outlinesA` a projde `\toclist`. Pak je nastaví na hodnotu `\outlinesB` a znovu projde `\toclist`.

opmac.tex
```
\def\outlines#1{\openref\ifx\toclist\empty
     \opwarning{\noexpand\outlines -- data unavailable. TeX me again}%
   \else
     {\let\tocline=\outlinesA
      \count0=0 \count1=0 \toclist % calculate numbers o childs
      \def\outlinelevel{#1}\let\tocline=\outlinesB
      \count0=0 \count1=0 \toclist}% create outlines
   \fi
}
```

V makru `\outlinesA {`⟨*odsazení*⟩`}{`⟨*font*⟩`}{`⟨*číslo*⟩`}{`⟨*text*⟩`}{`⟨*strana*⟩`}` počítáme potomky. Makro je navrženo tak, aby bylo snadno rozšiřitelné na libovolnou úroveň hloubky stromu, nicméně pro potřeby OPmac stačí hloubka tři (kapitoly, sekce, podsekce). Úroveň uzlu přečteme v parametru ⟨*odsazení*⟩. Pro kapitolu je ⟨*odsazení*⟩=0, pro sekci je ⟨*odsazení*⟩=1 a pro podsekci je ⟨*odsazení*⟩=2. Představme si vedle sebe řadu counterů `\count0:\count1:\count2`. Při sekvenčním čtení jednotlivých uzlů stromu si každý uzel zvětší v této pomyslné řadě hodnotu svého counteru o jedničku. Kapitoly zvětšují `\count0`, sekce `\count1`, podsekce `\count2`. Stačí tedy zvětšit `\count`⟨*odsazení*⟩. Řada counterů pak jednoznačně určuje zpracovávaný uzel. Uzly pro kapitoly mají přidělenu kontrolní sekvenci `ol:\the\count0` a uzly pro sekce mají přidělenu kontrolní sekvenci `ol:\the\count0:\the\count1`. Jsou to makra, jejichž obsahem je počet potomků daného uzlu. Makrem `\addoneol` ⟨*csname*⟩ zvětšíme obsah dané kontrolní sekvence o jedničku. Příkazem `\ifcase`⟨*odsazení*⟩ řešíme, kterému rodiči je třeba zvednout tuto hodnotu. Při nule (kapitola) nikomu, neboť daný uzel nemá rodiče. Při ⟨*odsazeni*⟩=1 zvětšíme o jedničku počet potomků nadřazené kapitole a při ⟨*odsazeni*⟩=2 nadřazené sekci. Asi by bylo přehlednější na začátku definovat všechny potřebné sekvence `ol:`⟨*něco*⟩ a nastavit jim hodnotu 0. Ovšem šetříme pamětí i časem, takže zakládáme sekvenci `ol:`⟨*něco*⟩ teprve v makru `\addoneol` a to tehdy, když je ji poprvé potřeba zvětšit o jedničku.

opmac.tex
```
\def\outlinesA#1#2#3#4#5{%
   \advance\count#1 by1
   \ifcase#1\or
     \addoneol{ol:\the\count0}\or
     \addoneol{ol:\the\count0:\the\count1}\fi
}
\def\addoneol#1{\isdefined{#1}%
   \iftrue \tmpnum=\csname#1\endcsname\relax
           \advance\tmpnum by1 \sxdef{#1}{\the\tmpnum}%
   \else \sxdef{#1}{1}%
   \fi
}
```

\outlines: 29–31 \outlinesA: 29 \addoneol: 29

V makru \outlinesB {⟨odsazení⟩}{⟨font⟩}{⟨číslo⟩}{⟨text⟩}{⟨strana⟩} vkládáme jednotlivou položku obsahu do záložek pomocí pdfTEXového primitivu \pdfoutline␣goto␣name{⟨lejblík⟩}␣count⟨potomci⟩␣{⟨text⟩}. Číslo ⟨potomci⟩ je opatřeno znaménkem mínus právě tehdy, když chceme, aby položka ve výchozím stavu nezobrazovala své potomky, ale jen trojúhelníček. Potomci se zobrazí až po kliknutí na trojúhelníček. V makru \outlinelevel máme makrem \outlines připravenu úroveň rozevření, kterou si uživatel přeje. Nejprve přičtením \count⟨odsazení⟩ dostaneme řadu \count0:\count1:\count2 do stejného stavu, jako v předchozím prvním průchodu a máme tím jednoznačně přidělen uzel stromu. Do \tmpnum vložíme údaj o počtu potomků daného uzlu. K tomu je potřeba rozvětvit výpočet příkazem \ifcase, protože pro různou úroveň uzlu máme údaj v různě definovaném makru. Příkazem \protectlist zastavíme expanze případných maker registrovaných pomocí \addprotect a definujeme vlnku jako mezeru (v záložce vypadá líp než vlnka). Dále pomocí \setcnvcodesA expandujeme \toasciidata. Pomocí \setlccodes\toasciidata připravíme \lccode znaků tak, aby \lowercase odstranil háčky a čárky. To vzápětí provedeme, ale nejprve ještě do toho může promluvit uživatel v makru \cnvhook, které je implicitně nastaveno na makro prázdné.

opmac.tex

```
\def\outlinesB#1#2#3#4#5{%
   \advance\count#1 by1
   \ifcase#1\tmpnum=\isdefined{ol:\the\count0}%
               \iftrue\csname ol:\the\count0\endcsname\else0\fi \or
            \tmpnum=\isdefined{ol:\the\count0:\the\count1}%
               \iftrue\csname ol:\the\count0:\the\count1\endcsname\else0\fi \or
            \tmpnum = 0 \fi
  \protectlist \def~{ }\setcnvcodesA
  \expandafter \setlccodes \toasciidata{}{}%
  \cnvhook \lowercase{\gdef\tmp{#4}}%
  \pdfoutline goto name{toc:#3} count
     \ifnum#1<\outlinelevel\space\else-\fi\tmpnum {\tmp}\relax
}
```

Makro \setcnvcodesA zkontroluje podle definovanosti \r, zda je zapnutý \csaccents a pokud je, expanduje \toasciidata. Makro \toasciidata potřebujeme expandovat, protože neobsahuje přímý zápis znaků. Důvod je zřejmý, nechceme, aby se soubor opmac.tex stal závislý na použitém kódování.

opmac.tex

```
\def\setcnvcodesA{\global\let\setcnvcodesA=\relax % I am working only once
  \ifx\r\undefined
     \gdef\toasciidata{}
     \opwarning{\noexpand\csaccents unused, CZ/SK outline-conversion is off}%
  \else
     \xdef\toasciidata{\toasciidata}%
  \fi
}
\def\toasciidata{%  Removes Czech+Slovak accents
       AA\'AA\"AA\'aa\" aaBBCC\v CC\v ccDD\v DD\v ddEE\'EE\v EE\'ee\v ee%
       FFGGHHII\'II\'iiJJKKLL\'LL\v LL\'ll\v llMMNN\v NN\v nnOO\'OO\"OO\^OO%
       \'oo\"oo\^ooPPQQRR\v RR\v rrSS\v SS\v ssTT\v TT\v ttUU\'UU\"UU\r UU%
       \'uu\"uu\r  uuVVWWXXYY\'YY\'yyZZ\v ZZ\v zz%
}
```

Na řádku 921 se makro \setlccodes spustí jako \setlccodes␣AAÄAÄÄáa...{}{}. Toto makro si odloupne dva parametry xy, provede \lccode‘x=‘y a v rekurzivním cyklu pokračuje v činnosti, dokud nenarazí na {}{}.

opmac.tex

```
\def\setlccodes#1#2{\if\relax#2\relax \else \lccode‘#1=‘#2 \expandafter \setlccodes \fi}
```

Makro \insertoutline {⟨text⟩} vloží jedinou položku do záložky. Pro tuto položku se předpokládá nulový počet potomků. Využití: uživatel může takto odkázat na začátek nebo konec dokumentu.

opmac.tex

```
\def\insertoutline#1{\pdfdest name{oul:#1} xyz\relax
   \pdfoutline goto name{oul:#1} count0 {#1}\relax
}
```

Pokud je dokument zpracován do DVI výstupu, je vhodné výše zmíněná makra deaktivovat:

\outlinesB: 29–30 \outlinelevel: 29–30 \setcnvcodesA: 30 \toasciidata: 30
\setlccodes: 22, 30 \insertoutline: 30–31

opmac.tex
```
\ifpdftex \else
  \def\outlines#1{\opwarning{DVI output has no outlines}\gdef\outlines##1{}}
  \let\insertoutline=\outlines
\fi
```

3.18 Verbatim

Verbatim výpisy budou odsazeny o \ttindent. Je nastaven na hodnotu \parindent v době čtení souboru a společně s \parindent by měl uživatel změnit i \ttindent. Čítač \ttline čísluje řádky běžného verbatim výstupu, čítač \viline čísluje řádky souboru čteného pomoci \verbinput. Souborový deskriptor \vifile bude přiřazen souboru v makru \verbinput.

opmac.tex
```
\newcount\ttline    \ttline=-1
\newcount\viline
\newread\vifile
```

Makra \setverb, \begtt ...\endtt jsou dokumentována v TBN, str. 29.

