

DocBy.T_EX – nástroj na dokumentování zdrojových kódů

Petr Olšák

www.olsak.net/docbytex.html

Obsah

1	Úvod	3
2	Pro uživatele	4
	2.1 Členění souborů	4
	<code>\module</code> ... 4	
	2.2 Příklad dokumentace modulu	5
	<code>\ins</code> ... 5, <code>dvojice</code> ... 5, <code>uzasna_funkce</code> ... 5	
	2.3 Jaký T _E X pro DocBy.T _E X?	6
	<code>\enc</code> ... 6, <code>\NOenc</code> ... 6, <code>\PDF</code> ... 6, <code>\DVI</code> ... 6	
	2.4 Vyhledávání slov encT _E Xem	7
	<code>\noactive</code> ... 7, <code>\onlyactive</code> ... 7	
	2.5 Generování rejstříku, obsahu, poznámek pod čarou a záložek	7
	<code>\dotoc</code> ... 7, <code>\doindex</code> ... 7, <code>\bye</code> ... 7, <code>\bookmarks</code> ... 7	
	2.6 Vkládání zdrojových textů podrobněji	8
	<code>\ifirst</code> ... 8, <code>\inext</code> ... 8, <code>\end</code> ... 8, <code>\empty</code> ... 8, <code>\nb</code> ... 8,	
	<code>\obrace</code> ... 8, <code>\cbrace</code> ... 8, <code>\percent</code> ... 8, <code>\inchquote</code> ... 8, <code>\lineno</code> ... 8,	
	<code>\skippingfalse</code> ... 8, <code>\skippingtrue</code> ... 8, <code>\count</code> ... 9	
	2.7 Odkazy na čísla řádků	9
	<code>\ilabel</code> ... 9	
	2.8 Verbatim ukázky pomocí <code>\begtt</code> / <code>\endtt</code> a palcových uvozovek	9
	<code>\begtt</code> ... 9, <code>\endtt</code> ... 9	
	2.9 Deklarace dokumentovaného slova	10
	<code>\dg</code> ... 10, <code>\dgn</code> ... 10, <code>\dgh</code> ... 10, <code>\dl</code> ... 10, <code>\dln</code> ... 10, <code>\dlh</code> ... 10,	
	<code>\iidg</code> ... 11, <code>\iidgh</code> ... 11, <code>\iidgn</code> ... 11, <code>\iidl</code> ... 11, <code>\iidlh</code> ... 11,	
	<code>\iidln</code> ... 11	
	2.10 Jmenné prostory	11
	<code>\namespace</code> ... 11, <code>\endnamespace</code> ... 11	
	2.11 Místo pro dokumentaci aplikačního rozhraní	11
	<code>\api</code> ... 11, <code>\apitext</code> ... 12	
	2.12 Sekce, sekcičky, část, titul	12
	<code>\sec</code> ... 12, <code>\subsec</code> ... 12, <code>\part</code> ... 12, <code>\title</code> ... 12, <code>\projectversion</code> ... 12,	
	<code>\author</code> ... 12, <code>\headtitle</code> ... 12, <code>\savetocfalse</code> ... 12, <code>\emptynumber</code> ... 12	
	2.13 Křížové odkazy	12
	<code>\label</code> ... 12, <code>\pgref</code> ... 12, <code>\numref</code> ... 12, <code>\ilink</code> ... 12, <code>\cite</code> ... 12,	
	<code>\labeltext</code> ... 13	
	2.14 Vkládání obrázků	13
	<code>\ifig</code> ... 13, <code>\figdir</code> ... 13	
	2.15 Výčty	13
	<code>\begitems</code> ... 13, <code>\enditems</code> ... 13, <code>\item</code> ... 13, <code>\itemno</code> ... 13	
3	Pro náročné	13
	3.1 Interní názvy	13
	<code>\titindex</code> ... 14, <code>\tittoc</code> ... 14, <code>\titmodule</code> ... 14, <code>\titversion</code> ... 14, <code>\opartname</code> ... 14	
	3.2 Vložené skupiny příkazů (hooks)	14
	<code>\begthook</code> ... 14, <code>\quotehook</code> ... 14, <code>\indexhook</code> ... 14, <code>\tohook</code> ... 14,	
	<code>\bookmarkshook</code> ... 14, <code>\outputhook</code> ... 14	
	3.3 Příkaz <code>\module</code> a <code>\ins</code>	15
	<code>\module</code> ... 15, <code>\docsuffix</code> ... 15, <code>\modulename</code> ... 15, <code>\ins</code> ... 15	
	3.4 Zelenající komentáře	15

	<code>\setlinecomment ... 15, \setlrcoment ...15, \linecomment ... 15, \leftcomment... 15,</code>	
	<code>\rightcomment ...15, \returntoBlack... 15</code>	
4	Pro designéry	16
4.1	Parametry a pomocná makra pro nastavení vzhledu	16
	<code>\hsize ...16, \vsize... 16, \parindent ...16, \width... 16, \bbf ...16,</code>	
	<code>\bbbff ...16, \btt ...16, \ttsmall... 16, \rmsmall ...16, \itsmall ...16,</code>	
	<code>\partfont ... 16, \setsmallprinting ...16, \ttstrut ... 16, \setnormalprinting... 16,</code>	
	<code>\Blue ...17, \Red... 17, \Brown ...17, \Green ...17, \Yellow ... 17, \Black ... 17,</code>	
	<code>\setcmykcolor ...17, \oriBlack ... 17, \rectangle ...17, \docbytex ... 17</code>	
4.2	Vzhled sekcí a podsekcí	17
	<code>\printsec ... 17, \printsecbelow... 17, \printsubsec ...18, \printsubsecbelow... 18,</code>	
	<code>\printpart ... 18, \printpartbelow ...18, \emptynumber ... 18</code>	
4.3	Titul, autor	18
	<code>\title ...18, \iititle... 18, \projectversion ... 19, \author ... 19</code>	
4.4	Hlavičky a patičky	19
	<code>\footline ... 19, \headline ...19, \normalhead ... 19, \noheadline... 19,</code>	
	<code>\headtitle ... 19, \headlinebox ... 19</code>	
4.5	Tisk cíle odkazu a odkazů pod čarou	20
	<code>\printdg... 20, \printdgininside ... 20, \printfnote... 20</code>	
4.6	Tisk údaje v obsahu a v rejstříku	20
	<code>\ptocline ... 20, \ptocsubline ... 20, \mydotfill ...20, \ptocentry ... 20,</code>	
	<code>\myldots... 21, \printindexentry ... 21, \separeright ... 21</code>	
4.7	Tisk zdrojového textu	21
	<code>\printiabove ...21, \printiline ...21, \printibelow ...21, \specrule ... 22,</code>	
	<code>\isnameprinted... 22</code>	
4.8	Tisk z prostředí <code>\begtt/\endtt</code>	22
	<code>\printvabove ...22, \printvline ... 22, \printvbelow ... 22</code>	
4.9	Vkládání obrázků	22
	<code>\figwidth... 22, \ifig... 22, \figdir... 22</code>	
4.10	Výčty	22
	<code>\begitems ... 22, \enditems ...22, \itemno ...22, \dbtitem... 22, \item ...22</code>	
5	Pro otrlé	23
5.1	Pomocná makra	23
	<code>\dbtwarning ...23, \defsec... 23, \edefsec ...23, \undef ...23, \nb ...23,</code>	
	<code>\obrace... 23, \lbrace ...23, \percent ...23, \inchquote... 23, \softinput ... 23,</code>	
	<code>\setverb... 23</code>	
5.2	Inicializace	23
	<code>\dbtversion ...24, \entextable ...24, \owordbuffer ...24, \noactive ... 24,</code>	
	<code>\emptysec ... 24, \sword ... 24, \onlyactive ...24, \oword ...24</code>	
5.3	Makra <code>\ifirst, \inext, \ilabel</code>	25
	<code>\lineno... 25, \ttlineno ... 25, \ifcontinue ...25, \infile ...25, \ifskipping ... 25,</code>	
	<code>\skippingfalse... 25, \skippingtrue ... 25, \ifirst... 25, \inputfilename ...25,</code>	
	<code>\inext ...25, \noswords ... 26, \readiparamwhy ...26, \startline ... 26, \stopline ...26,</code>	
	<code>\scaniparam ...26, \scaniparamA ...26, \scaniparamB... 26, \scaniparamC ... 26,</code>	
	<code>\insinternal ...26, \testline ...27, \nocontinue... 27, \returninsinternal... 27,</code>	
	<code>\readnewline ...27, \text ... 27, \etext... 27, \printilineA ...27, \lastline... 27,</code>	
	<code>\ilabel... 28, \ilabellist ...28, \ilabelee... 28, \testilabel ...28</code>	
5.4	Příkazy <code>\begtt, \endtt</code>	28
	<code>\begtt ...28, \startverb... 28, \runttloop ...28, \endttloop ...28,</code>	
	<code>\scannexttoken... 28</code>	
5.5	Jmenné prostory	29
	<code>\namespacemacro... 29, \namespace... 29, \locword ...29, \endnamespace... 29,</code>	
	<code>\ewrite... 29, \lword ... 30, \genlongword... 30, \refns ... 30, \refnsend ... 30,</code>	
	<code>\currns... 30</code>	
5.6	<code>\dg</code> a přátelé	30

