verze Jun. 2017

# OPmac – rozšiřující makra plainTEXu

**Petr Olšák**

www.olsak.net/opmac.html

## Obsah

# 1 Úvod

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

# 2 Uživatelská dokumentace

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

# 3 Technická dokumentace

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

## 3.1 Základní makra

Na začátku souboru `opmac.tex` zjistíme, zda není soubor čtený podruhé. V takovém případě čtení odmítneme. Ptáme se na to, zda je definováno makro `\OPmacversion`, které vzápětí definujeme.

`\OPmacversion`: 4

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

opmac.tex
```
\ifx\OPmacversion\undefined \else \endinput \fi
\def\OPmacversion{Jun. 2017}
\immediate\write16{This is OPmac (Olsak's Plain macros), version <\OPmacversion>}
```

Dva pracovní registry:

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

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

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

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

opmac.tex
```
\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` ⟨*makro*⟩, které zařídí vložení ⟨*makra*⟩ do seznamu.

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

opmac.tex
```
\ifx\pdfextension\undefined \else
   \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 směrován 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.

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

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

opmac.tex
```
\def\sdef#1{\expandafter\def\csname#1\endcsname}
\def\sxdef#1{\expandafter\xdef\csname#1\endcsname}
\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, 37, 54–55 `\addprotect`: 4, 6, 8, 11, 33, 36–37, 55 `\ifpdftex`: 4, 33–35, 38, 44–45
`\sdef`: 4, 12–13, 20, 25, 51, 53–54, 57 `\sxdef`: 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.

opmac.tex
```
\def\adef#1{\catcode`#1=13 \begingroup \lccode`\~=`#1\lowercase{\endgroup\def~}}
```

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

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

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

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

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

opmac.tex
```
\long\def\isnextchar#1#2#3{\begingroup\toks0={\endgroup#2}\toks1={\endgroup#3}%
   \let\tmp=#1\futurelet\next\isnextcharA
}
\def\isnextcharA{\the\toks\ifx\tmp\next0\else1\fi\space}
```

Makro `\eoldef \foo#1{`⟨*makro*⟩`}` 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.

opmac.tex
```
\def\eoldef#1{\def#1{\begingroup \catcode`\^^M=12 \eoldefA#1}%
   \expandafter\def\csname\string#1:M\endcsname}
{\catcode`\^^M=12 \gdef\eoldefA#1#2^^M{\endgroup\csname\string#1:M\endcsname{#2}}}
```

Makro `\maybebreak` ⟨*rozměr*⟩ 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 ⟨*rozměr*⟩ 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.

opmac.tex
```
\def\maybebreak{\afterassignment\maybebreakA\tmpdim=}
\def\maybebreakA{\ifvmode \vskip0pt plus\tmpdim \penalty-130 \vskip0pt plus-\tmpdim
    \else \hskip0pt plus\tmpdim \penalty-130 \hskip0pt plus-\tmpdim \fi \relax
}
```

---

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

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

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

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

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

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

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

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

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

opmac.tex
```
\def\,{\relax\ifmmode \mskip\thinmuskip \else \thinspace \fi}
```

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

opmac.tex
```
\addprotect\percent \addprotect\bslash \addprotect\, \addprotect\exfont
```

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 {⟨*string1*⟩}{⟨*string2*⟩} vymění v makru \tmpb veškeré výskyty ⟨*string1*⟩ za ⟨*string2*⟩. Pro tento účel definuje pracovní makra \replacestringsA a \replacestringsB se separátorem ⟨*string1*⟩. 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.

opmac.tex
```
\bgroup \catcode`!=3 \catcode`?=3
\gdef\replacestrings#1#2{\long\def\replacestringsA##1#1{\def\tmpb{##1}\replacestringsB}%
   \long\def\replacestringsB##1#1{\ifx!##1\relax \else\addto\tmpb{#2##1}%
      \expandafter\replacestringsB\fi}%     improved version <May 2016> inspired
   \expandafter\replacestringsA\tmpb?#1!#1% from pysyntax.tex by Petr Krajnik
   \long\def\replacestringsA##1?{\def\tmpb{##1}}\expandafter\replacestringsA\tmpb
}
\egroup
```

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

```
\widowpenalty=10000
\clubpenalty=10000
\showboxdepth=7
\showboxbreadth=30
```

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

opmac.tex

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

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

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

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

\def\multiskip{\medskip}      % space above and below \begmulti...\endmulti
```

\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 % ditance 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\sechook{}     % hook in \sec
115: \def\secchook{}    % 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 CSTEXu. Tím snadno vytvoříme i logo \csplain.

opmac.tex

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

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

opmac.tex

```
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}
```

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

opmac.tex

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

## 3.4 Velikosti fontů, řádkování

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

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

opmac.tex
```
\ifx\regfont\undefined
   \ifx\uselanguage\undefined % if this is etex.src, the following warning is suppressed
      \opwarning{No multilanguage support (csplain is recommended)}
   \fi
   % macros from csplain, file csfontsm.tex:
   \ifx\tenbi\undefined \font\tenbi=cmbxti10 \def\bi{\tenbi}\fi
   \def\letfont#1#2{\ifx#2=\expandafter\letfont\expandafter#1\else
     \expandafter\font\expandafter#1\expandafter\rfontskipat\fontname#2 \relax\space \fi}
   \def\rfontskipat#1{\ifx#1"\expandafter\rfskipatX\else\expandafter\rfskipatN\expandafter#1\fi}
   \def\rfskipatX #1" #2\relax{"\whichtfm{#1}"}  \def\rfskipatN #1 #2\relax{\whichtfm{#1}}
   \def\sizespec{}   \def\whichtfm#1{#1}
   \def\resizefont#1{\letfont#1#1\sizespec}
   \def\regfont#1{\expandafter\def\expandafter\resizeall\expandafter{\resizeall \resizefont#1}}
   \def\resizeall{}
   \regfont\tenrm \regfont\tenit \regfont\tenbf \regfont\tenbi \regfont\tentt
\fi
```

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

opmac.tex
```
\newdimen\ptunit    \ptunit=1pt
\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 načetl matematické makro ještě před zavoláním \input␣opmac.

opmac.tex
```
\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 \dgsize. 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.

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

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

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

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

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

opmac.tex
```
\def\fontsizex[#1]{\if$#1$\else
  \textfontsize[#1]%
  \tmpdim=0.7\fontdim \edef\tmpa{\expandafter\ignorept\the\tmpdim}%
  \tmpdim=0.5\fontdim \edef\tmpb{\expandafter\ignorept\the\tmpdim}%
  \edef\tmp{\noexpand\setmathsizes[\expandafter\ignorept\the\fontdim/\tmpa/\tmpb]}%
  \tmp \normalmath
  \fi
}
```

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

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

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

opmac.tex
```
\def\setbaselineskip[#1]{\if$#1$\else
  \tmpdim=#1\ptunit
  \baselineskip=\tmpdim \relax
  \ifx\baselineskipB\undefined \edef\baselineskipB{\the\baselineskip}\fi
  \bigskipamount=\tmpdim plus.33333\tmpdim minus.33333\tmpdim
  \medskipamount=.5\tmpdim plus.16666\tmpdim minus.16666\tmpdim
  \smallskipamount=.25\tmpdim plus.08333\tmpdim minus.08333\tmpdim
  \normalbaselineskip=\tmpdim
  \jot=.25\tmpdim
  \maxdepth=.33333\tmpdim
  \setbox\strutbox=\hbox{\vrule height.709\tmpdim depth.291\tmpdim width0pt}%
  \fi
}
```

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

opmac.tex
```
\def\withoutunit#1#2{\expandafter#1\expandafter[\expandafter\ignorept\the#2]}
```

Makra `\fontscalex` ⟨*factor*⟩, `\textfontscale` ⟨*factor*⟩ a `\scalebaselineskip` ⟨*factor*⟩ přepočítají ⟨*factor*⟩ podle aktuálního `\fontdim` resp. `\baselineskip` na absolutní jednotku a zavolají odpovídající makro definované před chvílí. Na řádku 199 je `#1` převedeno na `(#1/1000)pt`: Číslo 3277sp je $2^{16}/20$sp, tedy 1/20pt. 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`.

opmac.tex
```
\def\fontscalex[#1]{\if$#1$\else \ifnum#1=1000 \else
  \tmpdim=3277sp \tmpdim=#1\tmpdim \divide\tmpdim by50
  \tmpdim=\expandafter\ignorept\the\tmpdim \fontdim
  \withoutunit\fontsizex\tmpdim
  \fi\fi
}
\def\textfontscale[#1]{\if$#1$\else
  \tmpdim=#1pt \divide\tmpdim by1000
  \tmpdim=\expandafter\ignorept\the\tmpdim \fontdim
  \withoutunit\textfontsize\tmpdim
  \fi
}
\def\scalebaselineskip[#1]{\if$#1$\else \ifnum#1=1000 \else
```

---

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

```
  \tmpdim=3277sp \tmpdim=#1\tmpdim \divide\tmpdim by50
  \tmpdim=\expandafter\ignorept\the\tmpdim \baselineskip
  \withoutunit\setbaselineskip\tmpdim
  \fi\fi
}
```

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

opmac.tex

```
\def\thefontsize[#1]{\fontdim=#1\ptunit
  \expandafter\let \expandafter\thefont \the\font
  \edef\sizespec{at#1\ptunit}\def\dgsize{#1\ptunit}\resizefont\thefont
  \thefont \let\dgsize=\undefined \ignorespaces
}
\def\thefontscale[#1]{%
  \tmpdim=#1pt \divide\tmpdim by1000
  \tmpdim=\expandafter\ignorept\the\tmpdim \fontdim
  \withoutunit\thefontsize\tmpdim
}
```

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

opmac.tex

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

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

opmac.tex

```
\def\typobase{\ifx\baselineskipB\undefined \def\baselineskipB{12pt}\fi
   \ifx\fontdimB\undefined \def\fontdimB{10pt}\fi
   \baselineskip=\baselineskipB\relax \fontdim=\fontdimB\relax
}
```

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

opmac.tex

```
\def\em {\expandafter\ifx \the\font \tenit \additcorr \rm  \else
         \expandafter\ifx \the\font \tenbf \bi\aftergroup\afteritcorr\else
         \expandafter\ifx \the\font \tenbi \additcorr \bf  \else
         \it \aftergroup\afteritcorr\fi\fi\fi}
\def\additcorr{\ifdim\lastskip>0pt \skip0=\lastskip \unskip\/\hskip\skip0 \else\/\fi}
\def\afteritcorr{\def\tmp{\ifx\next..\else\ifx\next,,\else\/%
                          \expandafter\expandafter\expandafter\next\expandafter\fi\fi}%
   \afterassignment\tmp \let\next= }
```

Fontová makra zabezpečíme proti rozkladu v parametru \write.

opmac.tex

```
\addprotect\thefontsize   \addprotect\thefontscale
\addprotect\typosize      \addprotect\typoscale
\addprotect\textfontsize  \addprotect\textfontscale
\addprotect\em
```

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

opmac.tex
```
246: \def\fontfam{\par \input fontfam \fontfam}
```

## 3.5 Texty ve více jazycích

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

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

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

opmac.tex
```
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}
```

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

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

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

opmac.tex
```
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}
```

Je-li detekován místo CSplainu eTEX, nastavíme značky jazyků dle hodnoty \language v eTEXu:

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

Makro \isolangset {⟨dlouhý-název⟩}{⟨iso-zkratka⟩} přiřadí dlohému názvu jazyka (a související hodnotě registu \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 pořád na disku kvůli tomu uklízet smetí.