opmac.tex
```
\def\setverb{\def\do##1{\catcode`##1=12}\dospecials \catcode`\*=12 }
\def\begtt{\ttskip\bgroup \wipeepar
   \setverb \adef{ }{ }%
   \ifx\savedttchar\undefined \else \catcode\savedttchar=12 \fi
   \parindent=\ttindent
   \tthook\relax
   \ifnum\ttline<0 \else
     \tenrm \thefontscale[700]\let\sevenrm=\thefont
     \everypar={\global\advance\ttline by1
                \llap{\sevenrm\the\ttline\kern.9em}}\fi
   \def\par##1{\endgraf\ifx##1\egroup\else\penalty\ttpenalty\leavevmode\fi ##1}
   \obeylines \startverb}
{\catcode`\|=0 \catcode`\\=12
|gdef|startverb#1\endtt{|tt#1|egroup|ttskip|testparA}}
```

Makro \begtt očichá na konci své činnosti, zda se nachází pod \endtt prázdný řádek (alias \par). K tomu slouží makra \testparA (přeskočí mezeru, která za \endtt vždy je), \testparB (přečte následující znak pomocí \futurelet) a \testparC (ošetří, zda tento následující znak je \par).

opmac.tex
```
\def\testparA{\afterassignment\testparB\let\tmpa= }
\def\testparB{\futurelet\tmpa\testparC}
\def\testparC{\ifx\tmpa\par\else\afternoindent\fi}
```

Makro \activettchar pracuje podobně, jako makro \adef. Navíc potřebuje použít nově načtený znak ve své aktivní kategorii jako separátor vymezující konec parametru.

opmac.tex
```
\def\activettchar#1{%
   \ifx\savedttchar\undefined\else \catcode\savedttchar=\savedttcharc \fi
   \chardef\savedttchar=`#1%
   \chardef\savedttcharc=\catcode`#1%
   \lccode`\~=`#1
   \lowercase {\def~}{\leavevmode\hbox\bgroup\setverb\adef{ }{ }%
                      \intthook\tt\readverb}%
   \lowercase{\def\readverb ##1~}{##1\egroup}%
   \lccode`\~=0 \catcode\savedttchar=13
}
```

Makro \verbinput si pomocí \tmpa ověří, zda minule byl čten stejný soubor. Pokud ne, otevře soubor #2 ke čtení pomocí \openin a uloží do \vifilename jméno naposledy otevřeného souboru. Dále zkontroluje pomocí \ifeof, zda je možné ze souboru číst. Pokud ne, vypíše se varování a pomocí \skiptorelax se přeskočí zbytek obsahu makra až po \relax, takže se neprovede nic dalšího. Je-li soubor úspěšně otevřen nebo byl-li otevřen již minule, pustí se makro \verbinput do prozkoumání parametru #1 zapsaného v závorce před jménem souboru.

\ttline: 31, 33 \viline: 31–33 \vifile: 31–33 \setverb: 31–33 \begtt: 5, 31
\testparA: 31 \testparB: 31, 33 \testparC: 31 \activettchar: 31–32 \verbinput: 5, 31–32
\vifilename: 32–33 \skiptorelax: 32, 40

opmac.tex
```
\def\verbinput (#1) #2 {\par \def\tmpa{#2}%
   \ifx\vifilename\tmpa \else
      \openin\vifile=#2
      \global\viline=0 \global\let\vifilename=\tmpa
      \ifeof\vifile
         \opwarning{\noexpand\verbinput - file "#2" is unable to reading}
         \expandafter\expandafter\expandafter\skiptorelax
      \fi
   \fi
   \viscanparameter #1+\relax
}
\def\skiptorelax#1\relax{}
```

Cílem vyhodnocení parametru v závorce makra \verbinput jsou dva údaje: \vinolines bude obsahovat počet řádků, které je od začátku souboru nutno přeskočit, než se má zahájit přepisování řádků a \vidolines bude obsahovat počet řádků, které se mají přepsat ze souboru do dokumentu. Písmena vi na začátku těchto názvů představují zkratku pro verbinput. Vyšetření parametru ukončeného textem +\relax se v makru \viscanparameter větví na případ, kdy parametr obsahuje symbol + a použije se pak \viscanplus. Druhý případ, kdy uživatel nenapsal symbol plus (takže parametr #2 makra \viscanparameter je prázdný) je dále vyšetřen v makru \viscanminus. Obě makra si oddělí do svých parametrů první a druhou číslici (každá z nich může být prázdná) a nastaví podle zdokumentovaných pravidel pro zápis parametru odpovídající interní údaje \vinolines a \vidolines. Vychází přitom z předpokladu, že registr \viline obsahuje číslo naposledy přečteného řádku (nebo nulu, jsme-li na začátku souboru).

opmac.tex
```
\def \viscanparameter #1+#2\relax{%
   \if$#2$\viscanminus(#1)\else \viscanplus(#1+#2)\fi
}
\def\viscanplus(#1+#2+){%
   \if$#1$\tmpnum=\viline
   \else \ifnum#1<0 \tmpnum=\viline \advance\tmpnum by-#1
       \else \tmpnum=#1
             \advance\tmpnum by-1
             \ifnum\tmpnum<0 \tmpnum=0 \fi % (0+13) = (1+13)
   \fi \fi
   \edef\vinolines{\the\tmpnum}%
   \if$#2$\def\vidolines{0}\else\edef\vidolines{#2}\fi
   \doverbinput
}
\def\viscanminus(#1-#2){%
   \if$#1$\tmpnum=0
      \else \tmpnum=#1 \advance\tmpnum by-1 \fi
   \ifnum\tmpnum<0 \tmpnum=0 \fi  % (0-13) = (1-13)
   \edef\vinolines{\the\tmpnum}%
   \if$#2$\tmpnum=0
      \else \tmpnum=#2 \advance\tmpnum by-\vinolines \fi
   \edef\vidolines{\the\tmpnum}%
   \doverbinput
}
```

Makro \doverbinput provede samotnou práci: přeskočí \vinolines řádků a přepíše \vidolines řádků. To provede v prvním a druhém cyklu \loop. Než se k těmto cyklům dostane, musí udělat jisté přípravné práce. Nejprve odečte od \vinolines počet už přečtených řádků, protože při opakovaném čtení stejného souboru jej neotevíráme znova, jen přeskočíme příslušný menší počet řádků. Pokud ale se ukáže, že rozdíl je záporný (je potřeba se v souboru vracet dozadu), makro znovuotevře soubor ke čtení pomocí \openin a upraví podle toho příslušné údaje o řádcích. Pak zahájí skupinu, dále pomocí \setverb nastaví speciálním znakům kategorii 12 a pomocí \adef{␣}{␣} nastaví mezeře aktivní kategorii (bude expandovat na neaktivní mezeru jako \space) a také nastaví kategorii 12 znaku, který byl deklarován pomocí \activettchar. Připraví odsazení podle \ttindent a spustí uživatelský \tthook. Je-li potřeba tisknout čísla řádků, připraví si na to font \sevenrm, který má velikost rovnu 0,7 násobku základní velikosti. A pustí se do zmíněných dvou cyklů \loop. V obou cyklech se může stát, že narazíme nečekaně na

\vinolines: 32–33 \vidolines: 32–33 \viscanparameter: 32 \viscanplus: 32
\viscanminus: 32 \doverbinput: 32–33

konec souboru. To je ošetřeno testem \ifeof\vifile a následnou úpravou čítače \tmpnum tak, abychom okamžitě vyskočili z cyklu. Druhý cyklus obsahuje ještě jeden speciální rys: přeje-li si uživatel číst až do konce souboru, je nastaveno \vidolines na nulu a před zahájením cyklu je čítač \tmpnum nastaven na -1. Uvnitř cyklu je pak zajištěno, že v tomto případě není čítač zvětšován o jedničku. Po ukončení práce v těchto dvou cyklech je ukončena skupina, vložena mezera \ttskip a makrem \testparB se ověří, zda následuje prázdný řádek.

opmac.tex

```
\def\doverbinput{%
   \tmpnum=\vinolines
   \advance\tmpnum by-\viline
   \ifnum\tmpnum<0
      \openin\vifile=\vifilename\space
      \global\viline=0
   \else
      \edef\vinolines{\the\tmpnum}%
   \fi
   \ttskip\bgroup \wipeepar
   \setverb \adef{ }{ }%
   \ifx\savedttchar\undefined \else \catcode\savedttchar=12 \fi
   \parindent=\ttindent
   \tthook\relax
   \ifnum\ttline<-1 \else
      \tenrm \thefontscale[700]\let\sevenrm=\thefont \fi
   \tmpnum=0 \tt
   \loop \ifeof\vifile \tmpnum=\vinolines\space \fi
         \ifnum\tmpnum<\vinolines\space
         \vireadline \advance\tmpnum by1 \repeat      %% skip line
   \tmpnum=0 \ifnum\vidolines=0 \tmpnum=-1 \fi
   \loop \ifeof\vifile \tmpnum=\vidolines\space \fi
         \ifnum\tmpnum<\vidolines\space
         \vireadline \penalty\ttpenalty \viprintline  %% print line
             \ifnum\vidolines=0 \else\advance\tmpnum by1 \fi \repeat
   \egroup\ttskip\testparB
}
```

V prvním cyklu \loop v těle makra \doverbinput se opakovaně volá \vireadline, což je makro, které přečte další řádek ze souboru. V druhém cyklu se opakovaně volá \vireadline následované \viprintline. Toto makro vytiskne přečtený řádek do dokumentu. Před řádkem může být v \llap vytištěno číslo řádku. Záleží na hodnotě \ttline. Je to naprogramováno v souladu s uživatelskou dokumentací.