<code>\dg ... 30, \dl ... 30, \dgn ... 30, \dgh ... 30, \dln ... 30, \dlh ... 30,</code>	
<code>\dgp ... 30, \dparam ... 30, \nextdparam ... 30, \varparam ... 30, \gobblelast ... 31,</code>	
<code>\managebrackets ... 31, \printbrackets ... 31, \maybespace ... 31, \iidg ... 31,</code>	
<code>\iidl ... 31, \iidgh ... 31, \iidlh ... 31, \iidgn ... 32, \fword ... 32, \iidln ... 32,</code>	
<code>\flword ... 32</code>	
5.7 Speciální poznámky pod čarou	32
<code>\totalfoocount ... 33, \totalfoodim ... 33, \specfootnote ... 33, \refcoef ... 33,</code>	
<code>\gobblrest ... 33</code>	
5.8 Sekce, podsekce	34
<code>\secnum ... 34, \subsecnum ... 34, \sectitle ... 34, \ifsavetoc ... 34,</code>	
<code>\savetocfalse ... 34, \sec ... 34, \subsec ... 34, \secparam ... 34, \seclabel ... 34,</code>	
<code>\secparamA ... 34, \secparamB ... 34, \nolastspace ... 34, \setparamC ... 34, \iisec ... 34,</code>	
<code>\makelinks ... 34, \iisubsec ... 35, \partnum ... 35, \thepart ... 35, \part ... 35,</code>	
<code>\iipart ... 35</code>	
5.9 Odkazy, reference	35
<code>\savelink ... 35, \ilink ... 35, \linkskip ... 35, \savepglink ... 36, \pglink ... 36,</code>	
<code>\dopglink ... 36, \reflabel ... 36, \numref ... 36, \pgref ... 36, \labeltext ... 36,</code>	
<code>\writelabel ... 36, \writelabelinternal ... 36, \label ... 36, \cite ... 36, \api ... 37,</code>	
<code>\apitext ... 37, \bye ... 37, \setrefchecking ... 37, \ignoretorelax ... 38</code>	
5.10 Tvorba obsahu, rejstříku a záložek	38
<code>\addtext ... 38, \reffile ... 38, \reftocline ... 38, \tocbuffer ... 38, \dotocline ... 38,</code>	
<code>\istocsec ... 38, \refdg ... 38, \refapiword ... 38, \dotoc ... 39, \indexbuffer ... 39,</code>	
<code>\doindex ... 39, \doindexentry ... 39, \ignoretwo ... 40, \remakebackslash ... 40,</code>	
<code>\addbookmark ... 40, \currb ... 40, \currsecb ... 40, \bookmarks ... 40, \setoutline ... 40,</code>	
<code>\cnvbookmark ... 40, \nobraces ... 40, \nobrA ... 40</code>	
5.11 Abecední řazení rejstříku	41
<code>\ifAleB ... 41, \nullbuf ... 41, \return ... 41, \fif ... 41, \sortindex ... 41,</code>	
<code>\mergesort ... 41, \isAleB ... 42, \testAleB ... 42, \napercarky ... 42</code>	
5.12 Transformace seznamu stránek	42
<code>\refuseword ... 42, \listofpages ... 42, \dgnum ... 43, \apinum ... 43, \tempnum ... 43,</code>	
<code>\ifdash ... 43, \iffirst ... 43, \transf ... 43, \cykltransf ... 43</code>	
5.13 Více sloupců	43
<code>\begmulti ... 44, \calculatedimone ... 44</code>	
5.14 Závěrečná nastavení, kategorie	44
<code>\subori ... 44, \langleactive ... 45</code>	
6 Rejstřík	45

1 Úvod

DocBy.T_EX umožňuje jednoduše dokumentovat pomocí T_EXu zdrojové kódy programu napsaném v jazyce C případně v jakémkoli jiném jazyce.

Na rozdíl od Knuthova literárního programování tento nástroj nepoužívá žádné preprocesory nebo filtry pro oddělení informace pro člověka a pro počítač. Vycházím z toho, že programátor je zvyklý psát tyto informace odděleně a chce mít věci pod vlastní kontrolou. Rovněž mnozí programátoři uvítají, že mohou psát dokumentaci dodatečně, a přitom skoro nezasahovat do už napsaného (a možná odladěného) zdrojového kódu. Doba, kdy Knuth navrhoval literární programování, pokročila a tvůrce dokumentace dnes může mít zároveň ve více oknech otevřeno více textů. Některé jsou určeny pro člověka a jiné pro počítač. Nevnímám tedy tak hlasitou potřebu tyto informace slučovat do jednoho souboru, jako tomu bylo kdysi.

V první části (sekce 2) dokumentu seznamujeme čtenáře s použitím DocBy.T_EXu na uživatelské úrovni. V další sekci jsou dokumentovaná výchozí makra DocBy.T_EXu, u nichž se předpokládá, že je bude chtít náročný uživatel měnit, aby přizpůsobil chování DocBy.T_EXu obrazu svému. Dále následuje sekce 4 s dokumentací maker, která rovněž budou měněna, pokud uživatel bude chtít jiný vzhled dokumentu. V poslední sekci 5 je dokumentován kompletní DocBy.T_EX na implementační úrovni. Takže se tam můžete dočíst, jak makra fungují.

Tento dokument je zpracován DocBy.T_EXem, takže slouží mimo jiné jako ukázka, co je možné tímto nástrojem vytvořit.

2 Pro uživatele

2.1 Členění souborů

DocBy.T_EX je implicitně navržen pro dokumentování zdrojových kódů v jazyce C. Proto i následující ukázka dokumentuje hypotetický program napsaný v tomto jazyce. Chcete-li dokumentovat jiný jazyk, můžete implicitní chování DocBy.T_EXu pozměnit. Tomu je věnována sekce 3.

Předpokládá se, že zdrojové kódy programu jsou členěny na moduly. Každý modul je myšlenkově samostatná záležitost. Alespoň pro programátora. Každý modul má své jméno (například `cosi`) a je napsán v souborech `cosi.h` a `cosi.c`, případně v dalších. Tyto soubory se kompilují, aby vznikl `cosi.o` a v závěru kompilace se linkují všechny kompilované moduly do výsledného programu.

Chceme-li takové zdrojové kódy dokumentovat, připišeme ke každému modulu soubor s příponou `.d`, například `cosi.d`, který obsahuje dokumentaci k danému modulu. Dále založíme třeba soubor `program.tex`, ze kterého postupně načítáme dokumentace jednotlivých modulů pomocí příkazu `\module`. V „hlavním souboru“ `program.tex` můžeme též použít příkazy `\title` pro vyznačení názvu programu, `\author` se jménem autora programu a třeba `\dotoc` pro vytvoření obsahu a `\doindex` pro vygenerování rejstříku. Samozřejmě zde můžeme napsat třeba úvodní poznámky ke zdrojovým kódům programu a použít plno dalších vymezených příkazů (viz dále). Obsah souboru `program.tex` může vypadat třeba takto:

```
\input docby.tex
\title   Program lup -- dokumentace ke zdrojovým textům

\author  Progr a Mátor

\dotoc   % tady bude obsah

\sec    Členění zdrojových textů

Zdrojové texty programu "lup" jsou rozděleny do tří modulů.
V "base.c" jsou definovány pomocné funkce a v "base.h" jsou jejich
prototypy. Podobně ve "win.c" jsou funkce pro okenní záležitosti a
"win.h" obsahuje jejich prototypy. Konečně "main.c" obsahuje hlavní
funkci programu.
\module base
\module win
\module main
\doindex % v tomto místě bude sestaven rejstřík
\bye
```

V tomto příkladě jsme se rozhodli čtenáře dokumentace seznamovat s programem „zdola nahoru“, tedy od elementárních funkcí až k hotovému programu. Někdo možná preferuje cestu „shora dolů“ a může mít v dokumentaci napsáno:

```
\module main
\module win
\module base
\doindex
\bye
```

Oba přístupy jsou možné, protože dokumentace je automaticky provázána hyperlinky. Čtenář se kdykoli může podívat na dokumentaci té funkce, jejíž použití zrovna čte, a obráceně může projít výskyty veškerého použití funkce, když čte její dokumentaci.

2.2 Příklad dokumentace modulu

Soubor s dokumentací jednotlivého modulu budu pro tento případ značit `cosi.d`. Ten je načten příkazem `\module cosi_`. V souboru `cosi.d` je možno se literárně vyřadit a kdykoli vložit část existujícího zdrojového kódu programu se stejným jménem modulu. To provedeme příkazem `\ins c_keyword_`, který vloží do dokumentace část zdrojového kódu ze souboru `cosi.c`, která je vymezena pomocí slova `keyword`. Místo písmene `c` je možno použít `h` nebo jakoukoli jinou příponu souboru, ze kterého chceme vložit část do dokumentace. K vymezení částí, které se mají vložit, je nutno mít ve zdrojovém souboru text `//:_keyword`. Vše vysvětlí následující příklad.

Předpokládejme, že v souboru `cosi.d` máme napsánu tuto dokumentaci:

```
Struktura \dg dvojice se používá jako návratová hodnota funkce
"uzasna_funkce" a sdružuje dvě hodnoty typu "float".
\ins c dvojice

Funkce \dg [struct dvojice] uzasna_funkce() si vezme jeden parametr "p"
a vrátí ve struktuře "dvojice" dvojnásobek a trojnásobek tohoto parametru.
\ins c uzasna_funkce
```

V tomto případě je nutné, aby v souboru `cosi.c` existoval vymežující text `//:_dvojice` a text `//:_uzasna_funkce`. Tyto texty vymežují úseky, které se mají do dokumentace vložit. Soubor `cosi.c` může vypadat třeba takto:

```
#include <stdio.h>

//: dvojice

struct dvojice {
    float x, y;
};

//: uzasna_funkce

struct dvojice uzasna_funkce (float p)
{
    struct dvojice navrat;
    navrat.x = 2*p; // tady nasobim p dvema
    navrat.y = 3*p; // tady nasobim p tremi
    return navrat;
}
```

Výsledek po zpracování části dokumentace z `cosi.d` pak vypadá takto:

Struktura `dvojice` se používá jako návratová hodnota funkce `uzasna_funkce` a sdružuje dvě hodnoty typu `float`.

```
5: struct dvojice {
6:     float x, y;
7: };
```

`cosi.c`

Funkce `uzasna_funkce` si vezme jeden parametr `p` a vrátí ve struktuře `dvojice` dvojnásobek a trojnásobek tohoto parametru.

```
11: struct dvojice uzasna_funkce (float p)
12: {
13:     struct dvojice navrat;
14:     navrat.x = 2*p; // tady nasobim p dvema
15:     navrat.y = 3*p; // tady nasobim p tremi
16:     return navrat;
17: }
```

`cosi.c`

`struct dvojice: 5-6` `struct dvojice uzasna_funkce(): 5`

V ukázkovém zdrojovém kódu je první vložený úsek vymezen na začátku textem `//: dvojice` a na konci textem `//:`. Druhý úsek je vymezen textem `//: uzasna_funkce` a končí na konci souboru.