Je potřeba deklarovat souborové deskriptory \reffile a \testin:

opmac.tex
```
271: \newwrite\reffile
272: \newread\testin
```

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

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

\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.

opmac.tex

```
\def\inputref{
  \openin\testin=\jobname.ref
  \ifeof\testin \else
    \closein\testin
    \input \jobname.ref
    \fnotenum=0 \mnotenum=0
    \openrefA{\string\inputref}%
  \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, předefinuje \wref a vloží první řádek do souboru. Tím je zaručeno, že při příštím TEXování dokumentu je soubor neprázdný, takže jej OPmac rovnou přečte a znovu založí na začátku své činnosti. Nakonec se \openref zasebevraždí, aby se nemuselo při opakovaném volání obtěžovat vykonávat nějakou práci. Práce už je totiž hotova.

opmac.tex

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

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

Jako první je do REF souboru vložen příkaz \Xrefversion {⟨*číslo*⟩}. Pokud toto ⟨*číslo*⟩ mení 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.

opmac.tex

```
\def\REFversion{2}
\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 \sxdef{lab:⟨*lejblík*⟩}{⟨*číslo*⟩}.
- V místě \ref[⟨*lejblík*⟩] vytiskneme \csname␣lab:⟨*lejblík*⟩\endcsname, tedy ⟨*číslo*⟩.

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

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

\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␣l0:⟨lejblík⟩\endcsname) je od toho, aby OPmac pohlídal případné dvojí použití stejného ⟨lejblíku⟩ a upozornil na to.

opmac.tex

```
\def\label[#1]{\isdefined{l0:#1}%
  \iftrue \opwarning{duplicated label [#1], ignored}\else \xdef\lastlabel{#1}\fi
  \ignorespaces}
```

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

opmac.tex

```
\def\wlabel#1{%
  \ifx\lastlabel\undefined \else
     \dest[ref:\lastlabel]%
     \edef\tmp{\wref\Xlabel{{\lastlabel}{#1}}}\tmp
     \sxdef{lab:\lastlabel}{#1}\sxdef{l0:\lastlabel}{}%
     \global\let\lastlabel=\undefined
  \fi
}
```

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

opmac.tex

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

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

opmac.tex

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

## 3.8 Kapitoly, sekce, podsekce

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 napisem⟩
```

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 prametr, 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 uvedená výše. Pokud je dále třeba definice podsedkce \printsecc zahájena příkazy

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

pak se mohou dít tyto věci:

- Následuje-li podsekce těsně za sekcí, pak se vymaže spodní mezera od sekce 6pt␣plus1pt a místo ní se vloží mezera 8pt␣plus2pt. Mezery se tedy nesčítají. Navíc v tomto případě se nevloží penalta −200, takže mezi sekcí a podeskcí nedojde nikdy ke stránkovému zlomu.
- Následuje-li za sekcí obyčejný text, pak je pod sekcí a nad textem mezera 6pt␣plus␣1pt, která je nezlomitelná.
- Předchází-li před podseskcí obyčejný text, pak se vloží před nadpisem podsekce \penalty-200 následovaná \vskip␣8pt␣plus2pt. Tato mezera je ochotně zlomitelná (bonus −200), takže se může nadpis podsekce objevit na následující straně.

Je možné mezeru 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␣plus1pt jen její část 4pt␣plus1pt.

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

opmac.tex

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

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

```
\def\printsec#1{\par \norempenalty-400 \bigskip
  {\secfont \noindent \dotocnum{\thetocnum\quad}#1\nbpar}\insertmark{#1}%
  \nobreak \remskip\medskipamount \firstnoindent
}
\def\printsecc#1{\par \norempenalty-200 \medskip
  {\seccfont \noindent \dotocnum{\thetocnum\quad}#1\nbpar}%
  \nobreak \remskip\medskipamount \firstnoindent
}
```

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

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

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

opmac.tex

```
\def\titfont{\typobase\typoscale[\magstep4/\magstep4]\bfshape}
\def\chapfont{\typobase\typoscale[\magstep3/\magstep3]\bfshape}
\def\secfont{\typobase\typoscale[\magstep2/\magstep2]\bfshape}
\def\seccfont{\typobase\typoscale[\magstep1/\magstep1]\bfshape}
\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

```
\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 možnot použít jako prefix před stejnými makry s tím, že kapitola, sekce, podsekce nebude mít číslo. I nečíslovaní kapitola se může dostat do obsahu. Je-li obsah klikací, pořebuje mít svoje referenšní číslo pro vytvoření linku. K tomu slouží registr \nonumnum.

opmac.tex

```
\newif\ifnotoc \notocfalse  \def\notoc{\global\notoctrue}
\newif\ifnonum \nonumfalse  \def\nonum{\global\nonumtrue}
```

Makra \chap, \sec a \secc nastaví odpovídající čítače, dále vytvoří číslování pro tisk (sestávající z více čísel) v makrech \thechapnum, \thesecnum a \theseccnum. Aktuální čítač má vždy název \thetocnum. S touto hodnotou (nazávisle na tom, zde jde o kapitlu, sekci nebo podsekci, 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 \thesecnum: 17–19 \theseccnum: 17–18 \thetocnum: 15–18
\dotocnumafter: 17–18

opmac.tex
```
\eoldef\chap#1{\ifnonum\else \global\advance\chapnum by1 \fi
  \chaphook {\globaldefs=1 \secnum=0 \seccnum=0 \tnum=0 \fnum=0 \dnum=0}\relax
  \edef\thechapnum{\the\chapnum}\let\thetocnum=\thechapnum
  \def\dotocnumafter{\wtotoc0\bfshape{#1}}%
  \printchap{#1}\resetnonumnotoc
}
\eoldef\sec#1{\ifnonum\else \global\advance\secnum by1 \fi
  \sechook {\globaldefs=1 \seccnum=0 \tnum=0 \fnum=0 \dnum=0}\relax
  \edef\thesecnum{\othe\chapnum.\the\secnum}\let\thetocnum=\thesecnum
  \def\dotocnumafter{\wtotoc1\rm{#1}}%
  \printsec{#1}\resetnonumnotoc
}
\eoldef\secc#1{\ifnonum\else \global\advance\seccnum by1 \fi
  \secchook {}\relax
  \edef\theseccnum{\othe\chapnum.\the\secnum.\the\seccnum}\let\thetocnum=\theseccnum
  \def\dotocnumafter{\wtotoc2\rm{#1}}%
  \printsecc{#1}\resetnonumnotoc
}
```

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 ⟨úroveň⟩⟨font⟩{⟨text⟩}, které vytvoří

```
\write\reffile
  {\string\Xtoc{⟨úroveň⟩}{⟨font⟩}{⟨expandovaný thetocnum⟩}{⟨text⟩}{\the\pageno}}
```

Přitom ⟨úroveň⟩ 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.

opmac.tex
```
\def\wtotoc#1#2#3{% #1 = level, #2 = info, #3 = titletext
  \ifnotoc\else
      \def\act{\wref{\Xtoc{#1}{\noexpand#2}}}%
      \expandafter\act\expandafter{\expandafter{\thetocnum}{#3}{\the\pageno}}%
  \fi
}
```

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.

opmac.tex
```
\def\wcontents#1#2{% #1 = sequence to REF, #2 = titletext
  \ifnotoc\else
      \expandafter\wref\expandafter#1\expandafter
        {\expandafter{\thetocnum}{#2}{\the\pageno}}%
  \fi
}
```

Makro \dotocnum {⟨text⟩} 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 nezapíše 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.

opmac.tex
```
\def\dotocnum#1{%
  \leavevmode
     {\ifnonum \global\advance\nonumnum by1 \edef\thetocnum{!\the\nonumnum}\fi
      \wlabel\thetocnum \dest[toc:\tocilabel.\thetocnum]%
      \dotocnumafter}\ifnonum\else#1\fi
  \global\let\dotocnumafter=\relax
}
```

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

\wtotoc: 17 \wcontents: 17 \dotocnum: 15–18 \resetnonunotoc

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.

opmac.tex
```
392: \def\resetnonumnotoc{\global\notocfalse \global\nonumfalse
393:   \ifx\dotocnumafter\relax \else
394:     \opwarning{\noexpand\dotocnum unused in printchap/printsec/printsecc}\fi
395: }
```

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.

opmac.tex
```
396: \def\insertmark#1{\toks0={#1}\mark{{\ifnonum\else\thetocnum\fi} {\the\toks0}}}
```

Příklad použití plovoucího záhlaví v \headline:

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

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

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

Makro \remskip ⟨*velikost*⟩ je implimentováno jako \vskip⟨*velikost*⟩ následované smluvenou nezlomitelnou penaltou 11333. Makro \norempenalty pak podle této hodnoty poslední penalty v seznamu větví svou činnost. V registru \remskipamount je uložena naposledy vložená mezera z \remskip.

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

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 tisnout dokument jen se sekcemi bez kapitol. Číslo kapitoly se pak nezobrazuje jako 0., protože se nezobrazuje vůbec.

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

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

opmac.tex
```
407: \def\afternoindent{\global\everypar={\wipeepar\setbox7=\lastbox}}
408: \def\wipeepar{\global\everypar={}}
409: \let\firstnoindent=\afternoindent
```

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

opmac.tex
```
410: \def\nbpar{{\interlinepenalty=10000\endgraf}}
411: \def\nl{\hfil\break}
```

\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
\nbpar: 15–16, 18–19 \nl: 18, 54

## 3.9 Popisky, rovnice

Nejprve deklarujeme potřebné čítače:

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

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

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

Makro \caption /⟨typ⟩␣ zvedne čítač \⟨typ⟩num o jedničku, dále nastaví pružné mezery s odsazením \iindent a s centrováním posledního řádku (viz TBN str. 234), propojí pomocí \wlabel číslo s případným lejblíkem 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.

opmac.tex
```
\def\caption/#1 {\isdefined{#1num}%
   \iftrue \global\advance \csname #1num\endcsname by1
   \else   \opwarning{Unknown caption /#1}%
   \fi
   \bgroup
      \leftskip=\iindent plus1fil
      \rightskip=\iindent plus-1fil
      \parfillskip=0pt plus2fil
      \def\par{\nbpar\egroup}%
      \captionhook{#1}\noindent
      \wlabel{\csname the#1num\endcsname}%
      \printcaption{\mtext{#1}}{\csname the#1num\endcsname}%
}
```

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}

opmac.tex
```
\def\printcaption#1#2{{\bf#1 #2}\enspace}
```

Předefinujeme makro z plainTEXu \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.

opmac.tex
```
\expandafter\def\expandafter\endinsert\expandafter{\expandafter\par\endinsert}
```

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

opmac.tex
```
\def\eqmark{\global\advance\dnum by1
  \ifinner\else\eqno \fi
  \wlabel\thednum \thednum
}
```

## 3.10 Odrážky

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

opmac.tex
```
\newcount\itemnum  \itemnum=0
```

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

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

opmac.tex
```
\def\begitems{\par\iiskip\bgroup
  \itemnum=0 \adef*{\startitem}
  \advance\leftskip by\iindent
  \let\printitem=\normalitem
}
\def\enditems{\par\egroup\iiskip}
```

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

opmac.tex
```
\def\startitem{\par \advance\itemnum by1
   \itemhook \noindent\llap{\printitem}\ignorespaces}
\def\normalitem{$\bullet$\enspace}
```

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

opmac.tex
```
\def\style#1{\expandafter\let\expandafter\printitem\csname item:#1\endcsname
  \ifx\printitem\relax \let\printitem=\normalitem \fi
}
\sdef{item:o}{\raise.4ex\hbox{$\scriptscriptstyle\bullet$} }
\sdef{item:-}{- }
\sdef{item:n}{\the\itemnum. }
\sdef{item:N}{\the\itemnum) }
\sdef{item:i}{(\romannumeral\itemnum) }
\sdef{item:I}{\uppercase\expandafter{\romannumeral\itemnum}\kern.5em}
\sdef{item:a}{\athe\itemnum) }
\sdef{item:A}{\uppercase\expandafter{\athe\itemnum}) }
\sdef{item:x}{\raise.3ex\fullrectangle{.6ex} }
\sdef{item:X}{\raise.2ex\fullrectangle{1ex}\kern.5em}
```

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

opmac.tex
```
\def\fullrectangle#1{\hbox{\vrule height#1 width#1}}
```

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

opmac.tex
```
\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
   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
}
```

## 3.11 Tvorba obsahu

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

opmac.tex
```
\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 ⟨*úroveň*⟩⟨*info*⟩⟨*číslo*⟩⟨*text*⟩⟨*strana*⟩.