opmac.tex

```
\def\vireadline{\read\vifile to \tmp \global\advance\viline by1 }
\def\viprintline{\indent
   \ifnum \ttline<-1 \else
      \llap{\sevenrm\ifnum\ttline<0 \the\viline \else
               \global\advance\ttline by1 \the\ttline \fi \kern.9em}%
   \fi
   \tmp\par % print the line from \tmp
}
```

3.19 Jednoduchá tabulka

Tabulku makrem \table vytvoříme jako \vbox, ve kterém je \halign. Je tedy potřeba načíst deklaraci typu {llc|rr} a převést ji na deklaraci pro \halign. Tato deklarace obsahuje znak # a tento znak se obtížně přidává do těla maker. Nashromáždíme tedy postupně deklaraci pro \halign do registru typu \toks, který je nazvaný \tabdata. Dále definujeme interní \tabstrutA, který bude obsahovat uživatelův \tabstrut, ovšem přechodně budeme toto makro měnit. Také deklarujeme čítač \colnum, ve kterém budeme mít po přečtení deklarace uložen počet sloupců tabulky. Dále během skenování ⟨*deklarace*⟩ vytvoříme makro \ddlinedata, které bude obsahovat &\dditem␣&\dditem␣... (počet těchto dvojic bude roven $n-1$, kde n je počet sloupců). Pokud je v deklaraci dvojitá svislá čára, bude v makru \ddlinedata

\vireadline: 33 \viprintline: 33 \tabdata: 34 \tabstrutA: 34–35 \colnum: 34
\ddlinedata: 33–35

na příslušném místě ještě \vvitem. Makro \ddlinedata pak použijeme v \crli a v \tskip, Strýček Příhoda to může použít jinde a jinak. Konečně makro \vvleft je neprázdné, pokud úplně vlevo tabulky je dvojitá čára.

opmac.tex
```
\newtoks\tabdata
\def\tabstrutA{\tabstrut}
\newcount\colnum  \colnum=0
\def\ddlinedata{}
\def\vvleft{}
```

Makro \table {⟨*deklarace*⟩}{⟨*data*⟩} vypadá takto:

opmac.tex
```
\def\table#1#2{\vbox{\offinterlineskip
   \def\tmpa{}\scantabdata#1\relax
   \halign\expandafter{\the\tabdata\tabstrutA\cr#2\crcr}}}
```

Makro \scantabdata postupně čte znak po znaku z deklarace \table a podle přečteného znaku ukládá do \tabdata odpovídající úsek skutečné deklarace pro \halign. Volá přitom \addtabvrule nebo \addtabitem{\tabdeclare⟨*znak*⟩}.

opmac.tex
```
\def\scantabdata#1{%
   \ifx#1\relax\let\scantabdata=\relax
   \else\if#1|\addtabvrule
   \else\expandafter\ifx\csname tabdeclare#1\endcsname \relax
     \opwarning{tab-declare letter #1 unknown, ignored}%
        \else\expandafter \addtabitem\expandafter{\csname tabdeclare#1\endcsname}%
   \fi\fi\fi \scantabdata
}
```

OPmac předdefinuje tři ⟨*znaky*⟩ pro ⟨*deklaraci*⟩, sice ⟨*znaky*⟩ `c`, `l`, `r` v makrech \tabdeclarec, \tabdeclarel, \tabdeclarer.

opmac.tex
```
\def\tabdeclarec{\tabiteml\hfil##\unsskip\hfil\tabitemr}
\def\tabdeclarel{\tabiteml##\unsskip\hfil\tabitemr}
\def\tabdeclarer{\tabiteml\hfil##\unsskip\tabitemr}
```

Makro \unsskip vkládané na konec každé datové položky odebere mezeru, pokud má nenulovou základní velikost. Uživatelé totiž někdy dávají kolem datových položek mezery a někdy ne, přitom chtějí, aby se jim to chovalo stejně. Je náročné si pamatovat, že mezery před položkou jsou ignorovány primitivem \halign, ale mezery za položkou jsou podstatné. Tak raději i mezery za položkou uděláme nepodstatné.

opmac.tex
```
\def\unsskip{\ifdim\lastskip>0pt \unskip\fi}
```

Příklad: po deklaraci: {|cr|cl|} makro \scantabdata vytvoří:

```
tabdata: \vrule\tabiteml\hfil#\unsskip\hfil\tabitemr
        &\tabiteml\hfil#\unsskip\tabitemr \vrule\kern\vvkern\vrule
        &\tabiteml\hfil#\unsskip\hfil\tabitemr
        &\tabiteml#\unsskip\hfil\tabitemr\vrule
ddlinedata: &\dditem &\dditem\vvitem &\dditem &\dditem
```

Makra \addtabitem, \addtabdata a \addtabvrule vloží do \tabdata a \ddlinedata požadovaný údaj. Makro \addtabitem pozná podle \colnum=0, zda vkládá data pro první sloupec (nepřidává &) nebo pro další sloupce (přidává &). Makro \addtabvrule pozná podle \tmpa, zda před ním předchází další \vrule. Pokud ano, vloží dodatečnou mezeru \kern\vvkern a přidá \vvitem do \ddlinedata.

opmac.tex
```
\def\addtabitem#1{\ifnum\colnum>0 \addtabdata{&}\addto\ddlinedata{&\dditem}\fi
    \advance\colnum by1 \let\tmpa=\relax \expandafter\addtabdata\expandafter{#1}}
\def\addtabdata#1{\expandafter\tabdata\expandafter{\the\tabdata#1}}
\def\addtabvrule{\ifx\tmpa\vrule \addtabdata{\kern\vvkern}%
    \ifnum\colnum=0\def\vvleft{\vvitem}\else\addto\ddlinedata{\vvitem}\fi\fi
    \let\tmpa=\vrule \addtabdata{\vrule}}
```

\vvleft: 34–35 \table: 5, 33–34 \scantabdata: 34 \tabdeclarec: 34 \tabdeclarel: 34
\tabdeclarer: 34 \unsskip: 34 \addtabitem: 34 \addtabdata: 34 \addtabvrule: 34

Než se pustíme do výkladu dalších maker, předvedeme příklad, ve kterém je definováno další písmeno `P` pro ⟨*deklaraci*⟩. Písmeno `P` vymezí tabulkovou položku, jež má stanovenou šířku a delší text se láme do více řádků. Je možné si vyzkoušet třeba tento kód:

```
\newdimen\Pwidth
\def\tabdeclareP {\enskip\vtop{\hsize=\Pwidth \rightskip=0pt plus1fil
    \baselineskip=1.2em \lineskiplimit=0pt \noindent ##\tabstrut}\enskip}

\Pwidth=3cm \table{|c|P|}{\crl
   aaa & Tady je delší textík, který se nevejde na jeden řádek. \crl
   bb  & A tady je taky je něco delšího. \crl}
```

Pusťme se nyní do rozboru maker na ukončení řádků. Makro `\crl` přidá čáru pomocí `\noalign`. Makro `\crll` přidá dvojitou čáru pomocí `\noalign`.

opmac.tex

```
1093: \def\crl{\cr\noalign{\hrule}}
1094: \def\crll{\cr\noalign{\hrule\kern\hhkern\hrule}}
```

Makro `\crli` provede `\cr` a dále se vnoří do řádku tabulky, ve kterém klade postupně následující `\omit\tablinefil␣&\omit\tablinefil␣&...` Přitom v místě dvojité vertikální čáry naklade navíc `\tabvvline`. Makro `\tablinefil` vloží natahovací čáru na šířku celé položky a makro `\tabvvline` vloží dvě `\vrule` vzdáleny od sebe o `\vvkern`. Tím vzniká přetrzené místo v postupně tvořené lince. Ke správnému nakladení uvedených povelů použije makro `\crli` obsah makra `\ddlinedata` a vlevo přidává `\vvleft`. Před spuštěním makra `\ddlinedata` definuje odpovídajícím způsobem `\dditem` a `\vvitem`.

opmac.tex

```
1096: \def\crli{\cr \omit \gdef\dditem{\omit\tablinefil}\gdef\vvitem{\tabvvline}%
1097:    \vvleft\tablinefil\ddlinedata\cr}
1098: \def\crlli{\crli\noalign{\kern\hhkern}\crli}
1099: \def\tablinefil{\leaders\hrule\hfil}
1100: \def\tabvvline{\vrule\kern\vvkern\vrule}
```

Makro `\tskip` prostřednictvím `\tskipA` přechodně vyprázdní `\tabstrut` předefinováním `\tabstrutA` a také vyprázdní `\dditem` a `\vvitem`, aby po použití `\ddlinedata` vznikl řádek tabulky s prázdnými položkami. Řádek je vypodložený strutem stanovené výšky `\tmpdim`. Nakonec je potřeba vrátit `\tabstrutA` do původního stavu.

opmac.tex

```
1102: \def\tskip{\afterassignment\tskipA \tmpdim}
1103: \def\tskipA{\gdef\dditem{}\gdef\vvitem{}\gdef\tabstrutA{}%
1104:     \vrule height\tmpdim width0pt \ddlinedata\cr
1105:     \gdef\tabstrutA{\tabstrut}}
```

Globální změna šířek všech linek tvořených pomocí `\vrule` a `\hrule` je provedena makry `\rulewidth` a `\rulewidthA`. Myšlenka je dokumentována v TBN na str. 328.