Na pořadí úseků, které zahrnujeme ze zdrojového textu do dokumentace, nezáleží. Klidně jsme mohli dokumentaci začít od povídání o úžasné funkci (včetně vložení jejího kódu) a potom ještě dopsat, co to je ta struktura `dvojice` a následně vložit deklaraci této struktury.

Kdybychom před řádek `#include <stdio.h>` vložili třeba text `//: start`, bylo by možné příkazem `\ins c_start` vložit do dokumentace začátek souboru `cosi.c`, který v ukázce vložen není.

Všimněme si, že T_EX zapsal čísla řádků přesně podle toho, jak jsou ve zdrojovém kódu. Tj. počítal i přeskakovaný řádek `#include <stdio.h>` i přeskakované prázdné a vymezuující řádky.

Vymezení `//: keyword` se může v řádku nacházet kdekoli, není nutné, aby se vyskytovalo na začátku řádku. Řádek s tímto vymezením není do dokumentace zahrnut a pokud následuje za řádkem s vymezením prázdný řádek, ani ten není do dokumentace zahrnut.

Stejně tak koncové vymezení `//:` se může v řádku nacházet kdekoli a celý řádek s tímto vymezením není do dokumentace zahrnut. Pokud před tímto koncovým řádkem je prázdný řádek, ani ten není do dokumentace zahrnut.

Konečně za povšimnutí stojí použití příkazu `\dg` v dokumentaci. Za ním následuje slovo (separované mezerou), které dokumentujeme. Toto slovo se v dokumentaci výrazně označí (v PDF verzi červenou barvou navíc v barevném rámečku) a jakýkoli jiný výskyt takového slova ve zdrojovém textu nebo mezi uvozovkami `"..."` bude automaticky označen modrou barvou a bude klikací. Kliknutí na modrý výskyt slova kdekoli v dokumentaci vrátí čtenáře na červený výskyt, kde je slovo dokumentováno.

Dokumentované slovo může mít před sebou v hranatých závorkách text, který např. označuje typ funkce a za sebou může mít kulaté závorky `()`. Tím můžeme dát najevo, že dokumentujeme funkci. V místě dokumentace se neobjeví ani tento nepovinný text ani závorky, ale v poznámce pod čarou a v rejstříku se tyto informace vytisknou.

„Palcové uvozovky“ `"..."` vymezují kusy kódu uvnitř odstavce. Text takto uvozený je psán strojopisem a pokud se v něm vyskutují deklarovaná slova, tato slova automaticky modrají a stávají se klikatelnými odkazy. Text mezi těmito uvozovkami je navíc přepisován ve „verbatim“ módu T_EXu, tj. žádné znaky nemají speciální vlastnosti (s výjimkou koncové palcové uvozovky).

Na stránce, kde je slovo dokumentováno (pomocí `\dg`), je v poznámkách pod čarou slovo znovu zmíněno a vedle této zmínky je seznam všech stránek, na kterých se kdekoli v textu vyskytuje použití tohoto slova. Dále jsou všechna dokumentovaná slova zahrnuta do závěrečného abecedního rejstříku, který odkazuje jednak na stránku, kde je slovo dokumentováno, i na stránky se všemi výskyty slova.

2.3 Jaký T_EX pro DocBy.T_EX?

Aby fungovaly všechny výše uvedené vlastnosti, je potřeba použít pdfT_EX rozšířený o encT_EX. Dále je dle `\language` detekován jazyk, který se použije v automaticky generovaných slovech. DocBy.T_EX se ohlásí na terminálu například těmito slovy:

```
This is DocBy.TeX, version May 2014, modes: enc+PDF+ENG
```

DocBy.T_EX rozlišuje tři módy, každý může nabývat dvou stavů: mód `enc/NOenc`, dále mód `PDF/DVI` a konečně mód jazyka `ENG/CS`.

Mód `enc` se zapne, je-li detekována přítomnost encT_EXu. Pokud encT_EX není dostupný, vypíše o tom DocBy.T_EX varování a přejde do `NOenc` módu. V tomto módu nefunguje automatická detekce slov, která jsou dokumentována, takže tato slova nemodrají a nestávají se klikacími odkazy. V rejstříku pak také není seznam stránek se všemi výskyty slova, ale jen místo, kde je slovo dokumentováno. V tomto případě tedy je deaktivována nejdůležitější vlastnost DocBy.T_EXu, takže je žádoucí vynaložit jisté úsilí a encT_EX zprovoznit. V současných distribucích T_EXu bývá encT_EX v pdfT_EXu zahrnut a je aktivován například ve formátu `pdfcsplain`.

Mód `PDF` je detekován, pokud je použit pdfT_EX, jinak DocBy.T_EX přejde do módu `DVI` a napíše o tom varování na terminál. V módu `DVI` nefungují barvy ani klikací odkazy. Ovšem seznam stránek s použitím dokumentovaného slova se generuje, je-li přítomen encT_EX.

DocBy.T_EX detekuje mód jazyka `ENG` (angličtina), je-li `\language=0`. To je implicitní chování. Pokud například v `csplainu` nastavíte `\chyp` před `\input docby.tex`, DocBy.T_EX to vyhodnotí jako dokument v češtině (`CS`). Jiné jazyky nejsou zatím podporovány. V módu `ENG` jsou automaticky generované názvy „Contents“, „Index“ anglické, v módu `CS` jsou tyto názvy „Obsah“, „Rejstřík“ české.

V sekci 3.1 je řečeno, jak jsou tato slova generována a co tedy udělat, když chcete mít dokument v jiném jazyce.

2.4 Vyhledávání slov encT_EXem

Slova, která se stávají klikatelnými odkazy vyhledává encT_EX. Ten má zabudován tzv. „hladový algoritmus“. To znamená, že jsou-li dokumentována např. slova `abc` a `abcde`, pak text `abcdefg` zmordrá až po písmeno `e` a odkazuje na `abcde`, zatímco `abcdx` zmordrá až po písmeno `c` a odkazuje na `abc`. To bývá obvykle žádoucí. V encT_EXu není možno programovat vyhledávání podle regulárních výrazů, takže není možné jednoduše říci, aby encT_EX hledal jen slova, která jsou ohraničena mezerou, tečkou, závorkou, středníkem, atd. Místo toho encT_EX tupě vyhledá slovo třeba uvnitř jiného slova.

Může se tedy stát, že máme dokumentováno kratší slovo, které se objevuje jako část jiných nedokumentovaných slov. Například je dokumentována struktura `turn`, ale ve výpisech programu nechceme, aby v každém výskytu klíčového slova `return` zmordrala jeho část. V takovém případě je potřeba explicitně definovat `return` jako „normální“ nedokumentované slovo. K tomu slouží příkaz `\noactive{<slowo>}`, tedy například `\noactive{return}`. Tento příkaz globálně deklaruje `<slowo>` jako vyhledávané slovo (pro encT_EX), ale specifikuje jej jako neaktivní.

Může se také stát, že máme dokumentováno slovo, které se objevuje ve zdrojových textech i v jiném (nedokumentovaném) významu. Přitom dokumentované slovo poznáme podle toho, jak vypadá text před slovem a za slovem. Pak lze použít deklaraci `\onlyactive{<před>}{<slowo>}{<za>}`, která sama o sobě nedělá nic. Pokud ale vyznačíme `<slowo>` pomocí `\dg` (nebo podobného makra na dokumentování slov, viz sekce 2.9), pak bude `<slowo>` automaticky modrat jen tehdy, předchází-li mu text `<před>` a následuje text `<za>`. Texty `<před>` nebo `<za>` mohou být prázdné (ne oba současně) a k jednomu `<slowo>` můžeme napsat více různých deklarácí `\onlyactive`.

DocBy.T_EX aktivuje encT_EX (pomocí `\mubytein=1`) jen uvnitř skupiny, když zpracovává text mezi palcovými uvozovkami ("`...`") nebo při načítání zdrojového textu programu. Předpokládá se, že nepoužíváte encT_EX k dekódování UTF-8 kódu. Pokud používáte, zkuste si zapnout `\mubytein=1` pro celý dokument, ale na *vlastní riziko*. V takovém případě vám budou modrat slova nebo jejich části i v běžném textu a pokud je dokumentované slovo podmnožinou nějaké T_EXové sekvence, kterou používáte, pak se dočkáte nepříjemných chyb.

2.5 Generování rejstříku, obsahu, poznámek pod čarou a záložek

Generování rejstříku i obsahu probíhá v DocBy.T_EXu zcela automaticky. Pro vytvoření rejstříku není nutné používat externí program (DocBy.T_EX si slova abecedně zatřídí sám). Stačí tedy vložit na požadovaná místa příkazy `\dotoc` a `\doindex`. Upozorňuji, že rejstřík ani obsah nejsou správně vygenerovány po prvním průchodu T_EXu. Je potřeba T_EXovat dvakrát. Po druhém průchodu dojde zřejmě k přestránkování textu (protože je například vložen obsah). Je tedy nutné T_EXovat ještě jednou. Tři průchody T_EXem jsou (snad) dostačující. Slovo „snad“ vychází z problému s poznámkami pod čarou podrobně popsaném v sekci 5.7. Poznámky pod čarou se totiž průběžně v průchodech mění a ovlivňují zpětně vertikální sazbu. DocBy.T_EX proto provádí na konci zpracování v příkaze `\bye` kontrolu, zda nedošlo ke změnám v referencích. Je proto užitečné používat `\bye` místo `\end`. V závěru zpracování pak DocBy.T_EX vypíše zprávu `OK, all references are consistent` nebo vypíše varování, že některé reference jsou nekonzistentní a že je tedy potřeba T_EXovat znovu.

Další test konzistence můžeme provést například následujícím skriptem:

```
#!/bin/bash
cp dokument.ref dokument.r0
pdfcspain dokument.d
diff dokument.r0 dokument.ref
```

DocBy.T_EX se snaží (z důvodu záruky konvergence dokumentu) fixovat zpracování poznámek pod čarou po druhém průchodu. Pokud poté měníte rozsáhlé dokument, takže seznamy stránek vedle poznámek pod čarou jsou výrazně jiné délky, DocBy.T_EX to nepozná a může docházet k přeplnění nebo nenaplnění stránek. V takovém případě je rozumné vymazat soubor `.ref` a znovu spustit tři průchody.

