

# OPmac – rozšiřující makra plain $\TeX$ u

Petr Olšák

[www.olsak.net/opmac.html](http://www.olsak.net/opmac.html)

## Obsah

1	Úvod .....	3
2	Uživatelská dokumentace .....	3
3	Technická dokumentace .....	3
3.1	Základní makra .....	3
	<code>\OPmacversion</code> ... 3, <code>\tmpnum</code> ... 4, <code>\tmpdim</code> ... 4, <code>\opwarning</code> ... 4, <code>\addto</code> ... 4,	
	<code>\protectlist</code> ... 4, <code>\addprotect</code> ... 4, <code>\ifpdfTeX</code> ... 4, <code>\sdef</code> ... 4, <code>\sxdef</code> ... 4,	
	<code>\slet</code> ... 4, <code>\adef</code> ... 4, <code>\isdefined</code> ... 5, <code>\isinlist</code> ... 5, <code>\isnextchar</code> ... 5,	
	<code>\isnextcharA</code> ... 5, <code>\eoldef</code> ... 5, <code>\eoldefA</code> ... 5, <code>\maybebreak</code> ... 5, <code>\maybebreakA</code> ... 5,	
	<code>\uv</code> ... 6, <code>\percent</code> ... 6, <code>\bslash</code> ... 6, <code>\replacestrings</code> ... 6, <code>\replacestringsA</code> ... 6,	
	<code>\replacestringsB</code> ... 6	
3.2	Globální parametry .....	7
	<code>\iindent</code> ... 7, <code>\ttindent</code> ... 7, <code>\ttskip</code> ... 7, <code>\ttpenalty</code> ... 7, <code>\tthook</code> ... 7,	
	<code>\intthook</code> ... 7, <code>\ptthook</code> ... 7, <code>\iiskip</code> ... 7, <code>\itemhook</code> ... 7, <code>\bibskip</code> ... 7,	
	<code>\tabstrut</code> ... 7, <code>\tabiteml</code> ... 7, <code>\tabitemr</code> ... 7, <code>\vbkern</code> ... 7, <code>\hhkern</code> ... 7,	
	<code>\multiskip</code> ... 7, <code>\colsep</code> ... 8, <code>\mnoteindent</code> ... 8, <code>\mnotesize</code> ... 8, <code>\picdir</code> ... 8,	
	<code>\bibtexhook</code> ... 8, <code>\chaphook</code> ... 8, <code>\sechook</code> ... 8, <code>\secchhook</code> ... 8, <code>\cnvhook</code> ... 8,	
	<code>\prepghook</code> ... 8, <code>\pghook</code> ... 8, <code>\toclinehook</code> ... 8, <code>\fnotehook</code> ... 8, <code>\mnotehook</code> ... 8,	
	<code>\captionhook</code> ... 8	
3.3	Loga .....	8
	<code>\OPmac</code> ... 8, <code>\CS</code> ... 8, <code>\csplain</code> ... 8, <code>\LaTeX</code> ... 8, <code>\slantcorr</code> ... 8	
3.4	Velikosti fontů, řádkování .....	8
	<code>\resizefont</code> ... 8, <code>\sizespec</code> ... 8, <code>\resizeall</code> ... 8, <code>\regfont</code> ... 8, <code>\ptunit</code> ... 9,	
	<code>\fontdim</code> ... 9, <code>\regtfm</code> ... 9, <code>\whichtfm</code> ... 9, <code>\dgsiz</code> ... 9, <code>\ignorept</code> ... 9,	
	<code>\typosize</code> ... 9, <code>\typoscale</code> ... 9, <code>\fontsize</code> ... 9, <code>\textfontsize</code> ... 10,	
	<code>\setbaselineskip</code> ... 10, <code>\withoutunit</code> ... 10, <code>\fontscale</code> ... 10, <code>\textfontscale</code> ... 10,	
	<code>\scalebaselineskip</code> ... 10, <code>\thefontsize</code> ... 11, <code>\thefont</code> ... 11, <code>\thefontscale</code> ... 11,	
	<code>\magstep</code> ... 11, <code>\typobase</code> ... 11, <code>\baselineskipB</code> ... 11, <code>\fontdimB</code> ... 11, <code>\em</code> ... 11,	
	<code>\additcorr</code> ... 11, <code>\afteritcorr</code> ... 11, <code>\fontfam</code> ... 11	
3.5	Texty ve více jazycích .....	12
	<code>\mtext</code> ... 12, <code>\isolangset</code> ... 12	
3.6	REF soubor .....	12
	<code>\reffile</code> ... 12, <code>\testin</code> ... 12, <code>\wref</code> ... 12, <code>\wrefrelax</code> ... 12, <code>\inputref</code> ... 13,	
	<code>\openref</code> ... 13, <code>\openrefA</code> ... 13, <code>\Xrefversion</code> ... 13, <code>\REFversion</code> ... 13	
3.7	Lejblíky a odkazy .....	13
	<code>\label</code> ... 14, <code>\lastlabel</code> ... 14, <code>\wlabel</code> ... 14, <code>\ref</code> ... 14, <code>\pgref</code> ... 14,	
	<code>\Xlabel</code> ... 14	
3.8	Kapitoly, sekce, podsekce .....	14
	<code>\printchap</code> ... 15, <code>\printsec</code> ... 15, <code>\printsecc</code> ... 15, <code>\tit</code> ... 16, <code>\titfont</code> ... 16,	
	<code>\chapfont</code> ... 16, <code>\secfont</code> ... 16, <code>\seccfont</code> ... 16, <code>\bfshape</code> ... 16, <code>\chapnum</code> ... 16,	
	<code>\secnum</code> ... 16, <code>\seccnum</code> ... 16, <code>\nonumnum</code> ... 16, <code>\notoc</code> ... 16, <code>\nonum</code> ... 16,	
	<code>\chap</code> ... 16, <code>\sec</code> ... 16, <code>\secc</code> ... 16, <code>\thechapnum</code> ... 16, <code>\thesecnum</code> ... 16,	
	<code>\theseccnum</code> ... 16, <code>\thetocnum</code> ... 16, <code>\dotocnumafter</code> ... 16, <code>\wtotoc</code> ... 17,	
	<code>\wcontents</code> ... 17, <code>\dotocnum</code> ... 17, <code>\resetnonunotoc</code> ... 17, <code>\insertmark</code> ... 18,	
	<code>\remskip</code> ... 18, <code>\norempenalty</code> ... 18, <code>\remskipamount</code> ... 18, <code>\othe</code> ... 18,	
	<code>\afternoindent</code> ... 18, <code>\wipeepar</code> ... 18, <code>\firstnoindent</code> ... 18, <code>\nbp</code> ... 18, <code>\nl</code> ... 18	
3.9	Popisky, rovnice .....	19

<code>\tnum ... 19, \fnum ... 19, \dnum ... 19, \caption ... 19, \printcaption ... 19,</code> <code>\eqmark ... 19</code>	
<b>3.10</b> Odrážky .....	19
<code>\itemnum ... 19, \begitem ... 20, \enditem ... 20, \startitem ... 20, \printitem ... 20,</code> <code>\normalitem ... 20, \style ... 20, \fullrectangle ... 20, \athe ... 20</code>	
<b>3.11</b> Tvorba obsahu .....	20
<code>\toclist ... 20, \ifischap ... 20, \Xtoc ... 20, \Xchap ... 20, \Xsec ... 20,</code> <code>\Xsec ... 20, \tocline ... 21, \tocdotfill ... 21, \maketoc ... 21, \toclinkA ... 21</code>	
<b>3.12</b> Sestavení rejstříku .....	21
<code>\iindex ... 21, \ii ... 21, \iiA ... 21, \iiatsign ... 21, \iiB ... 21, \iiC ... 21,</code> <code>\iid ... 22, \iid ... 22, \Xindex ... 22, \iilist ... 22, \Xindexg ... 22,</code> <code>\firstdata ... 23, \seconddata ... 23, \firstdataA ... 23, \seconddataA ... 23,</code> <code>\XindexA ... 23, \XindexB ... 23, \iiendash ... 23, \pgfolioA ... 24, \pgfolioB ... 24,</code> <code>\makeindex ... 24, \printiipages ... 24, \prepii ... 24, \prepiiA ... 24, \iis ... 25,</code> <code>\iispeclist ... 25, \printii ... 25, \printiiA ... 25, \previi ... 25, \iiendash ... 25,</code> <code>\currii ... 25, \everyii ... 25, \scanprevii ... 25</code>	
<b>3.13</b> Abecední řazení rejstříku .....	25
<code>\sortingdata ... 26, \setignoredchars ... 26, \specsortingdatacs ... 26,</code> <code>\specsortingdatask ... 26, \setprimarysorting ... 27, \asciisorting ... 27,</code> <code>\specsortingdata ... 27, \setprimarysortingA ... 27, \sortingmessage ... 27,</code> <code>\setsecondarysorting ... 28, \preparesorting ... 28, \preparesortingA ... 28,</code> <code>\preparesortingB ... 28, \ifAleB ... 28, \isAleB ... 28, \testAleB ... 28,</code> <code>\testAleBsecondary ... 28, \testAleBsecondaryX ... 28, \dosorting ... 29, \mergesort ... 29,</code> <code>\gobbletoend ... 29, \sortreturn ... 29</code>	
<b>3.14</b> Více sloupců .....	30
<code>\begmulti ... 30, \endmulti ... 30, \corrsize ... 30, \makecolumns ... 30,</code> <code>\splitpart ... 30, \balancecolumns ... 31, \mullines ... 31</code>	
<b>3.15</b> Barvy .....	32
<code>\localcolor ... 32, \localcolortrue ... 32, \localcolorfalse ... 32, \longlocalcolor ... 32,</code> <code>\linecolor ... 32, \Blue ... 32, \Red ... 32, \Brown ... 32, \Green ... 32, \Yellow ... 32,</code> <code>\Cyan ... 32, \Magenta ... 32, \White ... 32, \Grey ... 32, \LightGrey ... 32,</code> <code>\Black ... 32, \setcmykcolor ... 32, \currentcolor ... 32, \pdfblackcolor ... 33,</code> <code>\ensureblacko ... 33, \ensureblackoA ... 33, \colorstackpush ... 33, \colorstackpop ... 33,</code> <code>\colorstackset ... 33, \draft ... 33, \draftbox ... 33</code>	
<b>3.16</b> Klikací odkazy .....	34
<code>\destactive ... 34, \destbox ... 34, \destheight ... 34, \dest ... 34, \linkactive ... 34,</code> <code>\link ... 34, \urllink ... 34, \toclink ... 35, \pglink ... 35, \citelink ... 35,</code> <code>\reflink ... 35, \ulink ... 35, \hyperlinks ... 35, \urlcolor ... 35, \tocilabel ... 35,</code> <code>\pgilabel ... 35, \pdfborder ... 35, \url ... 35, \urlfont ... 35, \urlskip ... 35,</code> <code>\urlbskip ... 35, \urlslashslash ... 35, \urlspecchar ... 36</code>	
<b>3.17</b> Outlines – obsah v záložce PDF dokumentu .....	36
<code>\outlines ... 36, \outlinesA ... 36, \addoneol ... 37, \outlinesB ... 37, \outlinesC ... 37,</code> <code>\outlinelevel ... 37, \setcnvcodesA ... 38, \toasciidata ... 38, \setlccodes ... 38,</code> <code>\insertoutline ... 38, \oulnum ... 38</code>	
<b>3.18</b> Verbatim .....	38
<code>\ttline ... 38, \viline ... 38, \vifile ... 38, \setverb ... 38, \begtt ... 38,</code> <code>\testparA ... 39, \testparB ... 39, \testparC ... 39, \printttline ... 39,</code> <code>\activettchar ... 39, \savedttchar ... 39, \savedttcharc ... 39, \verbinput ... 39,</code> <code>\vifilename ... 39, \skiptorelax ... 39, \vinolines ... 39, \vidolines ... 40,</code> <code>\viscanparameter ... 40, \viscanplus ... 40, \viscanminus ... 40, \doverbininput ... 40,</code> <code>\vireadline ... 41, \viprintline ... 41</code>	
<b>3.19</b> Jednoduchá tabulka .....	41
<code>\tabdata ... 41, \tabstrutA ... 41, \colnum ... 41, \ddlinedata ... 41, \vvleft ... 41,</code> <code>\table ... 41, \scantabdata ... 42, \scantabdataA ... 42, \scantabdataB ... 42,</code> <code>\scantabdataC ... 42, \scantabdataD ... 42, \tabdeclarec ... 42, \tabdeclarel ... 42,</code> <code>\tabdeclarer ... 42, \paramtabdeclarep ... 42, \unsskip ... 42, \addtabitem ... 42,</code>	

<code>\addtabdata ... 42</code> , <code>\addtabvrule ... 42</code> , <code>\crl ... 43</code> , <code>\crl1 ... 43</code> , <code>\crli ... 43</code> , <code>\tablinefil ... 43</code> , <code>\tabvline ... 43</code> , <code>\dditem ... 43</code> , <code>\vvitem ... 43</code> , <code>\crl1i ... 43</code> , <code>\tskip ... 43</code> , <code>\tskipA ... 43</code> , <code>\mspan ... 43</code> , <code>\mspanA ... 43</code> , <code>\mspanB ... 43</code> , <code>\rulewidth ... 44</code> , <code>\rulewidthA ... 44</code> , <code>\orihrule ... 44</code> , <code>\orivrule ... 44</code> , <code>\frame ... 44</code>	
<b>3.20</b> Vložení obrázku .....	44
<code>\picwidth ... 44</code> , <code>\picheight ... 44</code> , <code>\picw ... 44</code> , <code>\inspic ... 44</code> , <code>\inspicpage ... 44</code>	
<b>3.21</b> PDF transformace .....	44
<code>\pdfscale ... 44</code> , <code>\pdfrotate ... 45</code> , <code>\pdfrotateA ... 45</code> , <code>\smallcos ... 45</code> , <code>\smallsin ... 45</code>	
<b>3.22</b> Poznámky pod čarou a na okraji stránek .....	46
<code>\fnote ... 46</code> , <code>\fnotenum ... 46</code> , <code>\fnotemark ... 46</code> , <code>\fnotetext ... 46</code> , <code>\fnmarkx ... 46</code> , <code>\thefnote ... 46</code> , <code>\locfnum ... 46</code> , <code>\fnotenumlocal ... 46</code> , <code>\Xfnote ... 46</code> , <code>\runningfnotes ... 46</code> , <code>\mnotenum ... 46</code> , <code>\mnoteskip ... 46</code> , <code>\mnote ... 47</code> , <code>\mnoteA ... 47</code> , <code>\Xmnote ... 47</code> , <code>\fixmnotes ... 47</code> , <code>\mnotesfixed ... 47</code>	
<b>3.23</b> Bibliografické reference .....	47
<code>\auxfile ... 47</code> , <code>\bibmark ... 47</code> , <code>\bibnum ... 47</code> , <code>\lastcitenum ... 47</code> , <code>\cite ... 48</code> , <code>\nocite ... 48</code> , <code>\rcite ... 48</code> , <code>\savedcites ... 48</code> , <code>\citeA ... 48</code> , <code>\bibnn ... 49</code> , <code>\printsavedcites ... 49</code> , <code>\sortcitesA ... 49</code> , <code>\sortcitations ... 49</code> , <code>\sortcitesB ... 49</code> , <code>\sortcitesC ... 50</code> , <code>\sortcitesD ... 50</code> , <code>\citeB ... 50</code> , <code>\shortcitations ... 50</code> , <code>\printcite ... 50</code> , <code>\printdashcite ... 50</code> , <code>\citesep ... 50</code> , <code>\nonumcitations ... 51</code> , <code>\citelinkA ... 51</code> , <code>\etalchar ... 51</code> , <code>\ecite ... 51</code> , <code>\eciteB ... 51</code> , <code>\bib ... 51</code> , <code>\bibA ... 51</code> , <code>\bibB ... 51</code> , <code>\wbib ... 51</code> , <code>\Xbib ... 51</code> , <code>\lastbibnum ... 51</code> , <code>\printbib ... 51</code> , <code>\addcitelist ... 52</code> , <code>\citelist ... 52</code> , <code>\citeI ... 52</code> , <code>\writeaux ... 52</code> , <code>\writeXcite ... 52</code> , <code>\bibdata ... 52</code> , <code>\bibstyle ... 52</code> , <code>\citation ... 52</code> , <code>\usebibtex ... 52</code> , <code>\openauxfile ... 52</code> , <code>\readbbfile ... 52</code> , <code>\bibitem ... 53</code> , <code>\bibitemB ... 53</code> , <code>\bibitemC ... 53</code> , <code>\bibitemD ... 53</code> , <code>\genbbl ... 53</code> , <code>\usebbl ... 53</code> , <code>\Xcite ... 54</code> , <code>\usebib ... 54</code>	
<b>3.24</b> Úprava output rutiny .....	54
<code>\begoutput ... 54</code> , <code>\endoutput ... 54</code> , <code>\prephoffset ... 54</code> , <code>\opmacoutput ... 55</code> , <code>\doprotect ... 55</code> , <code>\prepage ... 55</code> , <code>\preboxcclv ... 55</code> , <code>\postboxcclv ... 55</code> , <code>\pagecontents ... 55</code> , <code>\Xpage ... 55</code> , <code>\lastpage ... 55</code>	
<b>3.25</b> Okraje .....	55
<code>\pgwidth ... 55</code> , <code>\pgheight ... 55</code> , <code>\shiftoffset ... 55</code> , <code>\margins ... 56</code> , <code>\rbmargin ... 56</code> , <code>\setpagedimens ... 56</code> , <code>\setpagedimensB ... 56</code> , <code>\setpagedimensA ... 56</code> , <code>\setpagedimensC ... 56</code> , <code>\magscale ... 57</code> , <code>\trueunit ... 57</code> , <code>\truedimen ... 57</code>	
<b>3.26</b> Závěr .....	57
<b>4</b> <b>Rejstřík</b> .....	57

## 1 Úvod

OPmac je balík jednoduchých doplňujících maker k plain $\TeX$ u umožňující uživatelům základní La $\TeX$ ovou 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  $\TeX$ u, tj. například aspoň zběžná orientace v  $\TeX$ booku 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.

---

```
\OPmacversion: 4
```

Je-li někdo překvapen, proč jsem nepoužil `\expandafter\endinput\fi`, může si prostudovat TBN, stranu 358, heslo `\endinput`.

```
7: \ifx\OPmacversion\undefined \else \endinput \fi
8: \def\OPmacversion{Aug. 2018}
9: \immediate\write16{This is OPmac (Olsak's Plain macros), version <\OPmacversion>}
```

Dva pracovní registry:

```
13: \newcount\tmpnum % auxiliary count
14: \newdimen\tmpdim % auxiliary dimen
```

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

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

Makro `\addto`  $\langle makro \rangle \{ \langle tokeny \rangle \}$  přidá na konec  $\langle makra \rangle$  dané  $\langle tokeny \rangle$ .

```
18: \long\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`  $\langle makro \rangle$ , které zařídí vložení  $\langle makra \rangle$  do seznamu.

```
20: \def\protectlist{}
21: \def\addprotect#1{\addto\protectlist{\doprotect#1}}
22: \addprotect~
```

OPmac užívá v makrech pro speciální vlastnosti PDF výstupu výhradně primitivy pdfTeXu. LuaTeX nám v roce 2016 přidělal starosti, protože předefinoval pdfTeXové primitivy. Proto při detekování nového LuaTeXu (to poznáme podle `\pdfextension`) nastavíme význam primitivu `\pdfoutput` do původního stavu a dále, na konci souboru maker (viz sekci 3.26), voláme speciální soubor `opmac-luatex.tex`, který nastaví další pdfTeXové primitivy podle původního významu.

```
24: \ifx\pdfextension\undefined \else
25: \let\pdfoutput=\outputmode \def\pdfcolorstackinit{\pdffeedback colorstackinit}\fi
```

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 směřován do DVI nebo PDF. Pozdější změna `\pdfoutput` může způsobit potíže. XeTeX sice není pdfTeX, ale po dobu čtení maker jej za pdfTeX budeme považovat a na konci čtení maker (viz sekci 3.26) to spravíme.

```
27: \newif\ifpdftex \pdftextrue
28: \ifx\pdfoutput\undefined \pdftexfalse \else \ifnum\pdfoutput=0 \pdftexfalse \fi \fi
29: \ifx\XeTeXversion\undefined \else \pdftextrue \fi
```

Makra `\sdef` a `\sxddef` umožňují pohodlně definovat kontrolní sekvence ohraničené pomocí `\csname... \endcsname`. Stejně tak `\slet` nastaví význam sekvencí ohraničených pomocí `\csname... \endcsname`.

```
31: \def\sdef#1{\expandafter\def\csname#1\endcsname}
32: \def\sxddef#1{\expandafter\xdef\csname#1\endcsname}
33: \def\slet#1#2{\expandafter\let\csname#1\expandafter\endcsname\csname#2\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`

---

```
\tmpnum: 18, 22, 24, 27–28, 30–32, 37, 40–42, 45 \tmpdim: 5, 8, 10–11, 34, 43, 45,
56 \opwarning: 4, 9, 14, 18–19, 21, 24, 27, 34–36, 38–39, 42, 44, 46–47, 49, 51–54, 56
\addto: 4, 6, 20, 22, 25, 29–30, 33, 41–43, 46, 49, 52, 54, 56 \protectlist: 4, 14,
37, 54–55 \addprotect: 4, 6, 8, 11, 33, 36–37, 55 \ifpdftex: 4, 33–35, 38, 44–45
\sdef: 4, 12–14, 20, 25, 51, 53–54, 57 \sxddef: 4, 12–14, 22, 37, 46–47, 49 \slet: 4
\adef: 5, 20, 39–41
```

jej přepíšeme na požadovaný znak. Dostaneme tím aktivní token s požadovanou ASCII hodnotou a tento token definujeme. `\lccode` nastavíme ve skupině, takže po ukončení skupiny se vrací k výchozí hodnotě.

```
35: \def\adef#1{\catcode'\#1=13 \begingroup \lccode'\#1\lowercase{\endgroup\def~}}
```

opmac.tex

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

```
37: \def\isdefined #1#2{\expandafter\ifx \csname#1\endcsname \relax
38:   \csname iffalse\expandafter\endcsname
39:   \else
40:   \csname iftrue\expandafter\endcsname
41:   \fi
42: }
```

opmac.tex

Makro `\isinlist`  $\langle list \rangle$   $\langle tokeny \rangle$  `\iftrue` zjistí, zda  $\langle tokeny \rangle$  jsou (jako string) obsaženy v makru  $\langle list \rangle$ . Přitom sežere `\iftrue` ze stejných důvodů, jak je uvedeno před chvílí.

```
43: \long\def\isinlist#1#2#3{\begingroup \long\def\tmp##1#2##2\end{\def\tmp{##2}%
44:   \ifx\tmp\empty \endgroup \csname iffalse\expandafter\endcsname \else
45:   \endgroup \csname iftrue\expandafter\endcsname \fi}% end of \def\tmp
46:   \expandafter\tmp#1\endlistsep#2\end
47: }
```

opmac.tex

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

```
48: \long\def\isnextchar#1#2#3{\begingroup\toks0={\endgroup#2}\toks1={\endgroup#3}%
49:   \let\tmp#1\futurelet\next\isnextcharA
50: }
51: \def\isnextcharA{\the\toks\ifx\tmp\next0\else1\fi\space}
```

opmac.tex

Makro `\eoldef`  $\langle foo \rangle$   $\langle makro \rangle$  pracuje jako `\def`, ale parametr `#1` je separován koncem řádku. Takže třeba

```
\eoldef\foo#1{param={#1}}
\foo tady je parametr
```

expanduje na `param={tady_ je_parametr}`. Implementace se opírá o to, že při `\eoldef\foo` se definují `\foo` a `\\foo:M`. Přitom `\foo` ve skupině pozmění `catcode` znaku pro konec řádku a spustí `\eoldefA` `\foo`. Toto makro načte parametr `#2` do konce řádku (po `^M`), dále ukončí skupinu a spustí `\\foo:M{parametr}`. Konečně `\\foo:M` vykoná to, co definoval uživatel.

```
53: \def\eoldef#1{\def#1{\begingroup \catcode'\^M=12 \eoldefA#1}%
54:   \expandafter\def\csname\string#1:M\endcsname}
55: {\catcode'\^M=12 \gdef\eoldefA#1#2^M{\endgroup\csname\string#1:M\endcsname{#2}}}
```

opmac.tex

Makro `\maybebreak`  $\langle rozměr \rangle$  umožní uživateli rozlomit řádek nebo stránku v místě použití. Pomocné marko `\maybebreakA` se spustí po načtení parametru. Zlom se uskuteční, chybí-li do konce řádku/stránky zhruba méně než  $\langle rozměr \rangle$  místa. Jinak se zlom neuskuteční a nestane se nic. Makro je závislé na módu `TeXu` (vertikální/horizontální). Chcete-li jím lámat stránky, pište třeba `\par\maybebreak3cm`. Makro využívá triku, že přičte a odečte stejnou hodnotu roztažitelnosti mezery, takže tyto dvě mezery těsně za sebou se (při nezlomení v `\penalty-130`) anulují.

```
57: \def\maybebreak{\afterassignment\maybebreakA\tmpdim=}
58: \def\maybebreakA{\ifvmode \vskipOpt plus\tmpdim \penalty-130 \vskipOpt plus-\tmpdim
59:   \else \hskipOpt plus\tmpdim \penalty-130 \hskipOpt plus-\tmpdim \fi \relax
60: }
```

opmac.tex

---

`\isdefined`: 5, 14, 19, 22, 27, 35, 37–38, 46–47, 49, 51, 53, 56    `\isinlist`: 5, 24, 42, 52–54  
`\isnextchar`: 5, 51, 53, 56    `\isnextcharA`: 5    `\eoldef`: 5, 16–17    `\eoldefA`: 5    `\maybebreak`: 5  
`\maybebreakA`: 5

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`.

```
61: \long\def\uv#1{\clqq#1\crqq}
```

opmac.tex

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`.

```
62: \let\=\undefined
```

opmac.tex

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 překlápí otazník, takže `\percent` expanduje na znak procento s kategorií 12.

```
63: {\lccode'\?='\% \lowercase{\gdef\percent{?}}
```

opmac.tex

Podobně je naprogramováno makro `\bslash`, které vytiskne obyčejné zpětné lomítko:

```
64: {\lccode'\?='\ \lowercase{\gdef\bslash{?}}
```

opmac.tex

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.

```
65: \def\,\{\relax\ifmmode \mskip\thinmuskip \else \thinspace \fi}
```

opmac.tex

Definovaná makra chceme při `\write` do souboru nechat v původním stavu:

```
66: \addprotect\percent \addprotect\bslash \addprotect\, \addprotect\exfont
```

opmac.tex

Makro `\exfont` se vyskytuje v souboru `exchars.tex` z CSplainu. Příkaz `\addprotect\exfont` zaprotektuje všechny znaky deklarované v tomto souboru naráz. Podrobnosti lze nalézt v uvedeném souboru.