opmac.tex

```
1107: \let\orihrule=\hrule  \let\orivrule=\vrule
1108: \def\rulewidth{\afterassignment\rulewidthA \tmpdim}
1109: \def\rulewidthA{\edef\hrule{\orihrule height\the\tmpdim}%
1110:                \edef\vrule{\orivrule width\the\tmpdim}}
```

Makro `\frame` {⟨*text*⟩} vloží vnější `\vbox{\hrule..\hrule}`. V něm se nachází další box `\hbox{\vrule\kern\vvkern..\kern\vvkern\vrule}` a v něm `\vbox{\kern\hhkern..\kern\hhkern}`. Nejvíce uvnitř je pak `\hbox{`⟨*text*⟩`}`. To by pro sazbu rámovaného textu stačilo, nicméně my ještě řešíme úpravu výsledného boxu tak, aby měl účaří ve stejném místě jako je účaří textu. Proto uložíme `\hbox{`⟨*text*⟩`}` do boxu0 a změříme mu hloubku. V proměnné `\tmpdim` spočítáme celkovou hloubku výsledného boxu. Ve výpočtu přičítáme výšku `\hrule`, která nemusí být 0.4pt. Proto si její výšku změříme v boxu1. Ve vnějším `\vbox`u po nakreslení spodní `\hrule` se pak vracíme pomocí `\kern-\tmpdim` na úroveň účaří a zde umístíme strut hloubky `\tmpdim`, aby sazba směrem dolů nepřečnívala, ale byla obsažena v hloubce výsledného boxu.

\crl: 35 \crll: 35 \crli: 34–35 \tablinefil: 35 \tabvvline: 35 \dditem: 33–35
\vvitem: 34–35 \tskip: 34–35 \tskipA: 35 \rulewidth: 35 \rulewidthA: 35 \orihrule: 35
\orivrule: 35 \frame: 36

opmac.tex
```
\def\frame#1{\setbox0=\hbox{#1}\setbox1=\vbox{\hrule}%
   \tmpdim=\dp0 \advance\tmpdim by\ht1 \advance\tmpdim by\hhkern
   \vbox{\hrule\hbox{\vrule\kern\vvkern
         \vbox{\kern\hhkern\box0\kern\hhkern}\kern\vvkern\vrule}%
         \hrule\kern-\tmpdim\hbox{\vrule depth\tmpdim width0pt}}}
```

3.20 Vložení obrázku

Nejprve deklarujeme \picw.

opmac.tex
```
\newdimen\picw   \picw=0pt
```

Makro \inspic je zkratka za použití primitivů \pdfximage, \pdfrefximage a \pdflastximage. Kdo si to má pořád pamatovat. Není-li aktivován PDF výstup, napíšeme jen varování a neprovedeme nic. Pokud budeme pracovat se stejnými obrázky v jednom dokumentu, bude vhodné možnosti primitivů využít naplno.

opmac.tex
```
\ifpdftex
  \def\inspic #1 {\ifdim\picw=0pt
       \opwarning{\noexpand\inspic -- the width \noexpand\picw is not set}%
     \else
       \hbox{\pdfximage width\picw {\picdir#1}\pdfrefximage\pdflastximage}%
     \fi
  }
\else
  \def\inspic #1 {\opwarning
     {The \noexpand\inspic is supported for PDF output only}}
\fi
```

3.21 PDF transformace

Makro \pdfscale {⟨*vodorovně*⟩}{⟨*svisle*⟩} pracuje jednoduše:

opmac.tex
```
\def\pdfscale#1#2{\pdfsetmatrix{#1 0 0 #2}}
```

Na druhé straně makro \pdfrotate ⟨*úhel*⟩ vytvoří \pdfsetmatrix{$\cos\varphi\ \ \sin\varphi\ \ -\sin\varphi\ \ \cos\varphi$}, což není jednoduché, protože funkce cos, sin nejsou v TEXu implementovány. Balíček `trig.sty` nabízí vyhodnocování těchto funkcí pomocí Taylorových polynomů, nicméně OPmac nechce být závislý na balíčcích a také chce ukázat alternativní způsob implementace. Makro \pdfrotate pracuje zhruba takto: je-li argument 0, neprovede nic, je-li argument 90, provede otočení o 90 stupňů. V ostatních případech zavolá makro \pdfrotateA, které rozloží argument na celou #1 a zlomkovou #2 část. V další části na řádcích 1149 až 1158 se zabývá jen celými stupni. Nejprve pomocí prvního a druhého \loop posune argument o celé násobky 360 stupňů tak, že poté je argument mezi 0 až 360 stupni, a přitom se hodnoty funkcí sin a cos nezměnily. Ve třetím \loop postupně snižuje argument o 90 stupňů a přitom dělá rotaci o 90 stupňů tak dlouho, až máme argument mezi nulou a devadesáti. Je-li dále argument větší než 44 stupňů, otočíme se o 45 a snížíme argument o 45. Je-li dále argument větší než 22, otočíme se o 22 a snížíme argument o 22. Nyní máme argument v množině $\{0, 1, 2, 3, \ldots, 22\}$. Pro každý prvek z této množiny argumentů máme předpřipraveny hodnoty funkcí cos a sin v makrech \smallcos a \smallsin. Použijeme je pro závěrečnou rotaci. Tím máme sazbu otočenou o celé stupně. Další část makra na řádcích 1161 až 1165 řeší jemné dotočení podle zlomkové části argumentu. V intervalu nula až jeden stupeň aproximujeme funkci cos konstantní jedničkou a funkci sin lineární funkcí $x \cdot \pi/180$. V daném rozmezí je to velmi dobrá aproximace.

opmac.tex
```
\def\pdfrotate#1{\tmpdim=#1pt
   \ifdim\tmpdim=0pt
   \else \ifdim\tmpdim=90pt \pdfsetmatrix{0 1 -1 0}%
         \else \edef\tmp{#1}\expandafter\pdfrotateA\tmp..\relax
   \fi   \fi
}
\def\pdfrotateA #1.#2.#3\relax{%
```

\picw: 36 \inspic: 36 \pdfscale: 27, 36 \pdfrotate: 27, 36–37 \pdfrotateA: 36
\smallcos: 37 \smallsin: 37

```
    \def\tmp##1.##2\relax {##1}%
    \tmpnum=\expandafter \tmp \the\tmpdim \relax % round
    \ifdim\tmpdim>0pt \def\tmpa{}\else\def\tmpa{-}\fi % save -
    \loop \ifnum\tmpnum<0 \advance\tmpnum by360 \repeat
    \loop \ifnum\tmpnum>360 \advance\tmpnum by-360 \repeat
    \loop \ifnum\tmpnum>90 \pdfrotate{90}\advance\tmpnum by-90 \repeat
    \ifnum\tmpnum=90 \pdfrotate{90}\else
       \ifnum\tmpnum>44 \pdfsetmatrix{.7071 .7071 -.7071 .7071}%
                        \advance\tmpnum by-45 \fi
       \ifnum\tmpnum>22 \pdfsetmatrix{.9272 .3746 -.3746 .9272}%
                        \advance\tmpnum by-22 \fi
       \ifnum\tmpnum>0
          \pdfsetmatrix{\smallcos \smallsin -\smallsin \smallcos}%
    \fi\fi
    \if$#2$\else % fraction part
       \tmpdim=.01745329pt % \pi/180
       \tmpdim=.#2\tmpdim  %
       \edef\tmp{\expandafter\ignorept\the\tmpdim\space}%
       \ifx\tmpa\empty \pdfsetmatrix{1 \tmp -\tmp 1}%
       \else           \pdfsetmatrix{1 -\tmp \tmp 1}%
    \fi\fi
}
\def\smallcos{.\ifcase\tmpnum \or9998\or9994\or9986\or9976\or9962\or9945\or
  9925\or9903\or9877\or9848\or9816\or9781\or9744\or9703\or9659\or9613\or
  9563\or9511\or9455\or9397\or9336\or9272\fi\space}
\def\smallsin{.\ifcase\tmpnum 0\or0175\or0359\or0523\or0698\or0872\or1045\or
  1219\or1391\or1564\or1736\or1908\or2079\or2250\or2419\or2588\or2756\or
  2924\or309\or3256\or342\or3584\or3746\fi\space}
```

Pro případ, že nepracujeme s PDF výstupem, definujeme klíčové primitivy pdfTEXu jako makra, která nedělají nic.

opmac.tex

```
\ifpdftex \else
  \def\pdfsetmatrix#1{} \def\pdfsave{} \def\pdfrestore{}
\fi
```

3.22 Poznámky pod čarou a na okraji stránek

Makro `\fnote` předpokládá, že správné číslo poznámky na dané stránce je připraveno v makru `\fn:`⟨*číslo*⟩, kde ⟨*číslo*⟩ je celkové číslo poznámky napříč celým dokumentem sledované globálním čítačem `\fnotenum`. Makro ohlásí svou existenci do REF souboru záznamem `\Xfnote` (bez parametru). Dále vytiskne značku pomocí `\fnmarkx` a ve skupině přejde na menší sazbu a zavolá plainTEXové makro `\vfootnote`, které vloží sazbu pomocí tzv. insertu (TBN, kapitola 6.7). PlainTEXové nastavení této třídy insertu není makrem OPmac nijak měněno.

opmac.tex

```
\newcount\fnotenum \fnotenum=0
\newcount\fnotenumlocal

\def\fnote#1{\global\advance \fnotenum by1
   \leavevmode\openref\wref\Xfnote{}%
   \isdefined{fn:\the\fnotenum}\iftrue
   \else\opwarning{unknown \noexpand\fnote mark. TeX me again}\fi
   \fnmarkx{\typoscale[800/800]\vfootnote\fnmarkx{#1}}%
}
```