Pro vytvoření záložek se strukturovaným obsahem v PDF výstupu slouží příkaz `\bookmarks`. Je zcela jedno, v které části dokumentu je tento příkaz napsaný, neboť sestaví strukturovaný seznam záložek prolinkovaný s dokumentem na základě údajů ze souboru `.ref`. Může se stát, že některé texty v záložkách nejsou optimálně čitelné. O možnostech, jak toto řešit, pojednává sekce 3.2.

2.6 Vkládání zdrojových textů podrobněji

Kromě jednoduchého příkazu `\ins` na vkládání zdrojových textů jsou k dispozici příkazy `\ifirst` a `\inext`, které nabízejí uživateli daleko více možností.

Příkaz `\ifirst{<soubor>}{<odkud>}{<kam>}{<jak>}` vloží do dokumentu část souboru `<soubor>` (plný název souboru včetně přípony) od prvního řádku, na kterém se vyskytuje text `<odkud>` po řádek, na kterém se vyskytuje text `<kam>`, nebo (pokud text `<kam>` nelze nalézt) po konec souboru. Neexistuje-li ani řádek s textem `<odkud>`, DocBy.T_EX vypíše pouze varování na terminál.

Příkaz `\ifirst` si své parametry nejprve expanduje a pak teprve použije. Aktivní vlnka v parametru expanduje na mezeru.

Parametr `<jak>` udává, zda se bude tisknout výchozí řádek (s textem `<odkud>`) a koncový řádek (s textem `<kam>`). Tento parametr obsahuje právě dva znaky (plus nebo minus) s následujícím významem:

```
jak: -- netiskne se výchozí ani koncový řádek
jak: +- tiskne se výchozí řádek a netiskne se koncový řádek
jak: -+ netiskne se výchozí řádek, tiskne se koncový řádek
jak: ++ tisknou se oba řádky
```

Je-li parametr `<odkud>` prázdný (zapišeme pomocí `{}`), tiskne se od začátku souboru. Je-li parametr `<kam>` prázdný, tiskne se jediný řádek. Je-li parametr `<kam>=\end`, tiskne se až do konce souboru. Koncový řádek v tomto případě neexistuje.

Má-li parametr `<odkud>` (nebo `<kam>`) hodnotu `\empty` (zapišeme pomocí `{\empty}`), tiskne se od (nebo do) prvního prázdného řádku. Parametr `<jak>` ovlivní jeho tisk.

Parametry `<odkud>` nebo `<kam>` mohou mít na svém začátku znak `^~B` (tím dáváme najevo, že text musí na řádku začínat) nebo na svém konci znak `^~E` (tím dáváme najevo, že text musí na řádku končit). Takže třeba `^~Btext^~E` znamená, že se vyhledává řádek, ve kterém je pouze `text` a nic jiného.

V parametrech `<odkud>` a `<kam>` se nesmějí vyskytovat speciální T_EXové znaky (speciální kategorie). Pro použití znaků `\`, `{`, `}`, `%` a `"` v těchto parametrech jsou v DocBy.T_EXu připraveny zástupné kontrolní sekvence `\nb`, `\obrace`, `\cbrace`, `\percent` a `\inchquote`. Sekvence pro další speciální znaky `#`, `$`, atd. si musíte vytvořit např. pomocí:

```
{\catcode'\# =12 \gdef\vezeni{#}}
```

Jsou-li parametry `<odkud>` a `<kam>` stejné, nebo oba texty jsou na stejném řádku, pak se při `<jak>=++` nebo `<jak>=+-` vytiskne tento jeden řádek. Při `<jak>=--` nebo `<jak>=---` se tiskne až do konce souboru nebo do dalšího výskytu textu `<kam>`.

Příkaz `\ifirst` si zapamatuje název čteného souboru a pozici posledního přečteného řádku v daném souboru. Pak je možné použít příkaz `\inext{<odkud>}{<kam>}{<jak>}`, který začíná hledat výchozí řádek s textem `<odkud>` od místa v souboru, kde naposledy skončilo čtení příkazem `\ifirst` nebo `\inext`. Parametry `<odkud>`, `<kam>` a `<jak>` mají stejný význam, jako u příkazu `\ifirst`.

V registru `\lineno` je po ukončení příkazu `\ifirst` nebo `\inext` číslo řádku, které bylo naposledy přečteno (třebaže tento řádek nebyl vytištěn). Pokud bylo dosaženo konce souboru, obsahuje `\lineno` počet řádků souboru. Pomocí `\ifeof\infile` je možné se zeptat, zda bylo dosaženo konce souboru.

Příklady

```
\ifirst {soubor.txt}{textik}{textik}{++} % vytiskne první výskyt řádku
                                         % obsahující slovo textik
\inext {textik}{textik}{++}             % vytiskne následující výskyt
                                         % řádku obsahující slovo textik
\ifirst {soubor.c}{//: odkud}{//:}{--} % analogie příkazu \ins
\ifirst {soubor.c}{funkce(){}}{++}     % tisk prototypu funkce
\ifirst {soubor.c}{funkce(){^~B\cbrace}}{++} % tisk celého kódu funkce
\ifirst {soubor.txt}{}{\end}{++}      % tisk celého souboru
\ifirst {soubor.txt}{}{\empty}{+-}    % tisk po prázdný řádek
```

Je-li první řádek, který se má tisknout, prázdný, netiskne se. Je-li poslední řádek, který se má tisknout, prázdný, také se netiskne. Toto je implicitní chování. Pokud napíšete `\skippingfalse`, uvedená inteligence je zrušena a přepisují se i prázdné řádky vpředu a vzadu. Příkazem `\skippingtrue` se vrátíte k původnímu nastavení.

Parametrům $\langle odkud \rangle$ a $\langle kam \rangle$ může předcházet text $\backslash\text{count}=\langle \text{číslo} \rangle$. Hodnota $\langle \text{číslo} \rangle$ označuje, kolikátý výskyt textu $\langle odkud \rangle$ nebo $\langle kam \rangle$ se má použít. Například $\{\backslash\text{count}=3\langle odkud \rangle\}$ znamená, že se má při vyhledávání $\langle odkud \rangle$ přeskočit dva jeho výskyty a začít prepisovat soubor až od výskytu třetího. Podobně $\{\backslash\text{count}=5\langle kam \rangle\}$ značí, že se při prepisování souboru ignorují čtyři výskyty $\langle kam \rangle$ a prepisování se zastaví až u výskytu pátého.

Implicitně, není-li $\backslash\text{count}=\langle \text{číslo} \rangle$ uvedeno, předpokládá se $\backslash\text{count}=1$.

Pokud je text $\langle odkud \rangle$ prázdný, pak $\backslash\text{count}$ označuje číslo řádku, na kterém se má zahájit výpis. Je-li prázdný parametr $\langle kam \rangle$, pak $\backslash\text{count}$ označuje počet prepisovaných řádků. Toto platí pro $\langle jak \rangle=++$ a pro $\backslash\text{skippingfalse}$. Při jiných hodnotách $\langle jak \rangle$ se uvedená čísla logicky posunou o jedničku. Při prázdném $\langle odkud \rangle$ nebo $\langle kam \rangle$ není mezera za $\backslash\text{count}=\langle \text{číslo} \rangle$ povinná. Příklady:

```
\skippingfalse
\ifirst {soubor.txt}{\count=20}{\count=10}{++} % tisk řádků 20 až 29
\ifirst {soubor.txt}{\count=2 \empty}{+-} % tisk po druhý prázdný řádek
\ifirst {soubor.txt}{\count=50}{\end}{++} % tisk od 50. řádku do konce
\ifirst {soubor.tex}{\count=5 \nb section}{\count=2 \nb section}{+-}
% tisk páté sekce z TeXového souboru
```

2.7 Odkazy na čísla řádků

Pomocí $\backslash\text{cite}[\langle \text{lejblík} \rangle]$ je možné odkazovat na číslo řádku ve výpisu zdrojového kódu. Tento příkaz se promění na skutečné číslo řádku. Před použitím příkazu $\backslash\text{ifirst}$ nebo $\backslash\text{inext}$ je nutné $\langle \text{lejblík} \rangle$ deklarovat příkazem $\backslash\text{ilabel}[\langle \text{lejblík} \rangle]\langle \text{text} \rangle$. Těchto příkazů může být před použitím $\backslash\text{ifirst}$ resp. $\backslash\text{inext}$ více. Na pořadí příkazů $\backslash\text{ilabel}$ před jedním $\backslash\text{ifirst}$ nebo $\backslash\text{inext}$ nezáleží.

Existují-li deklarované $\langle \text{lejblík} \rangle$ y a $\langle \text{text} \rangle$ y, pak příkaz $\backslash\text{ifirst}$ nebo $\backslash\text{inext}$ si všimá výskytu $\langle \text{text} \rangle$ u ve vkládaných řádcích. Pokud takový $\langle \text{text} \rangle$ najde, přiřadí číslo řádku odpovídajícímu $\langle \text{lejblík} \rangle$ u, takže příkaz $\backslash\text{cite}$ bude fungovat, jak má.

Parametr $\langle \text{lejblík} \rangle$ musí být jednoznačný v celém dokumentu. Příkaz $\backslash\text{cite}$ funguje dopředně i zpětně.

Příkazy $\backslash\text{ilabel}$ mají lokální působnost a spolupracují jen s nejbližším následujícím $\backslash\text{ifirst}$ a $\backslash\text{inext}$. Takže před použitím dalšího $\backslash\text{ifirst}$ resp. $\backslash\text{inext}$ je potřeba deklarovat další vyhledávané texty pomocí $\backslash\text{ilabel}$ znovu.

DocBy. \TeX nevypíše žádné varování, pokud nějaký $\langle \text{text} \rangle$ deklarováný v $\backslash\text{ilabel}$ nenajde. Ovšem při použití $\backslash\text{cite}$ se objeví varování, že není známý $\langle \text{lejblík} \rangle$ a toto varování nezmezí ani při opakovaném \TeX ování.