Makro `\replacestrings`  $\langle string1 \rangle \{ \langle string2 \rangle \}$  vymění v makru `\tmpb` veškeré výskyty  $\langle string1 \rangle$  za  $\langle string2 \rangle$ . Pro tento účel definuje pracovní makra `\replacestringsA` a `\replacestringsB` se separátorem  $\langle string1 \rangle$ . Jak to pracuje je ukázáno na příkladu níže. Před spuštěním `\replacestringsA` je třeba nejprve vyvrhnout obsah `\tmpb` do vstupní fronty pomocí `\expandafter`. V makru pracujeme s tokeny `!` a `?` kategorie 3, které slouží jako separátory. Předpokládáme, že takové nestandardní tokeny se ve zpracovávaném textu nikdy neobjeví, protože vykřičník a otazník mají normálně kategorii 12.

```
68: \bgroup \catcode'\!=3 \catcode'\?=3
69: \gdef\replacestrings#1#2{\long\def\replacestringsA##1#1{\def\tmpb{##1}\replacestringsB}%
70: \long\def\replacestringsB##1#1{\ifx!##1\relax \else\addto\tmpb{#2##1}%
71: \expandafter\replacestringsB\fi}% improved version <May 2016> inspired
72: \expandafter\replacestringsA\tmpb?#1!#1% from pysyntax.tex by Petr Krajník
73: \long\def\replacestringsA##1?{\def\tmpb{##1}}\expandafter\replacestringsA\tmpb
74: }
75: \egroup
```

opmac.tex

Jak to pracuje si ukážeme na příkladu `\replacestrings{XX}{YY}`, pokud máme v `\tmpb` uložen třeba text `ahaXXuffXXkonec`. Makra `\replacestringsA` a `\replacestringsB` jsou v takovém případě definována jako:

```
\def\replacestringsA #1XX{\def\tmpb{#1}\replacestringsB}
\def\replacestringsB #1XX{\ifx!#1\relax\else
\addto\tmpb{YY#1}\expandafter\replacestringsB\fi}%
```

a jednotlivé kroky zpracování probíhají takto:

---

```
\uv: 6 \percent: 6, 13, 52 \bslash: 6, 36 \replacestrings: 6–7, 28, 36 \replacestringsA: 6
\replacestringsB: 6
```



```

\replacestringsA ahaXXuffXXkonec?XX!XX
#1 = "aha" zbytek fronty = "uffXXkonec?XX!"
\def\tmpb{aha}
\replacestringsB uffXXkonec?XX!XX
#1 = "uff" zbytek fronty = "konec?XX!"
\addto\tmpb{YYuff}, tj. \tmpb obsahuje "ahaYYuff".
\replacestringsB konec?XX!XX
#1 = "konec?" zbytek fronty = "!XX"
\addto\tmpb{YYkonec?}, tj. \tmpb obsahuje "ahaYYuffYYkonec?"
\replacestringsB !XX
#1 = ! zbytek fronty prázdný, rekurze končí

```

Dále se předefinuje `\def\replacestringsA#1?{\def\tmpb{#1}}` a provede se

```

\replacestringsA ahaYYuffYYkonec?
#1 = "ahaYYuffYYkonec"
\def\tmpb{ahaYYuffYYkonec}

```

tedy tímto algoritmem odstraníme koncový otazník. Proč jsme ho tam vlastně dávali? Kdyby tam nebyl, tak by nesprávně fungovalo `\replacestrings{XX}{YY}` při `\tmpb` ve tvaru `ahaX`.

Makro `\replacestrings` je kompromisem mezi jednoduchostí a přijatelnými možnostmi. Nefunguje nad textem s nespárovanými `\if... \fi` a také při `\def\tmpb{aha}XX\replacestrings{XX}{YY}` se bohužel odstraní kučeravé závorky kolem `aha`. Můžete třeba přidat před každou dvojici takových závorek `\empty`, abyste měli jistotu, že závorky nezmizí.

### 3.2 Globální parametry

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

opmac.tex

```

79: \widowpenalty=10000
80: \clubpenalty=10000
81: \showboxdepth=7
82: \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

```

84: \newdimen\iindent \iindent=\parindent
85: % indentation of items, TOC, captions, list of bib. references
86: \newdimen\ttindent \ttindent=\parindent
87: % indentation in \begtt...\endtt and \verbinput
88:
89: \def\ttskip{\medskip} % space above and below \begtt, \verbinput
90: \mathchardef\ttpenalty=100 % penalty between lines in \begtt, \verbinput
91: \def\tthook{} % hook in \begtt, \verbinput
92: \def\intthook{} % hook in in-text verbatim
93: \def\ptthook{} % hook in \begtt, \verbinput for post-processing
94:
95: \def\iiskip{\medskip} % space above and below \begitems...\enditems
96: \def\itemhook{} % hook in \startitem
97: \def\bibskip{\smallskip} % space between bibitems
98:
99: \def\tabstrut{\strut} % strut in the \table
100: \def\tabiteml{\enspace} % left material before each \table item
101: \def\tabitemr{\enspace} % right material after each \table item
102: \def\vvkern{1pt} % space between vertical lines
103: \def\hhkern{1pt} % space between horizontal lines
104:
105: \def\multiskip{\medskip} % space above and below \begmulti...\endmulti

```

---

```

\iindent: 7, 19–21, 24–25, 51–53 \ttindent: 7, 38–41 \ttskip: 39–41 \ttpenalty: 39, 41
\tthook: 39–41 \intthook: 39 \ptthook: 41 \iiskip: 20 \itemhook: 20 \bibskip: 51, 53
\tabstrut: 41, 43–44 \tabiteml: 42, 44 \tabitemr: 42, 44 \vvkern: 42–44 \hhkern: 43–44
\multiskip: 30

```

```

106: \newdimen\colsep \colsep=2em % space between columns
107:
108: \newdimen\mnoteindent \mnoteindent=10pt % dítance between mnote and text
109: \newdimen\mnotesize \mnotesize=20mm % the width of the mnote paragraph
110:
111: \def\picdir{} % the directory with picture files
112: \def\bibtexhook{} % hook in \usebibtex and \usebbl macros
113: \def\chaphook{} % hook in \chap
114: \def\sechhook{} % hook in \sec
115: \def\secchhook{} % hook in \secc
116: \def\cnvhook{} % hook before conversion of outlines
117: \def\prepghook{} % hook before page building in \output routine
118: \def\pghook{} % next hook in \output routine
119: \def\toclinehook{} % hook in \tocline
120: \def\fnotehook{} % hook in \fnote
121: \def\mnotehook{} % hook in \mnote
122: \def\captionhook#1{} % hook in \caption (#1 is "t" or "f")

```

### 3.3 Loga

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

```

126: \def\OPmac{\leavevmode
127:   \lower.2ex\hbox{\thefontscale[1400]0\kern-.86em P{\em mac}}}
128: \def\CS{\cal C$\kern-.1667em\lower.5ex\hbox{\cal S$}}
129: \def\csplain{\CS plain}

```

opmac.tex

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

```

131: \def\LaTeX{\tmpdim=.42ex L\kern-.36em \kern\slantcorr % slant correction
132:   \raise\tmpdim\hbox{\thefontscale[710]A}%
133:   \kern-.15em \kern-\slantcorr \TeX}
134: \def\slantcorr{\expandafter\ignorept\the\fontdimen1\the\font\tmpdim}

```

opmac.tex

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

```

136: \addprotect\TeX \addprotect\OPmac \addprotect\CS \addprotect\LaTeX

```

opmac.tex

### 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 relikv, který nám ovšem napoví, že kontrolní sekvence je fontovým přepínačem.

`OPmac` si zjistí, zda je definovaný `\regfont` (tj. je detekován dostatečně nový `csplain`). Pokud ne, upozorní na nedostupnost vícejazyčné podpory na terminálu a potřebná makra pro fonty si definuje. Je to kopie kódu ze souboru `csfontsm.tex` z balíčku `CSplain`.

---

```

\colsep: 8, 30 \mnoteindent: 8, 47 \mnotesize: 8, 47 \picdir: 44 \bibtexhook: 52
\chaphook: 17, 46 \sechhook: 17 \secchhook: 17 \cnvhook: 37 \prepghook: 33, 55
\pghook: 54–56 \toclinehook: 21 \fnotehook: 46 \mnotehook: 47 \captionhook: 19
\OPmac: 8 \CS: 8 \csplain: 8 \LaTeX: 8 \slantcorr: 8 \resizefont: 9–11
\sIZESPEC: 9–11 \resizeall: 9–10 \regfont: 8–9

```



```

141: \ifx\regfont\undefined
142: \ifx\uselanguage\undefined % if this is etex.src, the following warning is suppressed
143: \opwarning{No multilanguage support (csplain is recommended)}
144: \fi
145: % macros from csplain, file csfontsm.tex:
146: \ifx\tenbi\undefined \font\tenbi=cmbxti10 \def\bi{\tenbi}\fi
147: \def\letfont#1#2{\ifx#2=\expandafter\letfont\expandafter#1\else
148: \expandafter\font\expandafter#1\expandafter\fontskipat\fontname#2 \relax\space \fi}
149: \def\rfontskipat#1{\ifx#1"\expandafter\rfskipatX\else\expandafter\rfskipatN\expandafter#1\fi}
150: \def\rfskipatX #1" #2\relax{"\whichtfm{#1}} \def\rfskipatN #1 #2\relax{\whichtfm{#1}}
151: \def\sizespec{} \def\whichtfm#1{#1}
152: \def\resizefont#1{\letfont#1#1\sizespec}
153: \def\regfont#1{\expandafter\def\expandafter\resizeall\expandafter{\resizeall \resizefont#1}}
154: \def\resizeall{}
155: \regfont\tenrm \regfont\tenit \regfont\tenbf \regfont\tenbi \regfont\tentt
156: \fi

```

Makra `\typosize`, `\fontsize`, `\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.

```

158: \newdimen\ptunit \ptunit=1pt
159: \newdimen\fontdim \fontdim=10pt

```

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 textové fonty. To nezmění vlastnosti maker v OPmac, pokud nové soubory maker správně předefinují makra `\setmathsizes` [*text*]/[*script*]/[*scriptscript*], `\normalmath` a `\boldmath`. Soubor `ams-math.tex` načteme jen tehdy, když není definováno `\normalmath`. Je totiž možné, že uživatel načtl matematické makro ještě před zavoláním `\input_opmac`.

```

161: \ifx\normalmath\undefined \input ams-math \fi % ams-math.tex is in csplain package

```

Po načtení souboru `ams-math.tex` disponujeme makry `\regtfm` na registraci různých metrik pro různé designované velikosti fontů a `\whichtfm` je definováno tak, aby expandovalo na svůj parametr nebo na metriku, která je registrována pro velikost `\dgsizex`. Registrace metrik CSfontů je rovněž provedena v souboru `ams-math.tex`.

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.

```

163: {\lccode'\?='p \lccode'\!='t \lowercase{\gdef\ignorept#1?!{#1}}}

```

Makra `\typosize` a `\typoscale` změni 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.

```

165: \def\typosize[#1/#2]{\fontsize[#1]\setbaselineskip[#2]\ignorespaces}
166: \def\typoscale[#1/#2]{\fontscallex[#1]\scalebaselineskip[#2]\ignorespaces}

```

Makro `\fontsize` [*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 `\fontsize` 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.

---

```

\ptunit: 9–11 \fontdim: 9–11 \regtfm \whichtfm: 9 \dgsizex: 10–11 \ignorept: 8–11, 45,
57 \typosize: 9, 11, 33 \typoscale: 9, 11, 16, 46 \fontsize: 9–10

```

```

168: \def\fontsize#1{\if$#1$\else
169:   \textfontsize#1%
170:   \tmpdim=0.7\fontdim \edef\tmpa{\expandafter\ignorept\the\tmpdim}%
171:   \tmpdim=0.5\fontdim \edef\tmpb{\expandafter\ignorept\the\tmpdim}%
172:   \edef\tmp{\noexpand\setmathsizes[\expandafter\ignorept\the\fontdim/\tmpa/\tmpb]}%
173:   \tmp \normalmath
174:   \fi
175: }

```

opmac.tex

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

```

176: \def\textfontsize#1{\if$#1$\else
177:   \fontdim=#1\ptunit \ifx\fontdimB\undefined \edef\fontdimB{\the\fontdim}\fi
178:   \let\dgsiz=\fontdim
179:   \edef\sizspec{at\the\fontdim}%
180:   \resizeall \rm \let\dgsiz=\undefined
181:   \fi
182: }

```

opmac.tex

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.

```

183: \def\setbaselineskip#1{\if$#1$\else
184:   \tmpdim=#1\ptunit
185:   \baselineskip=\tmpdim \relax
186:   \ifx\baselineskipB\undefined \edef\baselineskipB{\the\baselineskip}\fi
187:   \bigskipamount=\tmpdim plus.33333\tmpdim minus.33333\tmpdim
188:   \medskipamount=.5\tmpdim plus.16666\tmpdim minus.16666\tmpdim
189:   \smallskipamount=.25\tmpdim plus.08333\tmpdim minus.08333\tmpdim
190:   \normalbaselineskip=\tmpdim
191:   \jot=.25\tmpdim
192:   \maxdepth=.33333\tmpdim
193:   \setbox\strutbox=\hbox{\vrule height.709\tmpdim depth.291\tmpdim width0pt}%
194:   \fi
195: }

```

opmac.tex

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

```

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

```

opmac.tex

Makra `\fontscale` (*factor*), `\textfontscale` (*factor*) a `\scalebaselineskip` (*factor*) přepočítají *factor* podle aktuálního `\fontdim` resp. `\baselineskip` na absolutní jednotku a zavolají odpovídající makro definované před chvílí. Na řádce 199 je #1 převedeno na (#1/1000)pt: Číslo 3277sp je  $2^{16}/20\text{sp}$ , tedy  $1/20\text{pt}$ . Tato hodnota je nejprve vynásobena #1 a vydělena 50. Proč bylo číslo 1000 rozloženo na  $20 \times 50$ ? Aby nedošlo k přetečení hodnoty typu *dimen* při velkém #1.

```

198: \def\fontscale#1{\if$#1$\else \ifnum#1=1000 \else
199:   \tmpdim=3277sp \tmpdim=#1\tmpdim \divide\tmpdim by50
200:   \tmpdim=\expandafter\ignorept\the\tmpdim \fontdim
201:   \withoutunit\fontsize\tmpdim
202:   \fi\fi
203: }
204: \def\textfontscale#1{\if$#1$\else
205:   \tmpdim=#1pt \divide\tmpdim by1000
206:   \tmpdim=\expandafter\ignorept\the\tmpdim \fontdim
207:   \withoutunit\textfontsize\tmpdim
208:   \fi
209: }
210: \def\scalebaselineskip#1{\if$#1$\else \ifnum#1=1000 \else

```

opmac.tex

---

`\textfontsize`: 9–11    `\setbaselineskip`: 9–11    `\withoutunit`: 10–11    `\fontscale`: 9–10  
`\textfontscale`: 10–11    `\scalebaselineskip`: 9–10

```

211: \tmpdim=3277sp \tmpdim=#1\tmpdim \divide\tmpdim by50
212: \tmpdim=\expandafter\ignorept\the\tmpdim \baselineskip
213: \withoutunit\setbaselineskip\tmpdim
214: \fi\fi
215: }

```

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`.

```

216: \def\thefontsize[#1]{\fontdim=#1\ptunit
217: \expandafter\let \expandafter\thefont \the\font
218: \edef\sizespec[at#1\ptunit]\def\dgsize{#1\ptunit}\resizefont\thefont
219: \thefont \let\dgsize=\undefined \ignorespaces
220: }
221: \def\thefontscale[#1]{%
222: \tmpdim=#1pt \divide\tmpdim by1000
223: \tmpdim=\expandafter\ignorept\the\tmpdim \fontdim
224: \withoutunit\thefontsize\tmpdim
225: }

```

opmac.tex

PlainT<sub>E</sub>Xový `\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.

```

226: \def\magstep#1{\ifcase#1 1000\or1200\or1440\or1728\or2074\or2488\fi\space}

```

opmac.tex

Makro `\typobase` nastaví `\baselineskip` a `\fontdim` podle `\baselineskipB` a `\fontdimB`, což jsou makra, která mají uloženu základní velikost řádkování a základní velikost písma.

```

228: \def\typobase{\ifx\baselineskipB\undefined \def\baselineskipB{12pt}\fi
229: \ifx\fontdimB\undefined \def\fontdimB{10pt}\fi
230: \baselineskip=\baselineskipB\relax \fontdim=\fontdimB\relax
231: }

```

opmac.tex

Makro `\em` přepíná kontextově do odpovídající varianty a ve spolupráci s makry `\additcorr` a `\afteritcorr` přidává italskou korekci. Makro `\additcorr` si pomocí `\lastskip` zapamatuje poslední mezeru, pak ji odstraní, vloží italskou korekci a nakonec vrátí tu odstraněnou mezeru. Makro `\afteritcorr` se probudí k činnosti na konci skupiny a přidá italskou korekci, pokud nenásleduje tečka nebo čárka.

```

232: \def\em {\expandafter\ifx \the\font \tenit \additcorr \rm \else
233: \expandafter\ifx \the\font \tenbf \bi\aftergroup\afteritcorr\else
234: \expandafter\ifx \the\font \tenbi \additcorr \bf \else
235: \it \aftergroup\afteritcorr\fi\fi\fi}
236: \def\additcorr{\ifdim\lastskip>0pt \skip0=\lastskip \unskip\/\hskip\skip0 \else\/\fi}
237: \def\afteritcorr{\def\tmp{\ifx\next..\else\ifx\next,,\else\/%
238: \expandafter\expandafter\expandafter\next\expandafter\fi\fi}%
239: \afterassignment\tmp \let\next= }

```

opmac.tex

Fontová makra zabezpečíme proti rozkladu v parametru `\write`.

```

241: \addprotect\thefontsize \addprotect\thefontscale
242: \addprotect\typosize \addprotect\typoscale
243: \addprotect\textfontsize \addprotect\textfontscale
244: \addprotect\em

```

opmac.tex

Makro `\fontfam` je definováno v souboru `fontfam.tex`. Není účelné je zavádět přímo do OPmac, protože makro přečte také rozsáhlá data o fontech, která mohou zbytečně zatěžovat paměť, pokud uživatel `\fontfam` nikdy nepoužije. Takže tento makro soubor a data jsou přečteny až při prvním použití `\fontfam`. Uvedený soubor maker definuje `\fontfam`, takže na následujícím řádku nevidíte žádnou rekurzi.

---

```

\thefontsize: 11, 55 \thefont: 11, 39, 41 \thefontscale: 8, 11, 39, 41 \magstep: 11, 16
\typobase: 11, 16, 46 \baselineskipB: 10-11 \fontdimB: 10-11 \em: 8, 11, 52 \additcorr: 11
\afteritcorr: 11 \fontfam: 11-12, 54

```

```
246: \def\fontfam{\par \input fontfam \fontfam}
```

opmac.tex

### 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.

```
250: \def\mtext#1{\csname mt:#1:\csname lan:\the\language\endcsname\endcsname}
```

opmac.tex

Jednotlivé texty definujeme pomocí `\sdef{mt:{značka}:{jazyk}}` takto:

```
252: \sdef{mt:chap:en}{Chapter} \sdef{mt:chap:cs}{Kapitola} \sdef{mt:chap:sk}{Kapitola}
253: \sdef{mt:t:en}{Table} \sdef{mt:t:cs}{Tabulka} \sdef{mt:t:sk}{Tabu\ v lka}
254: \sdef{mt:f:en}{Figure} \sdef{mt:f:cs}{Obr\ 'azek} \sdef{mt:f:sk}{Obr\ 'azok}
```

opmac.tex

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`.

```
256: \ifx\r\undefined \csname csaccents\endcsname \fi
```

opmac.tex

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:

```
258: \sdef{lan:0}{en} \sdef{lan:100}{en} \sdef{lan:101}{en}
259: \sdef{lan:5}{cs} \sdef{lan:15}{cs} \sdef{lan:115}{cs}
260: \sdef{lan:6}{sk} \sdef{lan:16}{sk} \sdef{lan:116}{sk}
```

opmac.tex

Je-li detekován místo CSplainu eTeX, nastavíme značky jazyků dle hodnoty `\language` v eTeXu:

```
262: \ifx\uselanguage\undefined \def\isolangset#1#2{\else
263: \message{OPmac: etex.src macros detected}
264: \def\isolangset#1#2{\uselanguage#1}\sdef{lan:\the\language}{#2}%
265: \global\expandafter\chardef\csname#2Patt\endcsname=\language}}
266: \isolangset{USenglish}{en} \isolangset{czech}{cs} \isolangset{slovak}{sk}
267: \fi
```

opmac.tex

Makro `\isolangset` `{\dlouhý-název}{\iso-zkratka}` přiřadí dlouhému názvu jazyka (a související hodnotě registru `\language`) jeho zkratku dle ISO 639-1 při použití formátu generovaného z `etex.src`. Naopak, při použití CSplainu makro nedělá nic, protože ISO zkratky a jejich propojení na hodnoty `\language` jsou nastaveny přímo v CSplainu.

### 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 pořádk na disku kvůli tomu uklízet smetí.

Je potřeba deklarovat souborové deskriptory `\reffile` a `\testin`:

```
271: \newwrite\reffile
272: \newread\testin
```

opmac.tex

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.

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

opmac.tex

---

`\mtext`: 12, 15, 19    `\isolangset`: 12    `\reffile`: 12–13    `\testin`: 12–13, 52  
`\wref`: 12–14, 17, 21, 46–47, 51–54    `\wrefrelax`: 12–13, 51

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 pomocí makra `\openrefA` otevře k zápisu a připraví `\wref` do stavu, kdy toto makro bude ukládat data do souboru.

```

277: \def\inputref{
278:   \openin\testin=\jobname.ref
279:   \ifeof\testin \else
280:     \closein\testin
281:     \input \jobname.ref
282:     \fnotenum=0 \mnotenum=0
283:     \openrefA{\string\inputref}%
284:   \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 zavolá `\openrefA`, které REF soubor založí, pře-definuje `\wref` a vloží první řádek do souboru. Tím je zaručeno, že při příštím  $\TeX$ ování 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.

```

286: \def\openref{%
287:   \ifx\wref\wrefrelax \openrefA{\string\openref}\fi
288:   \gdef\openref{}%
289: }
290: \def\openrefA#1{%
291:   \immediate\openout\reffile=\jobname.ref
292:   \gdef\wref##1##2{\write\reffile{\string##1##2}}%
293:   \immediate\write\reffile {\percent\percent\space OPmac - REF file (#1)}%
294:   \immediate\wref\Xrefversion{\REFversion}%
295: }

```

Pro zápisy do REF souboru používáme tuto konvenci: první kontrolní sekvence na řádce je vždy tvaru `\X<název>`, takže máme přehled, která kontrolní sekvence pochází z REF souboru.

Jako první je do REF souboru vložen příkaz `\Xrefversion <číslo>`. Pokud toto `<číslo>` není rovno `\REFversion`, REF soubor se nepřečte. Tím je zaručeno, že OPmac nezkolabuje při čtení REF souboru kvůli tomu, že je zde zmatení verzí. Číslo verze `\REFversion` zvětším o jedničku vždy, když v budoucí verzi OPmac přidám nebo uberu v REF souboru nějakou funkci.

```

296: \def\REFversion{2}
297: \def\Xrefversion#1{\ifnum#1=\REFversion\relax \else \endinput \fi}

```

### 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 `\sxddef{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.

---

`\inputref`: 13, 57    `\openref`: 13–14, 21, 36, 46–47, 49, 52    `\openrefA`: 13    `\Xrefversion`: 13  
`\REFversion`: 13



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. Ostatní balast v kódu (kontrolující definovanost makra `\csname_10:⟨lejblík⟩\endcsname`) je od toho, aby OPmac pohlídal případné dvojí použití stejného *⟨lejblíku⟩* a upozornil na to.

```
301: \def\label[#1]{\isdefined{10:#1}%
302: \iftrue \opwarning{duplicated label [#1], ignored}\else \xdef\lastlabel{#1}\fi
303: \ignorespaces}
opmac.tex
```

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⟩*]).

```
305: \def\wlabel#1{%
306: \ifx\lastlabel\undefined \else
307: \dest[ref:\lastlabel]%
308: {\protectlist\edef\tmp{\wref\Xlabel{\lastlabel}{#1}}\expandafter}\tmp
309: \sxddef{lab:\lastlabel}{#1}\sxddef{10:\lastlabel}{}%
310: \global\let\lastlabel=\undefined
311: \fi
312: }
opmac.tex
```

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`.

```
313: \def\ref[#1]{\isdefined{lab:#1}%
314: \iftrue \reflink[#1]{\csname lab:#1\endcsname}%
315: \else ??\opwarning{label [#1] unknown. Try to TeX me again}\openref
316: \fi
317: }
opmac.tex
```

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`.

```
318: \def\pgref[#1]{\isdefined{pgref:#1}%
319: \iftrue \pglink{\csname pgref:#1\endcsname}%
320: \else ??\opwarning{pg-label [#1] unknown. Try to TeX me again}\openref
321: \fi
322: }
323: \def\Xlabel#1#2{\sdef{lab:#1}{#2}\sxddef{pgref:#1}{\the\lastpage}}
opmac.tex
```

### 3.8 Kapitoly, sekce, podsekce

Od verze OPmac Jul. 2013 jsou zcela přepracována pomocná makra pro návrh typografie kapitol, sekcí a podsekcí. Nyní má autor typografického návrhu lépe pod vlastní kontrolou, co se vkládá do vertikálního seznamu, což je pro programování možných stránkových zlomů a nezlomů důležité.

Autor typografie dokumentu by měl definovat pro tisk kapitoly, sekce a podsekce makra `\printchap`, `\printsec` a `\printsecc`. Makra mají jeden parametr `#1`, který obsahuje text titulku. Typická struktura každého takového makra je:

```
\par
⟨penalta obvykle záporná, neboli bonus, pro zlomení stránky před nadpisem⟩
⟨mezera před nadpisem⟩
{\nastavení fontu} \noindent \dotocnum{⟨značka⟩}#1\nbpar}
⟨případné vložení značky (insertmark) pro plovoucí záhlaví⟩
```

---

`\label`: 13–14, 34    `\lastlabel`: 14    `\wlabel`: 14, 17, 19    `\ref`: 13–14    `\pgref`: 14  
`\Xlabel`: 13–14, 55



`\nobreak` *(mezera pod nadpisem)*

Pro realizaci maker `\printchap`, `\printsec` a `\printsecc` může autor návrhu využít následujících interních maker OPmac:

- `\dotocnum{⟨značka⟩}` – umístí cíle odkazů, zařídí obsah, vytiskne *⟨značku⟩*
- `\thetocnum` – *⟨značka⟩*, např. 3.2.4 pro secc, 3.2 pro sec a 3 pro chap
- `\insertmark{⟨text⟩}` – vloží `\mark` s expandovaným `\thetocnum` a neexpandovaným *⟨textem⟩*
- `\nbpar` – jako `\par`, ale mezi řádky je nezlomitelná mezera
- `\remskip⟨velikost⟩` – mezera (pod nadpisem) odstranitelná následujícím `\norempenalty`
- `\norempenalty⟨číslo⟩` – vloží penaltu *⟨číslo⟩* jen pokud nepředchází `\remskip`

Aby fungoval obsah a cíle odkazů, je *nutné* použít `\dotocnum`. Parametr makra `\dotocnum` nemusí obsahovat jen `\thetocnum`, ale také tečky a mezery kolem *⟨značky⟩*. Předchází-li `\nonum`, makro `\dotocnum` nevytiskne celý svůj druhý parametr, tedy včetně případného „okolí“ značky. Návrh tisku sekce může vypadat takto:

```
\def\printsec#1{\par
  \norempenalty-500
  \vskip 12pt plus 2pt
  {\secfont \noindent \dotocnum{\thetocnum\quad}#1\nbpar}%
  \placemark{#1}%
  \nobreak \remskip 6pt plus 1pt
}
```

V tomto návrhu bude nad nadpisem penalta  $-500$  (bonus za zlomení nad nadpisem), dále je `12pt` mezera, pak je titulek `#1` vytištěný fontem `\secfont`. Před tímto titulkem je číselná značka oddělená od titulku mezerou `\quad`. Titulek může být na více řádcích. Protože je ukončen `\nbpar`, nebude povolen mezi jednotlivými řádky titulku řádkový zlom.