Makro `\fnotemark` přičte lokálně k `\fnotenum` svůj parametr a vytiskne odpovídající značku. Celá práce makra probíhá ve skupině, takže po ukončení makra se `\fnotenum` vrátí do své původní hodnoty.

opmac.tex

```
\def\fnotemark#1{{\advance\fnotenum by#1\relax
   \isdefined{fn:\the\fnotenum}\iftrue\thefnote
   \else$^?$\opwarning{unknown \string\fnotemark. TeX me again}\fi}%
}
```

\fnote: 37 \fnotenum: 10, 37–38 \fnotemark: 37

Makro \fnotetext teprve zvedne čítač \fnotenum globálně a vytiskne poznámku pomocí plainTEXového \vfootnote.

opmac.tex
```
\def\fnotetext#1{\global\advance \fnotenum by1 \openref\wref\Xfnote{}%
   {\typoscale[800/800]\vfootnote\fnmarkx{#1}}
}
```

Makro \fnmarkx vytiskne otazník nebo \thefnote. Předpokládá se, že si uživatel předefinuje \thefnote k obrazu svému. Lokální číslo poznámky na stránce má připraveno v makru \locfnum.

opmac.tex
```
\def\fnmarkx{\isdefined{fn:\the\fnotenum}\iftrue\thefnote\else$^?$\fi}
\def\thefnote{$^{\locfnum}$)}
\def\locfnum{\csname fn:\the\fnotenum\endcsname}
```

Při čtení REF souboru se pro každou stranu přečte nejprve \Xpage, což je makro, které pronuluje \fnotenumlocal. Makru \Xfnote tedy stačí pozvednout \fnotenumlocal o jedničku a pomocí \sxdef si tuto hodnotu zapamatovat v makru \fn:⟨*číslo*⟩.

opmac.tex
```
\def\Xfnote{\advance\fnotenumlocal by1 \advance\fnotenum by1
   \sxdef{fn:\the\fnotenum}{\the\fnotenumlocal}}
```

Registr \mnotenum. globálně čísluje okrajové poznámky a plní podobnou funkci, jako registr \fnotenum pro podčárové poznámky.

opmac.tex
```
\newcount\mnotenum    \mnotenum=0       % global counter of mnotes
```

Makro \mnote ve vertikálním módu založí box nulové výšky pomocí \mnoteA a vycouvá na původní místo sazby pomocí \vskip-\baselineskip. V odstavcovém módu toto makro nalepí box nulové výšky pod právě vytvořený řádek v odstavci. Víme, že \vadjust nalepí svůj materiál bez mezery pod tento řádek. My ovšem potřebujeme vycouvat nahoru na účaří řádku. To nejde snadno provést, protože hloubka řádku je proměnlivá. Proto do je řádku vložen \strut a předpokládá se, že nyní má řádek hloubku \dp\strutbox a o tento rozměr makro vycouvá nahoru. Vloží požadovaný box výšky nula na úrovni účaří a pak se vrátí na původní místo.

opmac.tex
```
\def\mnote#1{\ifvmode \mnoteA{#1}\nobreak\vskip-\baselineskip
   \else \strut\vadjust{\kern-\dp\strutbox \mnoteA{#1}\kern\dp\strutbox}%
   \fi
}
```

Makro \mnoteA si zjistí, zda je v makru \mn:⟨*číslo*⟩ uložen primitivní příkaz \left nebo \right. Podle toho pozná, zda má umístit poznámku doleva nebo doprava. Rovněž dá o sobě vědět do REF souboru vložením sekvence \Xmnote (bez parametru). Sazba musí v obou případech vyprodukovat box nulové výšky i hloubky. Proto je \vtop, uvnitř kterého je text poznámky zpracován, vložen přechodně do boxu0 a je mu pronulována hloubka. Nulová výška je zařízena pomocí \vbox␣to0pt{\vss\box0}. Vlastní sazbu poznámky zahajujeme pomocí \noindent s tím, že je připraven pružný \leftskip nebo \rightskip podle toho, zda poznámku klademe vlevo nebo vpravo. Při kladení vlevo musíme použít fill, abychom přeprali natahovací mezeru z \parfillskip.

opmac.tex
```
\def\mnoteA#1{\global\advance \mnotenum by1
   \isdefined{mn:\the\mnotenum}\iftrue
   \else\opwarning{unknown \noexpand\mnote side. TeX me again}\fi
   \edef\tmp{\csname mn:\the\mnotenum\endcsname}%
   \openref\wref\Xmnote{}%
   \expandafter\ifx\tmp \left
      \hbox to0pt{\kern-\mnotesize \kern-\mnoteindent
         \vbox to0pt{\vss \setbox0=\vtop{%
             \hsize=\mnotesize \leftskip=0pt plus 1fill\noindent#1}%
               \dp0=0pt \box0}\hss}%
   \else
      \hbox to0pt{\kern\hsize \kern\mnoteindent
         \vbox to0pt{\vss \setbox0=\vtop{%
             \hsize=\mnotesize \rightskip=0pt plus 1fil\noindent#1}%
```

\fnotetext: 38 \fnmarkx: 37–38 \thefnote: 37–38 \locfnum: 38 \fnotenumlocal: 27, 37–38
\Xfnote: 37–38 \mnotenum: 10, 38–39 \mnote: 38 \mnoteA: 38

```
1225:                \dp0=0pt \box0}\hss}%
1226:    \fi
1227: }
```

Makro \Xmnote pracuje během čtení REF souboru a využívá toho, že makro \Xpage nastavuje číslo právě procesované strany do registru \lastpge. Takže stačí použít \sxdef následujícím způsobem:

opmac.tex

```
1228: \def\Xmnote{\advance\mnotenum by1
1229:    \sxdef{mn:\the\mnotenum}{\ifodd\lastpage \right \else \left \fi}}
```

Makro \fixmnotes ⟨*token*⟩ předefinuje v cyklu všechny definované kontrolní sekvence \mn:⟨*číslo*⟩ tak, že budou obsahovat ⟨*token*⟩. Tím toto makro zlikviduje práci makra \Xmnote, ale o to nám právě jde v případě, když chceme mít poznámky na jednoznačně dané straně.

opmac.tex

```
1231: \def\fixmnotes#1{\tmpnum=0
1232:    \loop \advance\tmpnum by1
1233:          \isdefined{mn:\the\tmpnum}\iftrue \sxdef{mn:\the\tmpnum}{#1}\repeat}
```

3.23 Bibliografické reference

Nejprve uvedeme deklarace deskriptoru \auxfile a čítačů \bibnum a \lastcitenum.

opmac.tex

```
1237: \newwrite\auxfile                    % AUX file for BibTeX
1238: \newcount\bibnum                     % the bibitem counter
1239: \newcount\lastcitenum  \lastcitenum=0  % for \shortcitations
```

Makro \cite [⟨*lejblík1*⟩,⟨*lejblík2*⟩,...] si prostřednictvím \citeA zavolá \rcite{⟨*lejblík*⟩} pro každou jednotlivou položku oddělenou čárkou ve svém parametru. Makro \citeA v sobě skrývá fintu: parametr nemá jediný, ale dva #1#2. Protože první z nich je neseparovaný, ignorují se případné mezery za čárkou a #1 obsahuje první písmeno ⟨*lejblíku*⟩. Uvnitř makra měníme \citesep, což je čárka a mezera, která se má mezi údaji vytisknout. Makro \nocite [⟨*lejblík1*⟩,⟨*lejblík2*⟩,...] je definováno stejně, jen připraví jiný význam makra \docite, krz které budeme tisknout v \rcite potřebný údaj. Veškerá činnost maker \cite a \nocite probíhá uvnitř skupiny.

opmac.tex

```
1241: \def\cite[#1]{{[\chardef\tmpb=0 \citeA #1,,,%
1242:    \ifnum\tmpb>0 \printdashcite{\the\tmpb}\fi]}}
1243: \def\nocite[#1]{{\def\docite##1{}\citeA #1,,,}}
1244: \def\citeA #1#2,{\if#1,\else \rcite{#1#2}\expandafter\citeA\fi}
1245: \def\citesep{}
```

Makro \rcite {⟨*lejblík*⟩} řeší zhruba řečeno následující věci:

- Zjistí, zda je definován \csname␣bib:⟨*lejblík*⟩\endcsname. Pokud ano, vytiskne jeho hodnotu, pokud ne, vytiskne do textu otazníky a na terminál varování. Tato kontrolní sekvence začne být známá po použití \bib[⟨*lejblík*⟩] nebo \bibitem{⟨*lejblík*⟩}. Tato makra uloží odpovídající informaci do REF souboru, odkud ji při opakovaném TEXování vyzvedneme. Je to klasická činnost, kterou provozujeme i u ostatních křížových referencí.
- Uloží o sobě zprávu do bufferu \citelist. To použijeme v makrech \usebibtex nebo \usebbl.