Pokud se $\langle \text{text} \rangle$ vyskytuje ve více řádcích ukázky, je odkazován řádek s prvním výskytem.

V následující ukázce je čten již známý soubor `cosi.c` (viz kapitolu 2.2).

Na řádku $\backslash\text{cite}[\text{ufunkce}]$ je deklarovaná úžasná funkce.

```
\ilabel [ufunkce] {funkce (float)}
\ilabel [navratx] {navrat.x}
\ifirst {cosi.c}{\}{++}
```

Zvláště upozorňuji na geniální myšlenku na řádku $\backslash\text{cite}[\text{navratx}]$, kde je vstupní parametr vynásoben dvěma.

2.8 Verbatim ukázky pomocí $\backslash\text{begtt}/\backslash\text{endtt}$ a palcových uvozovek

Verbatim ukázky můžete do dokumentace vkládat pomocí $\backslash\text{begtt}$ a $\backslash\text{endtt}$. Ty jsou (na rozdíl od vkládaných souborů) napsány přímo ve zdrojovém textu \TeX u. Všechny řádky za $\backslash\text{begtt}$ jsou vloženy beze změn až po ukončovací $\backslash\text{endtt}$. Řádky nejsou číslovány a texty v nich nemodrají a nestávají se klikatelnými odkazy.

Následující sekce 3.2 a 4.8 obsahují informace, jak je možné toto implicitní chování změnit.

Verbatim ukázky uvnitř odstavce lze vymezit palcovými uvozovkami "...". V tomto prostředí probíhá tisk strojopisem a je aktivní $\text{enc}\TeX$, takže dokumentovaná slova se stávají automaticky odkazy na místo, kde je $\backslash\text{dg}$. Doporučuje se toto prostředí používat na výpisy veškerých částí kódů dokumentovaného programu, které jsou vloženy uvnitř textu v odstavci (analogie matematického prostředí $\$. . . \$$).

2.9 Deklarace dokumentovaného slova

Na deklaraci slova, které dokumentujeme, lze použít příkaz `\dg`, `\dgn`, `\dgh`, `\dl`, `\dln` nebo `\dlh`. Významy jednotlivých příkazů vysvětlíme později. Nejprve se věnujeme syntaxi parametrů. Všechny příkazy mají stejnou syntaxi, takže nebude vadit, když bude vyložena jen v souvislosti s příkazem `\dg`. Syntaxe je poněkud zvláštní. Účelem totiž bylo minimalizovat práci písaře, takže jsem se vyhnul kučeravým závorkám, parametr separuji podle mezery nebo něčeho jiného, atd.

Existují tyto možnosti syntaxe parametrů:

```
\dg <slovo>           % <slovo> separované mezerou
\dg [text] <slovo>  % navíc nepovinný "přední" <text>
\dg [text]<slovo>   % <slovo> může na [text] navazovat bez mezery
\dg <slovo>()        % <slovo> s dvojicí "()" separované mezerou
\dg [text]<slovo>() % kombinace předchozího
\dg <slovo>,         % <slovo> separované čárkou
\dg [text] <slovo>, % kombinace předchozího
\dg <slovo>(),       % <slovo> s dvojicí "()" separované čárkou
\dg [text]<slovo>(), % kombinace předchozího
\dg <slovo>.         % slovo separované tečkou
atd...
```

Obecně: za příkazem `\dg` může následovat nepovinná `[`. Pokud následuje, pak se přečte `<text>` až po ukončovací `]`. Parametr `<text>` může obsahovat mezery. Za ukončovací `]` může a nemusí být mezera. Pokud tam je, pak ji makro přesune před koncovou závorku `]`, takže `\dg [aha_]slovo` je totéž jako `\dg [aha_]slovo`. Dále následuje čtení parametru `<slovo>`. Tento parametr nesmí obsahovat mezery, čárku, tečku, středník a dvojtečku. Čtení parametru je ukončeno, jakmile se objeví mezera nebo čárka nebo tečka nebo středník nebo dvojtečka. Uvedená interpunkce není součástí parametru `<slovo>` a po zpracování parametru se vrátí do vstupní fronty, takže se běžně vytiskne. Nakonec se zjistí, zda přečtený parametr až po separátor není ve tvaru `<slovo>()`. Pokud je, pak symbol `()` se nepovažuje za součást parametru `<slovo>`, ale mluvíme o `<slovo>` následovaném dvojicí `()`.

Pozor, za separátorem typu čárka, tečka, středník a dvojtečka se musí vyskytnout mezera. Ne nutně ihned, ale dříve, než se objeví úsek textu, který má být přečten s jinými kategoriemi (např. "..."). Není tedy možné psát `\dg text,"...`. Pokud za separátorem mezera následuje znak `'` (obrácený apostrof), mezera ani tento znak se netiskne. To je možné využít například pro vložení nezlomitelné mezery nebo pro jiné účely: `\dg <slovo>_ '~<přilepený-text>` nebo `\dg <slovo>_'"..."`.

Příkazy `\dgh`, `\dgn`, `\dln`, `\dlh` separující mezeru netisknou nikdy, protože tyto příkazy většinou netisknou nic (viz níže).

Parametr `<slovo>` je dokumentované slovo. Pokud se takové `<slovo>` vyskytne někde jinde v dokumentu mezi "... " nebo ve vloženém zdrojovém kódu, automaticky zmodrá a stává se klikatelným odkazem na místo, kde je použito `\dg`. V místě použití `\dg` je slovo zvýrazněno červenou barvou. Je vytištěno samotné bez parametru `<text>` a bez případných závorek `()`. V poznámce pod čarou se vypíše `<slovo>` (červeně). Tam je i případný `<text>` (před slovem) a za ním je případná dvojice `()`. Vedle tohoto výpisu je seznam stránek s výskyty `<slova>`. V rejstříku se objeví něco podobného, jako v poznámce pod čarou. Rejstřík je řazen abecedně podle `<slova>`, nikoli podle `<text>`.

Příkaz `\dg` deklaruje `<slovo>` globálně. Bude na něj odkazováno v celém dokumentu.

Příkaz `\dgh` pracuje jako `\dg`, ale slovo nebude v místě `\dgh` vypsáno (`\dg hidden`). Bude tam jen cíl odkazů a `<slovo>` se objeví v poznámce a v rejstříku.

Příkaz `\dgn` způsobí, že první následující výskyt `<slova>` ve vypisovaném zdrojovém kódu se stane cílem všech ostatních odkazů, zčervená (tedy nezmodrá) a v místě tohoto výskytu se objeví příslušná poznámka pod čarou. Příkaz `\dgn` čteme jako `\dg next`, nebo `\dg následující`.

Příkaz `\dl` deklaruje `<slovo>` lokálně. Bude na něj odkazováno svým krátkým jménem `<slovo>` jen v místě stejného jmenného prostoru, typicky při dokumentaci jednoho modulu. Každý modul zahájený příkazem `\module` zavádí jmenný prostor tvaru `<slovo>./<název>`, kde `<název>` je jméno modulu. Slovo deklarované pomocí `\dl` žije ve dvou variantách. V krátké variantě jako `<slovo>` jen v rozsahu jednoho jmenného prostoru a v dlouhé variantě `<slovo>./<název>` žije globálně v celém dokumentu. Případný výskyt dlouhého názvu odkáže na místo deklarace napříč celým dokumentem.

Podrobněji o jmenných prostorech a možnosti jejich změny najdete v sekci 2.10.

Každé $\langle slovo \rangle$ musí být v dokumentu deklarováno nejvýše jednou, jinak DocBy.T_EX ohlásí chybu. V případě $\backslash dl$ musí existovat jednoznačný dlouhý název.

Příkaz $\backslash dlh$ je skrytý $\backslash dl$. Příkaz $\backslash dln$ znamená $\backslash dl$ next. Analogicky, jako příkazy $\backslash dgh$ a $\backslash dgn$.

Pokud někoho irituje vysoká inteligence těchto příkazů při čtení parametrů, může použít interní verzi příkazů s povinnými třemi parametry obalenými do kučeravých závorek: $\backslash iidg$, $\backslash iidgh$, $\backslash iidgn$, $\backslash iidl$, $\backslash iidlh$, $\backslash iidln$. Parametry vypadají takto: $\backslash iidg\{\langle před \rangle\}\{\langle slovo \rangle\}\{\langle za \rangle\}$. Pravda, tyto příkazy umožňují více než jejich krátké verze: umožňují do parametru $\langle slovo \rangle$ propašovat čárku, mezeru, středník atd. a do parametru $\langle za \rangle$ napsat cokoli, nejen kulaté závorky.

2.10 Jmenné prostory

Jmenný prostor je pravidlo, podle kterého se krátký název dokumentovaného $\langle slova \rangle$ transformuje při použití $\backslash dl$ na název dlouhý. Je možné jej nastavit nebo změnit pomocí příkazu $\backslash namespace$, který se použije takto: $\backslash namespace \{ \langle text1 \rangle \#1 \langle text2 \rangle \} \dots \backslash endnamespace$. Pokud je uvnitř tohoto prostředí použit příkaz $\backslash dl \langle slovo \rangle$, je slovu přidělen krátký název $\langle slovo \rangle$ a dlouhý název $\langle text1 \rangle \langle slovo \rangle \langle text2 \rangle$. Uvnitř takto deklarováného prostředí se všechny výskyty krátkého názvu $\langle slovo \rangle$ transformují na dlouhý název a jsou prolinkovány s odpovídajícím místem $\backslash dl$. Jmenný prostor je lokální uvnitř svého prostředí, takže vně prostředí se $\langle slovo \rangle$ chová, jakoby nebyl žádný příkaz $\backslash dl$ použit. Například uvnitř prostředí $\backslash namespace \{ \#1 //uff \} \dots \backslash endnamespace$ je ke každému slovu deklarovávanému pomocí $\backslash dl \langle slovo \rangle$ přidělen dlouhý název $\langle slovo \rangle //uff$ a výskyty $\langle slovo \rangle$ odkazují na místo $\backslash dl \langle slovo \rangle$.

Vně všech prostředí $\backslash namespace \dots \backslash endnamespace$ není jmenný prostor definován, takže tam není možné použít příkaz $\backslash dl$. Ovšem příkaz $\backslash module \langle název \rangle \llcorner$ nastaví jmenný prostor na $\{ \#1 . / \langle název \rangle \}$, takže uvnitř dokumentace modulu je možné používat příkaz $\backslash dl$.