Vysvětlíme si nyní na příkladu činnost a smysl `\remskip` a `\norempenalty`. Předpokládejme pro ilustraci definici `\printsec`, jako je uvedena výše. Pokud je dále třeba definice podsekce `\printsecc` zahájena příkazy

```
\par \norempenalty-200 \vskip 8pt plus2pt
```

pak se mohou dít tyto věci:

- Následuje-li podsekce těsně za sekci, pak se vymaže spodní mezera od sekce `6pt plus 1pt` a místo ní se vloží mezera `8pt plus 2pt`. Mezery se tedy nesčítají. Navíc v tomto případě se nevloží penalta  $-200$ , takže mezi sekci a podsekcí nedojde nikdy ke stránkovému zlomu.
- Následuje-li za sekci obyčejný text, pak je pod sekci a nad textem mezera `6pt plus 1pt`, která je nezlomitelná.
- Předchází-li před podsekcí obyčejný text, pak se vloží před nadpisem podsekce `\penalty-200` následovaná `\vskip 8pt plus 2pt`. Tato mezera je ochotně zlomitelná (bonus  $-200$ ), takže se může nadpis podsekce objevit na následující straně.

Je možné mezera pod nadpisem složit ze dvou druhů:

```
\def\printsec{%
  ...
  \nobreak \vskip 2pt \remskip 4pt plus1pt}
```

V tomto příkladě se odstraní při následující podsekci z celkové mezery `6pt plus 1pt` jen její část `4pt plus 1pt`.

Defaultní hodnoty maker `\printchap`, `\printsec` a `\printsecc` vypadají v OPmac takto:

```
327: \def\printchap#1{\vfill\supereject
328:   {\chapfont \noindent \mtext{chap} \dotocnum{\thetocnum}\par
329:     \nobreak\smallskip\noindent #1\nbpar}\mark{}}%
330: \nobreak \remskip\bigskipamount \firstnoindent
331: }
```

opmac.tex

`\printchap`: 14–18    `\printsec`: 14–18    `\printsecc`: 14–18

```

332: \def\printsec#1{\par \norempenalty-400 \bigskip
333:   {\secfont \noindent \dotocnum{\thetocnum\quad}#1\nbpar}\insertmark{#1}%
334:   \nobreak \remskip\medskipamount \firstnoindent
335: }
336: \def\printsecc#1{\par \norempenalty-200 \medskip
337:   {\seccfont \noindent \dotocnum{\thetocnum\quad}#1\nbpar}%
338:   \nobreak \remskip\medskipamount \firstnoindent
339: }

```

Příkazem `\firstnoindent` dávají tato makra najevo, že následující odstavec nebude mít odstavcovou zarážku.

**Upozornění:** Od verze Apr. 2016 jsou makra `\tit`, `\chap`, `\sec` a `\secc` definována pomocí `\eoldef`, tedy titulek ve zdrojovém kódu je ukončen koncem řádku a ne následujícím prázdným řádkem. Chcete-li mít delší titulek ve zdrojovém kódu rozdělen do více řádků, ukončete „pokračovací řádky“ symbolem `%`. Pokud chcete makra `\chap`, `\sec` atd. použít uvnitř vlastních maker, nelze je použít přímo. Můžete to ale vyřešit třeba takto:

```
\def\mymacro#1{... \csname\string\sec:M\endcsname{#1} ...}
```

Makro pro titul `\tit` počítá s tím, že bude titul na více řádcích. Sází ho tedy jako odstavec s pružnými `\leftskip` a `\rightskip`.

opmac.tex

```

340: \eoldef\tit#1{\vglue4em
341:   {\leftskip=0pt plus1fill \rightskip=\leftskip
342:   \titfont \noindent #1\par}%
343:   \nobreak\bigskip
344: }

```

Fonty pro titul, kapitoly a sekce `\titfont`, `\chapfont`, `\secfont` a `\seccfont` jsou definovány jako odpovídající zvětšení a nastavení tučného ductu. Ten je nastaven pomocí `\bfshape` jako `\bf` a navíc je ztotožněn `\tenit` s `\tenbi`, takže když nyní uživatel napíše `\it`, dostane tučnou kurzívu. Tučné varianty matematických fontů se zavedou až při použití matematického módu v nadpise (viz `\everymath`).

opmac.tex

```

345: \def\titfont{\typobase\typoscale[\magstep4/\magstep4]\bfshape}
346: \def\chapfont{\typobase\typoscale[\magstep3/\magstep3]\bfshape}
347: \def\secfont{\typobase\typoscale[\magstep2/\magstep2]\bfshape}
348: \def\seccfont{\typobase\typoscale[\magstep1/\magstep1]\bfshape}
349: \def\bfshape{\let\tenit=\tenbi \everymath\expandafter{\the\everymath\boldmath}\bf}

```

V další části této sekce je implementace maker `\chap`, `\sec` a `\secc`. Pro číslování kapitol, sekcí a podsekcí potřebujeme čítače a další registry:

opmac.tex

```
351: \newcount\chapnum \newcount\secnum \newcount\seccnum \newcount\nonumnum
```

Makro `\notoc` je možno použít jako prefix před `\chap`, `\sec`, `\secc` s tím, že se kapitola, sekce, podsekce nedostane do obsahu. Makro `\nonum` je možno použít jako prefix před stejnými makry s tím, že kapitola, sekce, podsekce nebude mít číslo. I nečíslování kapitola se může dostat do obsahu. Je-li obsah klikací, potřebuje mít svoje referenční číslo pro vytvoření linku. K tomu slouží registr `\nonumnum`.

opmac.tex

```

352: \newif\ifnotoc \notocfalse \def\notoc{\global\notoctrue}
353: \newif\ifnonum \nonumfalse \def\nonum{\global\nonumtrue}

```

Makra `\chap`, `\sec` a `\secc` nastaví odpovídající čítače, dále vytvoří číslování pro tisk (sestává jící z více čísel) v makrech `\thechapnum`, `\theseccnum` a `\theseccnum`. Aktuální čítač má vždy název `\thetocnum`. S touto hodnotou (nezávisle na tom, zde jde o kapitolu, sekcí nebo podsekcí, bude pracovat `\dotocnum`. Dále makra připraví obsah makra `\dotocnumafter`, což je proměnlivá část makra `\dotocnum`. Konečně uvedená makra `\chap`, `\sec` a `\secc` zavolají odpovídající makro `\printchap`, `\printsec` a `\printsecc`.

---

```

\tit: 16      \titfont: 16      \chapfont: 15–16      \secfont: 15–16      \seccfont: 16
\bfshape: 16–17, 21  \chapnum: 17      \secnum: 17      \seccnum: 17      \nonumnum: 16–17
\notoc: 16–18      \nonum: 15–18, 21  \chap: 8, 16–17      \sec: 8, 16–17      \secc: 8, 16–17
\thechapnum: 17–18  \theseccnum: 17–19  \theseccnum: 17–18  \thetocnum: 15–18
\dotocnumafter: 17–18

```

```

355: \eoldef\chap#1{\ifnonum\else \global\advance\chapnum by1 \fi
356: \chaphook {\globaldefs=1 \secnum=0 \seccnum=0 \tnum=0 \fnum=0 \dnum=0}\relax
357: \edef\thechapnum{\the\chapnum}\let\thetocnum=\thechapnum
358: \edef\theseccnum{\othe\chapnum.\the\secnum}%
359: \def\dotocnumafter{\wtotoc0\bfshape{#1}}%
360: \printchap{#1}\resetnonumnotoc
361: }
362: \eoldef\sec#1{\ifnonum\else \global\advance\secnum by1 \fi
363: \sechook {\globaldefs=1 \seccnum=0 \tnum=0 \fnum=0 \dnum=0}\relax
364: \edef\theseccnum{\othe\chapnum.\the\secnum}\let\thetocnum=\theseccnum
365: \def\dotocnumafter{\wtotoc1\rm{#1}}%
366: \printsec{#1}\resetnonumnotoc
367: }
368: \eoldef\secc#1{\ifnonum\else \global\advance\seccnum by1 \fi
369: \sechook {}\relax
370: \edef\theseccnum{\othe\chapnum.\the\secnum.\the\seccnum}\let\thetocnum=\theseccnum
371: \def\dotocnumafter{\wtotoc2\rm{#1}}%
372: \printsecc{#1}\resetnonumnotoc
373: }

```

V proměnlivé části makra `\dotocnum`, tedy v `\dotocnumafter`, se řeší uložení informací o kapitole, sekci nebo podsekci do obsahu. K tomu je využito makro `\wtotoc`  $\langle úroveň \rangle \langle font \rangle \langle text \rangle$ , které vytvoří

```

\write\reffile
  {\string\Xtoc{\langle úroveň \rangle}{\font}}{\langle expandovaný thetocnum \rangle}{\langle text \rangle}{\the\pageno}}

```

Přitom  $\langle úroveň \rangle$  je číslo rovné nule, jedná-li se o kapitolu, je rovné jedné, jedná-li se o sekci a je rovno dvěma, jedná-li se podsekci.

```

374: \def\wtotoc#1#2#3{% #1 = level, #2 = info, #3 = titletext
375:   \ifnotoc\else
376:     \def\act{\wref{\Xtoc{#1}{\noexpand#2}}}%
377:     \expandafter\act\expandafter{\expandafter{\thetocnum}{#3}{\the\pageno}}%
378:   \fi
379: }

```

Makro `\wcontents` je v kódu ponecháno pro zpětnou kompatibilitu (ukládalo do REF souboru údaje pro obsah `\Xchap`, `\Xsec` a `\Xsecc`). Někdy v roce 2016 je pravděpodobně zcela odstraním.

```

380: \def\wcontents#1#2{% #1 = sequence to REF, #2 = titletext
381:   \ifnotoc\else
382:     \expandafter\wref\expandafter#1\expandafter
383:       {\expandafter{\thetocnum}{#2}{\the\pageno}}%
384:   \fi
385: }

```

Makro `\dotocnum`  $\langle text \rangle$  umístí cíl odkazu do místa, které je od účaří vzdáleno o `\destheight`. Toto místo se kryje s horní hranou okna prohlížeče po kliknutí na odkaz. Dále toto makro vygeneruje data pro obsah pomocí předpřipraveného `\dotocnumafter`. Pokud předchází `\notoc`, makro nezapiše nic do obsahu. Pokud předchází `\nonum`, makro nevytiskne svůj parametr, takže je titulek bez čísla. Dále v takovém případě je pro hyperlinkové odkazy číslo `\nonumnum` uvozeno vykřičníkem, což navazuje v obsahu na makro `\toclinkA`.

```

386: \def\dotocnum#1{%
387:   \leavevmode
388:   {\ifnonum \global\advance\nonumnum by1 \edef\thetocnum{!\the\nonumnum}\fi
389:   \wlabel\thetocnum \dest[toc:\tocilabel.\thetocnum]%
390:   \dotocnumafter}\ifnonum\else#1\fi
391: \global\let\dotocnumafter=\relax
392: }

```

V makrech `\chap`, `\sec`, `\secc` je po zavolání `\printchap`, `\printsec` nebo `\printsecc` voláno `\resetnonunotoc`, které vrátí hodnotu přepínačů `\ifnonum` a `\ifnotoc` na implicitní hodnoty. Tím je zaručeno, že makra `\nonum` a `\notoc` ovlivní jen následující kapitolu, sekci nebo podsekci a fungují

jako prefixy. Makro navíc kvůli přechodu na verzi OPmac Jul. 2013 kontroluje, zda makra `\printchap`, `\printsec` a `\printsecc` skutečně použila makro `\dotocnum`. Pokud ne, náležitě na to upozorní.

```
393: \def\resetnonumtoc{\global\notocfalse \global\nonumfalse
394:   \ifx\dotocnumafter\relax \else
395:     \opwarning{\noexpand\dotocnum unused in printchap/printsec/printsecc}\fi
396: }
```

opmac.tex

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` `{(text)}` vloží do `\mark` data ve formátu `{(thetocnum)}_{(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.

```
397: \def\insertmark#1{\toks0={#1}\mark{\ifnonum\else\thetocnum\fi} {\the\toks0}}
```

opmac.tex

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, inspiруйте se v TBN na stranách 259 a 260.

Makro `\remskip` `(velikost)` je implimentováno jako `\vskip(velikost)` následované smluvenou nezlomitelnou penaltou 11333. Makro `\norempenalty` pak podle této hodnoty poslední penaltu v seznamu větví svou činností. V registru `\remskipamount` je uložena naposledy vložená mezerka z `\remskip`.

```
399: \newskip\remskipamount
400: \def\remskip{\afterassignment\remskipA \global\remskipamount}
401: \def\remskipA{\vskip\remskipamount \penalty11333 }
402: \def\norempenalty{\ifnum\lastpenalty=11333
403:   \vskip-\remskipamount \tmpnum=\else \removelastskip \penalty \fi}
```

opmac.tex

Makro `\othe` pracuje stejně jako primitiv `\the` s tím rozdílem, že nezobrazí nic (a sejme následující tečku), pokud je hodnota registru nulová. Tímto způsobem lze tisknout dokument jen se sekcemi bez kapitol. Číslo kapitoly se pak nezobrazuje jako 0., protože se nezobrazuje vůbec.

```
405: \def\othe#1.{\ifnum#1>0 \the#1.\fi}
406: \def\thechapnum{} \def\thesecnum{} \def\theseccnum{}
```

opmac.tex

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 výpisu.

```
408: \def\afternoindent{\global\everypar={\wipeepar\setbox7=\lastbox}}
409: \def\wipeepar{\global\everypar={}}
410: \let\firstnoindent=\afternoindent
```

opmac.tex

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

```
411: \def\nbp{{\interlinepenalty=10000\endgraf}}
412: \def\nl{\hfil\break}
```

opmac.tex

---

`\insertmark`: 15–16, 18    `\remskip`: 15–16, 18    `\norempenalty`: 15–16, 18    `\remskipamount`: 18  
`\othe`: 17–18    `\afternoindent`: 18, 39    `\wipeepar`: 18, 30, 39, 41    `\firstnoindent`: 15–16, 18  
`\nbp`: 15–16, 18–19    `\nl`: 18, 54

### 3.9 Popisky, rovnice

Nejprve deklaruujeme potřebné čítače:

```
416: \newcount\tnum \newcount\fnun \newcount\dnum
```

opmac.tex

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

```
418: \def\thetnum{\theseccnum.\the\tnum}
```

```
419: \def\thefnum{\theseccnum.\the\fnun}
```

```
420: \def\thednum{\the\dnum}
```

opmac.tex

Makro `\caption` /(*typ*)<sub>□</sub> 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 a vytiskne zahájení popisku makrem `\printcaption`. Makro `\caption` tím končí svou činnost a dále je zpracován odstavec s nastavenými `\leftskip`, `\rightskip`. Sekvence `\par` je předefinována tak, že první výskyt `\par` (alias prázdného řádku) ukončí skupinu a tím se všechna nastavení vrátí do původního stavu.

```
422: \def\caption/#1 {\isdefined{#1num}%
423:   \iftrue \global\advance \csname #1num\endcsname by1
424:   \else \opwarning{Unknown caption /#1}%
425:   \fi
426:   \bgroup
427:   \leftskip=\iindent plus1fil
428:   \rightskip=\iindent plus-1fil
429:   \parfillskip=0pt plus2fil
430:   \def\par{\nbpargroup}%
431:   \captionhook{#1}\noindent
432:   \wlabel{\csname the#1num\endcsname}%
433:   \printcaption{\mtext{#1}}{\csname the#1num\endcsname}%
434: }
```

opmac.tex

Makro `\printcaption` {*slovo*}{*číslo*} vytiskne zahájení popisku tabulky a obrázku. Za číslem implicitně není žádná interpunkce, jen `\enspace`. Je možné předefinovat `\printcaption` s interpunkcí třeba takto: `\def\printcaption#1#2{\bf#1 #2:\space}`

```
435: \def\printcaption#1#2{\bf#1 #2}\enspace}
```

opmac.tex

Předefinujeme makro z plain $\TeX$ u `\endinsert` tak, že dopředu vložíme `\par`. Pak bude možné těsně za odstavec zahájený pomocí `\caption` vkládat `\endinsert`. Těžko lze totiž přesvědčovat uživatele, aby tam dával prázdný řádek.

```
437: \expandafter\def\expandafter\endinsert\expandafter{\expandafter\par\endinsert}
```

opmac.tex

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`.

```
439: \def\eqmark{\global\advance\dnum by1
440:   \ifinner\else\eqno \fi
441:   \wlabel\thednum \thednum
442: }
```

opmac.tex

### 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 deklaruujeme čítač pro odrážky `\itemnum`.

```
446: \newcount\itemnum \itemnum=0
```

opmac.tex

---

`\tnum`: 17, 19    `\fnun`: 17, 19    `\dnum`: 17, 19    `\caption`: 8, 19    `\printcaption`: 19    `\eqmark`: 19  
`\itemnum`: 19–20

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

opmac.tex

```
448: \def\begitem{\par\iiskip\bgroup
449:   \itemnum=0 \adef*\startitem}
450:   \advance\leftskip by\iindent
451:   \let\printitem=\normalitem
452: }
453: \def\enditem{\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 `\begitem`s na `\normalitem`.

opmac.tex

```
455: \def\startitem{\par \advance\itemnum by1
456:   \itemhook \noindent\llap{\printitem}\ignorespaces}
457: \def\normalitem{${\bullet}$\enspace}
```

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

opmac.tex

```
459: \def\style#1{\expandafter\let\expandafter\printitem\csname item:#1\endcsname
460:   \ifx\printitem\relax \let\printitem=\normalitem \fi
461: }
462: \sdef{item:o}{\raise.4ex\hbox{${\scriptscriptstyle}\bullet$} }
463: \sdef{item:-}{- }
464: \sdef{item:n}{\the\itemnum. }
465: \sdef{item:N}{\the\itemnum} }
466: \sdef{item:i}{(\romannumeral\itemnum) }
467: \sdef{item:I}{\uppercase\expandafter{\romannumeral\itemnum}\kern.5em}
468: \sdef{item:a}{\athe\itemnum} }
469: \sdef{item:A}{\uppercase\expandafter{\athe\itemnum}} }
470: \sdef{item:x}{\raise.3ex\fullrectangle{.6ex} }
471: \sdef{item:X}{\raise.2ex\fullrectangle{1ex}\kern.5em}
```

Čtvereček kreslíme jako `\vrule` odpovídajících rozměrů makrem `\fullrectangle`  $\langle dimen \rangle$ .

opmac.tex

```
473: \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`  $\langle number \rangle$ .

opmac.tex

```
475: \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
476:   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
477: }
```

### 3.11 Tvorba obsahu

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

opmac.tex

```
481: \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, které jsou shodné, jako parametry makra `\Xtoc`  $\langle úroveň \rangle \langle info \rangle \langle číslo \rangle \langle text \rangle \langle strana \rangle$ .

opmac.tex

```
483: \def\Xtoc#1#2#3#4#5{\ifnum#1=0 \ischaptrue\fi \addto\toclist{\tocline{#1}{#2}{#3}{#4}{#5}}}
```

Makra `\Xchap`, `\Xsec` a `\Xsecc` přetrvávají v kódu jen pro zpětnou kompatibilitu a někdy v roce 2016 je odstraní.

---

```
\begitem: 7, 20   \enditem: 7, 20   \startitem: 7, 20   \printitem: 20   \normalitem: 20
\style: 20       \fullrectangle: 20   \athe: 20       \toclist: 20-21, 36   \ifischap: 20-21
\Xtoc: 17, 20-21  \Xchap: 17, 21   \Xsec: 17, 21   \Xsecc: 17, 21
```



```
484: \def\Xchap{\Xtoc0\bfshape} \def\Xsec{\Xtoc1\rm} \def\Xsecc{\Xtoc2\rm} opmac.tex
```

Makro `\tocline`  $\{\langle\acute{u}roveň\rangle\}\{\langle info\rangle\}\{\langle\acute{c}íslo\rangle\}\{\langle text\rangle\}\{\langle strana\rangle\}$  vytvoří řádek obsahu. Údaj  $\langle\acute{u}roveň\rangle$  je číslo 0 pro kapitolu, 1 pro sekci a 2 pro podsekcí. Údaj  $\langle info\rangle$  používá OPmac pro informaci o fontu, kterým se má tisknout řádek v obsahu. Řádek obsahu tiskneme jako odstavec, protože  $\langle text\rangle$  může být třeba delší. Registr `\leftskip` nastavíme jako součin  $\langle\acute{u}roveň\rangle$  krát `\iindent`. Pokud se v dokumentu vyskytují kapitoly, odsadíme ještě o další `\iindent`. Registr `\rightskip` nastavíme na  $2\iindent$ , aby delší  $\langle text\rangle$  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.

```
486: \def\tocline#1#2#3#4#5{\leftskip=#1\iindent \rightskip=2\iindent opmac.tex
487: \ifischap\advance\leftskip by\iindent\fi
488: \ifnum#1>1 \advance\leftskip by\iindent\fi
489: \toclinehook \noindent\llap{#2\toclink{#3}\enspace}%
490: {#2#4}\nobreak\tocdotfill\pglink{#5}\nobreak\hskip-2\iindent\null\par}}
491: \def\tocdotfill{\leaders\hbox to.8em{\hss.\hss}\hskip 1em plus1fill\relax}
```

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

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

Makro `\toclinkA` je citlivé na vykřičník jako první znak argumentu. Pokud tam je, vytiskne jen mezeru velikosti `0.8em`, jinak vytiskne argument. Vykřičník do argumentu dáme v případě kapitol a sekcí prefixovaných jako `\nonum`. V obsahu pak nechceme mít žádné číslo, jen příslušnou mezeru.

```
497: \def\toclinkA#1{\def\tmp##1!##2\end{\if^##1^\kern.8em \else##1\fi}\tmp##1!\end} opmac.tex
```

### 3.12 Sestavení rejstříku

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

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

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

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

Makro `\iiA` sejme vždy jedno slovo ze seznamu. Podle prázdného parametru poznáme, že jsme u konce a neděláme nic. Při výskytu  $\langle slovo\rangle=@$  (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.

```
505: \def\iiA #1,{\if$#1$\else\def\tmpa{#1}% opmac.tex
506: \ifx\tmpa\iiatsign \expandafter\iiB\tmp,,\else\iindex{#1}\fi
507: \expandafter\iiA\fi}
508: \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`.

---

```
\tocline: 8, 20–21, 36 \tocdotfill: 21 \maketoc: 21 \toclinkA: 17, 21, 35 \iindex: 21–22
\ii: 21–22 \iiA: 21 \iiatsign: 21 \iiB: 21–22 \iiC: 21–22
```

```

510: \def\iiB #1,{\if$#1$\else \iiC#1/\relax \expandafter\iiB\fi}
511: \def\iiC #1/#2\relax{\if$#2$\else\iindex{#2#1}\fi}

```

opmac.tex

Makro `\iid`  $\langle slovo \rangle$  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

```

513: \def\iid #1 {\leavevmode\iindex{#1}#1\futurelet\tmp\iid}
514: \def\iid{\ifx\tmp,\else\ifx\tmp.\else\space\fi\fi}

```

Při čtení REF souboru se vykonávají makra `\Xindex`  $\{\langle heslo \rangle\}\{\langle strana \rangle\}$ , která postupně vytvářejí makra tvaru `\, \langle heslo \rangle`, ve kterých je shromažďován seznam stránek pro dané  $\langle heslo \rangle$ . Kromě toho každé makro `\, \langle heslo \rangle` 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  $\text{\TeX}$ U). Každé `\, \langle heslo \rangle` na konci čtení REF souboru obsahuje dva údaje ve svorkách, první údaj obsahuje pomocná data a druhý obsahuje seznam stránek. Tedy `\, \langle heslo \rangle` je makro s obsahem  $\{\langle pomocná-data \rangle\}\{\langle seznam-stránek \rangle\}$ .

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`  $\{\langle heslo \rangle\}\{\langle strana \rangle\}$  je z tohoto důvodu poněkud sofistikovanější.

opmac.tex

```

516: \def\Xindex{\Xindexg,}
517: \def\Xindexg#1#2#3{\bgroup \def~{ }% #1=prefix, #2=index-item, #3=pageno
518:   \isdefined{#1#2}\iftrue
519:     \ifx~#3~\else
520:       \expandafter\firstdata \csname#1#2\endcsname \XindexA
521:       \ifnum#3=\tmpa % \ii on the same page
522:       \else
523:         \tmpnum=#3 \advance\tmpnum by\pgfolioB-1
524:         \expandafter\seconddata \csname#1#2\endcsname \XindexB
525:         \ifx\tmp\empty
526:           \sxddef{#1#2}\{#3/\}\{\pgfolioA{#3}\} % previous item: empty page
527:         \else
528:           \if\tmpb+% state: the pagelist ends by a pagenumber
529:             \ifnum\tmpnum=\tmpa % the consecutive page
530:               \sxddef{#1#2}\{#3/-}\{\tmp\iiendash}\}
531:             \else % the pages drop
532:               \sxddef{#1#2}\{#3/\}\{\tmp, \pgfolioA{#3}\}
533:             \fi
534:           \else % state: the pagelist ends by --
535:             \ifnum\tmpnum=\tmpa % the consecutive page
536:               \sxddef{#1#2}\{#3/-}\{\tmp}\}
537:             \else % the pages drop
538:               \sxddef{#1#2}\{#3/\}\{\tmp\pgfolioA{\tmpa}, \pgfolioA{#3}\}
539:             \fi\fi\fi\fi\fi
540:           \else % first occurrence of the index item #2
541:             \ifx~#3~\sxddef{#1#2}\{0/\}\}\}\else \sxddef{#1#2}\{#3/\}\{\pgfolioA{#3}\}\}\fi
542:             \ifx,#1
543:               \global \expandafter\addto \expandafter\iilist \csname#1#2\endcsname
544:             \else
545:               \isdefined{iilist:#1}\iftrue
546:                 \global\expandafter\addto \csname iilist:#1\endcsname \csname#1#2\endcsname
547:               \else \sxddef{iilist:#1}\{\expandafter\noexpand \csname#1#2\endcsname}\}
548:             \fi\fi\fi
549:             \egroup
550:           }
551: \def\iilist{} \def\iiendash{--}

```

Činnost makra `\Xindex` si vysvětlíme za chvíli podrobněji. Nyní jen uvedeme, že `\Xindex` je jen speciální variantou obecného makra `\Xindexg` při `#1=,`. Takže pracuje s kontrolními sekvencemi typu `\, \langle heslo \rangle`. Následující text popisuje jen tento případ. Makro `\Xindexg` je možné použít pro paralelní vytváření dalších seznamů stránek stejných hesel (např. seznam vyznačený kurzívou, tučně atd.). Jak to udělat je popsáno v OPmac triku 0072.

Údaje o stranách spojených s rejstříkovým heslem jsou ukládány do makra `\,<heslo>` a jsou rozděleny do dvou částí ve tvaru `{první}{druhy}`. Definujeme pomocné makro `\firstdata \,<heslo> \<cs>`, které expanduje na `\<cs> <první-datový-údaj-hesla>&`. Je-li třeba `\,aa` definováno jako `{první}{druhy}`, pak `\firstdata \,aa \cosi` expanduje na `\cosi první&`. Tím máme možnost vyzískat data z makra. Podobně makro `\seconddata \,<heslo> \<cs>` expanduje na `\<cs> <druhý-datový-údaj-hesla>&`. Jsou použita pomocná makra `\firstdataA` a `\seconddataA`.

opmac.tex

```
553: \def\firstdata#1#2{\expandafter\expandafter\expandafter #2\expandafter\firstdata#1}
554: \def\firstdataA#1#2{#1&}
555: \def\seconddata#1#2{\expandafter\expandafter\expandafter #2\expandafter\seconddata#1}
556: \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  $\TeX$ u. 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 řádce 520 a `\seconddata` na řádce 524. 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

```
558: \def\XindexA#1/#2&{\def\tmpa{#1}\let\tmpb=#2}
559: \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> == <empty> || <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>;
```

---

```
\firstdata: 22–24, 28   \seconddata: 22–24   \firstdataA: 23   \seconddataA: 23
\XindexA: 22–24   \XindexB: 22–24   \iiendash: 22
```

V makru `\Xindex` pracujeme ještě se dvěma pomocnými makry `\pgfolioA` a `\pgfolioB`. Pro kladné stránky se tato makra chovají stejně, jako kdyby tam vůbec nebyla. Ovšem v plainTeXu (viz maro `\folio`) se mohou stránkové číslice vyskytnout na začátku dokumentu záporné, v takovém případě se mají tisknout římskými číslicemi. Proto jsou uvedená makra definována poněkud chytřeji.