opmac.tex
```
\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 odstraním.

---

\begitems: 7, 20 \enditems: 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

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

Makro \tocline {⟨úroveň⟩}{⟨info⟩}{⟨číslo⟩}{⟨text⟩}{⟨strana⟩} vytvoří řádek obsahu. Údaj ⟨úroveň⟩ je číslo 0 pro kapitolu, 1 pro sekci a 2 pro podsekci. Údaj ⟨info⟩ používá OPmac pro informaci o fontu, kterým se má tisknout řádek v obsahu. Řádek obsahu tiskneme jako odstavec, protože ⟨text⟩ může být třeba delší. Registr \leftskip nastavíme jako součin ⟨úroveň⟩ krát \iindent. Pokud se v dokumentu vyskytují kapitoly, odsadíme ještě o další \iindent. Registr \rightskip nastavíme na 2\iindent, aby delší ⟨text⟩ se zalomil dřív než v místě, kde jsou stránkové číslice. Konec odstavce se stránkovou číslicí pak vytáhneme mimo tento rozsah pomocí \hskip-2\iindent. Odstavec pruží v \tocdotfill, protože tento výplněk má větší pružnost než \parfillskip. Registr \parfillskip má 1fil, zatímco \tocdotfill má pružnost 1fill. Makro \tocdotfill je implementováno pomocí \leaders jako opakované tečky, které budou lícovat pod sebou.

opmac.tex
```
\def\tocline#1#2#3#4#5{{\leftskip=#1\iindent \rightskip=2\iindent
   \ifischap\advance\leftskip by\iindent\fi
   \ifnum#1>1 \advance\leftskip by\iindent\fi
   \toclinehook \noindent\llap{#2\toclink{#3}\enspace}%
         {#2#4}\nobreak\tocdotfill\pglink{#5}\nobreak\hskip-2\iindent\null\par}}
\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.

opmac.tex
```
\def\maketoc{\par \ifx\toclist\empty
      \opwarning{\noexpand\maketoc -- data unavailable, TeX me again}\openref
   \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.

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

## 3.12 Sestavení rejstříku

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

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

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

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

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

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

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

\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

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

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

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

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

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

opmac.tex
```
\def\Xindex{\Xindexg,}
\def\Xindexg#1#2#3{\bgroup \def~{ }% #1=prefix, #2=index-item, #3=pageno
   \isdefined{#1#2}\iftrue
      \ifx^#3^\else
         \expandafter\firstdata \csname#1#2\endcsname \XindexA
         \ifnum#3=\tmpa % \ii on the same page
         \else
            \tmpnum=#3 \advance\tmpnum by\pgfolioB-1
            \expandafter\seconddata \csname#1#2\endcsname \XindexB
            \ifx\tmp\empty
               \sxdef{#1#2}{{#3/+}{\pgfolioA{#3}}} % previous item: empty page
            \else
               \if\tmpb+% state: the pagelist ends by a pagenumber
                  \ifnum\tmpnum=\tmpa  % the consecutive page
                     \sxdef{#1#2}{{#3/-}{\tmp\iiendash}}
                  \else                % the pages drop
                     \sxdef{#1#2}{{#3/+}{\tmp, \pgfolioA{#3}}}
                  \fi
               \else    % state: the pagelist ends by --
                  \ifnum\tmpnum=\tmpa  % the consecutive page
                     \sxdef{#1#2}{{#3/-}{\tmp}}
                  \else                % the pages drop
                     \sxdef{#1#2}{{#3/+}{\tmp\pgfolioA{\tmpa}, \pgfolioA{#3}}}
      \fi\fi\fi\fi\fi
   \else % first occurrence of the index item #2
      \ifx^#3^\sxdef{#1#2}{{0/+}{}}\else \sxdef{#1#2}{{#3/+}{\pgfolioA{#3}}}\fi
      \ifx,#1
         \global \expandafter\addto \expandafter\iilist \csname#1#2\endcsname
      \else
         \isdefined{iilist:#1}\iftrue
            \global\expandafter\addto \csname iilist:#1\expandafter\endcsname \csname#1#2\endcsname
         \else \sxdef{iilist:#1}{\expandafter\noexpand \csname#1#2\endcsname}
   \fi\fi\fi
   \egroup
}
\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 \,⟨heslo⟩. Následující text popisuje jen tento případ. Makro \Xindexg je možné použít pro pralelní vytváření dalších seznamů stránek stejných hesel (např. seznam vyznačený kurzívou, tučně atd.). Jak to udělat je popsáno v OPmac triku 0072.

---

\iid: 22 \iiD: 22 \Xindex: 21–24 \iilist: 22, 24, 29–30 \Xindexg: 22

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

opmac.tex
```
552: \def\firstdata#1#2{\expandafter\expandafter\expandafter #2\expandafter\firstdataA#1}
553: \def\firstdataA#1#2{#1&}
554: \def\seconddata#1#2{\expandafter\expandafter\expandafter #2\expandafter\seconddataA#1}
555: \def\seconddataA#1#2{#2&}
```

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

V makru `\Xindex` čteme `\firstdata` na řádku 519 a `\seconddata` na řádku 523. 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
```
557: \def\XindexA#1/#2&{\def\tmpa{#1}\let\tmpb=#2}
558: \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.

opmac.tex

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

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.

opmac.tex

```
\def\makeindex{\par
  \ifx\iilist\empty \opwarning{index data-buffer is empty. TeX me again}
    \else
    \bgroup
       \setprimarysorting
       \def\act##1{\ifx##1\relax \else
          \firstdata##1\XindexA \seconddata##1\XindexB
          \if\tmpb+%
             \preparesorting##1% converted item by sorting data in \tmpb
             \xdef##1{{\tmpb}{\tmp}}
          \else
             \preparesorting##1% converted item by sorting data in \tmpb
             \xdef##1{{\tmpb}{\tmp\pgfolioA{\tmpa}}}
          \fi
          \expandafter\act\fi}
       \expandafter \act \iilist \relax
    \egroup
    \dosorting  % sorting is in progress
    \bgroup
       \rightskip=0pt plus1fil \exhyphenpenalty=10000 \leftskip=\iindent
       \def\act##1{\ifx##1\relax \else \prepii##1%
                \seconddata##1\printiipages \expandafter\act \fi}
       \expandafter \act \iilist \relax
    \egroup
  \fi
}
```

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

opmac.tex

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

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

opmac.tex

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

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

---

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

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

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

Makro „print index item“ \printii ⟨heslo⟩& vytiskne jeden údaj do rejstříku. Makro projde prostřednictvím \printiiA jednotlivá podslova oddělená lomítkem a přepíše je do rejstříku odděleny mezerou. Přitom kontroluje, zda se podslova rovnají odpovídajícím podslovům 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 podslova 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.

opmac.tex
```
\def\printii #1&{\gdef\currii{#1}\noindent\everyii
   \hskip-\iindent \ignorespaces\printiiA#1//}
\def\printiiA #1/{\if^#1^\let\previi=\currii \else
   \expandafter\scanprevii\previi/&\def\tmpb{#1}\edef\tmpb{\meaning\tmpb}%
   \ifx\tmpa\tmpb \iiemdash \else#1 \gdef\previi{}\fi
   \expandafter\printiiA\fi
}
\def\iiemdash{\kern.1em---\space}
\def\everyii{}
```

Makro \makeindex nastavuje na řádku 582 lokálně parametry sazby odstavce v rejstříku. Vlevo budeme mít \leftskip rovný \iindent, ale první řádek posuneme o -\iindent (viz řádek kódu 606) 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 ⟨expanded-previi⟩& se podívá do \previi, odloupne z něj úsek před prvním lomítkem a tento úsek definuje jako \tmpa.

opmac.tex
```
\def\scanprevii#1/#2&{\def\previi{#2}\def\tmpa{#1}\edef\tmpa{\meaning\tmpa}}
```

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

opmac.tex
```
\def\previi{} % previous index item
```

## 3.13 Abecední řazení rejstříku

Nejprve se zaměříme na vytvoření makra \isAleB \,⟨heslo1⟩ \,⟨heslo2⟩, které rozhodne, zda je ⟨heslo1⟩ řazeno před ⟨heslem2⟩ 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 háčkovaná před stříškovaná před kroužkovaná a dále s nejnižší prioritou malá písmena před velká.

Nejprve připravíme data pro porovnávací algoritmus.

opmac.tex
```
\def\sortingdata{%
  /,{ },-,&,@,%
  aA\"a\"A\'a\'A,%
  bB,%
  cC,%
  \v c\v C,%
  dD\v d\v D,%
  eE\'e\'E\v e\v E,%
```

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

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

Mezi jednotlivými čárkami v makru \sortingdata jsou skupiny znaků, které se z hlediska prvního průchodu řadicím algoritmem nerozlišují. Jednotlivé znaky v \sortingdata se rozliší při případném druhém průchodu. Řazení znaků v \sortingdata 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 \specsortingdatacs a \specsortingdatask 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 \sortingdata 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 \sortingdata, 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\specsortingdatask {ch:^^T Ch:^^U CH:^^V
   dz:^^N Dz:^^O DZ:^^P d\v z:^^Q D\v z:^^R D\v Z:^^S}
```

\sortingdata: 25–28 \setignoredchars: 26–28 \specsortingdatacs: 26
\specsortingdatask: 26

Makra pro spřežky mají název ve tvaru \specsortingdata⟨*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 `^^I` a `^^M`, kterým plainTEX, resp. iniTEX, nastavuje speciální kategorie.