V rejstříku a v poznámce pod čarou se tisknou dlouhé názvy. Rejstřík abecedně řadí podle dlouhých názvů. V obsahu se tisknou názvy krátké.

Příklad práce se jmennými prostory:

```
\namespace {ju:#1} %% nastavuji namespace ju
Tady deklaruji slovo \dl aha.
Tady slovo "aha" automaticky odkazuje na místo deklarace.
Slovo "ju::aha" také odkazuje na místo deklarace.
\endnamespace
\namespace {hele:#1} %% nastavuji namespace hele
Tady znovu deklaruji slovo \dl aha.
Zde slovo "aha" odkazuje na lokální deklaraci uvnitř "hele"
\endnamespace %% ruším namespace
Zde slovo "aha" neodkazuje nikam, ale slova "ju::aha"
a "hele::aha" stále odkazují na místa, kde byla deklarována.
```

Prostředí $\backslash namespace \dots \backslash endnamespace$ je možné vnořovat, ovšem vnořená prostředí musejí mít jiný jmenný prostor než prostředí vnější. Prostředí jmenných prostorů pracují globálně nezávisle na $\backslash bgroup$, $\backslash egroup$. Příkaz $\backslash endnamespace$ použitý vně všech prostředí $\backslash namespace \dots \backslash endnamespace$ neudělá nic. Prostředí není nutné před příkazem $\backslash bye$ ukončovat.

2.11 Místo pro dokumentaci aplikačního rozhraní

Může se stát, že píšeme dokumentaci jednak pro uživatele, které zajímá způsob použití dokumentovaných funkcí a co zhruba dělají (tzv. API), ale nezajímá je, jak je funkce naprogramovaná. Druhá chceme mít dokumentován i způsob, jak funkce funguje uvnitř. V takovém případě musí dokumentované $\langle slovo \rangle$ odkazovat na dvě místa v dokumentu.

Místo, kde je podrobně $\langle slovo \rangle$ popsáno, je vymezeno příkazem $\backslash dg$ nebo podobným. Místo, kde slovo dokumentujeme pro uživatele (je-li toto místo odlišné od prvního místa), lze vyznačit příkazem $\backslash api \{ \langle slovo \rangle \}$. V místě použití $\backslash api \{ \langle slovo \rangle \}$ se nestane nic, jen se tam umístí neviditelný cíl odkazů. V obsahu se pak $\langle slovo \rangle$ objeví s odkazem na toto místo. V rejstříku se v seznamu stránek objeví jedna stránka podtržená: to je stránka, kde byl použit příkaz $\backslash api \{ \langle slovo \rangle \}$. Ovšem, aby se v rejstříku $\langle slovo \rangle$ vůbec objevilo, musí se někde v dokumentu vyskytovat i jeho plná deklarace pomocí $\backslash dg$ nebo podobných příkazů. Na stránce, kde je použito $\backslash dg$, je pod čarou vedle slova seznam stránek a rovněž je tam jedna stránka podtržená. Když čtete implementační popis pro $\langle slovo \rangle$, snadno se tedy dostanete na stránku, kde je API k tomuto $\langle slovu \rangle$. V rejstříku a obsahu jsou také slova, která byla deklarována pomocí $\backslash api$,

zleva vyznačena textem `\apitext`. Ten je implicitně nastaven na šipku. Můžete se podívat do rejstříku a do obsahu tohoto dokumentu. V tomto místě bylo použito `\api{\nb\api}`, zatímco skutečná definice příkazu `\api` je v sekci 5.9.

Je-li použito `\api{<slovo>}`, pak je možné se na místo odkazovat také pomocí `\cite[+(<slovo>)]`. Tato konstrukce se promění v číslo stránky, kde je dokumentováno API daného slova. Například v tomto dokumentu se `\cite[+\nb\api]` promění na: 11.

Pokud toto slovo má také svůj API cíl (vytvořený pomocí `\api`), pak se červený text (tištěný v místě `\dg`) stává aktivním odkazem na API cíl. Tam typicky čtenář najde výskyt slova, který je zase klikatelným odkazem na `\dg` cíl. Takže tyto dva cíle jsou prolinkovány křížem.

2.12 Sekce, sekcičky, část, titul

Sekce se uvozuji příkazem `\sec <název-sekce>\par`. Každá sekce může mít několik podsekcí (sekciček), které lze vyznačit příkazem `\subsec <název-podsekce>\par`. Symbol `\par` zde znamená, že název sekce či podsekce je oddělen od dalšího textu prázdným řádkem (viz ukázkou v 2.1).

Několik sekcí může tvořit část. Část je uvozena příkazem `\part <název-části>\par`. Části jsou automaticky označeny písmeny A,B,C,... a jsou vyznačeny výrazněji než sekce v místě začátku části i v obsahu. Části ale nenarušují číslování sekcí. Tj. sekce jsou číslovány od jedné napříč celým dokumentem bez ohledu na to, zda jsou nebo nejsou rozděleny na části.

Příkaz `\module<soubor>` automaticky založí sekci s názvem `Modul<soubor>` a deklaruje svůj jmenný prostor. Toto chování lze změnit, viz 3.1, 3.3.

Příkaz `\title<název>\par` vytiskne název dokumentu větším písmem a v rámečku. Je-li definováno makro `\projectversion`, bude jeho obsah vytištěn drobně vpravo nahoře doplněný zepředu textem `verze`. Pokud váš projekt nemá verzi, může se hodit třeba:

```
\def\projectversion{\the\day. \the\month. \the\year}
```

Příkaz `\author<text>\par` napíše do středu řádku tučně `<text>`, což bývá obvykle jméno autora (jména autorů).

Do záhlaví každé stránky se začne přepisovat zleva název aktuální sekce a zprava název dokumentu. Uživatel může text pro pravé záhlaví změnit změnou makra `\headtitle`.

Příkazy `\sec` a `\subsec` mohou mít v hranaté závorce nepovinný parametr `<lejblík>`. V takovém případě vypadají parametry takto: `\sec [<lejblík>](<název-sekce>\par`. Po takovém použití je možné se na sekci (podsekci) odkazovat příkazem `\cite[<lejblík>]`. Tento příkaz se promění v číslo odkazované sekce (podsekce) a navíc se stane aktivním odkazem.

Pomocí příkazu `\savetocfalse` lze před použitím příkazu `\sec` nebo `\subsec` zajistit, že název sekce se nedostane do obsahu a nebude mít své číslo. Místo čísla se vytiskne obsah makra `\emptynumber`, které je implicitně prázdné. Příkaz `\savetocfalse` ovlivní jen první následující `\sec` nebo `\subsec`.

2.13 Křížové odkazy

Cíl, kam směřuje odkaz, je potřeba vyznačit pomocí `<lejblíku>`. To je možné udělat v příkaze `\sec`, `\subsec` (viz předchozí sekci 2.12) nebo kdekoli v textu samostatným příkazem `\label[<lejblík>]`. Také je možné odkazovat na číslo řádku (viz sekci 2.7). Všechny lejblíky musejí být jednoznačné (bez ohledu na jejich typ) napříč celým dokumentem.

Příkaz `\pgref[<lejblík>]` expanduje na číslo strany, na které se vyskytuje cíl odkazu. Příkaz `\numref[<lejblík>]` expanduje v závislosti na typu cíle na:

- číslo sekce, je-li cílem sekce,
- dvojčíslí `<sekce>.<podsekce>`, je-li cílem podsekce,
- číslo řádku, je-li cílem řádek zdrojového kódu,
- prázdné makro, je-li `<lejblík>` deklarovaný pomocí `\label`.

Oba příkazy `\pgref` a `\numref` expandují na uvedené texty bez další inteligence. Tj. výstupní text se nestává klikatelným odkazem.

K aktivaci odkazu v PDF módu slouží makro `\ilink [<lejblík>]{<text>}`. Toto makro vytiskne modře `<text>`, který se stává klikatelným odkazem na cíl, deklarovaný pomocí `<lejblíku>`. Takže již známý příkaz `\cite[<lejblík>]` udělá zhruba to samé, jako `\ilink [<lejblík>]{\numref[<lejblík>]}`. Skutečný příkaz `\cite` navíc ověří, zda není `\numref[<lejblík>]` prázdné makro. Pokud je, obarví namísto výstupu `\numref` výstup makra `\pgref`.

Pokud `<lejblík>` jako argument příkazu `\pgref`, `\numref` nemá svůj cíl, příkaz `\pgref` expanduje na hodnotu `-1000` a `\numref` expanduje na prázdný výstup. Jsou to expanzní makra, takže v nich není implementován například tisk varování. Podívejte se na definici příkazu `\cite` (na straně 36), jak se dá tisk varování implementovat.

Makro `\module <jméno>` založí sekci s lejblíkem `m:<jméno>`, takže lze na ní pak odkazovat. Například si můžete vytvořit makro

```
\def\refmodul[#1]{\ilink[m:#1]{\tt#1}}
```

kteří aktivizuje svůj parametr, pokud tento je názvem nějakého modulu. Třeba `\refmodul[base]` vytiskne slovo `base` strojopisem a modře a stává se klikatelným odkazem na začátek sekce „Modul base“, pokud je tato sekce založena příkazem `\module`.

Makra `\dg`, `\dgn`, `\dgh` interně provedou příkaz `\label[@< slovo>]` a makra `\dl`, `\dln`, `\dlh` provedou příkaz `\label[@<dlohé-slovo>]`, kde `<dlohé-slovo>` je `<slovo>` po transformaci podle aktuálního jmenného prostoru. Na místa, kde jsou slova dokumentovaná, je tedy možné odkazovat například pomocí `\link[@<slovo>]{<slovo>_dokumentované_na_straně~\pgref[@<slovo>]}`.

Makro `\api{<slovo>}` interně provede `\label[+<slovo>]`, takže je možné na toto místo odkazovat třeba pomocí `\ilink[+<slovo>]{API:~<slovo>}`.