```
561: \def\pgfolioA#1{\ifnum#1<0 \romannumeral-\fi#1}
562: \def\pgfolioB{\ifnum\tmpnum<0-\fi}
```

opmac.tex

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 typu `\,<heslo>` vloží konverzi textu `<heslo>` do tvaru vhodného 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`.

```
564: \def\makeindex{\par
565:   \ifx\iilist\empty \opwarning{index data-buffer is empty. TeX me again}
566:   \else
567:   \bgroup
568:     \setprimarysorting
569:     \def\act##1{\ifx##1\relax \else
570:       \firstdata##1\XindexA \seconddata##1\XindexB
571:       \if\tmpb+%
572:         \preparesorting##1% converted item by sorting data in \tmpb
573:         \xdef##1{\tmpb}{\tmpb}
574:       \else
575:         \preparesorting##1% converted item by sorting data in \tmpb
576:         \xdef##1{\tmpb}{\tmpb\pgfolioA{\tmpa}}
577:       \fi
578:       \expandafter\act\fi}
579:   \expandafter \act \iilist \relax
580:   \egroup
581:   \dosorting % sorting is in progress
582:   \bgroup
583:   \rightskip=0pt plus1fil \exhyphenpenalty=10000 \leftskip=\iindent
584:   \def\act##1{\ifx##1\relax \else \prepii##1%
585:     \seconddata##1\printiipages \expandafter\act \fi}
586:   \expandafter \act \iilist \relax
587:   \egroup
588:   \fi
589: }
```

opmac.tex

Makro `\printiipages` sebere z `<druhého-datového-údaje>` seznam stránek a jednoduše je vytiskne.

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

opmac.tex

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 na sekvenci s názvem `\,<heslo>`, ve které je uloženo, co se má místo hesla vytisknout. Data těchto výjimek jsou připravena makrem `\iis`

```
592: \def\prepii #1{\isinlist \iispeclist #1\iftrue
593:   \expandafter\expandafter\expandafter \printii \csname\string#1\endcsname&%
594:   \else \expandafter\prepiiA\string #1&%
595:   \fi
596: }
597: \def\prepiiA #1#2#3&{\printii#3&}
```

opmac.tex

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.

---

`\pgfolioA`: 22, 24, 35    `\pgfolioB`: 22, 24    `\makeindex`: 24–25, 28    `\printiipages`: 24  
`\prepii`: 24    `\prepiiA`: 24

Makro `\iis`  $\langle heslo \rangle_{\square} \{ text \}$  vloží další údaj do slovníku výjimek pro hesla v rejstříku. Přesněji: vloží  $\langle heslo \rangle$  do `\iispeclist` a definuje sekvenci  $\langle \langle heslo \rangle$  jako  $\langle text \rangle$ .

```
599: \def\iis #1 #2{\bgroup \def~{ }%
600:   \global\expandafter\addto\expandafter\iispeclist\csname,#1\endcsname
601:   \global\sdef\expandafter\string\csname,#1\endcsname}{#2}%
602:   \egroup \ignorespaces
603: }
604: \def\iispeclist{}
```

opmac.tex

Makro „print index item“ `\printii`  $\langle heslo \rangle$  & vytiskne jeden údaj do rejstříku. Makro projde prostřednictvím `\printiiA` jednotlivá podřada oddělená lomítkem a přepíše je do rejstříku odděleny mezerou. Přitom kontroluje, zda se podřada rovnají odpovídajícím podřadům z předchozího hesla, které je uloženo v `\previi`. Toto porovnání je protaženo krz `\meaning`, protože nechceme porovnávat kategorie, ale jen stringy. Pokud se stringy rovnají, místo podřada se vloží `\iiemdash`, což je pomlka. Na konci činnosti se nastaví `\previi` na `\currii` (nové slovo se pro další zpracování stává předchozím) a vytiskne se seznam stránek. Makrem `\everyii` (implicitně je prázdné) dovolíme uživateli vstoupit do procesu tisku hesla. Může například psát `\def\everyii{\indent}`, pokud chce.

```
606: \def\printii #1&{\gdef\currii{#1}\noindent\everyii
607:   \hskip-\iindent \ignorespaces\printiiA#1//}
608: \def\printiiA #1/{\if~#1~\let\previi=\currii \else
609:   \expandafter\scanprevii\previi/&\def\tmpb{#1}\edef\tmpb{\meaning\tmpb}%
610:   \ifx\tmpa\tmpb \iiemdash \else#1 \gdef\previi{ }\fi
611:   \expandafter\printiiA\fi
612: }
613: \def\iiemdash{\kern.1em---\space}
614: \def\everyii{}
```

opmac.tex

Makro `\makeindex` nastavuje na řádce 583 lokálně parametry sazby odstavce v rejstříku. Vlevo budeme mít `\leftskip` rovný `\iindent`, ale první řádek posuneme o  $-\iindent$  (viz řádek kódu 607) takže první řádek je vystrčen doleva. Vpravo máme pružnou mezeru, aby se seznam čísel stran mohl rozumně lámat, když je moc dlouhý.

Pomocné makro `\scanprevii`  $\langle expanded-previi \rangle$  & se podívá do `\previi`, odloupne z něj úsek před prvním lomítkem a tento úsek definuje jako `\tmpa`.

```
616: \def\scanprevii#1/#2&{\def\previi{#2}\def\tmpa{#1}\edef\tmpa{\meaning\tmpa}}
```

opmac.tex

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

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

opmac.tex

### 3.13 Abecední řazení rejstříku

Nejprve se zaměříme na vytvoření makra `\isAleB`  $\langle heslo1 \rangle \langle heslo2 \rangle$ , které rozhodne, zda je  $\langle heslo1 \rangle$  řazeno před  $\langle heslo2 \rangle$  nebo ne. Výsledek zkoumání můžeme prověřit pomocí `\ifAleB`.

Pro porovnání dvou údajů vyžaduje norma dva průchody. V prvním (primárním 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 háčkováná před stříškováná před kroužkováná a dále s nejnižší prioritou malá písmena před velká.

Nejprve připravíme data pro porovnávací algoritmus.

```
621: \def\sortingdata{%
622:   /, { } , - , & , @ , %
623:   aA \ " a \ " A \ ' a \ ' A , %
624:   bB , %
625:   cC , %
626:   \v c \v C , %
627:   dD \v d \v D , %
628:   eE \ ' e \ ' E \v e \v E , %
```

opmac.tex

---

`\iis`: 24–25    `\iispeclist`: 24–25    `\printii`: 24–25    `\printiiA`: 25    `\previi`: 25  
`\iiemdash`: 25    `\currii`: 25    `\everyii`: 25    `\scanprevii`: 25

```

629: fF,%
630: gG,%
631: hH,%
632: ^^T^^U^^V,% ch Ch CH
633: iI\'i\'I,%
634: jJ,%
635: kK,%
636: lL\'l\'L\'v l\'v L,%
637: mM,%
638: nN\'n n\'n N,%
639: oO\'o\'O\'o\'O\'o\'O\'o\'O,%
640: pP,%
641: qQ,%
642: rR\'r\'R,%
643: \'v r\'v R,%
644: sS,%
645: \'v s\'v S,%
646: tT\'v t\'v T,%
647: uU\'u\'U\'u\'U\'u\'r u\'r U,%
648: vV,%
649: wW,%
650: xX,%
651: yY\'y\'Y,%
652: zZ,%
653: \'v z\'v Z,%
654: 0,1,2,3,4,5,6,7,8,9,\' .%
655: }
656: \def\setignoredchars{\setlccodes ,.;?!.:.'".|.(.)[.].<.>.=+.{|}}
657: \def\specsoringdatacs {ch:^^T Ch:^^U CH:^^V}
658: \def\specsoringdatask {ch:^^T Ch:^^U CH:^^V} % DZ etc. are sorted normally

```

Mezi jednotlivými čárkami v makru `\soringdata` jsou skupiny znaků, které se z hlediska prvního průchodu řadicím algoritmem nerozlišují. Jednotlivé znaky v `\soringdata` se rozliší při případném druhém průchodu. Řazení znaků v `\soringdata` odpovídá požadovanému abecednímu řazení.

Dále je makrem `\setignoredchars` vyjmenován seznam znaků, které se při řazení zcela ignorují (jakoby tam vůbec nebyly). Typicky jde o interpunkci. Makro nastaví všem těmto znakům pomocí `\setlccodes` kód tečky a tato tečka se posléze z porovnávaného textu odstraní. Seznam znaků je oddělen tečkou a ukončen dvojicí `{|}`. Seznam ignorovaných znaků odpovídá pravidlům českého řazení. Norma doporučuje sice pro případ, kdy se hesla neliší jinak než těmito znaky, nasadit další průchod řazení, ale pro jednoduchost a velkou výjimečnost takové situace toto v OPmac implementováno není.

Makra `\specsoringdatacs` a `\specsoringdatask` deklarují náhrady před použitím řadicího algoritmu. Jednotivá náhrada je deklarována jako `<string1>:<string2>` a je od další deklarace náhrady oddělena mezerou. Tímto způsobem jsou implementovány spřežky `ch`, `Ch` a `CH`, které se nahrazují znaky `^^T`, `^^U` a `^^V`. Ty vystupují v řadicím algoritmu jako jediný znak, a mají své místo v makru `\soringdata`. Slovenština má sice další spřežky `dz`, `Dz`, `DZ`, `dž`, `Dž`, `DŽ`, ale ty jsou řazeny těsně za `D`, takže jsou řazeny správně i za situace, kdy nejsou nahrazeny jediným znakem. Nicméně, pokud by někdo chtěl tyto spřežky (například pro ošetřování výjimek) použít jako samostatné znaky, může si definovat své makro `\soringdata`, které by obsahovalo

```

...
dD\'v d\'v D,%
^^N^^O^^P,% dz Dz DZ
^^Q^^R^^S,% dž Dž DŽ
eE\'e\'E\'v e\'v E,%
...

```

a dále definuje

```

\def\specsoringdatask {ch:^^T Ch:^^U CH:^^V
dz:^^N Dz:^^O DZ:^^P d\'v z:^^Q D\'v z:^^R D\'v Z:^^S}

```

`\soringdata`: 25–28    `\setignoredchars`: 26–28    `\specsoringdatacs`: 26  
`\specsoringdatask`: 26



Makra pro spřežky mají název ve tvaru `\specssortingdata`(*kód-jazyka*). Použije se makro odpovídající jazyku podle nastaveného dělení slov. Není-li makro pro použitý jazyk definováno, žádné náhrady se neprovedou. Je třeba upozornit (například uživatele maďarštiny), pokud by se v těchto spřežkách chtěli rozšoupnout, vyhněte se znakům  $\text{^I}$  a  $\text{^M}$ , kterým plain $\text{T}_{\text{E}}\text{X}$ , resp. ini $\text{T}_{\text{E}}\text{X}$ , nastavuje speciální kategorie.

Implementaci řadičích algoritmu zahájíme makrem `\setprimarysorting`, které se spustí jednou při sestavení rejstříku, přečte výše uvedená data a připraví odpovídající datové struktury pro první průchod řadičích algoritmu. Hlavní činností tohoto makra je, že připraví `\lccode` znaků vyjmenovaných v `\sortingdata` podle jejich vzestupného pořadí, přitom znakům v jedné skupině (oddělené čárkou) přiřadí stejné `\lccode`. Text před řazením pak budeme konvertovat použitím `\lowercase`.

opmac.tex

```

660: \def\setprimarysorting {%
661:   \isdefined{sortingdata\csname lan:\the\language\endcsname}\iftrue
662:     \expandafter \let\expandafter\sortingdata
663:       \csname sortingdata\csname lan:\the\language\endcsname\endcsname
664:     \xdef\sortingmessage{using \string\sortingdata\csname lan:\the\language\endcsname}%
665:   \else
666:     \xdef\sortingmessage{using internal \string\sortingdata}%
667:     \ifx\r\undefined
668:       \opwarning{noexpand\csaccents is unused, falling back to ASCII sorting}%
669:       \global\let\asciisorting=t%
670:     \fi
671:     \ifx\asciisorting\undefined
672:       \xdef\sortingdata{\sortingdata}% expand sorting data now
673:       \isdefined{specssortingdata\csname lan:\the\language\endcsname}\iftrue
674:         \xdef\specssortingdata{\csname specssortingdata\csname lan:\the\language\endcsname
675:           \endcsname\space}%
676:         \expandafter\setprimarysortingA \meaning\specssortingdata\relax
677:       \else \gdef\specssortingdata{}\fi
678:     \else
679:       \gdef\sortingdata{.}\gdef\specssortingdata{}\gdef\sortingmessage{ASCII}%
680:     \fi
681:     \def\act##1{\ifx##1.\else
682:       \ifx##1,\advance\tmpnum by1
683:       \else \lccode'##1=\tmpnum \fi
684:       \expandafter \act \fi}%
685:     \tmpnum=60 \expandafter \act\sortingdata \setignoredchars
686:   }
687: \def\setprimarysortingA#1->#2\relax{\gdef\specssortingdata{#2}}
688: \def\sortingmessage{ASCII default}

```

Vidíme, že makro `\setprimarysorting` nejprve expanduje `\sortingdata`, aby se realizovaly znaky typu `\v_l c` podle nastaveného kódování. Dělá to jen tehdy, když je definováno makro `\r`, tj. uvedené sekvence pro akcenty expandují na správné kódy. Rovněž pomocí `\let\asciisorting=t` je možné zabránit použití `\sortingdata` a řadičích algoritmus řadí podle ASCII.

Dále `\setprimarysorting` připraví makro `\specssortingdata` (bez přípony jazyka) z makra `\specssortingdata`(*aktuální-jazyk*). Nejprve je expanduje a pak pomocí triku s `\meaning` za spolupráce s makrem `\setprimarysortingA` převede všechny znaky v makru na kategorii 12, protože toto budeme pro řadičích algoritmus potřebovat.

Konečně se v makru `\setprimarysorting` připraví (za použití opakovaného volání `\act`) `\lccode` všech znaků zmíněných v `\sortingdata`. Povšimneme si, že pro první průchod dostanou stejný `\lccode` všechny znaky ve skupině mezi čárkami. Je to tím, že v makru `\setprimarysorting` se zvedá `\tmpnum` jen v místě čárky. Nejnižší hodnotu má mezera vyznačená v `\sortingdata` pomocí `{_}`. Tím je zaručeno, že kratší slovo je řazeno dříve 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 nepřčetlo makro s neseparovaným parametrem. Ovšem my budeme chtít mezeru přečíst. Makrem `\setignoredchars` se zcela nakonec nastaví ignorovaným znakům `\lccode` tečky.

Makro `\sortingmessage` ukládá informaci o použitém `\sortingdata`, což bude později vypsáno na terminál makrem `\dosorting`.

---

```

\setprimarysorting: 24, 27-28   \asciisorting: 27   \specssortingdata: 27-28
\setprimarysortingA: 27   \sortingmessage: 27, 29

```

Makro `\setsecondarysorting` se volá opakovaně a příležitostně pro případy, kdy jsou hesla z hlediska primárního řazení totožná. Nastaví jinak `\lccode` znaků. Tentokrát mají všechny znaky ze `\sortingdata` rozdílný `\lccode`, ve vzestupném pořadí.

```
690: \def\setsecondarysorting {\def\act##1{\ifx##1.\else
691:   \ifx##1,\else \advance\tmpnum by1 \lccode'##1=\tmpnum \fi
692:   \expandafter \act \fi}%
693:   \tmpnum=60 \expandafter \act\sortingdata \setignoredchars
694: }
```

opmac.tex

Makro `\preparesorting` se volá (s nastavenými parametry podle `\setprimarysorting`) pro každé heslo jednou. Heslo je uloženo v názvu kontrolní sekvence, která je parametrem makra `\preparesorting`. Data pro primární řazení jsou už připravena na řádcích 570 až 578 v makru `\makeindex`. V případech, kdy jsou dvě hesla shodná z hlediska primárního řazení (to nastane asi velmi výjimečně), je pro danou dvojici hesel znovu zavoláno makro `\preparesorting`, tentokrát s přednastavenými daty podle `\setsecondarysorting`. Makro `\preparesorting` má za úkol uložit výsledek své konverze do `\tmpb`.

```
696: \def\preparesorting#1{\expandafter\preparesortingA\string#1&}
697: \gdef\preparesortingA#1#2#3&{\xdef\tmpb{#3}%
698:   \expandafter\preparesortingB\specsoringdata.:{ }
699:   \lowercase\expandafter{\expandafter\gdef\expandafter\tmpb\expandafter{\tmpb}}%
700:   \replacestrings{.}{ }%
701: }
702: \def\preparesortingB#1#2:#3 {\ifx.#1\else \replacestrings{#1#2}{#3}\expandafter\preparesortingB\fi}
```

opmac.tex

Všimneme si, že `\preparesorting` vykonává jádro své činnosti v `\preparesortingA`, které přebere text hesla extrahovaný do parametru #3. Toto makro pomocí `\preparesortingB` opakovaně volá `\replacestrings`, aby nahradilo spřežky odpovídajícími náhradami. Dále pomocí `\lowercase` provede konverzi a konečně pomocí `\replacestrings{.}{ }` odstraní z hesla nejen tečky, ale i znaky vyjmenované v makru `\setignoredchars`.

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

```
704: \newif \ifAleB
```

opmac.tex

Makro `\isAleB` `\,<heslo1> \,<heslo2>` spustí `\testAleB` `<zkonvertované-heslo1>&\relax <zkonvertované-heslo2>&\relax \,<heslo1> \,<heslo2>`.

```
706: \def\isAleB #1#2{%
707:   \edef\tmp{\firstdata#1\empty\relax\firstdata#2\empty\relax \noexpand#1\noexpand#2}%
708:   \expandafter \testAleB \tmp
709: }
```

opmac.tex

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.

```
710: \def\testAleB #1#2\relax #3#4\relax #5#6{%
711:   \if #1#3\if #1&\testAleBsecondary #5#6%
712:     \else \testAleB #2\relax #4\relax #5#6%
713:     \fi
714:   \else \ifnum '#1<'#3 \AleBtrue \else \AleBfalse \fi
715:   \fi
716: }
```

opmac.tex

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`.

---

```
\setsecondarysorting: 28–29   \preparesorting: 24, 28–29   \preparesortingA: 28
\preparesortingB: 28       \ifAleB: 25, 28–29   \isAleB: 25, 28–29   \testAleB: 28
\testAleBsecondary: 28–29   \testAleBsecondaryX: 29
```

```

717: \def\testAleBsecondary#1#2{%
718:   \bgroup
719:   \setsecondarysorting
720:   \prepaesorting#1\let\tmpa=\tmpb \prepaesorting#2%
721:   \edef\tmp{\tmpa0\relax\tmpb1\relax}%
722:   \expandafter\testAleBsecondaryX \tmp
723:   \egroup
724: }
725: \def\testAleBsecondaryX #1#2\relax #3#4\relax {%
726:   \if #1#3\testAleBsecondaryX #2\relax #4\relax
727:   \else \ifnum '#1<'#3 \global\AleBtrue \else \global\AleBfalse \fi
728:   \fi
729: }

```

opmac.tex

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`.

```

730: \def\dosorting{%
731:   \message{Opmac: Sorting index (\sortingmessage)...}%
732:   \def\act##1{\ifx##1\relax\else \addto\iilist{##1,}%
733:     \expandafter\act\fi}%
734:   \edef\iilist{\expandafter}\expandafter\act \iilist\relax
735:   \edef\iilist{\expandafter}\expandafter\mergesort \iilist \end,\end
736: }

```

opmac.tex

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 `bdeikmz`. 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 748) 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. Makro `\sortreturn` ukončí činnost až po koncové `\relax` a dále vykoná svůj parametr.

```

738: \def\mergesort #1#2,#3{% by Miroslav Olsak
739:   \ifx,#1%           % prazdna-skupina,neco, (#2=neco #3=pokracovani)
740:   \addto\iilist{#2,}% % dvojice skupin vyresena
741:   \sortreturn{\fif\mergesort#3}% % \mergesort pokracovani
742:   \fi
743:   \ifx,#3%           % neco,prazna-skupina, (#1#2=neco #3=,)
744:   \addto\iilist{#1#2,}% % dvojice skupin vyresena
745:   \sortreturn{\fif\mergesort}% % \mergesort dalsi
746:   \fi
747:   \ifx\end#3%        % neco,konec (#1#2=neco)
748:   \ifx\empty\iilist  % neco=kompletni setrideny seznam
749:   \def\iilist{#1#2}%
750:   \sortreturn{\fif\fif\gobbletoend}% % koncim
751:   \else               % neco=posledni skupina nebo \end
752:   \sortreturn{\fif\fif % spojim \indexbuffer+necoa cele znova
753:     \edef\iilist{\expandafter}\expandafter\mergesort\iilist#1#2,#3}%
754:   \fi\fi             % zatriduji: p1+neco1,p2+neco2, (#1#2=p1+neco1 #3=p2)
755:   \isAleB #1#3\ifAleB % p1<p2

```

opmac.tex

---

`\dosorting`: 24, 27, 29    `\mergesort`: 29–30    `\gobbletoend`: 29–30    `\sortreturn`: 29–30

```

756: \addto\iilist{#1}% % p1 do bufferu
757: \sortreturn{\fif\mergesort#2,#3}% % \mergesort neco1,p2+neco2,
758: \else % p1>p2
759: \addto\iilist{#3}% % p2 do bufferu
760: \sortreturn{\fif\mergesort#1#2,}% % \mergesort p1+neco1,neco2,
761: \fi
762: \relax % zarazka, na ktere se zastavi \sortreturn
763: }
764: \def\sortreturn#1#2\fi\relax{#1} \def\fif{\fi}
765: \def\gobbletoend #1\end{}

```

Jádro `\mergesort` vidíme na řádcích 755 až 760. 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 739 až 745 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 zářezku `\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 `\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 obejít problém „dimension too large“.

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

opmac.tex

```

769: \newcount\mullines
770: \def\corrsiz #1{%% #1 := #1 + \splittopskip - \topskip
771: \advance #1 by \splittopskip \advance #1 by-\topskip
772: }
773: \def\begmulti #1 {\par\bgroup\wipepar\multiskip\penalty0 \def\Ncols{#1}
774: \setbox6=\vbox\bgroup\penalty0
775: %% \hsiz := Sirka sloupce = (\hsiz+colsep) / n - \colsep
776: \advance\hsiz by\colsep
777: \divide\hsiz by\Ncols \advance\hsiz by-\colsep
778: \mullines=0
779: \def\par{\ifhmode\endgraf\global\advance\mullines by\prevgraf\fi}%
780: }
781: \def\endmulti{\vskip-\prevdepth\vfil
782: \expandafter\egroup\expandafter\baselineskip\the\baselineskip\relax
783: \dimen0=.8\maxdimen \tmpnum=\dimen0 \divide\tmpnum by\baselineskip
784: \splittopskip=\baselineskip
785: \setbox1=\vsplit6 toOpt
786: %% \dimen1 := the free space on the page
787: \ifdim\pagegoal=\maxdimen \dimen1=\vsize \corrsiz{\dimen1}
788: \else \dimen1=\pagegoal \advance\dimen1 by-\pagetotal \fi
789: \ifdim \dimen1<2\baselineskip
790: \vfil\break \dimen1=\vsize \corrsiz{\dimen1} \fi
791: \ifnum\mullines<\tmpnum \dimen0=\ht6 \else \dimen0=.8\maxdimen \fi
792: \divide\dimen0 by\Ncols \relax
793: %% split the material to more pages?
794: \ifdim \dimen0>\dimen1 \splitpart
795: \else \balancecolumns \fi % only balancing
796: \multiskip\egroup

```

`\begmulti`: 7, 30    `\endmulti`: 7, 30    `\corrsiz`: 30–31    `\makecolumns`: 31    `\splitpart`: 30–31

```

797: }
798: \def\makecolumns{\bgroup % full page, destination height: \dimen1
799:   \vbadness=20000 \setbox1=\hbox{}\tmpnum=0
800:   \loop \ifnum\Ncols>\tmpnum
801:     \advance\tmpnum by1
802:     \setbox1=\hbox{\unhbox1 \vsplit6 to\dimen1 \hss}
803:   \repeat
804:   \hbox{\nobreak\vskip-\splittopskip \nointerlineskip
805:     \line{\unhbox1\unskip}
806:     \dimen0=\dimen1 \divide\dimen0 by\baselineskip \multiply\dimen0 by\Ncols
807:     \global\advance\mullines by-\dimen0
808:   \egroup
809: }
810: \def\splitpart{%
811:   \makecolumns % full page
812:   \vskip 0pt plus 1fil minus\baselineskip \break
813:   \ifnum\mullines<\tmpnum \dimen0=\ht6 \else \dimen0=.8\maxdimen \fi
814:   \divide\dimen0 by\Ncols \relax
815:   \ifx\balancecolumns\flushcolumns \advance\dimen0 by-.5\vsizel \fi
816:   \dimen1=\vsizel \corrsize{\dimen1}\dimen2=\dimen1
817:   \advance\dimen2 by-\Ncols\baselineskip
818:   %% split the material to more pages?
819:   \ifvoid6 \else
820:     \ifdim \dimen0>\dimen2 \expandafter\expandafter\expandafter \splitpart
821:     \else \balancecolumns % last balancing
822:   \fi \fi
823: }

```

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 zůstat 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 zazálohují materiál z boxu 6 do boxu 7 a jme se zkusit rozlomit box 6 na sloupce s výškou `\dimen0`. Pokud ale po rozlomení není výchozí box 6 zcela prázdný, makro zvětší krapánek (o  $0,2\text{\baselineskip}$ ) 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

```

824: \def\balancecolumns{\bgroup \setbox7=\copy6 % destination height: \dimen0
825:   \ifdim\dimen0>\baselineskip \else \dimen0=\baselineskip \fi
826:   \vbadness=20000
827:   \def\tmp{%
828:     \setbox1=\hbox{}\tmpnum=0
829:     \loop \ifnum\Ncols>\tmpnum
830:       \advance\tmpnum by1
831:       \setbox1=\hbox{\unhbox1
832:         \ifvoid6 \hbox to\wd6{\hss}\else \vsplit6 to\dimen0 \fi\hss}
833:     \repeat
834:     \ifvoid6 \else
835:       \advance \dimen0 by.2\baselineskip
836:       \setbox6=\copy7
837:       \expandafter \tmp \fi}\tmp
838:     \hbox{\nobreak\vskip-\splittopskip \nointerlineskip
839:       \hbox to\hsizel\unhbox1\unskip}%
840:   \egroup
841: }

```

Když je sazba plněna do boxu 6, může jí být tak moc, že se nedá změřit jeho výška pomocí `\dimen0=\ht6`. Box samotný sice může být vyšší než pět metrů, ale `\dimen0` nikoli: objeví se chyba „dimension too large“. Z toho důvodu je v makrech zavedena proměnná `\mullines`, která pomocí předefinovaného `\par` (na řádce 779) počítá počet řádků sazby. Je-li `\mullines` větší než `\tmpnum` (což při daném `\baselineskip` odpovídá  $0,8\text{\maxdimen}$ ), makro pracuje, jakoby výška boxu 6 byla  $0,8\text{\maxdimen}$ , tedy rozběhne se `\splitpart` a `\makecolumns`. Přitom makro `\makecolumns` snižuje hodnotu `\mullines`



o počet vytištěných řádků, takže příště už může být `\mullines` menší než `\tmpnum`. K tomu určitě na několika posledních stránkách dojde, takže nakonec `\balancecolumns` pracuje s přesnou výškou boxu 6.

### 3.15 Barvy

Až po verzi OPmac Nov. 2014 byly barvy implementovány pomocí `\pdfliteral` za použití maker, která sama implementují `\colorstack` pomocí REF souboru. V prosinci 2014 jsem se rozhodl tento kód z OPmac odstranit a využít přímo primitivní `\pdfcolorstack` (v pdfTeXu od verze 1.40). OPmac se tak zbavil asi 30 řádků poměrně komplikovaného kódu a ušetřil množství zápisů do REF souboru. Tyto změny jsou v souladu s myšlenkou „v jednoduchosti je síla“. Uvedené rozhodnutí není zcela zpětně kompatibilní, protože opouští možnost samostatného nastavení barvy pro tenké linky a pro text. Domnívám se, že to nevadí, protože pokud uživatel potřebuje elementární manipulaci s barvami, použije sám přímo `\pdfliteral`. Makra `\setcmykcolor` se nyní opírají o `\pdfcolorstack` a nastavují oba typy barev společně. Bylo sice možné inicializovat dva zásobníky barev (pro linky a pro text), ale to by fungovalo jen v pdfTeXu. Nikoli v XeTeXu. Cílem ovšem je, aby se barvy v pdfTeXu a XeTeXu chovaly pokud možno stejně. Navíc, když uživatel napíše barevně odmocninu, musí mít oba typy barev zapnuty současně na stejnou hodnotu, jinak má věčko odmocniny v jiné barvě než vodorovnou čáru. Je tedy i pro uživatele jednodušší tyto dva typy barev nerozlišovat.

Makro `\localcolor` (na rozdíl od předchozí verze) pouze nastavuje `\localcolortrue`. Podle `\localcolortrue` resp. `\localcolorfalse` se bude větvit činnost přepínačů barev, které ukládají aktuální barvu do zásobníku. To je tedy druhá mírná odlišnost od starší verze OPmac Nov. 2014, kdy makro `\localcolor` přímo ukládalo aktuální barvu do zásobníku barev, zatímco přepínače barev toto neřešily. Původní typické použití makra `\localcolor` není ve sporu s jeho novým významem.

opmac.tex