Implementaci řadicího algoritmu zahájíme makrem \setprimarysorting, které se spustí jednou při sestavení rejstříku, přečte výše uvedená data a připraví odpovídající datové stuktury pro první průchod řadicího 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

```
\def\setprimarysorting {%
   \isdefined{sortingdata\csname lan:\the\language\endcsname}\iftrue
      \expandafter \let\expandafter\sortingdata
         \csname sortingdata\csname lan:\the\language\endcsname\endcsname
      \xdef\sortingmessage{using \string\sortingdata\csname lan:\the\language\endcsname}%
   \else
      \xdef\sortingmessage{using internal \string\sortingdata}%
      \ifx\r\undefined
         \opwarning{\noexpand\csaccents is unused, falling back to ASCII sorting}%
         \global\let\asciisorting=t%
   \fi\fi
   \ifx\asciisorting\undefined
      \xdef\sortingdata{\sortingdata}% expand sorting data now
      \isdefined{specsortingdata\csname lan:\the\language\endcsname}\iftrue
         \xdef\specsortingdata{\csname specsortingdata\csname lan:\the\language\endcsname
            \endcsname\space}%
         \expandafter\setprimarysortingA \meaning\specsortingdata\relax
      \else \gdef\specsortingdata{}\fi
   \else
      \gdef\sortingdata{.}\gdef\specsortingdata{}\gdef\sortingmessage{ASCII}%
   \fi
   \def\act##1{\ifx##1.\else
      \ifx##1,\advance\tmpnum by1
      \else \lccode`##1=\tmpnum \fi
      \expandafter \act \fi}%
   \tmpnum=60 \expandafter \act\sortingdata \setignoredchars
}
\def\setprimarysortingA#1->#2\relax{\gdef\specsortingdata{#2}}
\def\sortingmessage{ASCII default}
```

Vidíme, že makro \setprimarysorting nejprve expanduje \sortingdata, aby se realizovaly znaky typu \v␣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 řadicí algoritmus řadí podle ASCII.

Dále \setprimarysorting připraví makro \specsortingdata (bez přípony jazyka) z makra \specsortingdata⟨*aktuální-jazyk*⟩. Nejrve je expanduje a pak pomocí triku s \meaning za spoluráce s makrem \setprimarysortingA převede všechny znaky v makru na kategorii 12, protože toto budeme pro řadicí 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 dřív než delší slovo se stejným začátkem, obsahující celé kratší slovo (ten tučňák≺tento). Je sice pravda, že ASCII hodnota mezery je ještě menší, ale my musíme mezeru někam šoupnout na jiný kód než 32, jinak by nám ji nepřečetlo makro s neseparovaným parametrem. Ovšem my budeme chtít mezeru přečíst. 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 \specsortingdata: 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.

opmac.tex

```
\def\setsecondarysorting {\def\act##1{\ifx##1.\else
     \ifx##1,\else \advance\tmpnum by1 \lccode`##1=\tmpnum \fi
     \expandafter \act \fi}%
  \tmpnum=60 \expandafter \act\sortingdata \setignoredchars
}
```

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 569 až 577 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.

opmac.tex

```
\def\preparesorting#1{\expandafter\preparesortingA\string#1&}
\gdef\preparesortingA#1#2#3&{\xdef\tmpb{#3}%
   \expandafter\preparesortingB\specsortingdata.:{}
   \lowercase\expandafter{\expandafter\gdef\expandafter\tmpb\expandafter{\tmpb}}%
   \replacestrings{.}{}%
}
\def\preparesortingB#1#2:#3 {\ifx.#1\else \replacestrings{#1#2}{#3}\expandafter\preparesortingB\fi}
```

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:

opmac.tex

```
\newif \ifAleB
```

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

opmac.tex

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

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

opmac.tex

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

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

---

\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

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

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

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

opmac.tex
```
\def\dosorting{%
   \message{Opmac: Sorting index (\sortingmessage)...}%
   \def\act##1{\ifx##1\relax\else \addto\iilist{##1,}%
             \expandafter\act\fi}%
   \edef\iilist{\expandafter}\expandafter\act \iilist\relax
   \edef\iilist{\expandafter}\expandafter\mergesort \iilist \end,\end
}
```

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

Na počátku jsou skupiny jednoprvkové (\iilist odděluje každou položku čárkou). Makro \mergesort v tomto případě projde seznam a vytvoří seznam zatříděných dvoupoložkových skupin, uložený zpětně v \iilist. V dalším průchodu znovu vyvrhne \iilist do vstupní fronty, vyprázdní ho a startuje znovu. Nyní vznikají čtyřpoložkové zatříděné skupiny. Pak osmipoložkové atd. V závěru (na řádku 747) 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.

opmac.tex
```
\def\mergesort #1#2,#3{% by Miroslav Olsak
   \ifx,#1%                      % prazdna-skupina,neco,  (#2=neco #3=pokracovani)
      \addto\iilist{#2,}%        % dvojice skupin vyresena
      \sortreturn{\fif\mergesort#3}%   % \mergesort pokracovani
   \fi
   \ifx,#3%                      % neco,prazna-skupina,  (#1#2=neco #3=,)
      \addto\iilist{#1#2,}%      % dvojice skupin vyresena
      \sortreturn{\fif\mergesort}%      % \mergesort dalsi
   \fi
   \ifx\end#3%                   % neco,konec (#1#2=neco)
      \ifx\empty\iilist                % neco=kompletni setrideny seznam
         \def\iilist{#1#2}%
         \sortreturn{\fif\fif\gobbletoend}%   % koncim
      \else                        % neco=posledni skupina nebo \end
         \sortreturn{\fif\fif       % spojim \indexbuffer+necoa cele znova
                     \edef\iilist{\expandafter}\expandafter\mergesort\iilist#1#2,#3}%
   \fi\fi                       % zatriduji: p1+neco1,p2+neco2, (#1#2=p1+neco1 #3=p2)
   \isAleB #1#3\ifAleB          % p1<p2
```

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

```
      \addto\iilist{#1}%       % p1 do bufferu
      \sortreturn{\fif\mergesort#2,#3}%         % \mergesort neco1,p2+neco2,
   \else                       % p1>p2
      \addto\iilist{#3}%       % p2 do bufferu
      \sortreturn{\fif\mergesort#1#2,}%         % \mergesort p1+neco1,neco2,
   \fi
   \relax % zarazka, na ktere se zastavi \sortreturn
}
\def\sortreturn#1#2\fi\relax{#1} \def\fif{\fi}
\def\gobbletoend #1\end{}
```

Jádro `\mergesort` vidíme na řádcích 754 až 759. 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 738 až 744 řeší případy, kdy je jedna ze skupin prázdná: je potřeba vložit do `\indexbuffer` zbytek neprázdné skupiny a přejít na další dvojici skupin. Ostatní řádky makra se vyrovnávají se skutečností, že zpracování narazilo na zarážku `\end,\end` a je tedy potřeba vystartovat další průchod.

## 3.14 Více sloupců

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

opmac.tex

```
\newcount\mullines
\def\corrsize #1{%% #1 := #1 + \splittopskip - \topskip
   \advance #1 by \splittopskip \advance #1 by-\topskip
}
\def\begmulti #1 {\par\bgroup\wipeepar\multiskip\penalty0 \def\Ncols{#1}
   \setbox6=\vbox\bgroup\penalty0
   %% \hsize := Sirka sloupce = (\hsize+\colsep) / n - \colsep
   \advance\hsize by\colsep
   \divide\hsize by\Ncols  \advance\hsize by-\colsep
   \mullines=0
   \def\par{\ifhmode\endgraf\global\advance\mullines by\prevgraf\fi}%
}
\def\endmulti{\vskip-\prevdepth\vfil
   \expandafter\egroup\expandafter\baselineskip\the\baselineskip\relax
   \dimen0=.8\maxdimen \tmpnum=\dimen0 \divide\tmpnum by\baselineskip
   \splittopskip=\baselineskip
   \setbox1=\vsplit6 to0pt
   %% \dimen1 := the free space on the page
   \ifdim\pagegoal=\maxdimen \dimen1=\vsize \corrsize{\dimen1}
   \else \dimen1=\pagegoal \advance\dimen1 by-\pagetotal \fi
   \ifdim \dimen1<2\baselineskip
     \vfil\break \dimen1=\vsize \corrsize{\dimen1} \fi
   \ifnum\mullines<\tmpnum \dimen0=\ht6 \else \dimen0=.8\maxdimen \fi
   \divide\dimen0 by\Ncols \relax
   %% split the material to more pages?
   \ifdim \dimen0>\dimen1 \splitpart
   \else \balancecolumns \fi  % only balancing
   \multiskip\egroup
```

\begmulti: 7, 30 \endmulti: 7, 30 \corrsize: 30–31 \makecolumns: 31 \splitpart: 30–31

```
}
\def\makecolumns{\bgroup % full page, destination height: \dimen1
   \vbadness=20000 \setbox1=\hbox{}\tmpnum=0
   \loop \ifnum\Ncols>\tmpnum
      \advance\tmpnum by1
      \setbox1=\hbox{\unhbox1 \vsplit6 to\dimen1 \hss}
   \repeat
   \hbox{}\nobreak\vskip-\splittopskip \nointerlineskip
   \line{\unhbox1\unskip}
   \dimen0=\dimen1 \divide\dimen0 by\baselineskip \multiply\dimen0 by\Ncols
   \global\advance\mullines by-\dimen0
   \egroup
}
\def\splitpart{%
   \makecolumns % full page
   \vskip 0pt plus 1fil minus\baselineskip \break
   \ifnum\mullines<\tmpnum \dimen0=\ht6 \else \dimen0=.8\maxdimen \fi
   \divide\dimen0 by\Ncols \relax
   \ifx\balancecolumns\flushcolumns \advance\dimen0 by-.5\vsize \fi
   \dimen1=\vsize \corrsize{\dimen1}\dimen2=\dimen1
   \advance\dimen2 by-\Ncols\baselineskip
   %% split the material to more pages?
   \ifvoid6 \else
      \ifdim \dimen0>\dimen2 \expandafter\expandafter\expandafter \splitpart
      \else \balancecolumns % last balancing
   \fi \fi
}
```

Výstup rozlomené sazby do sloupců probíhá ve dvou režimech: když je třeba sloupci zaplnit celou stránku, použijeme \makecolumns. Toto makro neřeší otázku, že může v kumulovaném boxu 6 zbýt nějaká sazba, protože se předpokládá, že lámání bude pokračovat na další straně. Pokud ale je na aktuální straně vícesloupcová sazba ukončena, použijeme propracovanější \balancecolumns. Toto makro si zazálohuje materiál z boxu 6 do boxu 7 a jme se zkoušeti rozlomit box 6 na sloupce s výškou \dimen0. Pokud ale po rozlomení není výchozí box 6 zcela prázdný, makro zvětší krapánek (o 0,2\baselineskip) požadovanou výšku, vrátí se k zálohované sazbě v boxu 7 a zkusí rozlomit znovu. To opakuje tak dlouho, dokud je box 6 prázdný.

opmac.tex

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

Když je sazba plněna do boxu 6, může ji být tak moc, že 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 řádku 778) počítá počet řádků sazby. Je-li \mullines větší než \tmpnum (což při daném \baselineskip odpovídá 0,8\maxdimen), makro pracuje, jakoby výška boxu 6 byla 0,8\maxdimen, tedy rozběhne se \splitpart a \makecolumns. Přitom makro \makecolumns snižuje hodnotu \mullines