DocBy.T_EX nenabízí kromě čísel sekcí, podsekcí a čísel řádků žádné další automatické číslování. Pokud tedy chcete implementovat např. číslování obrázků, čísla publikací atd., musíte si napsat makra vlastní. K tomu můžete využít makro `\labeltext[<lejblík>]{<text>}`, které uloží v horizontálním módu do sazby neviditelný cíl odkazu, a při dalším průchodu T_EXem expanduje makro `\numref` na `<text>`. Použití makra ukážeme na příkladě, ve kterém definujeme makro `\bib[<lejblík>]`. Toto makro zahájí sazbu další položky v seznamu literatury. Odkazovat na knihu pak lze pomocí `\cite[b:<lejblík>]`.

```
\newcount\bibnum
\def\bib[#1]{\par\advance\bibnum by1 \indent
  \llap{[\the\bibnum] } \labeltext[b:#1]{[\the\bibnum] \ignorespaces}}
```

2.14 Vkládání obrázků

Příkazem `\ifig <šířka>_<jméno-obrázku>` je možné vložit obrázek. Obrázek musí být připraven v souboru `fig/<jméno-obrázku>.eps` (v případě DVI módu) a v souboru `fig/<jméno-obrázku>.pdf` (v případě PDF módu). Adresář, kde DocBy.T_EX vyhledává obrázky (`fig/`), lze změnit předefinováním sekvence `\figdir`. Rozměr `<šířka>` je bez jednotky a udává poměr požadované šířky obrázku ku šířce sazby. Obrázek je umístěn zarovnan doleva na odstavcovou zarážku.

Máte-li připraven obrázek ve formátu `eps`, pak jej do `pdf` převedete příkazem

```
ps2pdf -dEPSCrop <jméno obrázku>.eps
```

2.15 Výčty

Seznam položek obklopite `\begitems` a `\enditems`. V tomto prostředí je text odsazen zleva o odstavcovou zarážku. Prostor lze vnořovat. Jednotlivou položku zahájíte pomocí `\item <značka>_<text>`, přitom `<značka>` se vystrčí vlevo od `<textu>`. Je-li `<značka>` hvězdička, promění se v puntík. Další možnost: `\item \the\itemno_<text>`, což vytvoří číslované výčty, v každém prostředí číslovány od jedné.

Makro plainu `\item` není předefinováno globálně, ale jen uvnitř `\begitems... \enditems`. Můžete tedy použít i makro plainu, pokud se vám koncept položek nabízený DocBy.T_EXem nelíbí.

3 Pro náročné

V této sekci jsou uvedeny a vysvětleny definice základních příkazů DocBy.T_EXu. Uživatel si může tyto definice změnit, pokud chce změnit chování DocBy.T_EXu. Pokud například pracuje s jiným programovacím jazykem, může si změnit makro `\docsuffix` nebo kompletně předefinovat makra `\module` a `\ins`.

3.1 Interní názvy

Příkazem `\doindex` vytvoří DocBy.T_EX automaticky novou sekci s názvem „Rejstřík“. Podobně při tvorbě obsahu nebo natažení modulu vzniká název „Obsah“ nebo „Modul“. Před názvem verze

v titulu při použití `\projectversion` se objeví slůvko „verze“. Část (vytvořená pomocí `\part`) má v záložkách uvozující text `>>CAST`. Tyto texty jsou definovány v makrech `\titindex`, `\tittoc`, `\titmodule`, `\titversion` a `\opartname`.

docby.tex

```

25: \def\titmodule{Module}
26: \def\tittoc{Table of Contents}
27: \def\titindex{Index}
28: \def\titversion{version }
29: \def\opartname{>> PART}
30: \ifx\chyp\undefined
31: \else \ifnum\language=0
32: \else
33: \def\titmodule{Modul}
34: \def\tittoc{Obsah}
35: \def\titindex{Rejstřík}
36: \def\titversion{verze }
37: \def\opartname{>> CAST}
38: \fi \fi

```

Za povšimnutí stojí, že jsou jinak tato makra definována při použití klasického plainu a jinak při použití csplainu. To ovšem neznamená, že uživatel si tyto názvy nemůže předefinovat ještě jinak, nezávisle na použitém formátu.

3.2 Vložené skupiny příkazů (hooks)

Některá složitější makra (`\begtt`, palcové uvozovky, `\ifirst`, `\inext`, `\doindex`, `\dotoc`) dovolují vkládat uživateli na začátku zpracování různé příkazy (tzv. hooks). Implicitně jsou tyto vložky prázdné:

docby.tex

```

42: \def\begthook{}
43: \def\quotehook{}
44: \def\indexhook{}
45: \def\tochhook{}
46: \def\bookmarkshook{}
47: \def\outpuhook{}

```

Makro `\begthook` je vloženo po založení skupiny a nastavení všech kategorií těsně před začátkem zpracování prostředí `\begtt... \endtt`. Makro `\quotehook` je vloženo po založení skupiny a nastavení všech kategorií těsně před začátkem zpracování prostředí "...". Makro `\indexhook` je vloženo makrem `\doindex` po založení sekce a před přechodem do sazby ve dvou sloupcích. V tomto dokumentu je v něm úvodní povídání k rejstříku. Makro `\tochhook` je vloženo makrem `\dotoc` po založení sekce před sazbou prvního řádku obsahu. Makro `\bookmarkshook` je vloženo uvnitř skupiny na začátku zpracování záložek. Je možné v něm nastavit expanze maker vyskytujících se v nadpisech na rozumnou hodnotu pro záložky. Pokud navíc nastavíte `\let\cnvbookmark=\lowercase`, budou všechny znaky pro záložky procházet filtrem `\lowercase`. Uvnitř `\bookmarkshook` je pak možné nastavit `\lccode` vybraným znakům (například pro odstranění háčků a čárek). Makro `\outpuhook` je vloženo na začátek výstupní rutiny. Je vhodné v něm nastavit vybrané příkazy na hodnotu `\relax`, aby se neexpandovaly do souboru `.ref`.

Příklady použití

```

\def\quotehook{\obeyspaces} % ve výpisech "..." budou normální mezery
\def\quotehook{\langleactive} % <text> se promění na <text>
\def\begthook{\mubytein=1} % mezi \begtt... \endtt bude aktivní encTeX
\def\begthook{\setsmallprinting} % ukázky \begtt... \endtt budou malé
\def\begthook{\catcode'\!=0} % mezi \begtt... \endtt fungují !prikazy
\def\indexhook{To čubrníte, jaký tu mám rejstřík.}
\def\outpuhook{\let\mylogo=\relax} % \mylogo nebude expandovat

```

```

\titindex: 14, 39   \tittoc: 14, 39   \titmodule: 14–15   \titversion: 14, 18–19
\opartname: 14, 40 \begthook: 14, 28   \quotehook: 14, 44–45   \indexhook: 14, 39
\tochhook: 14, 39  \bookmarkshook: 14, 40   \outpuhook: 14, 34

```


3.3 Příkaz `\module` a `\ins`

Uživatelská dokumentace k těmto příkazům je v sekci 2.1. Příkaz `\module` $\langle soubor \rangle_{\perp}$ načte soubor s názvem $\langle soubor \rangle \backslash \text{docsuffix}$, kde makro `\docsuffix` obsahuje příponu souboru včetně tečky.

```
51: \def\docsuffix {.d} % implicit filename extension (for \module command)
52: \def\module #1 {\endnamespace\namespace{##1./#1}\sec [m:#1] \titmodule\space #1 \par
53:   \def\modulename{#1} \input #1\docsuffix\relax
54: }
```

docby.tex

Příkaz `\module` vloží název čteného souboru (bez přípony) do pomocného makra `\modulename`. Toto makro pak využívá příkaz `\ins` $\langle přípona \rangle_{\perp} \langle text \rangle_{\perp}$.

```
55: \def\ins #1 #2 {\ifirst {\modulename.#1}{//: #2}{//:}{--}}
```

docby.tex

3.4 Zelenající komentáře

Příkazy `\ifirst` a `\inext` si také všímají (implicitně) C komentářů tvaru `//... $\langle eol \rangle$` a `/*... $\langle * \rangle$ */`. Tyto komentáře barví ve výpisu programu zeleně. Zrušit tuto vlastnost lze příkazem `\noactive $\langle string \rangle$` . Pomocí `\setlinecomment` $\{ \langle string \rangle \}$ lze nastavit nový typ komentářů, které budou barveny zeleně od $\langle string \rangle$ do konce řádku. Příkazy mají globální platnost. Například

```
\noactive{/*}\noactive{*/}\noactive{//}
\setlinecomment{\percent} \noactive{\nb\percent}}
```

nastaví komentáře podle zvyklostí v T_EXu a PostScriptu.

Příkazem `\setlrcoment` $\{ \langle levý \rangle \} \{ \langle pravý \rangle \}$ lze nastavit komentáře typu `/*... $\langle * \rangle$ */`.

Pro změnu vlastností obarvování komentářů stačí uvedená makra použít. Kdo chce vědět, jak jsou implementovaná, nechte čte dále.

```
59: \ifx\mubyte\undefined
60:   \def\setlinecomment#1{}
61:   \def\setlrcoment#1#2{}
62: \else
63:   \def\setlinecomment#1{\mubyte \linecomment ##0 #1\endmubyte}
64:   \def\setlrcoment#1#2{\mubyte \leftcomment ##0 #1\endmubyte
65:     \mubyte \rightcomment #2\endmubyte \gdef\rightcomment{#2\returntoBlack}}
66: \fi
```

docby.tex

Uvedené příkazy jsou prázdné v módu bez encT_EXu a při detekci encT_EXu zapíší informace do encT_EXové tabulky prostřednictvím primitiv `\mubyte... \endmubyte` .