Makro \rcite je naprogramováno zhruba takto

```
function rcite(⟨lejblík⟩) {
  if (⟨lejblík⟩ == '*') { ⟨zapiš do⟩ \citelist '*'; return; }
  if (\bib:⟨lejblík⟩ == nedef) {
    ⟨zapiš do⟩ \citelist ⟨lejblík⟩;
    ⟨na terminál:⟩ "Warning, cite [label] unknown";
    ⟨do tiskového výstupu:⟩ "??";
    \bib:⟨lejblík⟩ = empty;
    return;
  if (\bib:⟨lejblík⟩ == empty) {
    ⟨do tiskového výstupu:⟩ "??";
```

\Xmnote: 38–39 \fixmnotes: 39 \auxfile: 39, 41–43 \bibnum: 39, 41–43
\lastcitenum: 39–41 \cite: 39–43 \citeA: 39 \citesep: 39–41 \nocite: 39, 43
\rcite: 39–40

```
    return;
  }
  if (\bib:⟨lejblík⟩ končí znakem '&') {
    ⟨zapiš do⟩ \citelist ⟨lejblík⟩;
    ⟨odstraň znak & z obsahu makra⟩ \bib:⟨lejblík⟩;
  }
  ⟨tiskni obsah makra⟩ \bib:⟨lejblík⟩;
}
```

Výklad kódu: Protože chceme šetřit pamětí bufferu `\citelist`, zapisujeme tam každý ⟨*lejblík*⟩ jen jednou. Zda se nedeklarovaný ⟨*lejblík*⟩ vyskytl poprvé poznáme podle nedefinované hodnoty `\bib:`⟨*lejblík*⟩. Zda se vyskytl později znovu poznáme podle toho, že má hodnotu `empty`. Zda se deklarovaný ⟨*lejblík*⟩ vyskytl poprvé poznáme podle znaku `&` v jeho obsahu.

Návrh kódu v C-like notaci nyní převedeme do maker v TEXu:

opmac.tex

```
\def\rcite#1{%
   \if *#1\addcitelist{*}\expandafter \skiptorelax \fi
   \expandafter \ifx \csname bib:#1\endcsname \relax
      \addcitelist{#1}%
      \opwarning{The cite [#1] unknown. Try to TeX me again}%
      \docite{}\openref
      \expandafter\gdef\csname bib:#1\endcsname {}%
      \expandafter \skiptorelax \fi
   \expandafter \ifx \csname bib:#1\endcsname \empty
      \docite{}%
      \expandafter \skiptorelax \fi
   \def\bibnn##1{}%
   \if &\csname bib:#1\endcsname
      \addcitelist{#1}%
      \def\bibnn##1##2{##1}%
      \sxdef{bib:#1}{\csname bib:#1\endcsname}%
   \fi
   \docite{\csname bib:#1\endcsname}%
   \relax
}
```

Asi nejzajímavější vychytávka v tomto makru se týká testu na znak `&`. Implicitně při čtení REF souboru se do makra `bib:`⟨*lejblík*⟩ uloží `\bibnn{`⟨*hodnota*⟩`}&`. Příkaz `\if` za sebou totálně expanduje vše následující, takže nejprve narazí na `&`, pak se obsah `bib:`⟨*lejblík*⟩ expanduje prostřednictvím `\bibnn {`⟨*hodnota*⟩`}` na nic a za tímto „nic“ se zjeví druhý znak `&`, který se tedy přilepí na ten první. Ano, je pravda, že tyto dva znaky jsou stejné. Odstranění tohoto znaku probíhá znovu totální expanzí, tentokrát `\bibnn` první parametr ⟨*hodnota*⟩ zopakuje a druhý parametr se znakem `&` zahodí.

Než se pustíme do výkladu makra `\docite`, připravíme si makra `\printcite` ⟨*položka*⟩ a `\printdashcite` ⟨*položka*⟩. První z nich tiskne jednu položku oddělenou od případné další čárkou, druhé tiskne položku, před kterou předchází pomlčka vyznačující interval položek. Pointa makra `\printcite` je v tom, že si samo po prvním zavolání připraví separátor `\citesep` (který je na začátku činnosti `\cite` prázdný), takže při opakovaném volání `\printcite` se vytiskne i požadovaný separátor. Pomlčka v `\printdashcite` je schována do `\hbox`, aby nedocházelo těsně za ní ke zlomu řádku.

opmac.tex

```
\def\printcite#1{\citesep\citelink{#1}\def\citesep{,\hskip.2em\relax}}
\def\printdashcite#1{\hbox{--}\citelink{#1}}
```

Makro `\docite` ⟨*položka*⟩ vytiskne otazníky při prázdném parametru a jinak vytiskne prostřednictvím `\printcite` jednu ⟨*položku*⟩. Kromě toho řeší při nenulovém `\lastcitenum` slučování po sobě následujících čísel položek do intervalů. Naposledy vytištěnou položku uchovává v registru `\lastcitenum`. Při příštím zavolání zvětší `\lastcitenum` o jedničku a srovná ji s ⟨*položkou*⟩. Jsou-li si rovny, jde o následující položku v řadě a takovou položku netiskneme, nicméně si její hodnotu uchováme v `\tmpb`. Pokud je mezi souvislou řadou položek díra, tj. `\lastcitenum` se nerovná ⟨*položce*⟩, pak dovytiskneme předchozí interval pomocí `\printdashcite{\the\tmpb}` a následně vytiskneme i ⟨*položku*⟩. Makro `\shortcitations`

\bibnn: 40–41 \printcite: 40–41 \printdashcite: 39–41 \docite: 39–41
\shortcitations: 39, 41

jednoduše nastavuje \lastcitenum na nenulovou hodnotu a tím probudí k životu hlavní část makra \docite.

opmac.tex
```
\def\docite#1{\if$#1$??%
   \else
      \ifnum\lastcitenum=0   % only comma separated list
         \printcite{#1}%
      \else
         \ifx\citesep\empty  % first cite item
            \lastcitenum=#1\relax
            \printcite{#1}%
         \else               % next cite item
            \advance\lastcitenum by1
            \ifnum\lastcitenum=#1\relax % cosecutive cite item
               \mathchardef\tmpb=\lastcitenum
            \else  % there is a gap between cite items
               \lastcitenum=#1\relax
               \ifnum\tmpb=0 % previous items were printed
                  \printcite{#1}%
               \else
                  \printdashcite{\the\tmpb}\printcite{#1}\chardef\tmpb=0
   \fi\fi\fi\fi\fi
}
\def\shortcitations{\lastcitenum=1 }
```

Následuje kód makra \bib [⟨lejblík⟩], které prostřednictvím \wbib {⟨lejblík⟩}{⟨hodnota⟩} vloží do REF souboru propojené údaje o tom, jaké má ⟨lejblík⟩ přiřazeno číslo v seznamu literatury. Makro \wbib připojí před \wref příkaz \immediate právě tehdy, když \wref je ve stavu, kdy skutečně zapisuje do souboru REF. Makro \Xbib pracuje při čtení souboru REF a dělá to, co jsme si řekli už dříve: nastaví hodnotu makra \bib:⟨lejblík⟩ na \bibnn{⟨hodnota⟩}&.

opmac.tex
```
\def\bib[#1]{\par \ifnum\bibnum>0 \bibskip \fi
   \advance\bibnum by1
   \wbib{#1}{\the\bibnum}%
   \hangindent=\iindent \hangafter=1
   \noindent \dest[cite:\the\bibnum]{1.2em}%
   \hskip\iindent \llap{[\the\bibnum] }\ignorespaces
}
\def\wbib#1#2{\edef\tmp{\wref\Xbib{{#1}{#2}}}%
  \ifx\tmp\empty\else \immediate\tmp \fi
}
\def\Xbib#1#2{\sdef{bib:#1}{\bibnn{#2}&}}
```

Makro \addcitelist {⟨lejblík⟩} přidá do \citelist údaj ve tvaru \lcite[⟨lejblík⟩]. Hranaté závorky jsou použity proto, aby fungoval test \isinlist\citelist{[⟨lejblík⟩]}. Jak uvidíme za chvíli, makro \addcitelist změní během činnosti makra \usebibtex svůj význam na \writeaux, aby případné použití \cite až za \usebitex rovnou zapisovalo do AUX souboru. Podobně makro \addcitelist změní v makru \usebbl svůj význam \writeXcite {⟨lejblík⟩}, aby v příštím průchodu TEXem mělo makro \usebbl přehled i o výskytech \cite, které jsou napsány později, než \usebbl.

opmac.tex
```
\def\addcitelist#1{\global\addto\citelist{\lcite[#1]}}
\def\writeaux#1{\immediate\write\auxfile{\string\citation{#1}}}
\def\writeXcite#1{\openref\immediate\wref\Xcite{{#1}}}
\def\citelist{} \def\citelistB{}
```

Než se pustíme do výkladu maker \usebibtex, \genbbl a \usebbl, uvedeme stručně popis činnosti BibTEXu. Příkaz bibtex␣⟨dokument⟩ způsobí, že program bibtex se podívá do souboru ⟨dokument⟩.aux a tam si všímá sekvencí \bibdata {⟨bib-báze⟩}, \bibstyle {⟨bib-style⟩} a \citation {⟨lejblík⟩}. Na základě toho následně přečte soubor ⟨bib-báze⟩.bib se zdrojovými zápisy bibliografických údajů. Pro konverzi těchto zdrojových zápisů do výstupního souboru ⟨dokument⟩.bbl použije stylový soubor ⟨bib-style⟩.bst. Není-li mezi sekvencemi \citation uvedeno \citation{*}, program bibtex zahrne do výstupu jen ty bibliografické údaje, které mají ⟨lejblík⟩ shodný s některým z ⟨lejblíků⟩ uvedených

\bib: 39–41 \wbib: 41–43 \Xbib: 41 \addcitelist: 40–44 \citelist: 39–44
\writeaux: 41–42 \writeXcite: 41, 43–44 \bibdata: 42 \bibstyle: 42 \citation: 41–43

v parametrech sekvencí \citation. Každá sekvence \citation{⟨lejblík⟩} v souboru ⟨dokument⟩.aux typicky odpovídá jednomu použití příkazu \cite[⟨lejblík⟩].