\balancecolumns: 30–32 \mullines: 30–32

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
```
844: \newif\iflocalcolor \localcolorfalse
845: \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
```
847: % for backward compatibility:
848: \let\longlocalcolor=\localcolor  \let\locpgcolor=\relax
849: \def\linecolor#1{}
```

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

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

Makro \setcmykcolor {⟨*CMYK-barva*⟩} nastaví požadovanou barvu. Nejprve přepne makro \ensureblacko 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 \localocolortrue 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

opmac.tex
```
\def\setcmykcolor#1{\global\let\ensureblacko=\ensureblackoA
   \iflocalcolor \edef\currentcolor{#1}\colorstackpush\currentcolor \aftergroup\colorstackpop
   \else         \xdef\currentcolor{#1}\colorstackset\currentcolor \fi
}
```

Makro \currentcolor je nastaveno na výchozí hodnotu \pdfblackcolor

opmac.tex
```
\def\pdfblackcolor{0 0 0 1}
\xdef\currentcolor{\pdfblackcolor}
```

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

opmac.tex
```
\def\ensureblacko#1{#1}
\def\ensureblackoA#1{\colorstackpush\pdfblackcolor #1\colorstackpop}
```

Makra \colorstackpush {⟨*CMYK-barva*⟩} a \colorstackpop implementují práci se zásobníkem barev za použití odpovídajících TEXových primitivů. Není-li přítomen pdfTEX 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. Povšimněte si, že se současně pracuje s barvou textu ⟨c⟩␣⟨m⟩␣⟨y⟩␣⟨k⟩␣k i s barvou tenkých linek ⟨c⟩␣⟨m⟩␣⟨y⟩␣⟨k⟩␣K. Konečně makro \colorstackset {⟨*CMYK-barva*⟩} nastavuje barvu přímo s umístěním této bravy na vrchol zásobníku místo bravy předchozí.

opmac.tex
```
\ifx\pdfcolorstackinit\undefined
   \def\colorstackpush#1{\pdfliteral{#1 k #1 K}}
   \def\colorstackpop{\colorstackpush\currentcolor}
   \let\colorstackset=\colorstackpush
\else
   \mathchardef\colorstackcnt=\pdfcolorstackinit page {0 g 0 G}
   \def\colorstackpush#1{\pdfcolorstack\colorstackcnt push{#1 k #1 K}}
   \def\colorstackpop{\pdfcolorstack\colorstackcnt pop}
   \def\colorstackset#1{\pdfcolorstack\colorstackcnt set{#1 k #1 K}}
\fi
```

Makra \colorstackpush, \colorstackpop a \colorstackset jsou odpovídajícím způsobem předefinována v souboru opmac-xetex.tex, aby bylo možné pracovat s barvami i v XeTEXu.

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

opmac.tex
```
\addprotect\setcmykcolor  \addprotect\localcolor  \addprotect\longlocalcolor
```

Není-li použit pdfTEX, některá makra pro barvu deaktivujeme:

opmac.tex
```
\ifpdftex\else
   \def\setcmykcolor#1{} \def\pdfliteral#1{}
\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.

opmac.tex
```
\def\draft{\addto\prepghook{\draftbox{\tenbf DRAFT}\nointerlineskip}}
```

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

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

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

```
895:    \hss}\vss}}
```

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

opmac.tex

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

## 3.16 Klikací odkazy

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

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

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

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

opmac.tex

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

Makro \urllink [⟨typ⟩:⟨lejblík⟩]{⟨text⟩} 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 \ulink.

---

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

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

opmac.tex
```
\def\toclink#1{\toclinkA{#1}}
\def\pglink#1{\pgfolioA{#1}\relax}
\def\citelink#1#2{\leavevmode#2}
\def\reflink[#1]#2{\leavevmode#2}
\def\ulink[#1]#2{{\leavevmode#2}}
\def\urlcolor{}
```

Ovšem po použití makra \hyperlinks {⟨*barva-lok*⟩}{⟨*barva-url*⟩} se uvedená makra \toclink, \pglink, \citelink a \reflink probouzejí k životu. Zde je také definováno makro \urlcolor.

opmac.tex
```
\def\hyperlinks#1#2{%
   \let\dest=\destactive \let\link=\linkactive
   \def\toclink##1{\link[toc:\tocilabel.##1]{\localcolor#1}{\toclinkA{##1}}}%
   \def\pglink##1{\link[pg:\pgilabel.##1]{\localcolor#1}{\pgfolioA{##1}}}%
   \def\citelink##1##2{\link[cite:##1]{\localcolor#1}{##2}}%
   \def\reflink[##1]##2{\link[ref:##1]{\localcolor#1}{##2}}%
   \def\ulink[##1]##2{\urllink[url:##1]{##2}}%
   \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.

opmac.tex
```
\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 zařídí makro \pdfborder {⟨*typ*⟩}, které expanduje na `attr /Border[0␣0␣0]`, pokud není kontrolní sekvence \⟨*typ*⟩`border` definována. Jinak expanduje na `arrt /Border[0␣0␣.6]` a `/C` s obsahem podle \⟨*typ*⟩`border`.

opmac.tex
```
\def\pdfborder#1{\if^#1^\else \isdefined{#1border}\iftrue
   \if^\csname#1border\endcsname^\else attr{/C[\csname#1border\endcsname] /Border[0 0 .6]}\fi
   \else attr{/Border[0 0 0]}\fi\fi
}
```

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

opmac.tex
```
\ifpdftex \else
  \def\link[#1]#2#3{#3}
  \def\urllink[#1]#2{#2}
  \def\hyperlinks#1#2{\opwarning{No pdfTeX detected, \noexpand\hyperlinks ignored}}
\fi
```

Makro \url {⟨*text*⟩} se používá k tisku URL. Vytiskne ⟨*text*⟩ fontem \urlfont, přitom kolem znaků lomítko, 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. Dvojité lomítko \urlslashslash 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
\urlslashslash: 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 \hfil\break).

opmac.tex

```
\def\url#1{{\def\tmpb{#1}%
   \replacestrings{//}{{\urlskip\urlslashslash\urlbskip}}%
   \replacestrings{/}{{\urlskip/\urlbskip}}%
   \replacestrings{.}{{\urlskip.\urlbskip}}%
   \replacestrings{?}{{\urlskip?\urlbskip}}%
   \replacestrings{=}{{\urlskip=\urlbskip}}%
   \replacestrings{~}{{\char`\~}}%
   \replacestrings{_}{{\char`\_}}%
   \replacestrings{^}{{\char`\^}}%
   \replacestrings{\\}{\bslash}%
   \replacestrings{\{}{{\char`\{}}%
   \replacestrings{\}}{{\char`\}}}%
   \replacestrings{&}{{\urlbskip\char`\& \urlskip}}%
   \def\|{}\ulink[#1]{\urlfont\tmpb\null}%
}}
\def\urlfont{\tt \let\|=\urlspecchar}
\def\urlspecchar{\penalty10 }
\def\urlskip{\null\nobreak\hskip0pt plus0.05em\relax}
\def\urlbskip{\penalty100 \hskip0pt plus0.05em\relax}
\def\urlslashslash{/\urlskip/}
\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 \ulink.

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 změní 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

```
\def\outlines#1{\pdfcatalog{/PageMode/UseOutlines}\openref\ifx\toclist\empty
     \opwarning{\noexpand\outlines -- data unavailable. TeX me again}%
   \else
     \ifx\urlcolor\empty
        \opwarning{\noexpand\outlines doesn't work when \noexpand\hyperlinks isn't declared}\fi
     {\let\tocline=\outlinesA
      \count0=0 \count1=0 \toclist % calculate numbers o childs
      \def\outlinelevel{#1}\let\tocline=\outlinesB
      \count0=0 \count1=0 \toclist}% create outlines
   \fi
}
```

V makru \outlinesA {⟨*odsazení*⟩}{⟨*info*⟩}{⟨*číslo*⟩}{⟨*text*⟩}{⟨*strana*⟩} počítáme potomky. Makro je navrženo tak, aby bylo snadno rozšiřitelné na libovolnou úroveň hloubky stromu, nicméně pro potřeby OPmac stačí hloubka tři (kapitoly, sekce, podsekce). Úroveň uzlu přečteme v parametru ⟨*odsazení*⟩. Pro kapitolu je ⟨*odsazení*⟩=0, pro sekci je ⟨*odsazení*⟩=1 a pro podsekci je ⟨*odsazení*⟩=2. Představme si vedle sebe řadu counterů \count0:\count1:\count2. Při sekvenčním čtení jednotlivých uzlů stromu si

\urlspecchar: 36 \outlines: 36–38 \outlinesA: 36–37

každý uzel zvětší v této pomyslné řadě hodnotu svého counteru o jedničku. Kapitoly zvětšují `\count0`, sekce `\count1`, podsekce `\count2`. Stačí tedy zvětšit `\count`⟨*odsazení*⟩. Řada counterů pak jednoznačně určuje zpracovávaný uzel. Uzly pro kapitoly mají přidělenu kontrolní sekvenci `ol:\the\count0` a uzly pro sekce mají přidělenu kontrolní sekvenci `ol:\the\count0:\the\count1`. Jsou to makra, jejichž obsahem je počet potomků daného uzlu. Makrem `\addoneol` ⟨*csname*⟩ zvětšíme obsah dané kontrolní sekvence o jedničku. Příkazem `\ifcase`⟨*odsazení*⟩ řešíme, kterému rodiči je třeba zvednout tuto hodnotu. Při nule (kapitola) nikomu, neboť daný uzel nemá rodiče. Při ⟨*odsazeni*⟩=1 zvětšíme o jedničku počet potomků nadřazené kapitole a při ⟨*odsazeni*⟩=2 nadřazené sekci. Asi by bylo přehlednější na začátku definovat všechny potřebné sekvence `ol:`⟨*něco*⟩ a nastavit jim hodnotu 0. Ovšem šetříme pamětí i časem, takže zakládáme sekvenci `ol:`⟨*něco*⟩ teprve v makru `\addoneol` a to tehdy, když je ji poprvé potřeba zvětšit o jedničku.

opmac.tex

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

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 `\lccode` znaků tak, aby `\lowercase` odstranil háčky a čárky. To vzápětí provedeme, ale nejprve ještě do toho může promluvit uživatel v makru `\cnvhook`, které je implicitně nastaveno na makro prázdné.

opmac.tex

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

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` připravenu 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

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

\addoneol: 37 \outlinesB: 36–37 \outlinesC: 37 \outlinelevel: 36–37

Makro \setcnvcodesA zkontroluje, zda je uživatelem definováno makro \toasciidata⟨*iso-kód*⟩. Pokud je, použije ho jako \toasciidata ke konverzi akcentovaných znaků na neakcentované. Jinak podle definovanosti \r zkontroluje, zda je zapnutý \csaccents a pokud je, expanduje interní \toasciidata. Makro \toasciidata potřebujeme expandovat, protože neobsahuje přímý zápis znaků. Důvod je zřejmý, nechceme, aby se soubor opmac.tex stal závislý na použitém kódování.

opmac.tex

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

Na řádku 1007 se makro \setlccodes spustí jako \setlccodes␣AAĂAÄAáa...{}{}. Toto makro si odloupne dva parametry xy, provede \lccode‘x=‘y a v rekurzivním cyklu pokračuje v činnosti, dokud nenarazí na {}{}.

opmac.tex

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

Makro \insertoutline {⟨*text*⟩} vloží jedinou položku do záložky. Pro tuto položku se předpokládá nulový počet potomků. Využití: uživatel může takto odkázat na začátek nebo konec dokumentu. Jako lejblík je použito oul:⟨*oulnum*⟩, kde \oulnum průběžně zvětšujeme o jedničku.

opmac.tex