```
845: \newif\iflocalcolor \localcolorfalse
846: \let\localcolor=\localcolortrue
```

Makro `\longlocalcolor` dříve umožňovalo přechod barvy na další stránku, nyní je tato vlastnost přímo řešena díky `\pdfcolorstack`, takže netřeba rozlišovat mezi `\localcolor` a `\longlocalcolor`. Makro `\linecolor` nyní nedělá nic, protože nerozlišujeme mezi barvou linek a barvou textu. V původní verzi bylo prefixem pro barvy linek.

opmac.tex

```
848: % for backward compatibility:
849: \let\longlocalcolor=\localcolor \let\locpgcolor=\relax
850: \def\linecolor#1{}
```

Připravíme barevná makra `\Blue`, `\Red`, `\Brown`, `\Green`, `\Yellow`, `\Cyan`, `\Magenta`, `\White`, `\Grey`, `\LightGrey`, `\Black`. Uživatel si může definovat další.

opmac.tex

```
852: \def\Blue{\setcmykcolor{1 1 0 0}}
853: \def\Red{\setcmykcolor{0 1 1 0}}
854: \def\Brown{\setcmykcolor{0 0.67 0.67 0.5}}
855: \def\Green{\setcmykcolor{1 0 1 0}}
856: \def\Yellow{\setcmykcolor{0 0 1 0}}
857: \def\Cyan{\setcmykcolor{1 0 0 0}}
858: \def\Magenta{\setcmykcolor{0 1 0 0}}
859: \def\White{\setcmykcolor{0 0 0 0}}
860: \def\Grey{\setcmykcolor{0 0 0 0.5}}
861: \def\LightGrey{\setcmykcolor{0 0 0 0.2}}
862: \def\Black{\setcmykcolor{0 0 0 1}}
```

Makro `\setcmykcolor`  $\langle CMYK\text{-barva} \rangle$  nastaví požadovanou barvu. Nejprve přepne makro `\ensureblack` do aktivního stavu. V tomto stavu makro setrvá právě tehdy, když je v dokumentu použit aspoň jednou přepínač barvy. Dále makro `\setcmykcolor` nastaví při `\localcolorfalse` barvu přímo a při `\localcolortrue` barvu vloží do zásobníku a pomocí `\aftergroup` zajistí návrat k původní hodnotě. Navíc nastaví na odpovídající hodnotu makro `\currentcolor`.

---

```
\localcolor: 32–33, 35 \localcolortrue: 32 \localcolorfalse: 32 \longlocalcolor: 32–33
\linecolor: 32 \Blue: 32 \Red: 32 \Brown: 32 \Green: 32 \Yellow: 32 \Cyan: 32
\Magenta: 32 \White: 32 \Grey: 32 \LightGrey: 32–33 \Black: 32 \setcmykcolor: 32–33
\currentcolor: 33
```



```

864: \def\setcmykcolor#1{\global\let\ensureblacko=\ensureblackoA
865:   \iflocalcolor \edef\currentcolor{#1}\colorstackpush\currentcolor \aftergroup\colorstackpop
866:   \else          \xdef\currentcolor{#1}\colorstackset\currentcolor \fi
867: }

```

Makro `\currentcolor` je nastaveno na výchozí hodnotu `\pdfblackcolor`

```

868: \def\pdfblackcolor{0 0 0 1}
869: \xdef\currentcolor{\pdfblackcolor}

```

Makro `\ensureblacko`  $\langle sazba \rangle$  je použito pro sazbu záhlaví a zápatí ve výstupní rutině v makru `\opmacoutput`. Implicitně se `\ensureblacko`  $\langle sazba \rangle$  chová stejně jako samotná  $\langle sazba \rangle$ , ale po použití přepínače barvy `\setcmykcolor` začne fungovat jako `\ensureblackoA`, což zajistí bravu  $\langle sazby \rangle$  v černém. Je to provedeno tak, že je na začátku  $\langle sazby \rangle$  alokována nová úroveň zásobníku barev s výchozí černou barvou a na konci  $\langle sazby \rangle$  je tato úroveň zásobníku ukončena.

```

870: \def\ensureblacko#1{#1}
871: \def\ensureblackoA#1{\colorstackpush\pdfblackcolor #1\colorstackpop}

```

Makra `\colorstackpush`  $\langle CMYK\text{-barva} \rangle$  a `\colorstackpop` implementují práci se zásobníkem barev za použití odpovídajících T<sub>E</sub>Xových primitivů. Není-li přítomen pdfT<sub>E</sub>X ve verzi aspoň 1.40, je barva nastavena pomocí `\pdfliteral` (což v komplikovanějších případech při přechodu na další stránky nefunguje správně), jinak je použit `\pdfcolorstack`, který je inicializován pomocí `\pdfcolorstackinit`. Pověšněte si, že se současně pracuje s barvou textu  $\langle c \rangle \langle m \rangle \langle y \rangle \langle k \rangle$  i s barvou tenkých linek  $\langle c \rangle \langle m \rangle \langle y \rangle \langle k \rangle$ . Konečně makro `\colorstackset`  $\langle CMYK\text{-barva} \rangle$  nastavuje barvu přímo s umístěním této bravu na vrchol zásobníku místo bravu předchozí.

```

873: \ifx\pdfcolorstackinit\undefined
874:   \def\colorstackpush#1{\pdfliteral{#1 k #1 K}}
875:   \def\colorstackpop{\colorstackpush\currentcolor}
876:   \let\colorstackset=\colorstackpush
877: \else
878:   \mathchardef\colorstackcnt=\pdfcolorstackinit page {0 g 0 G}
879:   \def\colorstackpush#1{\pdfcolorstack\colorstackcnt push{#1 k #1 K}}
880:   \def\colorstackpop{\pdfcolorstack\colorstackcnt pop}
881:   \def\colorstackset#1{\pdfcolorstack\colorstackcnt set{#1 k #1 K}}
882: \fi

```

Makra `\colorstackpush`, `\colorstackpop` a `\colorstackset` jsou odpovídajícím způsobem předefinována v souboru `opmac-xetex.tex`, aby bylo možné pracovat s barvami i v XeT<sub>E</sub>Xu.

Přepínače barev stejně jako makra `\localcolor` nebo `\longlocalcolor` se mohou vyskytnout v nadpise. Takže je potřeba je zabezpečit proti rozsypání.

```

884: \addprotect\setcmykcolor \addprotect\localcolor \addprotect\longlocalcolor

```

Není-li použit pdfT<sub>E</sub>X, některá makra pro barvu deaktivujeme:

```

886: \ifpdf\else
887:   \def\setcmykcolor#1{} \def\pdfliteral#1{}
888: \fi

```

Makro `\draft` vloží do `\prepghook` 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.

```

890: \def\draft{\addto\prepghook{\draftbox{\tenbf DRAFT}\nointerlineskip}}

```

V makru `\draftbox`  $\langle text \rangle$  je  $\langle text \rangle$  otočen o 55 stupňů, zvětšen desetkrát a vtištěn v barvě `\LightGrey`. K tomu jsou využity PDF transformace souřadnic.

```

891: \def\draftbox#1{\vbox to0pt{\setbox0=\hbox{\typosize[10/]{#1}}%
892:   \kern.5\vszise \kern4\wd0 \hbox to0pt{\kern.5\hszise \kern-2.5\wd0
893:   \pdfsave \pdfrotate{55}\pdfscale{10}{10}%
894:   \hbox to0pt{\localcolor\LightGrey \box0\hss}%
895:   \pdfrestore

```

---

`\pdfblackcolor`: 33    `\ensureblacko`: 32–33, 54–55    `\ensureblackoA`: 33    `\colorstackpush`: 33  
`\colorstackpop`: 33    `\colorstackset`: 33    `\draft`: 33–34    `\draftbox`: 33

```
896: \hss}\vss}}
```

Když není použit pdfTeX, barvy nefungují, takže makro `\draft` deaktivujeme.

opmac.tex

```
898: \ifpdf\else
899: \def\draft{\opwarning{\string\draft: Grey color is possible in pdfTeX only}}
900: \fi
```

### 3.16 Klikací odkazy

Makro `\deactive` [ $\langle typ \rangle : \langle lejblík \rangle$ ] založí cíl odkazu jen tehdy, když je  $\langle lejblík \rangle$  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 prostě vloží `\destbox`. Makro `\destbox` [ $\langle typ \rangle : \langle lejblík \rangle$ ] vytvoří box nulové výšky a z něj vystrčí nahoru cíl klikacího odkazu vzdálený od účarí o `\destheight`. Interně použije pdfTeXový primitiv `\pdfdest` s parametrem `xyz`, což charakterizuje obvyklou možnost chování PDF prohlížeče při odsokou na cíl. Podrobněji viz manuál k pdfTeXu. PDF prohlížeče většinou lícuji horní hranu okna přesně s místem cíle, je tedy potřeba cíl umístit poněkud výše, abychom viděli i odkazovaný text. K tomu právě slouží obsah makra `\destheight`.

opmac.tex

```
904: \def\destheight{1.4em}
905: \def\deactive[#1:#2]{\if$#2$\else\ifvmode
906: \tmpdim=\prevdepth \prevdepth=-1000pt
907: \destbox[#1:#2]\prevdepth=\tmpdim
908: \else \destbox[#1:#2]%
909: \fi\fi
910: }
911: \def\destbox[#1]{\vbox to0pt{\kern-\destheight \pdfdest name{#1} xyz\vss}}
912: \def\dest[#1]{}
```

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

Někdy je účelné v režimu „draft“ dokumentu tisknout v místě cílů odkazů jména lejblíků, aby autor viděl, jaké lejblíky použil a lépe se mu dílo modifikovalo. Stačí předefinovat pro tento režim makro `\destbox` třeba takto:

```
\def\destbox[#1#2:#3]{\vbox to0pt{\kern-\destheight
\pdfdest name{#1#2:#3} xyz\relax
\if#1r\llap{\labelfont[\detokenize\expandafter{#3}]\vss \else
\if#1c\vss\llap{\labelfont[\detokenize\expandafter{\tmpb}] } \kern-\prevdepth
\else \vss \fi\fi}}
\def\labelfont{\localcolor\Red\tt\thefontsize[10]}
```

Při tomto řešení budou lejblíky z `\label` tištěny nahoru v místě cíle zatímco lejblíky z `\bib` a `\bibitem` budou tištěny vedle položky se seznamem literatury. V obou případech budou lejblíky zelené a díky `\llap` neovlivní polohu ostatní sazby.

Klikací text vytvoří makro `\linkactive` [ $\langle typ \rangle : \langle lejblík \rangle \{ \langle barva \rangle \} \{ \langle text \rangle \}$ ]. Makro používá pdfTeXový primitiv `\pdfstartlink`, ve kterém je vymezena výška a hloubka aktivní plochy. Nakonec přepne na požadovanou  $\langle barvu \rangle$  (pokud není černá), vytiskne aktivní  $\langle text \rangle$  a přepne zpět na černou barvu. PdfTeXový primitiv `\pdfendlink` ukončí sazbu aktivního textu. K použití je připraveno makro `\link`, které dostane hodnotu `\linkactive` při `\hyperlinks`, jinak pouze přepíše svůj argument.

opmac.tex

```
914: \def\linkactive[#1:#2]#3#4{\leavevmode\pdfstartlink height.9em depth.3em
915: \pdfborder{#1} goto name{#1:#2}\relax {#3#4}\pdfendlink
916: }
917: \def\link[#1]#2#3{\leavevmode{#3}}
```

Makro `\urllink` [ $\langle typ \rangle : \langle lejblík \rangle \{ \langle text \rangle \}$ ] pracuje analogicky jako `\link`. Jen navíc přidává některé atributy do PDF výstupu a pracuje s barvou `\urlcolor`. Toto makro vytvoří externí odkaz. Je použito v makru `\url` prostřednictvím makra `\link`.

---

`\deactive`: 34–35    `\destbox`: 34    `\destheight`: 17, 34, 55    `\dest`: 14, 17, 34–35, 51, 53, 55  
`\linkactive`: 34–35    `\link`: 34–35    `\urllink`: 35

```

919: \def\urllink[#1:#2]#3{\let~=\relax \let\=\relax \let\{=\relax \let\}=\relax
920:   \leavevmode\pdfstartlink height.9em depth.3em
921:   \pdfborder[#1]user{/Subtype/Link/A <</Type/Action/S/URI/URI(#2)>>}\relax
922:   {\def~{\nobreak\space}\urlcolor#3}\pdfendlink}%
923: }

```

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

```

924: \def\toclink#1{\toclinkA{#1}}
925: \def\pglink#1{\leavevmode{\pgfolioA{#1}}}
926: \def\citelink#1#2{\leavevmode{#2}}
927: \def\reflink[#1]#2{\leavevmode{#2}}
928: \def\ulink[#1]#2{\leavevmode{#2}}
929: \def\urlcolor{}

```

Ovšem po použití makra `\hyperlinks`  $\langle\text{barva-lok}\rangle\langle\text{barva-url}\rangle$  se uvedená makra `\toclink`, `\pglink`, `\citelink` a `\reflink` probouzejí k životu. Zde je také definováno makro `\urlcolor`.

```

931: \def\hyperlinks#1#2{%
932:   \let\dest=\deactive \let\link=\linkactive
933:   \def\toclink##1{\link[toc:\tocilabel.##1]{\localcolor#1}{\toclinkA{##1}}}%
934:   \def\pglink##1{\link[pg:\pgilabel.##1]{\localcolor#1}{\pgfolioA{##1}}}%
935:   \def\citelink##1##2{\link[cite:##1]{\localcolor#1}{##2}}%
936:   \def\reflink[##1]##2{\link[ref:##1]{\localcolor#1}{##2}}%
937:   \def\ulink[##1]##2{\urllink[url:##1]{##2}}%
938:   \def\urlcolor{\localcolor#2}%

```

Makro `\toclink` čte parametr ve formátu „číslo kapitoly, sekce, kapitoly.sekce atd.“. Makro `\pglink` zase vyžaduje svůj parametr jen jako číslo strany. Když je dokument rozdělen do bloků a v každém je samostatné číslování stran, respektive bloky obsahují samostatné číslování sekcí, je potřeba rozlišit mezi těmito bloky, aby interní odkaz při `\hyperlinks` v dokumentu byl jednoznačný. Proto jsou zavedena (implicitně prázdná) makra `\tocilabel` a `\pgilabel`. Každý blok v dokumentu by pak měl mít svůj vlastní `\tocilabel` a `\pgilabel` o což se musí programátor maker postarat sám.

```

940: \def\tocilabel{} \def\pgilabel{}

```

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 zřídí makro `\pdfborder`  $\langle\text{typ}\rangle$ , které expanduje na `attr /Border[0_0_0]`, pokud není kontrolní sekvence `\langle\text{typ}\rangle border` definována. Jinak expanduje na `arrrt /Border[0_0_0.6]` a `/C` s obsahem podle `\langle\text{typ}\rangle border`.

```

942: \def\pdfborder#1{\if^#1^else \isdefined{#1border}\iftrue
943:   \if^#1border\endcsname^else attr{/C[csname#1border\endcsname] /Border[0 0 .6]}\fi
944:   \else attr{/Border[0 0 0]}\fi\fi
945: }

```

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

```

947: \ifpdf\else
948:   \def\link[#1]#2#3{#3}
949:   \def\urllink[#1]#2{#2}
950:   \def\hyperlinks#1#2{\opwarning{No pdfTeX detected, \noexpand\hyperlinks ignored}}
951: \fi

```

Makro `\url`  $\langle\text{text}\rangle$  se používá k tisku URL. Vytiskne  $\langle\text{text}\rangle$  fontem `\urlfont`, přitom kolem znaků lomítka, tečka a dalších přidává nulovou mezeru s dodatečnou mírnou roztažitelností `\urlskip`. Mezera vpravo od těchto znaků je navíc zlomitelná s penaltou definovanou v makru `\urlbskip`. Dvojitě lomítka `\urlslashtslash` má zlomitelnou mezeru jen na konci. Makro `\|` je lokálně definováno jako

---

```

\toclink: 21, 35   \pglink: 14, 21, 35   \citelink: 35, 50–51   \reflink: 14, 35   \ulink: 34–36
\hyperlinks: 34–36   \urlcolor: 34–36   \tocilabel: 17, 35, 37   \pgilabel: 35, 55
\pdfborder: 34–35   \url: 34, 36, 52   \urlfont: 36   \urlskip: 36   \urlbskip: 36
\urlslashtslash: 36

```

prázdné, ale při `\urlfont` nabývá hodnoty `\urlspecchar`. Takže ve skutečném odkaze se neprojeví, ale při tisku ano. Uživatel si může `\urlspecchar` definovat dle svých představ (například jako `\hf{il}{break}`).

opmac.tex

```

953: \def\url#1{\def\tmpb{#1}%
954:   \replacestrings{/}{\urlskip\urlslashslash\urlbskip}}%
955:   \replacestrings{/}{\urlskip\urlbskip}}%
956:   \replacestrings{.}{\urlskip.\urlbskip}}%
957:   \replacestrings?}{\urlskip?\urlbskip}}%
958:   \replacestrings={}{\urlskip=\urlbskip}}%
959:   \replacestrings~}{\char`\~}}%
960:   \replacestrings_{}{\char`\_}}%
961:   \replacestrings^}{\char`\^}}%
962:   \replacestrings\}{\char`\backslash}}%
963:   \replacestrings\{}{\char`\{}}%
964:   \replacestrings\}{\char`\}}%
965:   \replacestrings&}{\urlbskip\char`\&\urlskip}}%
966:   \def\|{\urllink[#1]{\urlfont\tmpb\|}}%
967: }%
968: \def\urlfont{\tt \let\|=\urlspecchar}
969: \def\urlspecchar{\penalty10 }
970: \def\urlskip{\null\nobreak\hskip0pt plus0.05em\relax}
971: \def\urlbskip{\penalty100 \hskip0pt plus0.05em\relax}
972: \def\urlslashslash{/urlskip/}
973: \addprotect\url

```

Makro `\url{<text>}` pracuje tak, že uloží `<text>` do `\tmpb` a nechá vyměnit příslušné znaky uvnitř `\tmpb` pomocí `\replacestrings`. Nakonec vytiskne `<text>` prostřednictvím `\urlink`.

Aktivní vlnku lze v `<textu>` vyměnit za `\char`\~`. Podobně lze řešit některé další znaky, ale ne všechny: procento, backlash. U těchto znaků bychom nejprve museli vyměnit jejich kategorie. Pak by ale makro `\url` nefungovalo uvnitř parametrů jiných maker. V zájmu jednoduchosti makra `\url` to neděláme. Takže pokud uživatel má v URL znak procento, musí psát `\%` nebo si změni kategorie sám. Podobná poznámka platí pro znaky `{, }, \, #` a `$`.

### 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

```

977: \def\outlines#1{\pdfcatalog{/PageMode/UseOutlines}\openref\ifx\toclist\empty
978:   \opwarning{\noexpand\outlines -- data unavailable. TeX me again}%
979:   \else
980:     \ifx\urlcolor\empty
981:       \opwarning{\noexpand\outlines doesn't work when \noexpand\hyperlinks isn't declared}\fi
982:       {\let\tocline=\outlinesA
983:        \count0=0 \count1=0 \toclist % calculate numbers o childs
984:        \def\outlinelevel{#1}\let\tocline=\outlinesB
985:        \count0=0 \count1=0 \toclist}% create outlines
986:     \fi
987: }

```

V makru `\outlinesA` `{<odsazení>}{<info>}{<číslo>}{<text>}{<strana>}` počítáme potomky. Makro je navrženo tak, aby bylo snadno rozšířitelné na libovolnou úroveň hloubky stromu, nicméně pro potřeby OPmac stačí hloubka tři (kapitoly, sekce, podsekce). Úroveň uzlu přečteme v parametru `<odsazení>`. Pro kapitoly 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 zpracováváný uzel. Uzly pro kapitoly mají přidělenou kontrolní sekvenci `ol:\the\count0` a uzly pro sekce mají přidělenou 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 `<odsazení>=1` zvětšíme o jedničku počet potomků nadřazené kapitole a při `<odsazení>=2` nadřazené sekci. Asi by bylo přehlednější na začátku definovat všechny potřebné sekvence `ol:<něco>` a nastavit jim hodnotu 0. Ovšem šetříme paměť i časem, takže zakládáme sekvenci `ol:<něco>` teprve v makru `\addoneol` a to tehdy, když je ji poprvé potřeba zvětšit o jedničku.

opmac.tex

```

988: \def\outlinesA#1#2#3#4#5{%
989:   \advance\count#1 by1
990:   \ifcase#1\or
991:     \addoneol{ol:\the\count0}\or
992:     \addoneol{ol:\the\count0:\the\count1}\fi
993: }
994: \def\addoneol#1{\isdefined{#1}%
995:   \iftrue \tmpnum=\csname#1\endcsname\relax
996:     \advance\tmpnum by1 \sxddef{#1}{\the\tmpnum}%
997:   \else \sxddef{#1}{1}%
998:   \fi
999: }
```

V makru `\outlinesB <odsazení>{<info>}{<číslo>}{<text>}{<strana>}` vkládáme položku obsahu do záložek. 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 `\lcode` 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

```

1000: \def\outlinesB#1#2#3#4#5{%
1001:   \advance\count#1 by1
1002:   \ifcase#1\tmpnum=\isdefined{ol:\the\count0}%
1003:     \iftrue\csname ol:\the\count0\endcsname\else0\fi \or
1004:     \tmpnum=\isdefined{ol:\the\count0:\the\count1}%
1005:     \iftrue\csname ol:\the\count0:\the\count1\endcsname\else0\fi \or
1006:     \tmpnum = 0 \fi
1007:   \protectlist \def~{ } \setcnvcodesA
1008:   \ifx\toasciidata\empty \let\lowercase=\relax \else \expandafter\setlccodes\toasciidata~{} \fi
1009:   \cnvhook \lowercase{\gdef\tmp#4}%
1010:   \outlinesC{#1}{toc:\tocilabel.#3}{\ifnum#1<\outlinelevel\space\else-\fi}{\tmpnum}{\tmp}%
1011: }
```

Makro `\outlinesC <úroveň>{<lejblík>}{<minus>}{<potomci>}{<text>}` konečně zavolá primitivní `\pdfoutline_goto_name{<lejblík>}_count<minus><potomci>_{<text>}`. a vytvoří lejblík v dané úrovni zanoření na základě předpočítaného údaje `<potomci>`, který obsahuje počet potomků právě vkládané záložky. Údaj `<minus>` je (po expanzi) prázdný, pokud nechceme mít potomky skryté a obsahuje znak minus, pokud chceme mít potomky ve výchozím stavu skryté. Připomínám, že v makru `\outlinelevel` máme makrem `\outlines` připravenou úroveň rozevření, kterou si uživatel přeje. Makro `\outlinesC` je připraveno k předefinování v modulu `opmac-xetex.tex`, který pro vytvoření záložek používá `\special` a nepotřebuje znát údaj `<potomci>`. Příslušný `\special` využije přímo údaj `<úroveň>`.

opmac.tex

```

1012: \def\outlinesC#1#2#3#4#5{\pdfoutline goto name{#2} count #3#4{#5}\relax}
```







```

1050: \def\setverb{\frenchspacing\def\do##1{\catcode'##1=12}\dospecials \catcode'\*=12 }
1051: \def\begtt{\par \vskip\parskip \ttskip \bgroup \wipepar
1052:   \setverb \adef{ }{\ }%
1053:   \ifx\savedttcharundefined \else \catcode\savedttchar=12 \fi
1054:   \parindent=\ttindent \parskip=0pt
1055:   \tthook\relax
1056:   \ifnum\ttline<0 \else
1057:     \tenrm \thefontscale[700]\let\sevenrm=\thefont
1058:     \everypar\expandafter{\the\everypar \global\advance\ttline by1 \printttline}\fi
1059:   \def\par##1{\endgraf\ifx##1\egroup\else\penalty\ttpenalty\leavevmode\fi ##1}%
1060:   \obeylines \startverb}
1061: {\catcode'\|=0 \catcode'\|=12
1062: |gdef|startverb#1\endtt[|tt|ptthook#1|egroup|par|ttskip|testparA|}

```

opmac.tex

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`).

```

1063: \def\testparA{\expandafter\testparB\romannumeral-'\.}
1064: \def\testparB{\futurelet\tmpa\testparC}
1065: \def\testparC{\ifx\tmpa\par\else\afternoindent\fi}

```

opmac.tex

Makro `\printttline` vytiskne číslo řádku.

```

1067: \def\printttline{\llap{\sevenrm\the\ttline\kern.9em}}

```

opmac.tex

Makro `\activettchar` pracuje podobně, jako makro `\adef`. Navíc potřebuje použít nově načtený znak ve své aktivní kategorii jako separátor vymezující konec parametru. Do sekvencí `\savedttchar` a `\savedttcharc` je uložena ASCII hodnota znaku a jeho původní kategorie.

```

1069: \def\activettchar#1{%
1070:   \ifx\savedttcharundefined\else \catcode\savedttchar=\savedttcharc \fi
1071:   \chardef\savedttchar='#1%
1072:   \chardef\savedttcharc=\catcode'#1%
1073:   \bgroup\lccode'\~='#1%
1074:   \lowercase {\egroup\def~}{\leavevmode\hbox\bgroup\setverb\adef{ }{\ }%
1075:     \intthook\tt\readverb}%
1076:   \bgroup\lccode'\~='#1\lowercase{\egroup\def\readverb ##1~}{##1\egroup}%
1077:   \catcode'#1=13
1078: }

```

opmac.tex

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.