Makro \usebibtex {⟨bib-báze⟩}{⟨bst-styl⟩} otevře soubor AUX prostřednictvím \openauxfile {⟨bib-báze⟩}{⟨bst-styl⟩}. Napíše tam tedy požadovaná data pro BibTEX. Dále z \citelist přepíše do AUX souboru lejbílky ve formátu \citation{⟨lejblík⟩}. Nakonec se uvnitř skupiny pustí do čtení souboru BBL prostřednictvím makra \readbblfile.

opmac.tex

```
\def\usebibtex#1#2{%
   \openref \openauxfile{#1}{#2}%
   \def\lcite[##1]{\writeaux{##1}}\citelist
   \global\let\addcitelist=\writeaux
   \bgroup \readbblfile{\jobname}\egroup
}
\def\openauxfile#1#2{%
   \immediate\openout\auxfile=\jobname.aux
   \immediate\write\auxfile
      {\percent\percent\space Opmac: AUX file reserved for bibtex only}%
   \immediate\write\auxfile{\string\bibdata{#1}}%
   \immediate\write\auxfile{\string\bibstyle{#2}}%
}
```

Makro \readbblfile {⟨soubor⟩} vyzkouší, zda je ⟨soubor⟩.bbl připraven ke čtení. Pokud ne, podá o tom odpovídající zprávu na terminál. Jinak nastaví čítač \bibnum na nulu a (vědomo si toho, že je spuštěno ve skupině) pustí se do lokálních re-definic LaTEXových konstrukcí, které se typicky v BBL souborech používají. Nastaví \leftskip na \iindent a spustí \bibtexhook. Konečně načte soubor BBL.

opmac.tex

```
\def\readbblfile #1{%
  \openin\testin=#1.bbl
  \ifeof\testin
    \opwarning{.bbl file doesn't exist.
               Use the ``bibtex #1'' command}%
  \else
    \closein\testin
    \bibnum=0
    \def\begin##1##2{}\def\end##1{}% LaTeX environment
    \def\newcommand##1 {}%
    \def\httpAddr##1{\url{http:##1}}\def\\{\hfill\break}%
    \def\newblock{\hskip .11em plus.33em minus.07em}%
    \def\mbox{\leavevmode\hbox}\let\em=\it
    \leftskip=\iindent
    \bibtexhook\relax
    \input #1.bbl
    \par
  \fi
}
```

V BBL souboru se vyskytují povely \bibitem. Za každým z nich se možná objeví parametr v hranaté závorce [⟨značka⟩] a následně je uveden {⟨lejblík⟩}. Pak na dalších řádcích jsou bibliografická data jednoho záznamu ukončená prázdným řádkem. Objeví-li se [⟨značka⟩], dává tím BibTEX najevo, že se má tato ⟨značka⟩ použít místo běžného číslování záznamů. Následuje kód, který takové údaje přečte, vytiskne a vloží do REF souboru o tom zprávu prostřednictvím \wref{⟨lejblík⟩}{⟨hodnota⟩}. Rozlišují se dva režimy tisku: není-li přítomna [⟨značka⟩] (makro \tmpa je prázdné), pak pomocí \llap vytiskneme [⟨číslo⟩]. Jinak se posuneme o -\iindent a tiskneme [⟨značku⟩] následovanou mezerou. Pak se vytisknou další údaje bibliografického záznamu.

opmac.tex

```
\def\bibitem{\isnextchar[{\bibitemB}{\def\tmpa{}\bibitemC}}
\def\bibitemB[#1]{\def\tmpa{#1}\bibitemC}
\def\bibitemC#1{\bibitemD{#1}}
\def\bibitemD#1{\ifnum\bibnum>0 \bibskip \fi
   \advance\bibnum by1
   \noindent \dest[cite:\if$\tmpa$\the\bibnum\else\tmpa\fi]{1.2em}%
   \if$\tmpa$\llap{[\the\bibnum] }\wbib{#1}{\the\bibnum}\else
```

\usebibtex: 6, 39, 41–42 \openauxfile: 42–43 \readbblfile: 42–43 \bibitem: 39, 42–43
\bibitemB: 42–43 \bibitemC: 42–43 \bibitemD: 42–43

```
                \hskip-\iindent [\tmpa]\wbib{#1}{\tmpa}\enskip
   \fi
   \ignorespaces
}
```

Makro \genbbl {⟨bib-báze⟩}{⟨bst-style⟩} otevře AUX soubor a zapíše do něj údaje potřebné pro BibTEX včetně \citation{*}. Poté se makro pokusí přečíst výstup z BibTEXu pomocí \readbblfile. V tomto případě pracuje \bibitem ve zvláštním režimu, kdy netiskne ⟨hodnoty⟩, ale ⟨lejblíky⟩. Z toho důvodu je předefinováno makro \bibitemC.

opmac.tex

```
\def\genbbl#1#2{\openauxfile{#1}{#2}%
   \immediate\write\auxfile{\string\citation{*}}%
   \bgroup
     \iindent=4em
     \def\bibitemC##1{\ifnum\bibnum>0 \bibskip \fi
        \advance\bibnum by1
        \noindent \llap{[##1]\enspace}\ignorespaces
     }%
     \readbblfile{\jobname}%
   \egroup
}
```

Makro \usebbl /⟨typ⟩␣⟨bbl-file⟩ spustí jiné makro s názvem \bbl:⟨typ⟩. Tři taková makra jsou definována pomocí \sdef. První \bbl:a je jednoduché: prostě projde BBL soubor a vytiskne údaje z něj. Druhé makro \bbl:b projde BBL soubor v režimu, při kterém jsou bibliografická data každého záznamu (až po prázdný řádek alias \par) přečtena do parametru #2 makra \bibitemC. Celý údaj je pak vytištěn jen za předpokladu, že [⟨lejblík⟩] je přítomen v seznamu \citelist. Třetí makro \bbl:c pracuje jako druhé až na to, že údaj netiskne, ale zapamatuje si ho do makra \bb:⟨lejblík⟩. Po takovém projití BBL souboru ještě projde \citelist, kde se \lcite[⟨lejblík⟩] promění v \bb:⟨lejblík⟩, takže se záznam vytiskne. Nyní ale v pořadí, v jakém jsou ⟨lejblíky⟩ zařazeny do \citelist.

opmac.tex

```
\def\usebbl/#1 #2 {\isdefined{bbl:#1}%
   \iftrue \csname bbl:#1\endcsname {#2}\else
      \opwarning{\string\usebbl/#1 #2 ... the ‘#1’ type undefined}%
   \fi
}
\sdef{bbl:a}#1{\bgroup \readbblfile{#1}\egroup}

\sdef{bbl:b}#1{\bgroup
     \let\lcite=\relax \xdef\citelist{\citelist\citelistB}%
     \def\bibitemC##1 ##2\par{%
        \isinlist\citelist{[##1]}\iftrue \bibitemD{##1}##2\par\fi}%
     \readbblfile{#1}%
     \global\let\addcitelist=\writeXcite
  \egroup
}
\sdef{bbl:c}#1{\bgroup
     \let\lcite=\relax \xdef\citelist{\citelist\citelistB}%
     \def\bibitemC##1 ##2\par{%
        \isinlist\citelist{[##1]}\iftrue
           \ifx\tmpa\empty \sdef{bb:##1}{\bibitemD{##1}##2\par}%
           \else \toks0={##2\par}%
                 \edef\tmpa{\noexpand\sdef{bb:##1}{% \tmpa have to expand
                      \noexpand\bibitemB[\tmpa]{##1}\the\toks0}}\tmpa
        \fi\fi}%
     \readbblfile{#1}%
     \def\bibitemC##1{\bibitemD{##1}}%
     \def\lcite[##1]{\csname bb:##1\endcsname}\citelist
     \global\let\addcitelist=\writeXcite
  \egroup
}
```

Za zmínku stojí ještě práce uvedených maker s \citelist. Před výskytem makra \usebbl se lejblíky z \cite a \nocite hromadí v \citelist. Ovšem další \cite a \nocite se mohou vyskytovat

\genbbl: 41, 43 \usebbl: 6, 39, 41, 43–44

za příkazem \usebbl. Pokud se tak stane, pracuje \addcitelist nyní ve významu \writeXcite a uloží potřebnou informaci do REF souboru. Při dalším TEXování se tato informace přečte makrem \Xcite {⟨lejblík⟩} z REF souboru takto:

opmac.tex
```
1393: \def\Xcite#1{\addto\citelistB{\lcite[#1]}}
```

To tedy znamená, že se uloží do seznamu \citelistB. Konečně makra \bbl:b a \bbl:c si dva seznamy \citelist a \citelistB před svou činností spojí do seznamu jediného nazvaného \citelist.

3.24 Úprava output rutiny

Místo původního makra \plainoutput používá OPmac makro \opmacoutput.

opmac.tex
```
1398: \output={\opmacoutput}
```

OPmac mění output rutinu proti originální \plainoutput jen v nejnutnějších věcech. Řeší tyto dva problémy:

- Místo přímého \shipout nechá nejprve box sestavit jako \box0, pak provede \protectlist a pak provede \shipout\box0.
- Mezi \makeheadline a \pagebody vkládá \prepage.