```
\newcount\oulnum
\def\insertoutline#1{\global\advance\oulnum by1
   \pdfdest name{oul:\the\oulnum} xyz\relax
   \pdfoutline goto name{oul:\the\oulnum} count0 {#1}\relax
}
```

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

opmac.tex

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

## 3.18 Verbatim

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

opmac.tex

```
\newcount\ttline    \ttline=-1
\newcount\viline
\newread\vifile
```

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

---

\setcnvcodesA: 37–38 \toasciidata: 37–38 \setlccodes: 26, 37–38 \insertoutline: 38
\oulnum: 38 \ttline: 38–39, 41 \viline: 38–41 \vifile: 38–41 \setverb: 39–41
\begtt: 7, 39, 41

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

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

opmac.tex
```
\def\testparA{\expandafter\testparB\romannumeral-`\.}
\def\testparB{\futurelet\tmpa\testparC}
\def\testparC{\ifx\tmpa\par\else\afternoindent\fi}
```

Makro \printttline vytiskne číslo řádku.

opmac.tex
```
\def\printttline{\llap{\sevenrm\the\ttline\kern.9em}}
```

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.

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

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

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

Cílem vyhodnocení parametru v závorce makra \verbinput jsou dva údaje: \vinolines bude obsahovat počet řádků, které je od začátku souboru nutno přeskočit, než se má zahájit přepisování

\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 verbinput. Vyšetření parametru ukončeného textem +\relax se v makru \viscanparameter větví na případ, kdy parametr obsahuje symbol + a použije se pak \viscanplus. Druhý případ, kdy uživatel nenapsal symbol plus (takže parametr #2 makra \viscanparameter je prázdný) je dále vyšetřen v makru \viscanminus. Obě makra si oddělí do svých parametrů první a druhou číslici (každá z nich může být prázdná) a nastaví podle zdokumentovaných pravidel pro zápis parametru odpovídající interní údaje \vinolines a \vidolines. Vychází přitom z předpokladu, že registr \viline obsahuje číslo naposledy přečteného řádku (nebo nulu, jsme-li na začátku souboru).

opmac.tex

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

Makro \doverbinput provede samotnou práci: přeskočí \vinolines řádků a přepíše \vidolines řádků. To provede v prvním a druhém cyklu \loop. Než se k těmto cyklům dostane, musí udělat jisté přípravné práce. Nejprve odečte od \vinolines počet už přečtených řádků, protože při opakovaném čtení stejného souboru jej neotevíráme znova, jen přeskočíme příslušný menší počet řádků. Pokud ale se ukáže, že rozdíl je záporný (je potřeba se v souboru vracet dozadu), makro znovuotevře soubor ke čtení pomocí \openin a upraví podle toho příslušné údaje o řádcích. Pak zahájí skupinu, dále pomocí \setverb nastaví speciálním znakům kategorii 12 a pomocí \adef{␣}{␣} nastaví mezeře aktivní kategorii (bude expandovat na neaktivní mezeru jako \space) a také nastaví kategorii 12 znaku, který byl deklarován pomocí \activettchar. Připraví odsazení podle \ttindent a spustí uživatelský \tthook. Je-li potřeba tisknout čísla řádků, připraví si na to font \sevenrm, který má velikost rovnu 0,7 násobku základní velikosti. A pustí se do zmíněných dvou cyklů \loop. V obou cyklech se může stát, že narazíme nečekaně na konec souboru. To je ošetřeno testem \ifeof\vifile a následnou úpravou čítače \tmpnum tak, abychom okamžitě vyskočili z cyklu. Druhý cyklus obsahuje ještě jeden speciální rys: přeje-li si uživatel číst až do konce souboru, je nastaveno \vidolines na nulu a před zahájením cyklu je čítač \tmpnum nastaven na -1. Uvnitř cyklu je pak zajištěno, že v tomto případě není čítač zvětšován o jedničku. Po ukončení práce v těchto dvou cyklech je ukončena skupina, vložena mezera \ttskip a makrem \testparB se ověří, zda následuje prázdný řádek.

opmac.tex

```
\def\doverbinput{%
   \tmpnum=\vinolines
   \advance\tmpnum by-\viline
   \ifnum\tmpnum<0
      \openin\vifile=\vifilename\space
      \global\viline=0
   \else
```

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

```
1123:       \edef\vinolines{\the\tmpnum}%
1124:    \fi
1125:    \vskip\parskip \ttskip \bgroup \wipeepar
1126:    \setverb \adef{ }{\ }%
1127:    \ifx\savedttchar\undefined \else \catcode\savedttchar=12 \fi
1128:    \parindent=\ttindent \parskip=0pt
1129:    \tthook\relax
1130:    \ifnum\ttline<-1 \else
1131:      \tenrm \thefontscale[700]\let\sevenrm=\thefont
1132:      \everypar\expandafter{\the\everypar \glob\advance\ttline by1 \printttline}\fi
1133:    \def\par##1{\endgraf\ifx##1\egroup\else\penalty\ttpenalty\leavevmode\fi ##1}%
1134:    \obeylines \tmpnum=0 \lccode`\~=`\^^M \lowercase{\def\tmpb{~}}%
1135:    \loop \ifeof\vifile \tmpnum=\vinolines\space \fi
1136:          \ifnum\tmpnum<\vinolines\space
1137:          \vireadline \advance\tmpnum by1 \repeat      %% skip line
1138:    \ifnum\ttline=-1 \ttline=\viline \let\glob=\relax \else\let\glob=\global \fi
1139:    \tmpnum=0 \ifnum\vidolines=0 \tmpnum=-1 \fi
1140:    \ifeof\vifile \tmpnum=\vidolines\space \fi
1141:    \loop \ifnum\tmpnum<\vidolines\space
1142:            \vireadline
1143:            \ifeof\vifile \tmpnum=\vidolines\space \else \viprintline \fi %% print line
1144:            \ifnum\vidolines=0 \else\advance\tmpnum by1 \fi
1145:            \repeat
1146:    \tt\expandafter\ptthook\tmpb\egroup\par\ttskip\testparA
1147: }
```

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

opmac.tex

```
1148: \def\vireadline{\read\vifile to \tmp \global\advance\viline by1 }
1149: \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 uživatelův `\tabstrut`, ovšem přechodně budeme toto makro měnit. Také deklarujeme čítač `\colnum`, ve kterém budeme mít po přečtení deklarace uložen počet sloupců tabulky. Dále během skenování ⟨*deklarace*⟩ vytvoříme makro `\ddlinedata`, které bude obsahovat `&\dditem␣&\dditem␣...` (počet těchto dvojic bude roven $n-1$, kde $n$ je počet sloupců). Pokud je v deklaraci dvojitá svislá čára, bude v makru `\ddlinedata` 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

```
1153: \newtoks\tabdata
1154: \def\tabstrutA{\tabstrut}
1155: \newcount\colnum
1156: \def\ddlinedata{}
1157: \def\vvleft{}
```

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

opmac.tex

```
1159: \def\table{\vbox\bgroup \catcode`\|=12 \tableA}
1160: \def\tableA#1#2{\offinterlineskip \colnum=0 \def\tmpa{}\tabdata={}\scantabdata#1\relax
1161:    \halign\expandafter{\the\tabdata\cr#2\crcr}\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*⟩}.

opmac.tex

```
\def\scantabdata#1{\let\next=\scantabdata
   \ifx\relax#1\let\next=\relax
   \else\ifx|#1\addtabvrule
      \else\isinlist{123456789}#1\iftrue \def\next{\scantabdataC#1}%
          \else \expandafter\ifx\csname tabdeclare#1\endcsname \relax
                \expandafter\ifx\csname paramtabdeclare#1\endcsname \relax
                   \opwarning{tab-declarator "#1" unknown, ignored}%
                \else \def\next{\expandafter \scantabdataB \csname paramtabdeclare#1\endcsname}\fi
             \else \def\next{\expandafter\scantabdataA \csname tabdeclare#1\endcsname}%
   \fi\fi\fi\fi \next
}
```

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.

opmac.tex

```
\def\scantabdataA#1{\addtabitem \expandafter\addtabdata\expandafter{#1\tabstrutA}\scantabdata}
\def\scantabdataB#1#2{\addtabitem\expandafter\addtabdata\expandafter{#1{#2}\tabstrutA}\scantabdata}
\def\scantabdataC {\def\tmpb{}\afterassignment\scantabdataD \tmpnum=}
\def\scantabdataD#1{\loop \ifnum\tmpnum>0 \advance\tmpnum by-1 \addto\tmpb{#1}\repeat
   \expandafter\scantabdata\tmpb
}
```

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.

opmac.tex

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

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

opmac.tex

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

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

```
tabdata: \vrule\tabiteml\hfil#\unsskip\hfil\tabitemr\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

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

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

opmac.tex
```
\def\crl{\crcr\noalign{\hrule}}
\def\crll{\crcr\noalign{\hrule\kern\hhkern\hrule}}
```

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

opmac.tex
```
\def\crli{\crcr \omit
   \gdef\dditem{\omit\tablinefil}\gdef\vvitem{\kern\vvkern\vrule}\gdef\vvitemA{\vrule}%
   \vvleft\tablinefil\ddlinedata\crcr}
\def\crlli{\crli\noalign{\kern\hhkern}\crli}
\def\tablinefil{\leaders\hrule\hfil}
```

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

opmac.tex
```
\def\tskip{\afterassignment\tskipA \tmpdim}
\def\tskipA{\gdef\dditem{}\gdef\vvitem{}\gdef\vvitemA{}\gdef\tabstrutA{}%
    \vbox to\tmpdim{}\ddlinedata \crcr \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 \crll: 43 \crli: 41, 43 \tablinefil: 43 \tabvvline: 43 \dditem: 41, 43
\vvitem: 41–43 \crlli: 43 \tskip: 41, 43 \tskipA: 43 \mspan: 44 \mspanA: 44
\mspanB: 44

opmac.tex
```
\def\mspan{\omit \tabdata={\tabstrut}\let\tmpa=\relax \afterassignment\mspanA \mscount=}
\def\mspanA[#1]{\loop \ifnum\mscount>1 \csname span\endcsname \omit \advance\mscount by-1 \repeat
   \mspanB#1\relax}
\def\mspanB#1{\ifx\relax#1\def\tmpa{\def\tmpa####1}%
   \expandafter\tmpa\expandafter{\the\tabdata\ignorespaces}\expandafter\tmpa\else
   \ifx |#1\ifx\tmpa\vrule\addtabdata{\kern\vvkern}\fi \addtabdata{\vrule}\let\tmpa=\vrule
   \else \let\tmpa=\relax
      \ifx c#1\addtabdata{\tabiteml\hfil\ignorespaces##1\unsskip\hfil\tabitemr}\fi
      \ifx l#1\addtabdata{\tabiteml\ignorespaces##1\unsskip\hfil\tabitemr}\fi
      \ifx r#1\addtabdata{\tabiteml\hfil\ignorespaces##1\unsskip\tabitemr}\fi
   \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.