```

1080: \def\verbinput (#1) #2 {\par \def\tmpa{#2}%
1081:   \ifx\vifilename\tmpa \else
1082:     \openin\vifile=#2
1083:     \global\viline=0 \global\let\vifilename=\tmpa
1084:     \ifeof\vifile
1085:       \opwarning{\noexpand\verbinput - file "#2" is unable to reading}
1086:       \expandafter\expandafter\expandafter\skiptorelax
1087:     \fi
1088:   \fi
1089:   \viscanparameter #1+\relax
1090: }
1091: \def\skiptorelax#1\relax{}

```

opmac.tex

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

---

```

\testparA: 39, 41   \testparB: 39–40   \testparC: 39   \printttline: 39, 41
\activettchar: 39–40   \savedttchar: 39, 41   \savedttcharc: 39   \verbinput: 7, 38–39
\vifilename: 39–40   \skiptorelax: 39, 49   \vinolines: 40–41

```

řádků a `\vidolines` bude obsahovat počet řádků, které se mají přepsat ze souboru do dokumentu. Písmena `vi` na začátku těchto názvů představují zkratku pro `verbatim`. 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

```

1093: \def \viscanparameter #1+#2\relax{%
1094:   \if$#2$\viscanminus(#1)\else \viscanplus(#1+#2)\fi
1095: }
1096: \def\viscanplus(#1+#2){%
1097:   \if$#1$\tmpnum=\viline
1098:   \else \ifnum#1<0 \tmpnum=\viline \advance\tmpnum by-#1
1099:     \else \tmpnum=#1
1100:     \advance\tmpnum by-1
1101:     \ifnum\tmpnum<0 \tmpnum=0 \fi % (0+13) = (1+13)
1102:   \fi \fi
1103:   \edef\vinolines{\the\tmpnum}%
1104:   \if$#2$\def\vidolines{0}\else\edef\vidolines{#2}\fi
1105:   \doverbinput
1106: }
1107: \def\viscanminus(#1-#2){%
1108:   \if$#1$\tmpnum=0
1109:   \else \tmpnum=#1 \advance\tmpnum by-1 \fi
1110:   \ifnum\tmpnum<0 \tmpnum=0 \fi % (0-13) = (1-13)
1111:   \edef\vinolines{\the\tmpnum}%
1112:   \if$#2$\tmpnum=0
1113:   \else \tmpnum=#2 \advance\tmpnum by-\vinolines \fi
1114:   \edef\vidolines{\the\tmpnum}%
1115:   \doverbinput
1116: }

```

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 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

```

1117: \def\doverbinput{%
1118:   \tmpnum=\vinolines
1119:   \advance\tmpnum by-\viline
1120:   \ifnum\tmpnum<0
1121:     \openin\vifile=\vifilename\space
1122:     \global\viline=0
1123:   \else

```

---

`\vidolines`: 40–41    `\viscanparameter`: 39–40    `\viscanplus`: 40    `\viscanminus`: 40  
`\doverbinput`: 40–41

```

1124: \edef\vinolines{\the\tmpnum}%
1125: \fi
1126: \vskip\parskip \ttskip \bgroup \wipeepar
1127: \setverb \adef{ }\ }%
1128: \ifx\savedttcharundefined \else \catcode\savedttchar=12 \fi
1129: \parindent=\ttindent \parskip=Opt
1130: \tthook\relax
1131: \ifnum\ttline<-1 \else
1132: \tenrm \thefontscale[700]\let\sevenrm=\thefont
1133: \everypar\expandafter{\the\everypar \glob\advance\ttline by1 \printttline}\fi
1134: \def\par#1{\endgraf\ifx#1\egroup\else\penalty\ttpenalty\leavevmode\fi #1}%
1135: \obeylines \tmpnum=0 \lccode'\~='^M \lowercase{\def\tmpb{~}}%
1136: \loop \ifeof\vfifile \tmpnum=\vinolines\space \fi
1137: \ifnum\tmpnum<\vinolines\space
1138: \vireadline \advance\tmpnum by1 \repeat %% skip line
1139: \ifnum\ttline=-1 \ttline=\viline \let\glob=\relax \else\let\glob=\global \fi
1140: \tmpnum=0 \ifnum\vidolines=0 \tmpnum=-1 \fi
1141: \ifeof\vfifile \tmpnum=\vidolines\space \fi
1142: \loop \ifnum\tmpnum<\vidolines\space
1143: \vireadline
1144: \ifeof\vfifile \tmpnum=\vidolines\space \else \viprintline \fi %% print line
1145: \ifnum\vidolines=0 \else\advance\tmpnum by1 \fi
1146: \repeat
1147: \tt\expandafter\ptthook\tmpb\egroup\par\ttskip\testpara
1148: }

```

V prvním cyklu `\loop` v těle makra `\doverbininput` se opakovaně volá `\vireadline`, což je makro, které přečte další řádek ze souboru. V druhém cyklu se opakovaně volá `\vireadline` následované `\viprintline`. Toto makro vloží přečtený řádek do `\tmpb`. Nakonec se `\tmpb` vytiskne stejným způsobem, jako při přečtení textu makrem `\begtt`.

opmac.tex

```

1149: \def\vireadline{\read\vfifile to \tmp \global\advance\viline by1 }
1150: \def\viprintline{\expandafter\addto\expandafter\tmpb\expandafter{\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 uživatelský `\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` 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

```

1154: \newtoks\tabdata
1155: \def\tabstrutA{\tabstrut}
1156: \newcount\colnum
1157: \def\ddlinedata{}
1158: \def\vvleft{}

```

Makro `\table` `{\langle deklarace \rangle}{\langle data \rangle}` vypadá takto:

opmac.tex

```

1160: \def\table{\vbox\bgroup \catcode'\|=12 \tableA}
1161: \def\tableA#1#2{\offinterlineskip \colnum=0 \def\tmpa{}\tabdata={}\scantabdata#1\relax
1162: \halign\expandafter{\the\tabdata\cr#2\cr}\egroup}

```

---

```

\vireadline: 41 \viprintline: 41 \tabdata: 41–44 \tabstrutA: 41–43 \colnum: 41–43
\ddlinedata: 41–43 \vvleft: 41, 43 \table: 7, 41–42

```

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⟩}`.

```

1164: \def\scantabdata#1{\let\next=\scantabdata
1165:   \ifx\relax#1\let\next=\relax
1166:   \else\ifx|#1\addtabvrule
1167:     \else\isinlist{123456789}#1\iftrue \def\next{\scantabdataC#1}%
1168:     \else \expandafter\ifx\csname tabdeclare#1\endcsname \relax
1169:       \expandafter\ifx\csname paramtabdeclare#1\endcsname \relax
1170:         \opwarning{tab-declarator "#1" unknown, ignored}%
1171:         \else \def\next{\expandafter \scantabdataB \csname paramtabdeclare#1\endcsname}\fi
1172:         \else \def\next{\expandafter\scantabdataA \csname tabdeclare#1\endcsname}%
1173:       \fi\fi\fi\fi \next
1174: }

```

opmac.tex

Pomocná makra `\scantabdataA` a `\scantabdataB` řeší případy, kdy deklarátor nemá nebo má parametr. Dále makra `\scantabdataC` a `\scantabdataD` se starají o případné opakování úseku deklarace.

```

1175: \def\scantabdataA#1{\addtabitem \expandafter\addtabdata\expandafter{#1\tabstrutA}\scantabdata}
1176: \def\scantabdataB#1#2{\addtabitem\expandafter\addtabdata\expandafter{#1{#2}\tabstrutA}\scantabdata}
1177: \def\scantabdataC {\def\tmpb{ }\afterassignment\scantabdataD \tmpnum=}
1178: \def\scantabdataD#1{\loop \ifnum\tmpnum>0 \advance\tmpnum by-1 \addto\tmpb{#1}\repeat
1179:   \expandafter\scantabdata\tmpb
1180: }

```

opmac.tex

OPmac předdefinuje čtyři *⟨deklarátory⟩* pro sloupce tabulky, sice *⟨znaky⟩* c, l, r, p v makrech `\tabdeclarec`, `\tabdeclarel`, `\tabdeclarer` a `\paramtabdeclarep`. Je-li deklarátor bez parametru, je třeba definovat `\tabdeclare⟨znak⟩` a je-li s parametrem, je třeba definovat `\paramtabdeclare⟨znak⟩`. V případě typu p přidáváme na konec odstavce (do posledního řádku) strut nulové výšky, ale hloubku má podle `\tabstrutA`.

```

1181: \def\tabdeclarec{\tabiteml\hfil#\unsskip\hfil\tabitemr}
1182: \def\tabdeclarel{\tabiteml#\unsskip\hfil\tabitemr}
1183: \def\tabdeclarer{\tabiteml\hfil#\unsskip\tabitemr}
1184: \def\paramtabdeclarep#1{\tabiteml\vtop{\hsize=#1\relax \baselineskip=\normalbaselineskip
1185:   \lineskiplimit=0pt \noindent#\unsskip \vbox to0pt{\vss\hbox{\tabstrutA}}}\tabitemr}

```

opmac.tex

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.

```

1187: \def\unsskip{\ifdim\lastskip>0pt \unskip\fi}

```

opmac.tex

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

```

tabdata: \vrule\tabiteml\hfil#\unsskip\hfil\tabitemr\tabstrutA
         &\tabiteml\hfil#\unsskip\tabitemr \vrule\kern\vvkern\vrule\tabstrutA
         &\tabiteml\hfil#\unsskip\hfil\tabitemr\tabstrutA
         &\tabiteml#\unsskip\hfil\tabitemr\vrule\tabstrutA
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`.

---

```

\scantabdata: 41–42   \scantabdataA: 42   \scantabdataB: 42   \scantabdataC: 42
\scantabdataD: 42   \tabdeclarec: 42   \tabdeclarel: 42   \tabdeclarer: 42
\paramtabdeclarep: 42   \unsskip: 42, 44   \addtabitem: 42–43   \addtabdata: 42–44
\addtabvrule: 42–43

```

```

1188: \def\addtabitem{\ifnum\colnum>0 \addtabdata{&}\addto\ddlinedata{\dditem}\fi
1189:   \advance\colnum by1 \let\tmpa=\relax}
1190: \def\addtabdata#1{\tabdata\expandafter{\the\tabdata#1}}
1191: \def\addtabvrule{%
1192:   \ifx\tmpa\vrule \addtabdata{\kern\vvkern}%
1193:   \ifnum\colnum=0 \addto\vleft{\vvitem}\else\addto\ddlinedata{\vvitem}\fi
1194:   \else\ifnum\colnum=0 \addto\vleft{\vvitemA}\else\addto\ddlinedata{\vvitemA}\fi\fi
1195:   \let\tmpa=\vrule \addtabdata{\vrule}}

```

Než se pustíme do výkladu dalších maker, předvedeme příklad, ve kterém je definován deklarátor F pro centrovanou položku, kde text je v rámečku (deklarátor bez parametru) a dále definujeme analogii deklarátoru p s parametrem (bude se jmenovat V), který umístí odstavce různé vysoké vedle sebe vertikálně centrovaně.

```

\def\tabdeclareF{\tabiteml\hfil\frame{##\unsskip}\hfil\tabitemr}
\def\paramtabdeclareV#1{\tabiteml{\$\vcenter{\hsize=#1
  \baselineskip=\normalbaselineskip \lineskiplimit=0pt
  \noindent\ vbox{\hbox{\tabstrutA}\kern-\prevdepth}##\unsskip
  \vbox to0pt{\vss\hbox{\tabstrutA}}}\$}\tabitemr}
\def\tabstrut{\vrule height 20pt depth10pt width0pt}

\table{V{3cm\raggedright} V{4cm}} {delší text & text \cr text & delší text}

```

Pustíme se nyní do rozboru maker na ukončení řádků. Makro `\crl` přidá čáru pomocí `\noalign`. Makro `\crl1` přidá dvojitou čáru pomocí `\noalign`.

```

1197: \def\crl{\crrc\noalign{\hrule}}
1198: \def\crl1{\crrc\noalign{\hrule\kern\hhkern\hrule}}

```

Makro `\crl1` provede `\cr` a dále se vnoří do řádku tabulky, ve kterém klade postupně následující `\omit\tablinefil` a `\omit\tablinefil`. Přitom v místě dvojitě 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 přetržené místo v postupně tvořené lince. Ke správnému naklazení uvedených povelů použije makro `\crl1` obsah makra `\ddlinedata` a vlevo přidává `\vleft`. Před spuštěním makra `\ddlinedata` definuje odpovídajícím způsobem `\dditem` a `\vvitem`. Makro `\crl1` sestává ze dvou `\crl1` oddělených od sebe vertikální mezerou vloženou pomocí `\noalign`.

```

1200: \def\crl1{\crrc \omit
1201:   \gdef\dditem{\omit\tablinefil}\gdef\vvitem{\kern\vvkern\vrule}\gdef\vvitemA{\vrule}%
1202:   \vleft\tablinefil\ddlinedata\crrc}
1203: \def\crl1i{\crl1\noalign{\kern\hhkern}\crl1}
1204: \def\tablinefil{\leaders\hrule\hfil}

```

Makro `\tskip` prostřednictvím `\tskipA` přechodně vyprázdní `\tabstrut` předdefinováním `\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.

```

1219: \def\tskip{\afterassignment\tskipA \tmpdim}
1220: \def\tskipA{\gdef\dditem{} \gdef\vvitem{} \gdef\vvitemA{} \gdef\tabstrutA{}%
1221:   \vbox to\tmpdim{\ddlinedata \crrc \noalign{\gdef\tabstrutA{\tabstrut}}}

```

Makro `\mspan` `<číslo>[<deklarace>]{<text>}` překoná `<číslo>` sloupců a dále `<text>` v tomto prostoru formátuje podle `<deklarace>`. K tomu účelu provede `\multispan` pomocí `\loop` v `\mspanA` a dále připraví tělo formátovacího makra postupným čtením deklarace pomocí `\mspanB`. V závěru je toto tělo použito v makru `\tmpa`, které je nakonec spuštěno.

---

```

\crl: 43   \crl1: 43   \crl1i: 41, 43   \tablinefil: 43   \tabvvline: 43   \dditem: 41, 43
\vvitem: 41–43   \crl1i: 43   \tskip: 41, 43   \tskipA: 43   \mspan: 44   \mspanA: 44
\mspanB: 44

```



```

1223: \def\mspan{\omit \tabdata={\tabstrut}\let\tmpa=\relax \afterassignment\mspanA \mscount=}
1224: \def\mspanA[#1]{\loop \ifnum\mscount>1 \csname span\endcsname \omit \advance\mscount by-1 \repeat
1225:   \mspanB#1\relax}
1226: \def\mspanB#1{\ifx\relax#1\def\tmpa{\def\tmpa###1}%
1227:   \expandafter\tmpa\expandafter{\the\tabdata\ignorespaces}\expandafter\tmpa\else
1228:   \ifx|#1\ifx\tmpa\vrule\adddtabdata{\kern\vvkern}\fi \adddtabdata{\vrule}\let\tmpa=\vrule
1229:   \else \let\tmpa=\relax
1230:   \ifx c#1\adddtabdata{\tabiteml\hfil\ignorespaces##1\unsskip\hfil\tabitemr}\fi
1231:   \ifx l#1\adddtabdata{\tabiteml\ignorespaces##1\unsskip\hfil\tabitemr}\fi
1232:   \ifx r#1\adddtabdata{\tabiteml\hfil\ignorespaces##1\unsskip\tabitemr}\fi
1233:   \fi \expandafter\mspanB \fi}

```

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

```

1236: \let\orihrule=\hrule \let\orivrule=\vrule
1237: \def\rulewidth{\afterassignment\rulewidthA \drulewidth}
1238: \def\rulewidthA{\edef\hrule{\orihrule height\the\drulewidth}%
1239:   \edef\vrule{\orivrule width\the\drulewidth}}

```

Makro `\frame` `{\text}` vloží vnější `\hbox{\vrule\vnitřek\vrule}`. Uvnitř tohoto boxu se nachází `\vtop{\langledalší\rule\vvkern\hrule}`, takže `\langledalší\rule` zůstává na účarí. Přitom `\langledalší\rule` je `\vbox{\hrule\kern\vvkern\langledalší2\rule}`, takže `\langledalší2\rule` zůstává na účarí. V tuto chvíli jsou již vytvořeny čáry vlevo, vpravo, nahoře i dole. Konečně `\langledalší2\rule` je `\hbox{\kern\hhkern\langledalší2\rule\kern\hhkern}`.

```

1241: \long\def\frame#1{%
1242:   \hbox{\vrule\vtop{\vbox{\hrule\kern\vvkern
1243:     \hbox{\kern\hhkern\relax#1\kern\hhkern}%
1244:   }}\kern\vvkern\hrule}\vrule}}

```

### 3.20 Vložení obrázku

Nejprve deklarujeme `\picwidth` a `\picheight`. Z důvodu zpětné kompatibility je dále ztotožněn `\picwidth` se sekvencí `\picw`.

```

1248: \newdimen\picwidth \picwidth=0pt \let\picw=\picwidth
1249: \newdimen\picheight \picheight=0pt

```

Makro `\inspic` je zkratka za použití primitiv `\pdfximage`, `\pdfrefximage` a `\pdflastximage`. Kdo si to má pořádkem pamatovat. Není-li aktivován PDF výstup, napíšeme jen varování a neprovedeme nic.

```

1251: \ifpdf\text
1252:   \def\inspic #1 {\hbox{%
1253:     \pdfximage \ifdim\picwidth=0pt \else width\picwidth\fi
1254:     \ifdim\picheight=0pt \else height\picheight\fi \inspicpage {\picdir#1}%
1255:     \pdfrefximage\pdflastximage}}
1256: \else
1257:   \def\inspic #1 {\opwarning
1258:     {The \noexpand\inspic is supported for PDF output only}}
1259: \fi
1260: \def\inspicpage{}

```

Makro `\inspicpage` může při natažení PDF obsahovat text `page\langlenumber\rule`. Pak se jako obrázek použije odpovídající strana PDF dokumentu.

### 3.21 PDF transformace

Makro `\pdfscale` `{\vodorovně}``{\svisle}` pracuje jednoduše:

```

1264: \def\pdfscale#1#2{\pdfsetmatrix{#1 0 0 #2}}

```

---

```

\rulewidth: 44 \rulewidthA: 44 \orihrule: 44 \orivrule: 44 \frame: 44
\picwidth: 44 \picheight: 44 \picw: 44 \inspic: 44 \inspicpage: 44
\pdfscale: 33, 44

```



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  $\text{\TeX}$  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 1276 až 1285 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` nezmění. 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  $\{0, 1, 2, 3, \dots, 22\}$ . 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 1288 až 1292 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.

```

1266: \def\pdfrotate#1{\tmpdim=#1pt
1267:   \ifdim\tmpdim=0pt
1268:     \else \ifdim\tmpdim=90pt \pdfsetmatrix{0 1 -1 0}%
1269:       \else \edef\tmp{#1}\expandafter\pdfrotateA\tmp..\relax
1270:     \fi \fi
1271: }
1272: \def\pdfrotateA #1.#2.#3\relax{%
1273:   \def\tmp##1.##2\relax {##1}%
1274:   \tmpnum=\expandafter \tmp \the\tmpdim \relax % round
1275:   \ifdim\tmpdim>Opt \def\tmpa{} \else \def\tmpa{-} \fi % save -
1276:   \loop \ifnum\tmpnum<0 \advance\tmpnum by360 \repeat
1277:   \loop \ifnum\tmpnum>360 \advance\tmpnum by-360 \repeat
1278:   \loop \ifnum\tmpnum>90 \pdfrotate{90}\advance\tmpnum by-90 \repeat
1279:   \ifnum\tmpnum=90 \pdfrotate{90} \else
1280:     \ifnum\tmpnum>44 \pdfsetmatrix{.7071 .7071 -.7071 .7071}%
1281:       \advance\tmpnum by-45 \fi
1282:     \ifnum\tmpnum>22 \pdfsetmatrix{.9272 .3746 -.3746 .9272}%
1283:       \advance\tmpnum by-22 \fi
1284:     \ifnum\tmpnum>0
1285:       \pdfsetmatrix{\smallcos \smallsin -\smallsin \smallcos}%
1286:     \fi \fi
1287:     \if$#2$\else % fraction part
1288:       \tmpdim=.01745329pt % \pi/180
1289:       \tmpdim=#2\tmpdim %
1290:       \edef\tmp{\expandafter\ignorept\the\tmpdim\space}%
1291:       \ifx\tmpa\empty \pdfsetmatrix{1 \tmp -\tmp 1}%
1292:       \else \pdfsetmatrix{1 -\tmp \tmp 1}%
1293:     \fi \fi
1294: }
1295: \def\smallcos{.\ifcase\tmpnum \or9998\or9994\or9986\or9976\or9962\or9945\or
1296: 9925\or9903\or9877\or9848\or9816\or9781\or9744\or9703\or9659\or9613\or
1297: 9563\or9511\or9455\or9397\or9336\or9272\fi\space}
1298: \def\smallsin{.\ifcase\tmpnum 0\or0175\or0359\or0523\or0698\or0872\or1045\or
1299: 1219\or1391\or1564\or1736\or1908\or2079\or2250\or2419\or2588\or2756\or
1300: 2924\or309\or3256\or342\or3584\or3746\fi\space}

```

Pro případ, že nepracujeme s PDF výstupem, definujeme klíčové primitivy pdf $\text{\TeX}$ u jako makra, která nedělají nic.

```

1302: \ifpdf\else
1303:   \def\pdfsetmatrix#1{} \def\pdfsave{} \def\pdfrestore{}
1304: \fi

```

---

`\pdfrotate`: 33, 45    `\pdfrotateA`: 45    `\smallcos`: 45    `\smallsin`: 45

### 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 `\tfootnote`, které vloží sazbu pomocí tzv. insertu (TBN, kapitola 6.7). PlainTeXové nastavení této třídy insertu není makrem OPmac nijak měněno.

opmac.tex

```
1308: \newcount\fnotenum \fnotenum=0
1309: \newcount\fnotenumlocal
1310: \newif\iflocfnum \locfnumtrue
1311:
1312: \long\def\fnote#1{\global\advance \fnotenum by1
1313:   \leavevmode\openref\wref\Xfnote{}}%
1314:   \iflocfnum \isdefined{fn:\the\fnotenum}\iftrue
1315:     \else\opwarning{unknown \noexpand\fnote mark. TeX me again}\fi\fi
1316:   \fnmarkx{\fnotehook\typobase\typoscale[800/800]\tfootnote\fnmarkx{#1}}%
1317: }
```

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

```
1318: \def\fnotemark#1{\advance\fnotenum by#1\relax
1319:   \isdefined{fn:\the\fnotenum}\iftrue\thefnote
1320:   \else$~?$\opwarning{unknown \string\fnotemark. TeX me again}\fi}%
1321: }
```

Makro `\fnotetext` teprve zvedne čítač `\fnotenum` globálně a vytiskne poznámku pomocí plainTeXového `\tfootnote`.

opmac.tex

```
1322: \long\def\fnotetext#1{\global\advance \fnotenum by1 \openref\wref\Xfnote{}}%
1323:   {\everypar={}\fnotehook\typobase\typoscale[800/800]\tfootnote\fnmarkx{#1}}%
1324: }
```

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

```
1325: \def\fnmarkx{\isdefined{fn:\the\fnotenum}\iftrue\thefnote\else$~?$\fi}
1326: \def\thefnote{$~{\locfnum}$}
1327: \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 `\sxdif` si tuto hodnotu zapamatovat v makru `\fn:⟨číslo⟩`.

opmac.tex

```
1329: \def\Xfnote{\advance\fnotenumlocal by1 \advance\fnotenum by1
1330:   \sxdif{fn:\the\fnotenum}{\the\fnotenumlocal}}
```

Makro `\runningfnotes` vypne lokální číslování poznámek na každé stránce. Místo toho se budou poznámky číslovat podle registru `\fnotenum`. Ten se zvětšuje o jedničku v celém dokumentu. Chcete-li mít poznámky číslované zvlášť například v každé kapitole, je nutno navíc resetovat tento čítač například pomocí `\addto\chaphook{\global\fnotenum=0}`.

opmac.tex

```
1332: \def\runningfnotes{\locfnumfalse\def\locfnum{\the\fnotenum}\def\fnmarkx{\thefnote}}
```

Registr `\mnotenum` globálně čísluje okrajové poznámky a plní podobnou funkci, jako registr `\fnotenum` pro podčárové poznámky. Registr `\mnoteskip` udává hodnotu vertikálního posunu poznámky.

opmac.tex

```
1334: \newcount\mnotenum \mnotenum=0 % global counter of mnotes
1335: \newdimen\mnoteskip \mnoteskip=0pt
```

---

```
\fnote: 8, 46, 54   \fnotenum: 13, 46   \fnotemark: 46, 54   \fnotetext: 46   \fnmarkx: 46
\thefnote: 46     \locfnum: 46     \fnotenumlocal: 46, 55   \Xfnote: 46, 55   \runningfnotes: 46
\mnotenum: 13, 46–47   \mnoteskip: 46–47
```

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 `\adjust` 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.

```

1337: \long\def\mnote#1{\ifvmode \hbox{\vbox to\ht\strutbox{\mnoteA{#1}}\nobreak\vskip-\baselineskip
1338:   \else \strut\adjust{\kern-\dp\strutbox \mnoteA{#1}\kern\dp\strutbox}%
1339:   \fi
1340: }

```

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_0to0pt{\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`.

```

1341: \long\def\mnoteA#1{\global\advance \mnotenum by1
1342:   \ifx\mnotesfixed\undefined
1343:     \isdefined{mn:\the\mnotenum}\iftrue
1344:       \else\opwarning{unknown \noexpand\mnote side. TeX me again}\fi
1345:       \edef\tmp{\csname mn:\the\mnotenum\endcsname}%
1346:       \openref\wref\Xmnote{\ifvmode\nobreak\fi
1347:       \else \let\tmp=\mnotesfixed \fi
1348:       \expandafter\ifx\tmp \left
1349:         \hbox to0pt{\kern-\mnotesize \kern-\mnoteindent
1350:           \vbox to0pt{\vss \setbox0=\vtop{\hsize=\mnotesize
1351:             \leftskip=0pt plus 1fill \rightskip=0pt {\mnotehook\noindent#1\endgraf}}}%
1352:           \dp0=0pt \box0 \kern\mnoteskip \global\mnoteskip=0pt}\hss}%
1353:         \else
1354:           \hbox to0pt{\kern\hsize \kern\mnoteindent
1355:             \vbox to0pt{\vss \setbox0=\vtop{\hsize=\mnotesize
1356:               \rightskip=0pt plus 1fil \leftskip=0pt {\mnotehook\noindent#1\endgraf}}}%
1357:             \dp0=0pt \box0 \kern\mnoteskip \global\mnoteskip=0pt}\hss}%
1358:         \fi
1359: }

```

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

```

1360: \def\Xmnote{\advance\mnotenum by1
1361:   \sxddef{mn:\the\mnotenum}{\ifodd\lastpage \right \else \left \fi}}

```

Makro `\fixmnotes` (*token*) definuje interní makro `\mnotesfixed` s obsahem `\left` nebo `\right` podle přání uživatele. Makro `\mnoteA` se pak na definovanost `\mnotesfixed` ptá a pokud je definované, nepoužije údaje přečtené ze souboru.

```

1363: \def\fixmnotes#1{\def\mnotesfixed{#1}}

```

### 3.23 Bibliografické reference

Nejprve uvedeme deklarace deskriptoru `\auxfile`, stringu `\bibmark` a čítačů `\bibnum` a `\lastcitenum`.