První úprava zabrání expanzi maker přidaných pomocí \addprotect v době práce příkazu \shipout, tedy v době, kdy expandují parametry \write. Těmto makrům je přidělen prostřednictvím \doprotect ⟨*makro*⟩ pro tento okamžik význam \relax. Je potřeba ještě vysvětlit, proč bylo nutné sestavit nejprve \box0 a teprve poté jej poslat ven pomocí \shipout. Je to z toho důvodu, že v době sestavování \box0 jsou expandována \headline a \footline a pro ten případ ještě chceme, aby všechna makra správně expandovala.

opmac.tex
```
1400: \def\opmacoutput{\def\nl{ }%
1401:   \setbox0=\vbox{\makeheadline\prepage\pagebody\makefootline}%
1402:   \pghook \protectlist
1403:   \shipout\box0 \advancepageno
1404:   \ifnum\outputpenalty>-20000 \else\dosupereject\fi
1405: }
1406: \def\doprotect#1{\let#1=\relax}
```

Druhá úprava vkládá makro \prepage, které zajistí tisk stránkové číslice do \reffile (to je docela užitečné) a kromě toho nastaví správnou barvu textu strany pomocí PDF speciálu.

opmac.tex
```
1408: \def\prepage{\wref\Xpage{{\the\pageno}}%
1409:    \dest[pg:\the\pageno]{25pt}%
1410:    \csname pg:\the\pageno\endcsname
1411: }
```

Když budou barvy přecházet na další stranu, jistě nechceme mít obarvenu stránkovou číslici. Je tedy potřeba do \footline vložit \Black. Když bude uživatel měnit velikost fontů v dokumentu, jistě nechce mít stránkovou číslici pokaždé jinak velkou. Proto je do \footline vloženo \thefontsize. Je nastaveno pevně na 10pt. Předpokládáme, že pokud bude někdo chtít jinak velkou stránkovou číslici, jednoduše si \footline nastaví podle svého. Jinak je \footline shodná s původním nastavením v plainTEXu.

opmac.tex
```
1413: \footline={\locpgcolor\Black \hss\tenrm\thefontsize[10]\the\pageno\hss}
```

3.25 Okraje

V registrech \pgwidth a \pgheight budeme mít po zavolání \setpagedimens šířku a výšku strany. V registru \shiftoffset budeme mít případný rozdíl okrajů mezi levou a pravou stránkou.

opmac.tex
```
1418: \newdimen\pgwidth  \newdimen\pgheight  \pgwidth=0pt
1419: \newdimen\shiftoffset
```

\Xcite: 41, 44 \opmacoutput: 44 \doprotect: 3–4, 44 \prepage: 26, 44 \footline: 44
\pgwidth: 44–46 \pgheight: 44–46 \shiftoffset: 44–45

Makro \margins /⟨typ⟩␣⟨formát⟩␣(⟨levý⟩,⟨pravý⟩,⟨horní⟩,⟨dolní⟩)⟨jednotka⟩␣ si nastaví registry \pgwidth a \pgheight prostřednictvím \setpagedimens a dále v souladu s uživatelskou dokumentací nastaví potřebné okraje. V makru \tmp je schována jednotka, kterou uživatel taky může zapomenout napsat. V takovém případě vypíšeme varování a doplníme jednotku mm. Jakmile měníme \hoffset nebo \voffset, nastavíme je nejprve na -1in (tím se dostaneme na okraj papíru) a pak budeme požadovanou velikost okraje k těmto registrům přidávat. Nemohu za to, že Knutha napadla taková ne příliš podařená myšlenka dát výchozí bod sazby kamsi doprostřed papíru umístěný pomocí ujetých jednotek. Za zmínku stojí ještě dvě myšlenky. Makro \rbmargin \h(v)offset\h(v)size{⟨okraj⟩} provede výpočet hodnoty \hoffset nebo \voffset v případě, že je dána protější hodnota okraje než je okraj přímo nastavitelný pomocí \h(v)offset. A konečně posun okraje při přechodu z pravé na levou stránku \shiftoffset počítáme jako \pgwidth - \hsize - 2*⟨levý⟩ což dá stejnou hodnotu jako ⟨pravý⟩-⟨levý⟩. Změna \hoffset o tuto hodnotu je provedena v makru \pghook, tedy v \output rutině, schována do skupiny, takže po ukončení \output rutiny se vrátí \hoffset na původní hodnotu.

opmac.tex

```
\def\margins/#1 #2 (#3,#4,#5,#6)#7 {\def\tmp{#7}%
   \ifx\tmp\empty
      \opwarning{\string\margins: missing unit, mm inserted}\def\tmp{mm}\fi
   \addto\tmp{\relax}%
   \setpagedimens #2 % setting \pgwidth, \pgheight
   \ifdim\pgwidth=0pt \else
      \hoffset=-1\trueunit in \voffset=-1\trueunit in
      \if$#3$\if$#4$\tmpdim=\pgwidth \advance\tmpdim -\hsize
                    \divide\tmpdim by2 \advance\hoffset \tmpdim % left=right
            \else  \rbmargin\hoffset\hsize{#4\tmp}% only right margin
            \fi
      \else  \if$#4$\advance\hoffset #3\tmp   % only left margin
            \else  \hsize=\pgwidth           % left+right margin
                   \advance\hsize -#3\tmp \advance\hsize -#4\tmp
                   \advance\hoffset #3\tmp
      \fi\fi
      \if$#5$\if$#6$\tmpdim=\pgheight \advance\tmpdim -\vsize
                    \divide\tmpdim by2 \advance\voffset \tmpdim % top=bottom
            \else  \rbmargin\voffset\vsize{#6\tmp}% only bottom margin
            \fi
      \else  \if$#6$\advance\voffset #5\tmp   % only top margin
            \else  \vsize=\pgheight          % top+bottom margin
                   \advance\vsize -#5\tmp \advance\vsize -#6\tmp
                   \advance\voffset #5\tmp
      \fi\fi
      \if 1#1\else \if 2#1% double-page layout
         \shiftoffset=\pgwidth \advance\shiftoffset -\hsize
         \advance\shiftoffset -2\hoffset \advance\shiftoffset -2in
         \addto\pghook{\ifodd\pageno \else \advance\hoffset \shiftoffset \fi}
      \else \opwarning{use \string\margins/1 or \string\margins/2}%
   \fi\fi\fi
}
\def\rbmargin#1#2#3{\advance#1\pgwidth \advance#1-#2 \advance#1-#3}
```

Makro \setpagedimens ⟨formát⟩␣ spustí \setpagedimensA ⟨šířka⟩␣⟨výška⟩/ a k tomu musí dopředu vyexpandovat obsah makra \pgs:⟨formát⟩. To provedeme pomocí tří \expandafter.

opmac.tex

```
\def\setpagedimens#1 {\isdefined{pgs:#1}\iftrue
   \expandafter\expandafter\expandafter \setpagedimensA \csname pgs:#1\endcsname&%
   \else \opwarning{page specification "#1" is undefined}\fi}
\def\setpagedimensA (#1,#2)#3&{\pgwidth=#1\trueunit#3 \pgheight=#2\trueunit#3\relax
    \ifx\pdfpagewidth\undefined \else
        \pdfpagewidth=\pgwidth \pdfpageheight=\pgheight \fi}
```

Jednotlivé ⟨formáty⟩ papíru je potřeba deklarovat.

opmac.tex

```
\sdef{pgs:a3}{(297,420)mm}  \sdef{pgs:a4}{(210,297)mm}  \sdef{pgs:a5}{(148,210)mm}
\sdef{pgs:a3l}{(420,297)mm} \sdef{pgs:a4l}{(297,210)mm} \sdef{pgs:a5l}{(210,148)mm}
\sdef{pgs:b5}{(176,250)mm}  \sdef{pgs:letter}{(8.5,11)in}
```

\margins: 45 \rbmargin: 45 \setpagedimens: 44–45 \setpagedimensA: 45–46

Makro \magscale [⟨factor⟩] zvětší/zmenší sazbu nastavením registru \mag a definuje dosud prázdné makro \trueunit hodnotou true, aby později při činnosti makra \setpagedimensA zůstaly zachovány rozměry stránek. Pokud ale je makro \magscale spuštěno až po nastavení velikosti stránek, jsou tyto velikosti dodatečně korigovány na „true“ jednotky pomocí makra \truedimen.

opmac.tex

```
\def\trueunit{}
\def\magscale[#1]{\mag=#1\def\trueunit{true}%
   \ifdim\pgwidth=0pt \else \truedimen\pgwidth \truedimen\pgheight \fi
   \ifx\pdfpagewidth\undefined \else
       \truedimen\pdfpagewidth \truedimen\pdfpageheight
       \pdfhorigin=1truein \pdfvorigin=1truein % Origin is independent off \mag
   \fi}
\def\truedimen#1{#1=\expandafter\ignorept\the#1truept }
```

3.26 Závěr

Nakonec pomocí \inputref přečteme REF soubor (pokud existuje) a vrhneme se na zpracování dokumentu, který nám připravil uživatel. Přeji dobré pořízení.

opmac.tex

```
\inputref
\endinput
```

4 Rejstřík

Tučně je označena strana, kde je slovo dokumentováno, pak následuje seznam všech stran, na kterých se slovo vyskytuje.