opmac.tex
```
\let\orihrule=\hrule  \let\orivrule=\vrule
\def\rulewidth{\afterassignment\rulewidthA \drulewidth}
\def\rulewidthA{\edef\hrule{\orihrule height\the\drulewidth}%
                \edef\vrule{\orivrule width\the\drulewidth}}
```

Makro `\frame` {⟨*text*⟩} vloží vnější `\hbox{\vrule`⟨*vnitřek*⟩`\vrule}`. Uvnitř tohoto boxu se nachází `\vtop{`⟨*další*⟩`\kern\vvkern\hrule}`, takže ⟨*další*⟩ zůstává na účaří. Přitom ⟨*další*⟩ je `\vbox{\hrule\kern\vvkern`⟨*další2*⟩`}`, takže ⟨*další2*⟩ zůstává na účaří. V tuto chvíli jsou již vytvořeny čáry vlevo, vravo, nahoře i dole. Konečně ⟨*další2*⟩ je `\hbox{\kern\hhkern`⟨*text*⟩`\kern\hhkern}`.

opmac.tex
```
\long\def\frame#1{%
   \hbox{\vrule\vtop{\vbox{\hrule\kern\vvkern
      \hbox{\kern\hhkern#1\kern\hhkern}%
   }\kern\vvkern\hrule}\vrule}}
```

## 3.20 Vložení obrázku

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

opmac.tex
```
\newdimen\picwidth    \picwidth=0pt   \let\picw=\picwidth
\newdimen\picheight   \picheight=0pt
```

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

opmac.tex
```
\ifpdftex
  \def\inspic #1 {\hbox{%
      \pdfximage \ifdim\picwidth=0pt \else width\picwidth\fi
                 \ifdim\picheight=0pt \else height\picheight\fi \inspicpage {\picdir#1}%
      \pdfrefximage\pdflastximage}}
\else
  \def\inspic #1 {\opwarning
     {The \noexpand\inspic is supported for PDF output only}}
\fi
\def\inspicpage{}
```

Makro `\inspicpage` může při natažení PDF obsahovat text `page`⟨*number*⟩. Pak se jako obrázek použije odpovídající strana PDF dokumentu.

## 3.21 PDF transformace

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

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

---

Na druhé straně makro \pdfrotate ⟨*úhel*⟩ vytvoří \pdfsetmatrix{cos φ sin φ −sin φ cos φ}, což není jednoduché, protože funkce cos, sin nejsou v TEXu implementovány. Balíček trig.sty nabízí vyhodnocování těchto funkcí pomocí Taylorových polynomů, nicméně OPmac nechce být závislý na balíčcích a také chce ukázat alternativní způsob implementace. Makro \pdfrotate pracuje zhruba takto: je-li argument 0, neprovede nic, je-li argument 90, provede otočení o 90 stupňů. V ostatních případech zavolá makro \pdfrotateA, které rozloží argument na celou #1 a zlomkovou #2 část. V další části na řádcích 1275 až 1284 se zabývá jen celými stupni. Nejprve pomocí prvního a druhého \loop posune argument o celé násobky 360 stupňů tak, že poté je argument mezi 0 až 360 stupni, a přitom se hodnoty funkcí sin a cos nezměnily. Ve třetím \loop postupně snižuje argument o 90 stupňů a přitom dělá rotaci o 90 stupňů tak dlouho, až máme argument mezi nulou a devadesáti. Je-li dále argument větší než 44 stupňů, otočíme se o 45 a snížíme argument o 45. Je-li dále argument větší než 22, otočíme se o 22 a snížíme argument o 22. Nyní máme argument v množině $\{0, 1, 2, 3, \ldots, 22\}$. Pro každý prvek z této množiny argumentů máme předpřipraveny hodnoty funkcí cos a sin v makrech \smallcos a \smallsin. Použijeme je pro závěrečnou rotaci. Tím máme sazbu otočenou o celé stupně. Další část makra na řádcích 1287 až 1291 řeší jemné dotočení podle zlomkové části argumentu. V intervalu nula až jeden stupeň aproximujeme funkci cos konstantní jedničkou a funkci sin lineární funkcí $x \cdot \pi/180$. V daném rozmezí je to velmi dobrá aproximace.

opmac.tex

```
\def\pdfrotate#1{\tmpdim=#1pt
   \ifdim\tmpdim=0pt
   \else \ifdim\tmpdim=90pt \pdfsetmatrix{0 1 -1 0}%
         \else \edef\tmp{#1}\expandafter\pdfrotateA\tmp..\relax
   \fi   \fi
}
\def\pdfrotateA #1.#2.#3\relax{%
   \def\tmp##1.##2\relax {##1}%
   \tmpnum=\expandafter \tmp \the\tmpdim \relax % round
   \ifdim\tmpdim>0pt \def\tmpa{}\else\def\tmpa{-}\fi % save -
   \loop \ifnum\tmpnum<0 \advance\tmpnum by360 \repeat
   \loop \ifnum\tmpnum>360 \advance\tmpnum by-360 \repeat
   \loop \ifnum\tmpnum>90 \pdfrotate{90}\advance\tmpnum by-90 \repeat
   \ifnum\tmpnum=90 \pdfrotate{90}\else
      \ifnum\tmpnum>44 \pdfsetmatrix{.7071 .7071 -.7071 .7071}%
                       \advance\tmpnum by-45 \fi
      \ifnum\tmpnum>22 \pdfsetmatrix{.9272 .3746 -.3746 .9272}%
                       \advance\tmpnum by-22 \fi
      \ifnum\tmpnum>0
         \pdfsetmatrix{\smallcos \smallsin -\smallsin \smallcos}%
   \fi\fi
   \if$#2$\else % fraction part
      \tmpdim=.01745329pt % \pi/180
      \tmpdim=.#2\tmpdim  %
      \edef\tmp{\expandafter\ignorept\the\tmpdim\space}%
      \ifx\tmpa\empty \pdfsetmatrix{1 \tmp -\tmp 1}%
      \else           \pdfsetmatrix{1 -\tmp \tmp 1}%
   \fi\fi
}
\def\smallcos{.\ifcase\tmpnum \or9998\or9994\or9986\or9976\or9962\or9945\or
  9925\or9903\or9877\or9848\or9816\or9781\or9744\or9703\or9659\or9613\or
  9563\or9511\or9455\or9397\or9336\or9272\fi\space}
\def\smallsin{.\ifcase\tmpnum 0\or0175\or0359\or0523\or0698\or0872\or1045\or
  1219\or1391\or1564\or1736\or1908\or2079\or2250\or2419\or2588\or2756\or
  2924\or309\or3256\or342\or3584\or3746\fi\space}
```

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

opmac.tex

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

\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 \vfootnote, které vloží sazbu pomocí tzv. insertu (TBN, kapitola 6.7). PlainTEXové nastavení této třídy insertu není makrem OPmac nijak měněno.

opmac.tex
```
\newcount\fnotenum \fnotenum=0
\newcount\fnotenumlocal
\newif\iflocfnum \locfnumtrue

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

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

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

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

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

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

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

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

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

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
```
\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
```
\newcount\mnotenum    \mnotenum=0       % global counter of mnotes
\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 \vadjust nalepí svůj materiál bez mezery pod tento řádek. My ovšem potřebujeme vycouvat nahoru na účaří řádku. To nejde snadno provést, protože hloubka řádku je proměnlivá. Proto do je řádku vložen \strut a předpokládá se, že nyní má řádek hloubku \dp\strutbox a o tento rozměr makro vycouvá nahoru. Vloží požadovaný box výšky nula na úrovni účaří a pak se vrátí na původní místo.

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

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

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

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

opmac.tex
```
\def\Xmnote{\advance\mnotenum by1
   \sxdef{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.

opmac.tex
```
\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

opmac.tex
```
\newwrite\auxfile                    % AUX file for BibTeX
\newcount\bibnum                     % the bibitem counter
\newtoks\bibmark                     % the bibmark used if \nonumcitations
\newcount\lastcitenum  \lastcitenum=0  % for \shortcitations
```

Makro \cite [⟨lejblík1⟩,⟨lejblík2⟩,...] si opakovaně zavolá \citeA⟨lejblík-i⟩, 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.

opmac.tex
```
\def\cite[#1]{{\citeA#1,,,[\printsavedcites]}}
\def\nocite[#1]{{\citeA#1,,,}}
\def\rcite[#1]{{\citeA#1,,,\printsavedcites}}
\def\savedcites{}
```

Makro \citeA ⟨lejblík⟩, řeší zhruba řečeno následující věci:

- Zjistí, zda je definován \csname␣bib:⟨lejblík⟩\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:⟨lejblík⟩\endcsname bude obsahovat ⟨číslo-citace⟩ po použití \bib[⟨lejblík⟩] nebo \bibitem{⟨lejblík⟩}. Tato makra uloží odpovídající informaci do REF souboru, odkud ji při opakovaném TEXování vyzvedneme. Je to klasická činnost, kterou provozujeme i u ostatních křížových referencí.
- Uloží o sobě zprávu do bufferu \citelist. To použijeme v makrech \usebibtex nebo \usebbl.

Makro \citeA je naprogramováno zhruba takto

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

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

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

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

opmac.tex

```
\def\citeA #1#2,{\if#1,\else
   \if *#1\addcitelist{*}\expandafter \skiptorelax \fi
   \isdefined{bib:#1#2}\iftrue \else
      \addcitelist{#1#2}%
      \opwarning{The cite [#1#2] unknown. Try to TeX me again}\openref
      \addto\savedcites{?,}\def\sortcitesA{}\lastcitenum=0
      \expandafter\gdef\csname bib:#1#2\endcsname {}%
      \expandafter \skiptorelax \fi
   \expandafter \ifx \csname bib:#1#2\endcsname \empty
      \addto\savedcites{?,}\def\sortcitesA{}\lastcitenum=0
      \expandafter \skiptorelax \fi
   \def\bibnn##1{}%
   \if &\csname bib:#1#2\endcsname
      \addcitelist{#1#2}%
      \def\bibnn##1##2{##1}%
      \sxdef{bib:#1#2}{\csname bib:#1#2\endcsname}%
   \fi
   \edef\savedcites{\savedcites \csname bib:#1#2\endcsname,}%
   \relax
   \expandafter\citeA\fi
}
```

Makro snímá svůj parametr jako `#1#2,` aby mohly být ⟨*lejblíky*⟩ 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 `bib:`⟨*lejblík*⟩ uloží `\bibnn{`⟨*hodnota*⟩`}&`. Příkaz `\if` za sebou totálně expanduje vše následující, takže nejprve narazí na `&`, pak se obsah `bib:`⟨*lejblík*⟩ expanduje prostřednictvím `\bibnn {`⟨*hodnota*⟩`}` na nic a za tímto „nic“ se zjeví druhý znak `&`, který se tedy přilepí na ten první. Ano, je pravda, že tyto dva znaky jsou stejné. Odstranění tohoto znaku probíhá znovu totální expanzí, tentokrát `\bibnn` první parametr ⟨*hodnota*⟩ zopakuje a druhý parametr se znakem `&` zahodí.

Makro `\printsavedcites` případně setřídí seznam `\savedcites` 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.

opmac.tex