---

```

\mnote: 8, 47   \mnoteA: 47   \Xmnote: 47, 55   \fixmnotes: 47   \mnotesfixed: 47
\auxfile: 48, 52–53   \bibmark: 48, 51, 53–54   \bibnum: 48, 51–53   \lastcitenum: 48–51

```

```

1367: \newwrite\auxfile           % AUX file for BibTeX
1368: \newcount\bibnum           % the bibitem counter
1369: \newtoks\bibmark           % the bibmark used if \nonumcitations
1370: \newcount\lastcitenum \lastcitenum=0 % for \shortcitations

```

Makro `\cite` [ $\langle lejblík1 \rangle, \langle lejblík2 \rangle, \dots$ ] si opakovaně zavolá `\citeA` $\langle lejblík-i \rangle$ , kde se připraví čísla citovaných publikací do lokálně tvořeného seznamu `\savedcites`. Poté zavolá `\printsavedcites`, které lokálně tvořený seznam čísel vytiskne. Kromě toho makro `\citeA` udělá plno dalších potřebných věcí, jak uvidíme za chvíli. Makro `\nocite` se chová jako `\cite` až na to, že se nic netiskne. Makro `\rcite` vytiskne čísla publikací, ale bez hranatých závorek kolem. Makro `\savedcites` je globálně prázdné a zaplní se vždy znovu uvnitř skupiny vymezené makrem `\cite` nebo `\nocite` nebo `\rcite`.

```

1372: \def\cite[#1]{\citeA#1,,,\printsavedcites}}
1373: \def\nocite[#1]{\citeA#1,,,\}
1374: \def\rcite[#1]{\citeA#1,,,\printsavedcites}}
1375: \def\savedcites{}

```

Makro `\citeA`  $\langle lejblík \rangle$ , řeší zhruba řečeno následující věci:

- Zjistí, zda je definován `\csname_bib:\langle lejblík \rangle\endcsname`. Pokud ano, přidá obsah tohoto makra (což je číslo citovaného záznamu) do `\savedcites`. Pokud ne, přidá do `\savedcites` otazník a na terminál vypíše varování. Kontrolní sekvence `\csname_bib:\langle lejblík \rangle\endcsname` bude obsahovat  $\langle číslo-citace \rangle$  po použití `\bib[\langle lejblík \rangle]` nebo `\bibitem{\langle lejblík \rangle}`. Tato makra uloží odpovídající informaci do REF souboru, odkud ji při opakovaném T<sub>E</sub>Xování 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 `\citeA` je naprogramováno zhruba takto

```

function citeA(\langle lejblík \rangle) {
  if (\langle lejblík \rangle == '*') { \zapiš do \citelist '*'; return; }
  if (\bib:\langle lejblík \rangle == nedef) {
    \přidej do \citelist \langle lejblík \rangle;
    \na terminál: "Warning, cite [label] unknown";
    \přidej do \savedcites "?,";
    \lokálně vypni třídění a zkracování seznamu \savedcites;
    \bib:\langle lejblík \rangle = empty;
    return;
  }
  if (\bib:\langle lejblík \rangle == empty) {
    \přidej do \savedcites "?,";
    \lokálně vypni třídění a zkracování seznamu \savedcites;
    return;
  }
  if (\bib:\langle lejblík \rangle končí znakem '&') {
    \přidej do \citelist \langle lejblík \rangle;
    \odstraň znak & z obsahu makra \bib:\langle lejblík \rangle;
  }
  \přidej do \savedcites \expandovaný "\bib:\langle lejblík \rangle,";
}

```

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

Návrh kódu v C-like notaci nyní převedeme do maker v T<sub>E</sub>Xu:

---

`\cite`: 48, 50, 52, 54    `\nocite`: 48, 51, 54    `\rcite`: 48    `\savedcites`: 48–51    `\citeA`: 48–49, 51

```

1377: \def\citeA #1#2,{\if#1,\else
1378:   \if *#1\addcitelist{*}\expandafter \skiptorelax \fi
1379:   \isdefined{bib:#1#2}\iftrue \else
1380:     \addcitelist{#1#2}%
1381:     \opwarning{The cite [#1#2] unknown. Try to TeX me again}\openref
1382:     \addto\savecites{?,}\def\sortcitesA{}\lastcitenum=0
1383:     \expandafter\gdef\csname bib:#1#2\endcsname {}%
1384:     \expandafter \skiptorelax \fi
1385:   \expandafter \ifx \csname bib:#1#2\endcsname \empty
1386:     \addto\savecites{?,}\def\sortcitesA{}\lastcitenum=0
1387:     \expandafter \skiptorelax \fi
1388:   \def\bibnn##1{%
1389:     \if &\csname bib:#1#2\endcsname
1390:       \addcitelist{#1#2}%
1391:       \def\bibnn##1##2{##1}%
1392:       \sxddef{bib:#1#2}{\csname bib:#1#2\endcsname}%
1393:     \fi
1394:     \edef\savecites{\savecites \csname bib:#1#2\endcsname,}%
1395:     \relax
1396:     \expandafter\citeA\fi
1397: }

```

Makro snímá svůj parametr jako #1#2, aby mohly být *lejbíky* odděleny před čárkou mezerou, která je neseparovaným parametrem #1 ignorována. Asi nejzajímavější vychytávka v tomto makru se týká testu na znak &. Implicitně při čtení REF souboru se do makra `\bibnn{<hodnota>}&`. Příkaz `\if` za sebou totálně expanduje vše následující, takže nejprve narazí na &, pak se obsah `\bibnn{<hodnota>}` 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.

Makro `\printsavedcites` případně setřídí seznam `\savecites` podle velikosti zavoláním `\sortcitesA` a dále opakovaně na jednotlivé prvky seznamu zavolá makro `\citeB`, které prvky seznamu vytiskne a případně je zkrátí pomocí intervalů (místo 3,4,5 píše 3--5). Pomocnou proměnnou `\tmpb` využije makro `\citeB`, jak uvidíme později při výkladu tohoto makra.

```

1398: \def\printsavedcites{\sortcitesA
1399:   \chardef\tmpb=0 \expandafter\citeB\savecites,%
1400:   \ifnum\tmpb>0 \printdashcite{\the\tmpb}\fi
1401: }

```

Makro `\sortcitesA` seřadí seznam `\savecites` podle velikosti. Takže třeba 4,7,3,5, se promění na 3,4,5,7,. Implicitně je definováno jako prázdné makro, takže řazení se neprovede. Nicméně uživatel ho použitím makra `\sortcitations` v hlavičce svého dokumentu probudí k životu.

Oživené `\sortcitesA` nejprve vyvrhne do čtecí fronty obsah `\savecites` ukončený další čárkou (máme zde dvě čárky vedle sebe) a následně spustí `\sortcitesB`, které postupně odebírá jednotlivé prvky ze čtecí fronty, předává je do nově tvořeného setříděného seznamu, kam je vkládá na správné místo. Výchozí hodnota nově tvořeného seznamu obsahuje číslo 300000, které bude vždy na konci seznamu, protože se předpokládá větší než jakýkoli tříděný prvek. Zajímavý trik s `\edef\savecites{... \expandafter}` způsobí, že se `\savecites` nejprve vyvrhne (po aplikaci dvou `\expandafter`) do čtecí fronty a teprve poté dostane novou hodnotu pomocí `\edef`. Na konci makra `\sortcitesA` ze seznamu odebereme koncové číslo 300000.

```

1402: \def\sortcitesA{}
1403: \def\sortcitations{%
1404:   \def\sortcitesA{\edef\savecites{300000, \expandafter}\expandafter\sortcitesB\savecites,%
1405:     \def\tmpa####1300000,{\def\savecites{####1}}\expandafter\tmpa\savecites}%
1406: }
1407: \def\sortcitesB #1,{\if $#1$%
1408:   \else
1409:     \mathchardef\tmpa=#1
1410:     \edef\savecites{\expandafter}\expandafter\sortcitesC \savecites\end

```

---

`\bibnn`: 49, 51    `\printsavedcites`: 48–49    `\sortcitesA`: 49, 51    `\sortcitations`: 49, 51  
`\sortcitesB`: 49–50



```

1411: \expandafter\sortcitesB
1412: \fi
1413: }

```

Vložení prvku do zatříděného seznamu probíhá pomocí `\sortcitesC`, což je makro, které nově tvořený seznam, který je nyní také vyvržen ve čtecí frontě, projde zleva doprava, dokud nenarazí na číslo větší než vkládané. Při té činnosti opakovaně sbírá hodnoty a vkládá je zpět do `\savedcites`. Je-li zařazovaný prvek `\tmpa` menší než odebraný prvek z fronty, vloží se pomocí `\sortcitesD` do `\savedcites` původní `\savedcites` následovaný `\tmpa` následovaný testovaným prvkem následovaný zbytkem vstupní fronty (až po `\end`).

opmac.tex

```

1414: \def\sortcitesC#1,{\ifnum\tmpa<#1\edef\tmpa{\the\tmpa,#1}\expandafter\sortcitesD
1415: \else\edef\savedcites{\savedcites#1,}\expandafter\sortcitesC\fi}
1416: \def\sortcitesD#1\end{\edef\savedcites{\savedcites\tmpa,#1}}

```

Makro `\citeB` (*položka*), ukončí činnost při prázdném parametru, jinak se po vytištění (*položky*) zavolá znova. Vytiskne dva otazníky, je-li parametrem otazník, 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` jednoduše nastavuje `\lastcitenum` na nenulovou hodnotu a tím probudí k životu hlavní část makra `\citeB`.

opmac.tex

```

1418: \def\citeB#1,{\if$#1$\else
1419: \if?#1\relax??%
1420: \else
1421: \ifnum\lastcitenum=0 % only comma separated list
1422: \printcite{#1}%
1423: \else
1424: \ifx\citesep\empty % first cite item
1425: \lastcitenum=#1\relax
1426: \printcite{#1}%
1427: \else % next cite item
1428: \advance\lastcitenum by1
1429: \ifnum\lastcitenum=#1\relax % cosecutive cite item
1430: \mathchardef\tmpb=\lastcitenum
1431: \else % there is a gap between cite items
1432: \lastcitenum=#1\relax
1433: \ifnum\tmpb=0 % previous items were printed
1434: \printcite{#1}%
1435: \else
1436: \printdashcite{\the\tmpb}\printcite{#1}\chardef\tmpb=0
1437: \fi\fi\fi\fi\fi
1438: \expandafter\citeB\fi
1439: }
1440: \def\shortcitations{\lastcitenum=1 }

```

Činnost `\cite` je konečně završena voláním maker `\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 upraví separátor `\citesep`, který je globálně a tedy na začátku činnosti `\cite` prázdný. Při opakovaném volání `\printcite` se tedy vytiskne i požadovaný separátor. Pomlčka v `\printdashcite` je schována do `\hbox`, aby nedocházelo těsně za ní ke zlomu řádku.

opmac.tex

```

1442: \def\printcite#1{\citesep\citelink{#1}{\citelinkA{#1}}\def\citesep{,\hskip.2em\relax}}
1443: \def\printdashcite#1{\ifmmode-\else\hbox{--}\fi\citelink{#1}{\citelinkA{#1}}}
1444: \def\citesep{}

```

---

`\sortcitesC`: 49–50    `\sortcitesD`: 50    `\citeB`: 49–50    `\shortcitations`: 48, 50–51  
`\printcite`: 50    `\printdashcite`: 49–50    `\citesep`: 50



Při použití `\nonumcitations` potlačíme případné předchozí `\shortcitations` a `\sortcitations` a dále nastavíme `\citelinkA` na jinou, než implicitní prázdnou hodnotu. Makro `\citelinkA` vytiskne `\bim:⟨číslo-citace⟩`, tedy značku citace (je to nastaveno v `\Xbib`). Není-li značka citace známá, vypíšeme varování a tiskneme `⟨číslo-citace⟩`. Makro `\etalchar` je potřebné při použití BibTeXového stylu alpha.

```

1446: \def\nonumcitations{\lastcitenum=0\def\sortcitesA{}\def\etalchar##1{$^{\##1}$}%
1447:   \def\citelinkA##1{\isdefined{bim:##1}\iftrue \cename bim:##1\endcename
1448:     \else ##1\opwarning{noexpand\nonumcitations + empty bibmark. Maybe bad BibTeX style}\fi}
1449: }
1450: \def\citelinkA{}

```

opmac.tex

Makro `\ecite` [`⟨lejblík⟩`]{`⟨text⟩`} nejprve provede `\citeA#1,,,`, tedy vlastně `\nocite` [`⟨lejblík⟩`] a pak si `\eciteB` vyzvedne ze `\savedcites` první údaj před čárkou, tedy `⟨číslo-citace⟩`, a uloží do #1. V #2 je případný zbytek ze `\savedcites` a dále v #3 pokračuje `⟨text⟩`. Makro vytiskne jen `⟨text⟩`, když je odkaz nedefinován, jinak vytiskne `⟨text⟩` prostřednictvím makra `\citelink`.

opmac.tex

```

1452: \def\ecite[#1]{\bgroup\citeA#1,,,\expandafter\eciteB\savedcites;}
1453: \def\eciteB#1,#2;#3{\if?#1\relax #3\else \citelink{#1}{#3}\fi\egroup}

```

Následuje kód makra `\bib` [`⟨lejblík⟩`]. Nejprve je ošetřeno, zda je použit zkrácený nebo rozšířený zápis `\bib` [`⟨lejblík⟩`]<sub>□=□</sub>{`⟨značka⟩`}. Případná mezera před rovnítkem je odstaněna pomocí triku s `\romannumeral`, který při záporném čísle expanduje na prázdný výsledek, ale případná mezera za `'\.` při skenování tohoto čísla je pozřena. Při zkráceném zápisu makra `\bib` (bez rovnítko) se zavolá `\bibB` s prázdným `\bibmark`, v druhém případě se `\bibmark` nejprve naplní prostřednictvím makra `\bibA`. Makro `\bibB` vloží prostřednictvím `\wbib` [`⟨lejblík⟩`]{`⟨číslo-citace⟩`}{`⟨značka⟩`} do REF souboru propojené údaje o tom, jaké má `⟨lejblík⟩` přiřazeno `⟨číslo-citace⟩` v seznamu literatury. Makro `\tmpb` je naplněno `⟨lejblíkem⟩` pro případné použití v `\dest` (při draft módu) nebo pro použití v makru `\printbib`. Makro `\wbib` připojí před `\wref` příkaz `\immediate`, aby byly zapsány do REF souboru aktuální hodnoty parametrů.

opmac.tex

```

1455: \def\bib[#1]{\def\tmpb{\isnextchar={\bibA[#1]}{\bibmark={}\bibB[#1]}}%
1456:   \expandafter\tmpb\romannumeral-'.' % ignore optional space
1457:   \def\bibA[#1]=#2{\bibmark={#2}\bibB[#1]}
1458:   \def\bibB[#1]{\par \ifnum\bibnum>0 \bibs\skip \fi
1459:     \advance\bibnum by1
1460:     \noindent \def\tmpb{#1}\wbib{#1}{\the\bibnum}{\the\bibmark}%
1461:     \printbib \ignorespaces
1462: }
1463: \def\wbib#1#2#3{\dest[cite:\the\bibnum]%
1464:   \ifx\wref\wrefrelax\else \immediate\wref\Xbib{#1}{#2}{#3}\fi}

```

Makro `\Xbib` pracuje při čtení souboru REF a dělá to, co jsme si řekli už dříve: nastaví hodnotu makra `\bib:⟨lejblík⟩` na `\bibnn{⟨číslo-citace⟩&}`. Dále definuje `\bim:⟨číslo-citace⟩` jako třetí parametr, který je při použití `\bib` prázdný, ale při čtení \*.bbl souboru vygenerovaného pomocí `alpha.bst` nebo `apalike.bst` tam bude uložena `⟨značka⟩`. Dále `\Xbib` definuje `\lastbibnum` jako `⟨číslo-citace⟩`, takže po přečtení REF souboru obsahuje největší použité `⟨číslo-citace⟩`. To se může hodit, pokud designér chce odsadit seznam literatury podle šířky největšího čísla citace.

opmac.tex

```

1466: \def\Xbib#1#2#3{\sdef{bib:#1}{\bibnn{#2}&}\if^#3^{\else\sdef{bim:#2}{#3}\fi\def\lastbibnum{#2}}

```

Makro `\printbib` se vloží na začátek každého záznamu v seznamu literatury. Implicitně vytiskne `\the\bibnum` v hranaté závorce a při `\nonumcitations` netiskne nic. V obou případech nastaví odsazení druhého a dalších řádků odstavce na `\iindent`. Designér si může toto makro předefinovat dle svého uvážení.

opmac.tex

```

1468: \def\printbib{\hangindent=iindent
1469:   \ifx\citelinkA\empty \noindent\hskip\iindent \llap{[\the\bibnum] }%
1470:   \else \noindent \fi
1471: }

```

---

```

\nonumcitations: 48, 51, 54   \citelinkA: 50–51, 54   \etalchar: 51   \ecite: 51   \eciteB: 51
\bib: 34, 48, 51   \bibA: 51   \bibB: 51   \wbib: 51, 53   \Xbib: 51   \lastbibnum: 51
\printbib: 51, 53

```

Makro `\addcitetlist`  $\langle\text{lejblík}\rangle$  přidá do `\citetlist` údaj ve tvaru `\citeI`  $[\langle\text{lejblík}\rangle]$ . Hranaté závorky jsou použity proto, aby fungoval test `\isinlist\citetlist{[\langle\text{lejblík}\rangle]}`. Jak uvidíme za chvíli, makro `\addcitetlist` změní během činnosti makra `\usebibtex` svůj význam na `\writeaux`, aby případné použití `\cite` až za `\usebibtex` rovnou zapisovalo do AUX souboru. Podobně makro `\addcitetlist` změní v makru `\usebbl` svůj význam `\writeXcite`  $\langle\text{lejblík}\rangle$ , aby v příštím průchodu  $\text{\TeX}$ em mělo makro `\usebbl` přehled i o výskytech `\cite`, které jsou napsány později, než `\usebbl`.

opmac.tex

```
1473: \def\addcitetlist#1{\global\addto\citetlist{\citeI[#1]}}
1474: \def\writeaux#1{\immediate\write\auxfile{\string\citation{#1}}}
1475: \def\writeXcite#1{\openref\immediate\wref\Xcite{#1}}
1476: \def\citetlist{} \def\citetlistB{}
```

Než se pustíme do výkladu maker `\usebibtex`, `\genbbl` a `\usebbl`, uvedeme stručně popis činnosti  $\text{Bib}\text{\TeX}$ u. Příkaz `bibtex`  $\langle\text{dokument}\rangle$  způsobí, že program `bibtex` se podívá do souboru  $\langle\text{dokument}\rangle$ .aux a tam si všimá sekvencí `\bibdata`  $\langle\text{bib-báze}\rangle$ , `\bibstyle`  $\langle\text{bib-style}\rangle$  a `\citation`  $\langle\text{lejblík}\rangle$ . Na základě toho následně přečte soubor  $\langle\text{bib-báze}\rangle$ .bib se zdrojovými zápisy bibliografických údajů. Pro konverzi těchto zdrojových zápisů do výstupního souboru  $\langle\text{dokument}\rangle$ .bbl použije stylový soubor  $\langle\text{bib-style}\rangle$ .bst. Není-li mezi sekvencemi `\citation` uvedeno `\citation{*}`, program `bibtex` zahrne do výstupu jen ty bibliografické údaje, které mají  $\langle\text{lejblík}\rangle$  shodný s některým z  $\langle\text{lejblíků}\rangle$  uvedených v parametrech sekvencí `\citation`. Každá sekvence `\citation`  $\langle\text{lejblík}\rangle$  v souboru  $\langle\text{dokument}\rangle$ .aux typicky odpovídá jednomu použití příkazu `\cite`  $\langle\text{lejblík}\rangle$ .

Makro `\usebibtex`  $\langle\text{bib-báze}\rangle$   $\langle\text{bst-styl}\rangle$  otevře soubor AUX prostřednictvím `\openauxfile`  $\langle\text{bib-báze}\rangle$   $\langle\text{bst-styl}\rangle$ . Napíše tam tedy požadovaná data pro  $\text{Bib}\text{\TeX}$ . Dále z `\citetlist` přepíše do AUX souboru lejblíky ve formátu `\citation`  $\langle\text{lejblík}\rangle$ . Nakonec se uvnitř skupiny pustí do čtení souboru BBL prostřednictvím makra `\readbblfile`.

opmac.tex

```
1478: \def\usebibtex#1#2{%
1479:   \openref \openauxfile{#1}{#2}%
1480:   \def\citeI[##1]{\writeaux{##1}}\citetlist
1481:   \global\let\addcitetlist=\writeaux
1482:   \bgroup \readbblfile{\jobname}\egroup
1483: }
1484: \def\openauxfile#1#2{%
1485:   \immediate\openout\auxfile=\jobname.aux
1486:   \immediate\write\auxfile
1487:     {\percent\percent\space Opmac: AUX file reserved for bibtex only}%
1488:   \immediate\write\auxfile{\string\bibdata{#1}}%
1489:   \immediate\write\auxfile{\string\bibstyle{#2}}%
1490: }
```

Makro `\readbblfile`  $\langle\text{soubor}\rangle$  vyzkouší, zda je  $\langle\text{soubor}\rangle$ .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  $\text{La}\text{\TeX}$ ových konstrukcí, které se typicky v BBL souborech používají. Nastaví `\leftskip` na `\iindent` a spustí `\bibtexhook`. Konečně načte soubor BBL.

opmac.tex

```
1491: \def\readbblfile #1{%
1492:   \openin\testin=#1.bbl
1493:   \ifeof\testin
1494:     \opwarning{The '#1.bbl' file doesn't exist. Use 'bibtex'..}%
1495:   \else
1496:     \closein\testin
1497:     \bibnum=0
1498:     \long\def\begin##1\bibitem{\bibitem}\def\end##1{}% LaTeX environment
1499:     \def\httpAddr##1{\url{http:##1}}\def\{\hfill\break}%
1500:     \def\newblock{\hskip .11em plus.33em minus.07em}%
1501:     \def\mbox{\leavevmode\hbox}\def\emph##1{\it##1}%
1502:     \parindent=\iindent \bibtexhook\relax
1503:     \input #1.bbl
1504:     \par
1505:   \fi
```

---

`\addcitetlist`: 49, 52–54    `\citetlist`: 48, 52–54    `\citeI`: 52–54    `\writeaux`: 52  
`\writeXcite`: 52–54    `\bibdata`: 52    `\bibstyle`: 52    `\citation`: 52–53    `\usebibtex`: 8, 48, 52  
`\openauxfile`: 52–53    `\readbblfile`: 52–54

```
1506: }
```

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 může 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}{číslo-citace}{značka}`. Makro `\tmpb` je naplněno *lejblíkem* pro případné použití v `\dest` (při draft módu) nebo pro použití v makru `\printbib`.

opmac.tex

```
1507: \def\bibitem{\isnextchar[{\bibitemB}{\bibmark={}\bibitemC}}
1508: \def\bibitemB[#1]{\bibmark=#1}\bibitemC}
1509: \def\bibitemC##1{\bibitemD{##1}}
1510: \def\bibitemD#1{\par\ifnum\bibnum>0 \bibs skip \fi
1511: \advance\bibnum by1
1512: \noindent \def\tmpb{##1}\wbib{##1}{\the\bibnum}{\the\bibmark}%
1513: \printbib \ignorespaces
1514: }
```

Makro `\genbbl` {*bib-báze*}{*bst-style*} otevře AUX soubor a zapíše do něj údaje potřebné pro BibTeX včetně `\citation{*}`. Poté se makro pokusí přecíst 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

```
1515: \def\genbbl#1#2{\openauxfile{#1}{#2}%
1516: \immediate\write\auxfile{\string\citation{*}}%
1517: \bgroup
1518: \iindent=4em
1519: \def\bibitemC##1{\par\ifnum\bibnum>0 \bibs skip \fi
1520: \advance\bibnum by1
1521: \noindent \hangindent=\parindent
1522: \indent \llap{##1}\enspace}\ignorespaces
1523: }%
1524: \readbblfile{\jobname}%
1525: \egroup
1526: }
```

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 `\citeI` [*lejblík*] promění v `\bb:lejblík`, takže se záznam vytiskne. Nyní ale v pořadí, v jakém jsou *lejblíky* zařazeny do `\citelist`.

opmac.tex

```
1527: \def\usebbl/#1 #2 {\isdefined\bbl:#1}%
1528: \iftrue \csname bbl:#1\endcsname {#2}\else
1529: \opwarning{\string\usebbl/#1 #2 ... the '#1' type undefined}%
1530: \fi
1531: }
1532: \sdef\bbl:a#1{\bgroup \readbblfile{#1}\egroup}
1533:
1534: \sdef\bbl:b#1{\bgroup
1535: \let\citeI=\relax \xdef\citelist{\citelist\citelistB}%
1536: \def\bibitemC##1 ##2\par{%
1537: \isinlist\citelist{##1}}\iftrue \bibitemD{##1}##2\par\fi}%
1538: \readbblfile{#1}%
1539: \global\let\addcitelist=\writeXcite
1540: \egroup
1541: }
1542: \sdef\bbl:c#1{\bgroup
```

`\bibitem`: 34, 48, 52–53    `\bibitemB`: 53–54    `\bibitemC`: 53–54    `\bibitemD`: 53–54    `\genbbl`: 52–53  
`\usebbl`: 8, 48, 52–54

```

1543: \ifx\citelinkA\empty \else
1544: \opwarning{\string\nonumcitations: don't use \string\usebbl/c}\fi
1545: \let\citeI=\relax \xdef\citelist{\citelist\citelistB}%
1546: \def\bibitemC##1 ##2\par{%
1547: \isinlist\citelist{##1}}\iftrue
1548: \if^{\the\bibmark}\sdef{bb:##1}{\bibitemD{##1}##2\par}%
1549: \else \toks0={##2\par}%
1550: \edef\tmpa{\noexpand\sdef{bb:##1}{\the\bibmark have to expand
1551: \noexpand\bibitemB[\the\bibmark]{##1}\the\toks0}}\tmpa
1552: \fi\fi}%
1553: \readbblfile{##1}%
1554: \def\bibitemC##1{\bibitemD{##1}}%
1555: \def\citeI[##1]{\csname bb:##1\endcsname}\citelist
1556: \global\let\addcitelist=\writeXcite
1557: \egroup
1558: }

```

Za zmínku stojí ještě práce uvedených maker s `\citelist`. Před výskytem makra `\usebbl` se lejbličky z `\cite` a `\nocite` hromadí v `\citelist`. Ovšem další `\cite` a `\nocite` se mohou vyskytovat za příkazem `\usebbl`. Pokud se tak stane, pracuje `\addcitelist` nyní ve významu `\writeXcite` a uloží potřebnou informaci do REF souboru. Při dalším  $\TeX$ ování se tato informace přečte makrem `\Xcite` `{\lejblik}` z REF souboru takto:

```

1559: \def\Xcite#1{\addto\citelistB{\citeI[#1]}}

```

opmac.tex

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`.

Makro `\usebib` je definováno v souboru maker (modulu) `opmac-bib.tex`. Tuto sadu maker není účelné zahrnout přímo do OPmac, protože je závislá na externím balíčku `librarian.tex`. Soubor maker tedy zavádíme až v případě, že uživatel skutečně použil makro `\usebib`. Je použit stejný trik, jako v případě makra `\fontfam`.

```

1561: \def\usebib{\par \input opmac-bib \usebib}

```

opmac.tex

Uživatel nicméně může makro soubor na začátku svého dokumentu volat explicitně pomocí `\inputoopmac-bib`.

### 3.24 Úprava output rutiny

OPmac mění output rutinu proti originální `\plainoutput` jen v nejnútnejších věcech. Řeší následující problémy:

- Místo přímého `\shipout` nechá nejprve box sestavit jako `\box0`, pak provede `\protectlist` a pak provede `\shipout\box0`. Tím jsou zabezpečeny tzv. protektované příkazy při `\write`.
- Pomocí `\ensureblacko` jsou řešeny barvy záhlaví, zápatí, `\topins` a `\footins`.
- Je vložen `\pghook` po sestavení boxů, ale před `\shipout`. Implicitně je `\pghook` prázdný. Mění jej makro `\margins` pro účely pravolevého střídání okrajů.
- Makro `\pagecontents` obsahuje navíc `\prepage` (kvůli odkazům na stránku).

Místo původního makra `\plainoutput` používá OPmac makro `\opmacoutput`, která je obklopeno makry `\begoutput` a `\endoutput`. Makro `\begoutput` zapíše do REF souboru údaj o čísle strany a předefinuje makra, která se mohou vyskytnout v záhlaví či zápatí stránky, pokud od nich chceme, aby se chovaly jinak než obvykle. Makro `\endoutput` je prázdné a je určeno pro strýčka Příhodu. Makro `\prephoffset` je rovněž implicitně prázdné, spouští se v `\begoutput` a může v něm být nastaveno střídání okrajů pro liché a sudé stránky, viz též makro `\margins`.

```

1565: \output={\begoutput \opmacoutput \endoutput}
1566: \def\begoutput{%
1567: \immediate\wref\Xpage{{\the\pageno}}%
1568: \def\nl{ } \def\fnote##1{\def\fnotemark##1{}}%
1569: \prephoffset
1570: }

```

opmac.tex

---

`\Xcite`: 52, 54    `\usebib`: 54    `\begoutput`: 54    `\endoutput`: 54–55    `\prephoffset`: 54–56

```
1571: \def\endoutput{} \def\prephoffset{}
```

Makro `\opmacoutput` se chová analogicky, jako `\plainoutput`. Rozdíl je v tom, že nejprve sestaví celou stranu do boxu0 a v té době expandují makra v `\headline` a `\footline`. Pak spustí `\pghook` a `\protectlist`. Makro `\protectlist` nastaví díky `\doprotect` kontrolní sekvence označené jako `\addprotect<sekvence>` na `\relax`, takže během `\shipout` (tedy během expanze záznamů `\write`) se nebudou expandovat. Další činnost je zcela shodná s činností makra `\plainoutput`.

opmac.tex

```
1573: \def\opmacoutput{%
1574:   \setbox0=\vbox{\prepghook\ensureblacko{\makeheadline}\pagebody\ensureblacko{\makefootline}}%
1575:   \pghook \protectlist
1576:   \shipout\box0 \advancepageno
1577:   \ifnum\outputpenalty>-20000 \else\dosupereject\fi
1578: }
1579: \def\doprotect#1{\let#1=\relax}
```

Barvy jsou v textu nastaveny pomocí `\pdfcolorstack`, takže na začátku následující strany začíná barva, která skončila na straně předchozí. My ale nechceme, aby barva textu ovlivnila barvu záhlaví a zápatí. Proto je sazba `\makeheadline` a `\makefootline` realizována pomocí makra `\ensureblacko`.

Makro `\prepage` se spustí na začátku `\pagecontents` a zajistí uložení cíle pro odskok podle čísla strany. Makra `\preboxcclv` a `\postboxcclv` se spustí na začátku a na konci sazby boxu 255, jsou prázdná a zůstávají v kódu pro zachování zpětné kompatibility.

opmac.tex

```
1580: \def\prepage{\def\destheight{25pt}\dest[pg:\pgilabel.\the\pageno]}
1581: \def\preboxcclv{} \def\postboxcclv{}
```

OPmac předefinováá makro `\pagecontents` z plainTeXu tak, že přidává makra `\prepage`, `\preboxcclv` a `\postboxcclv`. Také obsah boxů `\topins` a `\footins` tiskne pomocí `\ensureblacko`.

opmac.tex

```
1583: {\catcode'\@=11
1584: \gdef\pagecontents{\prepage % dest of pageno
1585:   \ifvoid\topins\else\ensureblacko{\unvbox\topins}\fi
1586:   \preboxcclv
1587:   \dimen@=\dp@cclv \unvbox@cclv % open up \box255
1588:   \postboxcclv
1589:   \ifvoid\footins\else % footnote info is present
1590:     \vskip\skip\footins
1591:     \ensureblacko{\footnoterule \unvbox\footins}\fi
1592:   \ifr@ggedbottom \kern-\dimen@ \vfil \fi
1593: }}
```

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

```
1595: \footline={\hss\tenrm\thefontsize[10]\folio\hss}
```

Makro `\Xpage` z REF souboru nastavuje `\lastpage` a `\fnotenumlocal`. S těmito registry také spolupracují makra `\Xlabel`, `\Xmnote` a `\Xfnote`.

opmac.tex

```
1597: \newcount\lastpage \lastpage=0 % the last page of the document
1598: \def\Xpage#1{\lastpage=#1 \fnotenumlocal=0 }
```

### 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

```
1602: \newdimen\pgwidth \newdimen\pgheight \pgwidth=0pt
1603: \newdimen\shiftoffset
```

---

```
\opmacoutput: 33, 54–55   \doprotect: 4, 55   \prepage: 54–55   \preboxcclv: 55
\postboxcclv: 55       \pagecontents: 54–55   \Xpage: 46–47, 54–55   \lastpage: 14, 47, 55
\pgwidth: 55–57       \pgheight: 55–57   \shiftoffset: 55–56
```



Makro `\margin`  $\langle typ \rangle \langle formát \rangle \langle levý \rangle, \langle pravý \rangle, \langle horní \rangle, \langle dolní \rangle \langle jednotka \rangle$  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 \langle okraj \rangle` 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 * \langle levý \rangle` což dá stejnou hodnotu jako  $\langle pravý \rangle - \langle levý \rangle$ . 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

```

1605: \def\margin/#1 #2 (#3,#4,#5,#6)#7 {\def\tmp{#7}%
1606:   \ifx\tmp\empty
1607:     \opwarning{\string\margin: missing unit, mm inserted}\def\tmp{mm}\fi
1608:     \addto\tmp{\relax}%
1609:     \setpagedimens #2 % setting \pgwidth, \pgheight
1610:     \ifdim\pgwidth=0pt \else
1611:       \hoffset=-1\trueunit in \voffset=-1\trueunit in
1612:       \if$#3$\if$#4$\tmpdim=\pgwidth \advance\tmpdim -\hsize
1613:         \divide\tmpdim by2 \advance\hoffset \tmpdim % left=right
1614:       \else \rbmargin\hoffset\hsize{#4\tmp}% only right margin
1615:       \fi
1616:     \else \if$#4$\advance\hoffset #3\tmp % only left margin
1617:       \else \hsize=\pgwidth % left+right margin
1618:       \advance\hsize -#3\tmp \advance\hsize -#4\tmp
1619:       \advance\hoffset #3\tmp
1620:     \fi\fi
1621:     \if$#5$\if$#6$\tmpdim=\pgheight \advance\tmpdim -\vsize
1622:       \divide\tmpdim by2 \advance\voffset \tmpdim % top=bottom
1623:     \else \rbmargin\voffset\vsize{#6\tmp}% only bottom margin
1624:     \fi
1625:     \else \if$#6$\advance\voffset #5\tmp % only top margin
1626:       \else \vsize=\pgheight % top+bottom margin
1627:       \advance\vsize -#5\tmp \advance\vsize -#6\tmp
1628:       \advance\voffset #5\tmp
1629:     \fi\fi
1630:     \if 1#1\shiftoffset=0pt \def\prephoffset{}\else \if 2#1% double-page layout
1631:       \shiftoffset=\pgwidth \advance\shiftoffset -\hsize
1632:       \advance\shiftoffset -2\hoffset \advance\shiftoffset -2in
1633:       \def\prephoffset{\ifodd\pageno \else \advance\hoffset \shiftoffset \fi}%
1634:     \else \opwarning{use \string\margin/1 or \string\margin/2}%
1635:     \fi\fi\fi
1636: }
1637: \def\rbmargin#1#2#3{\advance#1\pgwidth \advance#1-#2 \advance#1-#3}

```

Makro `\setpagedimens`  $\langle formát \rangle$  spustí `\setpagedimensB`  $\langle šířka \rangle, \langle výška \rangle \langle jednotka \rangle$ , pokud je prvním znakem  $\langle formátu \rangle$  závorka, jinak spustí `\setpagedimensA`, což je makro, které použije definovaný formát, ten expanduje a zavolá `\setpagedimensB`. Pomocné makro `\setpagedimensC`  $\langle reg \rangle = \langle num \rangle : \langle jednotka \rangle$  přiřadí do  $\langle reg \rangle$  daný rozměr.

opmac.tex

```

1639: \def\setpagedimens{\isnextchar{\setpagedimensB}{\setpagedimensA}}
1640: \def\setpagedimensA#1 {\isdefined\pgs:#1}\iftrue
1641:   \expandafter\expandafter\expandafter\setpagedimensB \csname \pgs:#1\expandafter\endcsname\space
1642: \else \opwarning{page specification "#1" is undefined}\fi
1643: \def\setpagedimensB (#1,#2)#3 {\setpagedimensC\pgwidth=#1:#3 \setpagedimensC\pgheight=#2:#3
1644:   \ifx\pdfpagewidth\undefined \else
1645:     \pdfpagewidth=\pgwidth \pdfpageheight=\pgheight \fi}
1646: \def\setpagedimensC #1=#2:#3 {#1=#2\ifx`#3`^tmp\else#3\fi\relax\truedimen#1}

```

---

`\margin`: 54, 56    `\rbmargin`: 56    `\setpagedimens`: 55–56    `\setpagedimensB`: 56  
`\setpagedimensA`: 56–57    `\setpagedimensC`: 56



Jednotlivé (*formáty*) papíru je potřeba deklarovat.

```
1648: \sdef{pgs:a3}{(297,420)mm} \sdef{pgs:a4}{(210,297)mm} \sdef{pgs:a5}{(148,210)mm}
1649: \sdef{pgs:a3l}{(420,297)mm} \sdef{pgs:a4l}{(297,210)mm} \sdef{pgs:a5l}{(210,148)mm}
1650: \sdef{pgs:b5}{(176,250)mm} \sdef{pgs:letter}{(8.5,11)in}
```

opmac.tex

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`.

```
1652: \def\trueunit{}
1653: \def\magscale[#1]{\mag=#1\def\trueunit{true}%
1654:   \ifdim\pgwidth=0pt \else \truedimen\pgwidth \truedimen\pgheight \fi
1655:   \ifx\pdfpagewidth\undefined \else
1656:     \truedimen\pdfpagewidth \truedimen\pdfpageheight
1657:     \ifx\pdfhorigin\undefined\else
1658:       \pdfhorigin=1truein \pdfvorigin=1truein % Origin is independent off \mag
1659:     \fi\fi}
1660: \def\truedimen#1{\ifx\trueunit\empty \else#1=\expandafter\ignorept\the#1truept \fi}
```

opmac.tex

### 3.26 Závěr

V případě, že je použit XeTeX, načteme dodatečná makra ze souboru `opmac-xetex.tex`. Tato makra nahrazují některá makra z OPmac XeTeX-specifickou variantou nebo emulují pdfTeXové primitivy. V případě, že je použit nový LuaTeX, načteme makra `opmac-luatex.tex`, která rekonstruují pdfTeXové primitivy dle původního významu. Nakonec pomocí `\inputref` přečteme REF soubor (pokud existuje) a vrhneme se na zpracování dokumentu, který nám připravil uživatel. Přeji dobré pořízení.

```
1664: \ifxXeTeXversion\undefined \else \pdftruefalse \input opmac-xetex \fi
1665: \ifxpdfextension\undefined \else \input opmac-luatex \fi
1666: \inputref
1667: \endinput
```

opmac.tex

## 4 Rejstřík

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

<code>\activettchar</code> : 39, 40	<code>\begmulti</code> : 30, 7
<code>\addcitelist</code> : 52, 49, 53–54	<code>\begoutput</code> : 54
<code>\additcorr</code> : 11	<code>\begtt</code> : 38, 7, 39, 41
<code>\addoneol</code> : 37	<code>\bfshape</code> : 16, 17, 21
<code>\addprotect</code> : 4, 6, 8, 11, 33, 36–37, 55	<code>\bib</code> : 51, 34, 48
<code>\addtabdata</code> : 42, 43–44	<code>\bibA</code> : 51
<code>\addtabitem</code> : 42, 43	<code>\bibB</code> : 51
<code>\addtabvrule</code> : 42, 43	<code>\bibdata</code> : 52
<code>\addto</code> : 4, 6, 20, 22, 25, 29–30, 33, 41–43, 46, 49, 52, 54, 56	<code>\bibitem</code> : 53, 34, 48, 52
<code>\adef</code> : 4, 5, 20, 39–41	<code>\bibitemB</code> : 53, 54
<code>\afteritcorr</code> : 11	<code>\bibitemC</code> : 53, 54
<code>\afternoindent</code> : 18, 39	<code>\bibitemD</code> : 53, 54
<code>\asciisorting</code> : 27	<code>\bibmark</code> : 47, 48, 51, 53–54
<code>\athe</code> : 20	<code>\bibnn</code> : 49, 51
<code>\auxfile</code> : 47, 48, 52–53	<code>\bibnum</code> : 47, 48, 51–53
<code>\balancecolumns</code> : 31, 30, 32	<code>\bibskip</code> : 7, 51, 53
<code>\baselineskipB</code> : 11, 10	<code>\bibstyle</code> : 52
<code>\begitem</code> : 20, 7	<code>\bibtexhook</code> : 8, 52
	<code>\Black</code> : 32

---

`\magscale`: 57    `\trueunit`: 56–57    `\truedimen`: 56–57

\Blue: **32**  
\Brown: **32**  
\bslash: **6**, 36  
\caption: **19**, 8  
\captionhook: **8**, 19  
\chap: **16**, 8, 17  
\chapfont: **16**, 15  
\chaphook: **8**, 17, 46  
\chapnum: **16**, 17  
\citation: **52**, 53  
\cite: **48**, 50, 52, 54  
\citeA: **48**, 49, 51  
\citeB: **50**, 49  
\citeI: **52**, 53–54  
\citelink: **35**, 50–51  
\citelinkA: **51**, 50, 54  
\citelist: **52**, 48, 53–54  
\citesep: **50**  
\cnvhook: **8**, 37  
\colnum: **41**, 42–43  
\colorstackpop: **33**  
\colorstackpush: **33**  
\colorstackset: **33**  
\colsep: **8**, 30  
\corrsize: **30**, 31  
\crl: **43**  
\crli: **43**, 41  
\crll: **43**  
\crlli: **43**  
\CS: **8**  
\csplain: **8**  
\currentcolor: **32**, 33  
\currii: **25**  
\Cyan: **32**  
\dditem: **43**, 41  
\ddlinedata: **41**, 42–43  
\dest: **34**, 14, 17, 35, 51, 53, 55  
\destactive: **34**, 35  
\destbox: **34**  
\destheight: **34**, 17, 55  
\dgsize: **9**, 10–11  
\dnum: **19**, 17  
\doprotect: **55**, 4  
\dosorting: **29**, 24, 27  
\dotocnum: **17**, 15–16, 18  
\dotocnumafter: **16**, 17–18  
\dovertbininput: **40**, 41  
\draft: **33**, 34  
\draftbox: **33**  
\ecite: **51**  
\eciteB: **51**  
\em: **11**, 8, 52  
\enditems: **20**, 7  
\endmulti: **30**, 7  
\endoutput: **54**, 55  
\ensureblacko: **33**, 32, 54–55  
\ensureblackoA: **33**  
\eoldef: **5**, 16–17  
\eoldefA: **5**  
\eqmark: **19**  
\etalchar: **51**  
\everyii: **25**  
\firstdata: **23**, 22, 24, 28  
\firstdataA: **23**  
\firstnoindent: **18**, 15–16  
\fixmnotes: **47**  
\fnmarkx: **46**  
\fnote: **46**, 8, 54  
\fnotehook: **8**, 46  
\fnotemark: **46**, 54  
\fnotenum: **46**, 13  
\fnotenumlocal: **46**, 55  
\fnotetext: **46**  
\fnum: **19**, 17  
\fontdim: **9**, 10–11  
\fontdimB: **11**, 10  
\fontfam: **11**, 12, 54  
\fontscalex: **10**, 9  
\fontsize: **9**, 10  
\frame: **44**  
\fullrectangle: **20**  
\genbbl: **53**, 52  
\gobbletoend: **29**, 30  
\Green: **32**  
\Grey: **32**  
\hhkern: **7**, 43–44  
\hyperlinks: **35**, 34, 36  
\ifAleB: **28**, 25, 29  
\ifischap: **20**, 21  
\ifpdfTeX: **4**, 33–35, 38, 44–45  
\ignorept: **9**, 8, 10–11, 45, 57  
\ii: **21**, 22  
\iiA: **21**  
\iiatsign: **21**  
\iiB: **21**, 22  
\iiC: **21**, 22  
\iid: **22**  
\iid: **22**  
\iiemdash: **25**  
\iiendash: **23**, 22  
\iiilist: **22**, 24, 29–30  
\iiindent: **7**, 19–21, 24–25, 51–53  
\iiindex: **21**, 22  
\iiis: **25**, 24  
\iiiskip: **7**, 20  
\iiispeclist: **25**, 24  
\inputref: **13**, 57  
\insertmark: **18**, 15–16  
\insertoutline: **38**  
\inspic: **44**  
\inspicpage: **44**  
\intthook: **7**, 39

**\isAleB:** 28, 25, 29  
**\isdefined:** 5, 14, 19, 22, 27, 35, 37–38, 46–47, 49, 51, 53, 56  
**\isinlist:** 5, 24, 42, 52–54  
**\isnextchar:** 5, 51, 53, 56  
**\isnextcharA:** 5  
**\isolangset:** 12  
**\itemhook:** 7, 20  
**\itemnum:** 19, 20  
**\label:** 14, 13, 34  
**\lastbibnum:** 51  
**\lastcitenum:** 47, 48–51  
**\lastlabel:** 14  
**\lastpage:** 55, 14, 47  
**\LaTeX:** 8  
**\LightGrey:** 32, 33  
**\linecolor:** 32  
**\link:** 34, 35  
**\linkactive:** 34, 35  
**\localcolor:** 32, 33, 35  
**\localcolorfalse:** 32  
**\localcolortrue:** 32  
**\locfnum:** 46  
**\longlocalcolor:** 32, 33  
**\Magenta:** 32  
**\magscale:** 57  
**\magstep:** 11, 16  
**\makecolumns:** 30, 31  
**\makeindex:** 24, 25, 28  
**\maketoc:** 21  
**\margins:** 56, 54  
**\maybebreak:** 5  
**\maybebreakA:** 5  
**\mergesort:** 29, 30  
**\mnote:** 47, 8  
**\mnoteA:** 47  
**\mnotehook:** 8, 47  
**\mnoteindent:** 8, 47  
**\mnotenum:** 46, 13, 47  
**\mnotesfixed:** 47  
**\mnotesize:** 8, 47  
**\mnoteskip:** 46, 47  
**\mspan:** 43, 44  
**\mspanA:** 43, 44  
**\mspanB:** 43, 44  
**\mtext:** 12, 15, 19  
**\mullines:** 31, 30, 32  
**\multiskip:** 7, 30  
**\nbparg:** 18, 15–16, 19  
**\nl:** 18, 54  
**\nocite:** 48, 51, 54  
**\nonum:** 16, 15, 17–18, 21  
**\nonumcitations:** 51, 48, 54  
**\nonumnum:** 16, 17  
**\norempenalty:** 18, 15–16  
**\normalitem:** 20  
**\notoc:** 16, 17–18  
**\openauxfile:** 52, 53  
**\openref:** 13, 14, 21, 36, 46–47, 49, 52  
**\openrefA:** 13  
**\OPmac:** 8  
**\opmacoutput:** 55, 33, 54  
**\OPmacversion:** 3, 4  
**\opwarning:** 4, 9, 14, 18–19, 21, 24, 27, 34–36, 38–39, 42, 44, 46–47, 49, 51–54, 56  
**\orihrule:** 44  
**\orivrule:** 44  
**\othe:** 18, 17  
**\oulnum:** 38  
**\outlinelevel:** 37, 36  
**\outlines:** 36, 37–38  
**\outlinesA:** 36, 37  
**\outlinesB:** 37, 36  
**\outlinesC:** 37  
**\pagecontents:** 55, 54  
**\paramtabdeclarep:** 42  
**\pdfblackcolor:** 33  
**\pdfborder:** 35, 34  
**\pdfrotate:** 45, 33  
**\pdfrotateA:** 45  
**\pdfscale:** 44, 33  
**\percent:** 6, 13, 52  
**\pgfolioA:** 24, 22, 35  
**\pgfolioB:** 24, 22  
**\pgheight:** 55, 56–57  
**\pghook:** 8, 54–56  
**\pgilabel:** 35, 55  
**\pglink:** 35, 14, 21  
**\pgref:** 14  
**\pgwidth:** 55, 56–57  
**\picdir:** 8, 44  
**\picheight:** 44  
**\picw:** 44  
**\picwidth:** 44  
**\postboxcclv:** 55  
**\preboxcclv:** 55  
**\prepage:** 55, 54  
**\prepresorting:** 28, 24, 29  
**\prepresortingA:** 28  
**\prepresortingB:** 28  
**\prepghook:** 8, 33, 55  
**\prephoffset:** 54, 55–56  
**\prepii:** 24  
**\prepiiA:** 24  
**\prepii:** 25  
**\printbib:** 51, 53  
**\printcaption:** 19  
**\printchap:** 15, 14, 16–18  
**\printcite:** 50  
**\printdashcite:** 50, 49  
**\printii:** 25, 24

\printiiA: 25  
\printiipages: 24  
\printitem: 20  
\printsavedcites: 49, 48  
\printsec: 15, 14, 16–18  
\printsecc: 15, 14, 16–18  
\printttline: 39, 41  
\protectlist: 4, 14, 37, 54–55  
\ptthook: 7, 41  
\ptunit: 9, 10–11  
\rbmargin: 56  
\rcite: 48  
\readbblfile: 52, 53–54  
\Red: 32  
\ref: 14, 13  
\reffile: 12, 13  
\reflink: 35, 14  
\REFversion: 13  
\regfont: 8, 9  
\regtfm: 9  
\remskip: 18, 15–16  
\remskipamount: 18  
\replacestrings: 6, 7, 28, 36  
\replacestringsA: 6  
\replacestringsB: 6  
\resetnonunotoc: 17  
\resizeall: 8, 9–10  
\resizefont: 8, 9–11  
\rulewidth: 44  
\rulewidthA: 44  
\runningfnotes: 46  
\savedcites: 48, 49–51  
\savedttchar: 39, 41  
\savedttcharc: 39  
\scalebaselineskip: 10, 9  
\scanprevii: 25  
\scantabdata: 42, 41  
\scantabdataA: 42  
\scantabdataB: 42  
\scantabdataC: 42  
\scantabdataD: 42  
\sdef: 4, 12–14, 20, 25, 51, 53–54, 57  
\sec: 16, 8, 17  
\secc: 16, 8, 17  
\seccfont: 16  
\secchok: 8, 17  
\seccnum: 16, 17  
\secfont: 16, 15  
\sechok: 8, 17  
\secnum: 16, 17  
\seconddata: 23, 22, 24  
\seconddataA: 23  
\setbaselineskip: 10, 9, 11  
\setcmkcolor: 32, 33  
\setcnvcodesA: 38, 37  
\setignoredchars: 26, 27–28  
\setlccodes: 38, 26, 37  
\setpagedimens: 56, 55  
\setpagedimensA: 56, 57  
\setpagedimensB: 56  
\setpagedimensC: 56  
\setprimarysorting: 27, 24, 28  
\setprimarysortingA: 27  
\setsecondarysorting: 28, 29  
\setverb: 38, 39–41  
\shiftoffset: 55, 56  
\shortcitations: 50, 48, 51  
\sizespec: 8, 9–11  
\skiptorelax: 39, 49  
\slantcorr: 8  
\slet: 4  
\smallcos: 45  
\smallsin: 45  
\sortcitations: 49, 51  
\sortcitesA: 49, 51  
\sortcitesB: 49, 50  
\sortcitesC: 50, 49  
\sortcitesD: 50  
\sortingdata: 26, 25, 27–28  
\sortingmessage: 27, 29  
\sortreturn: 29, 30  
\specsortingdata: 27, 28  
\specsortingdatacs: 26  
\specsortingdatask: 26  
\splitpart: 30, 31  
\startitem: 20, 7  
\style: 20  
\sxdef: 4, 12–14, 22, 37, 46–47, 49  
\tabdata: 41, 42–44  
\tabdeclarec: 42  
\tabdeclarel: 42  
\tabdeclarer: 42  
\tabiteml: 7, 42, 44  
\tabitemr: 7, 42, 44  
\tablinefil: 43  
\tabstrut: 7, 41, 43–44  
\tabstrutA: 41, 42–43  
\tabvvl: 43  
\testAleB: 28  
\testAleBsecondary: 28, 29  
\testAleBsecondaryX: 28, 29  
\testin: 12, 13, 52  
\testparA: 39, 41  
\testparB: 39, 40  
\testparC: 39  
\textfontscale: 10, 11  
\textfontsize: 10, 9, 11  
\thechapnum: 16, 17–18  
\thefnote: 46  
\thefont: 11, 39, 41  
\thefontscale: 11, 8, 39, 41

\thefontsize: **11**, 55  
\theseccnum: **16**, 17–18  
\thesecnum: **16**, 17–19  
\thetocnum: **16**, 15, 17–18  
\tit: **16**  
\titfont: **16**  
\tmpdim: **4**, 5, 8, 10–11, 34, 43, 45, 56  
\tmpnum: **4**, 18, 22, 24, 27–28, 30–32, 37, 40–42, 45  
\tnum: **19**, 17  
\toasciidata: **38**, 37  
\tocdotfill: **21**  
\tocilabel: **35**, 17, 37  
\tocline: **21**, 8, 20, 36  
\toclinehook: **8**, 21  
\toclink: **35**, 21  
\toclinkA: **21**, 17, 35  
\toclist: **20**, 21, 36  
\truedimen: **57**, 56  
>trueunit: **57**, 56  
\tskip: **43**, 41  
\tskipA: **43**  
\tthook: **7**, 39–41  
\ttindent: **7**, 38–41  
\ttline: **38**, 39, 41  
\ttpenalty: **7**, 39, 41  
\ttskip: **7**, 39–41  
\typobase: **11**, 16, 46  
\typoscale: **9**, 11, 16, 46  
\typosize: **9**, 11, 33  
\ulink: **35**, 34, 36  
\unsskip: **42**, 44  
\url: **35**, 34, 36, 52  
\urlbskip: **35**, 36  
\urlcolor: **35**, 34, 36  
\urlfont: **35**, 36  
\urllink: **34**, 35  
\urlskip: **35**, 36  
\urlslashslash: **35**, 36  
\urlspecchar: **36**  
\usebbl: **53**, 8, 48, 52, 54  
\usebib: **54**  
\usebibtex: **52**, 8, 48  
\uv: **6**  
\verbinput: **39**, 7, 38  
\vidolines: **40**, 41  
\vifile: **38**, 39–41  
\vifilename: **39**, 40  
\viline: **38**, 39–41  
\vinolines: **39**, 40–41  
\viprintline: **41**  
\vireadline: **41**  
\viscanminus: **40**  
\viscanparameter: **40**, 39  
\viscanplus: **40**  
\vvitem: **43**, 41–42  
\vvkern: **7**, 42–44  
\vvleft: **41**, 43  
\wbib: **51**, 53  
\wcontents: **17**  
\whichtfm: **9**  
\White: **32**  
\wipeepar: **18**, 30, 39, 41  
\withoutunit: **10**, 11  
\wlabel: **14**, 17, 19  
\wref: **12**, 13–14, 17, 21, 46–47, 51–54  
\wrefrelax: **12**, 13, 51  
\writeaux: **52**  
\writeXcite: **52**, 53–54  
\wtotoc: **17**  
\Xbib: **51**  
\Xchap: **20**, 17, 21  
\Xcite: **54**, 52  
\Xfnote: **46**, 55  
\Xindex: **22**, 21, 23–24  
\XindexA: **23**, 22, 24  
\XindexB: **23**, 22, 24  
\Xindexg: **22**  
\Xlabel: **14**, 13, 55  
\Xmnote: **47**, 55  
\Xpage: **55**, 46–47, 54  
\Xrefversion: **13**  
\Xsec: **20**, 17, 21  
\Xsecc: **20**, 17, 21  
\Xtoc: **20**, 17, 21  
\Yellow: **32**