```
\def\printsavedcites{\sortcitesA
   \chardef\tmpb=0 \expandafter\citeB\savedcites,%
   \ifnum\tmpb>0 \printdashcite{\the\tmpb}\fi
}
```

Makro `\sortcitesA` seřadí seznam `\savedcites` 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 `\savedcites` 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\savedcites{...\expandafter}` způsobí, že se `\savedcites` 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`.

opmac.tex

```
\def\sortcitesA{}
\def\sortcitations{%
  \def\sortcitesA{\edef\savedcites{300000,\expandafter}\expandafter\sortcitesB\savedcites,%
                  \def\tmpa####1300000,{\def\savedcites{####1}}\expandafter\tmpa\savedcites}%
}
\def\sortcitesB #1,{\if $#1$%
  \else
     \mathchardef\tmpa=#1
     \edef\savedcites{\expandafter}\expandafter\sortcitesC \savedcites\end
```

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

```
      \expandafter\sortcitesB
   \fi
}
```

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

```
\def\sortcitesC#1,{\ifnum\tmpa<#1\edef\tmpa{\the\tmpa,#1}\expandafter\sortcitesD
                   \else\edef\savedcites{\savedcites#1,}\expandafter\sortcitesC\fi}
\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

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

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

opmac.tex
```
\def\nonumcitations{\lastcitenum=0\def\sortcitesA{}\def\etalchar##1{$^{##1}$}%
   \def\citelinkA##1{\isdefined{bim:##1}\iftrue \csname bim:##1\endcsname
      \else ##1\opwarning{\noexpand\nonumcitations + empty bibmark. Maybe bad BibTeX style}\fi}
}
\def\citelinkA{}
```

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
```
\def\ecite[#1]{\bgroup\citeA#1,,,\expandafter\eciteB\savedcites;}
\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⟩]␣=␣{⟨značka⟩}. Případná mezera před rovnítkem je odstaněna pomocí triku s \rommannumeral, který při záporném čísle expanduje na prázdný výsledek, ale případná mezera za ‘\. při skenování tohoto čísla je pozřena. Při zkráceném zápisu makra \bib (bez rovnítka) 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
```
\def\bib[#1]{\def\tmp{\isnextchar={\bibA[#1]}{\bibmark={}\bibB[#1]}}%
   \expandafter\tmp\romannumeral-‘\.} % ignore optional space
\def\bibA[#1]=#2{\bibmark={#2}\bibB[#1]}
\def\bibB[#1]{\par \ifnum\bibnum>0 \bibskip \fi
   \advance\bibnum by1
   \noindent \def\tmpb{#1}\wbib{#1}{\the\bibnum}{\the\bibmark}%
   \printbib \ignorespaces
}
\def\wbib#1#2#3{\dest[cite:\the\bibnum]%
   \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
```
\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
```
\def\printbib{\hangindent=\iindent
   \ifx\citelinkA\empty \noindent\hskip\iindent \llap{[\the\bibnum] }%
   \else \noindent \fi
}
```

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

opmac.tex

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

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

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

opmac.tex

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

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

opmac.tex

```
\def\readbblfile #1{%
  \openin\testin=#1.bbl
  \ifeof\testin
    \opwarning{The ‘#1.bbl’ file doesn’t exist. Use ‘bibtex’..}%
  \else
    \closein\testin
    \bibnum=0
    \long\def\begin##1\bibitem{\bibitem}\def\end##1{}% LaTeX environment
    \def\httpAddr##1{\url{http:##1}}\def\\{\hfill\break}%
    \def\newblock{\hskip .11em plus.33em minus.07em}%
    \def\mbox{\leavevmode\hbox}\def\emph##1{{\it##1}}%
    \parindent=\iindent \bibtexhook\relax
    \input #1.bbl
    \par
  \fi
```

\addcitelist: 49, 52–54 \citelist: 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

```
1505: }
```

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

```
1506: \def\bibitem{\isnextchar[{\bibitemB}{\bibmark={}\bibitemC}}
1507: \def\bibitemB[#1]{\bibmark={#1}\bibitemC}
1508: \def\bibitemC#1{\bibitemD{#1}}
1509: \def\bibitemD#1{\par\ifnum\bibnum>0 \bibskip \fi
1510:    \advance\bibnum by1
1511:    \noindent \def\tmpb{#1}\wbib{#1}{\the\bibnum}{\the\bibmark}%
1512:    \printbib \ignorespaces
1513: }
```

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

opmac.tex

```
1514: \def\genbbl#1#2{\openauxfile{#1}{#2}%
1515:    \immediate\write\auxfile{\string\citation{*}}%
1516:    \bgroup
1517:      \iindent=4em
1518:      \def\bibitemC##1{\par\ifnum\bibnum>0 \bibskip \fi
1519:         \advance\bibnum by1
1520:         \noindent \hangindent=\parindent
1521:         \indent \llap{[##1]\enspace}\ignorespaces
1522:      }%
1523:      \readbblfile{\jobname}%
1524:    \egroup
1525: }
```

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

```
1526: \def\usebbl/#1 #2 {\isdefined{bbl:#1}%
1527:    \iftrue \csname bbl:#1\endcsname {#2}\else
1528:       \opwarning{\string\usebbl/#1 #2 ... the ‘#1’ type undefined}%
1529:    \fi
1530: }
1531: \sdef{bbl:a}#1{\bgroup \readbblfile{#1}\egroup}
1532:
1533: \sdef{bbl:b}#1{\bgroup
1534:      \let\citeI=\relax \xdef\citelist{\citelist\citelistB}%
1535:      \def\bibitemC##1 ##2\par{%
1536:         \isinlist\citelist{[##1]}\iftrue \bibitemD{##1}##2\par\fi}%
1537:      \readbblfile{#1}%
1538:      \global\let\addcitelist=\writeXcite
1539:   \egroup
1540: }
1541: \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

```
1542:      \ifx\citelinkA\empty \else
1543:          \opwarning{\string\nonumcitations: don't use \string\usebbl/c}\fi
1544:      \let\citeI=\relax \xdef\citelist{\citelist\citelistB}%
1545:      \def\bibitemC##1 ##2\par{%
1546:         \isinlist\citelist{[##1]}\iftrue
1547:            \if^\the\bibmark^\sdef{bb:##1}{\bibitemD{##1}##2\par}%
1548:            \else \toks0={##2\par}%
1549:                  \edef\tmpa{\noexpand\sdef{bb:##1}{% \the\bibmark have to expand
1550:                        \noexpand\bibitemB[\the\bibmark]{##1}\the\toks0}}\tmpa
1551:         \fi\fi}%
1552:      \readbblfile{#1}%
1553:      \def\bibitemC##1{\bibitemD{##1}}%
1554:      \def\citeI[##1]{\csname bb:##1\endcsname}\citelist
1555:      \global\let\addcitelist=\writeXcite
1556:   \egroup
1557: }
```

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

opmac.tex

```
1558: \def\Xcite#1{\addto\citelistB{\citeI[#1]}}
```

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

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.

opmac.tex

```
1560: \def\usebib{\par \input opmac-bib \usebib}
```

Uživatel nicméně může makro soubor na začátku svého dokumentu volat explicitně pomocí \input␣opmac-bib.

## 3.24 Úprava output rutiny

OPmac mění output rutinu proti originální \plainoutput jen v nejnutnějších 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 implictně prázdné, spouští se v \begoutput a může v něm být nastaveno střídání okrajů pro liché a sudé stránky, viz též makro \margins.

opmac.tex

```
1564: \output={\begoutput \opmacoutput \endoutput}
1565: \def\begoutput{%
1566:    \immediate\wref\Xpage{{\the\pageno}}%
1567:    \def\nl{ }\def\fnote##1{}\def\fnotemark##1{}%
1568:    \prephoffset
1569: }
```

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

```
1570: \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

```
1572: \def\opmacoutput{%
1573:   \setbox0=\vbox{\prepghook\ensureblacko{\makeheadline}\pagebody\ensureblacko{\makefootline}}%
1574:   \pghook \protectlist
1575:   \shipout\box0 \advancepageno
1576:   \ifnum\outputpenalty>-20000 \else\dosupereject\fi
1577: }
1578: \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

```
1579: \def\prepage{\def\destheight{25pt}\dest[pg:\pgilabel.\the\pageno]}
1580: \def\preboxcclv{}  \def\postboxcclv{}
```

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

opmac.tex

```
1582: {\catcode`\@=11
1583: \gdef\pagecontents{\prepage % dest of pageno
1584:   \ifvoid\topins\else\ensureblacko{\unvbox\topins}\fi
1585:   \preboxcclv
1586:   \dimen@=\dp\@cclv \unvbox\@cclv % open up \box255
1587:   \postboxcclv
1588:   \ifvoid\footins\else % footnote info is present
1589:     \vskip\skip\footins
1590:     \ensureblacko{\footnoterule \unvbox\footins}\fi
1591:   \ifr@ggedbottom \kern-\dimen@ \vfil \fi
1592: }}
```

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

```
1594: \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

```
1596: \newcount\lastpage  \lastpage=0  % the last page of the document
1597: \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

```
1601: \newdimen\pgwidth  \newdimen\pgheight  \pgwidth=0pt
1602: \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 \margins /⟨typ⟩␣⟨formát⟩␣(⟨levý⟩,⟨pravý⟩,⟨horní⟩,⟨dolní⟩)⟨jednotka⟩␣ si nastaví registry \pgwidth a \pgheight prostřednictvím \setpagedimens a dále v souladu s uživatelskou dokumentací nastaví potřebné okraje. V makru \tmp je schována jednotka, kterou uživatel taky může zapomenout napsat. V takovém případě vypíšeme varování a doplníme jednotku mm. Jakmile měníme \hoffset nebo \voffset, nastavíme je nejprve na -1in (tím se dostaneme na okraj papíru) a pak budeme požadovanou velikost okraje k těmto registrům přidávat. Nemohu za to, že Knutha napadla taková ne příliš podařená myšlenka dát výchozí bod sazby kamsi doprostřed papíru umístěný pomocí ujetých jednotek. Za zmínku stojí ještě dvě myšlenky. Makro \rbmargin \h(v)offset\h(v)size{⟨okraj⟩} provede výpočet hodnoty \hoffset nebo \voffset v případě, že je dána protější hodnota okraje než je okraj přímo nastavitelný pomocí \h(v)offset. A konečně posun okraje při přechodu z pravé na levou stránku \shiftoffset počítáme jako \pgwidth - \hsize - 2*⟨levý⟩ což dá stejnou hodnotu jako ⟨pravý⟩-⟨levý⟩. Změna \hoffset o tuto hodnotu je provedena v makru \pghook, tedy v \output rutině, schována do skupiny, takže po ukončení \output rutiny se vrátí \hoffset na původní hodnotu.

opmac.tex

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

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

opmac.tex

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

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

Jednotlivé ⟨*formáty*⟩ papíru je potřeba deklarovat.

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

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

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

### 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.

opmac.tex
```
\ifx\XeTeXversion\undefined \else \pdftexfalse \input opmac-xetex \fi
\ifx\pdfextension\undefined \else \input opmac-luatex \fi
\inputref
\endinput
```

## 4 Rejstřík

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