Příkazy `\linecomment` a `\leftcomment` se díky encT_EXu automaticky vloží před detekovanou sekvenci znaků. Tyto příkazy nastaví barvu textu na zelenou:

```
68: \def\linecomment {\let\Black=\Green \Green}
69: \def\leftcomment {\global\let\Black=\Green \Green}
```

docby.tex

Na druhé straně příkaz `\rightcomment` potřebuje vypnout zelenou barvu až po přeskočení detekované sekvence. Proto encT_EX v tomto případě detekovanou sekvenci zruší a příkaz `\rightcomment` má za úkol ji vrátit do sazby zpět a teprve poté pomocí `\returntoBlack` se vrátit k černé barvě.

```
71: \def\returntoBlack {\global\let\Black=\oriBlack \Black}
```

docby.tex

Je potřeba vysvětlit, proč přepínače barev jsou tak komplikovaně zapsány. Přepínač totiž v PDF zapíná barvu nezávisle na skupině a barva textu se drží tak dlouho, dokud není použit jiný přepínač barvy. Každý tisk řádku kódu je uveden přepínačem `\Black`, takže při poznámce „do konce řádku“ stačí jen přepnout na `\Green`. Ovšem uvnitř komentáře se může objevit link obalený příkazy `\Blue... \Black` (viz např. řádek 51 v předchozí sekci). Pak ale chceme, aby `\Black` vrátil barvu `\Green`. Proto je provedeno predefinování pomocí `\let`. Toto predefinování je lokální. Protože řádek je tištěn uvnitř skupiny, je další řádek už černý.

```
\module: 4, 5, 10–13, 15, 24   \docsuffix: 13, 15   \modulename: 15   \ins: 5, 6, 8, 13, 15
\setlinecomment: 15–16   \setlrcoment: 15–16   \linecomment: 15, 26, 44   \leftcomment: 15, 26,
44   \rightcomment: 15   \returntoBlack: 15, 26, 44
```

Při tisku komentáře, který má úvodní a koncový znak a může přesáhnout jeden řádek, musíme globálně předefinovat `\Black` na `\Green`, aby i další řádky (uvozené příkazem `\Black`) byly zelené. Koncový znak komentáře pak musí uvést barvy do původního stavu.

DocBy.T_EX inicializuje poznámky podle pravidel jazyka C:

docby.tex

```
73: \setlinecomment{//} \setlrcoment{/*}{*/}
```

4 Pro designéry

Následuje dokumentace definic maker ovlivňující vzhled dokumentu. Jejich předefinování může způsobit změnu vzhledu podle požadavku uživatele. Místo komplikovaných maker s množstvím parametrů pro řízení vzhledu jsou zde jednoduchá dobře dokumentovaná makra pro jedno použití. Předpokládá se, že při potřebě jiného vzhledu dokumentu je uživatel předefinuje.

Makra zabývající se vzhledem dokumentu jsou pokud možno oddělena od složitosti ostatních maker, ve kterých probíhá hlavní zpracování DocBy.T_EXu. To umožňuje designérovi zaměřit se jen na programování vzhledu a neutopit se v různých cyklech a rekurzích interních maker DocBy.T_EXu.

Typicky jsou makra pro vzhled ve dvou verzích: pro pdfT_EX a bez pdfT_EXu. To je důvod, proč ve výpisech se často vyskytuje test `\ifx\pdfoutput\undefined`.

4.1 Parametry a pomocná makra pro nastavení vzhledu

Velikost `\hsize` ani `\vsize` neměníme. Buď si ji nastaví uživatel, nebo se převezme velikost z plainu (vhodné pro papír letter) či csplainu (vhodné pro papír A4). Nastavujeme ale větší `\parindent`, neboť chceme do proužku vymezeného `\parindent` dát podbarvené čtverečky u názvů sekcí.

docby.tex

```
77: \parindent=30pt
```

Připravíme si „zúženou šířku“ `\nwidth` využitou např. jako šířka záhlaví:

docby.tex

```
79: \newdimen\nwidth \nwidth=\hsize \advance\nwidth by-2\parindent
```

Příkazem plainu `\raggedbottom` nastavíme pružnost stránky dole, a ne mezi jednotlivými řádky. Nastavením `\exhyphenpenalty=10000` zakážeme zlom za pomlčkou (v tisku rozsahu stránek, např. 11–13, takový zlom působí rušivě).

docby.tex

```
81: \raggedbottom
82: \exhyphenpenalty=10000
```

Zavedeme potřebné fonty `\bbf`, `\bbbf`, `\btt`, `\ttsmall`, `\rmsmall`, `\itsmall` a `\partfont`.

docby.tex

```
84: \font\bbf=csb10 at12pt
85: \font\bbbf=csb10 at14.4pt
86: \font\btt=cstt12
87: \font\ttsmall=cstt8
88: \font\rmsmall=csr8
89: \font\itsmall=csti8
90: \font\partfont=csb10 at80pt
```

Makro `\setsmallprinting` přepne do malého strojopisu, připraví `\ttstrut` vhodné velikosti a pomocí `\offinterlineskip` připraví tisk řádků v režimu, kdy se o sebe opírají. Hodnota `\parskip` je nastavena na `-1pt`, aby docházelo k mírnému překrývání a nevznikaly v tisku nebo na obrazovce pruhy. Analogicky pracuje makro `\setnormalprinting`.

docby.tex

```
92: \def\setsmallprinting{\ttsmall \let\it=\itsmall \let\rm=\rmsmall
93:   \def\ttstrut{\vrule height8pt depth3pt width0pt}%
94:   \offinterlineskip \parskip=-1pt\relax
95: }
96: \def\setnormalprinting{\tt \baselineskip=0pt \hfuzz=4em
97:   \def\ttstrut{\vrule height10pt depth3pt width0pt}%
```

```
\hsize: 16, 21–22, 33, 39, 43   \vsize: 32, 44   \parindent: 16, 21–23, 33, 39, 43   \nwidth: 16, 19, 33
\bbf: 16, 18   \bbbf: 16, 18–20   \btt: 16, 18   \ttsmall: 16, 20–21, 33   \rmsmall: 16, 18–20, 33
\itsmall: 16   \partfont: 16, 18   \setsmallprinting: 14, 16, 21–22   \ttstrut: 16, 21–22
\setnormalprinting: 16, 22
```

```
98: \offinterlineskip \parskip=-1pt\relax
99: }
```

V návrhu vzhledu pracuji jen s barvami `\Blue`, `\Red`, `\Brown`, `\Green`, `\Yellow` a `\Black`. Pokud budete chtít další barvy, definujte si je.

docby.tex

```
101: \ifx\pdfoutput\undefined
102:   \def\setcmykcolor#1{}
103: \else
104:   \def\setcmykcolor#1{\special{PDF:#1 k}}
105: \fi
106: \def\Blue{\setcmykcolor{0.9 0.9 0.1 0}}
107: \def\Red{\setcmykcolor{0.1 0.9 0.9 0}}
108: \def\Brown{\setcmykcolor{0 0.85 0.87 0.5}}
109: \def\Green{\setcmykcolor{0.9 0.1 0.9 0.2}}
110: \def\Yellow{\setcmykcolor{0.0 0.0 0.3 0.03}}
111: \def\Black{\setcmykcolor{0 0 0 1}}
112: \let\oriBlack=\Black
```

Barvy jsou definovány pomocí makra `\setcmykcolor`, které je v případě DVI výstupu nastaveno na prázdné makro a v případě PDF výstupu je použit PDF `\special`. Takže příkazy `\Brown` atd. je možné použít i ve verzi maker pro DVI, ovšem v této verzi neudělají nic. Barva `\oriBlack` je konstantně černá barva. Některá makra totiž normální `\Black` předefinovávají a pak se potřebují vrátit pomocí `\oriBlack` ke skutečné černé barvě.

Makro `\rectangle` \langle *výška* \rangle \langle *hloubka* \rangle \langle *šířka* \rangle \langle *obsah* \rangle vytvoří rámeček o stanovených rozměrech se stanoveným obsahem. V PDF verzi je rámeček ve tvaru plného žlutého obdélníku na kterém se nachází \langle *obsah* \rangle zatímco v DVI verzi se vytvoří obrysový rámeček. Pozor: parametr \langle *obsah* \rangle musí obsahovat přepínač barvy, jinak nebude v PDF verzi viditelný. Na druhé straně makro `\rectangle` se postará o návrat do „normální“ černé barvy.

docby.tex

```
114: \ifx\pdfoutput\undefined
115:   \def\rectangle#1#2#3#4{\vbox to0pt{\vss\hrule\kern-.3pt
116:     \hbox to#3{\vrule height#1 depth#2\hss#4\hss\vrule}%
117:     \kern-.3pt\hrule\kern-#2\kern-.1pt}}
118: \else
119:   \def\rectangle#1#2#3#4{\vbox to0pt{\vss\hbox to#3{%
120:     \rlap{\Yellow \vrule height#1 depth#2 width#3}%
121:     \hss#4\Black\hss}\kern-#2}}
122: \fi
```

Nakonec připravíme makro `\docbytex` jako zkratku pro logo DocBy. \TeX u.

docby.tex

```
124: \def\docbytex {\leavevmode\hbox{DocBy}. \TeX}
```

4.2 Vzhled sekcí a podsekcí

Makra `\printsec` \langle *nadpis* \rangle a `\printsecbelow`, jsou volána z makra pro vytvoření sekce `\sec` a mají za úkol vytisknout nadpis. Ostatní problematika, kterou musí řešit makro `\sec` (reference do obsahu, cílové reference, čísla sekcí, plovoucí záhlaví atd.) je zde odstíněna a nemusíme se jí v tuto chvíli zabývat.

Musíme ale dodržet následující úmluvu: Na začátku makra `\printsec` přejdeme pro jistotu do vertikálního módu, pak vložíme potřebné mezery, pak vložíme text nadpisu. V okamžiku, kdy přejdeme do horizontálního módu, vložíme makro `\makelinks`, které zajistí umístění cílů odkazů. Nakonec přejdeme do vertikálního módu příkazem `\par` a *nevkládáme žádné další vertikální mezery*. Makro `\sec` vloží pod vytištěný nadpis do horizontálního seznamu další prvky a posléze zavolá `\printsecbelow`. Tam teprve vložíme mezery obvykle blokované proti zlomu pomocí `\nbreak`. Základní řazení vertikálního seznamu v \TeX u totiž vypadá takto: box, (whatsit, mark, atd.), penalty, glue. O objekty uvedené v závorce se postará `\sec`, my zde řešíme jen box (v makru `\printsec`), a dále penaltu a glue (v makru `\printsecbelow`).

```
\Blue: 15, 17, 20, 35   \Red: 17, 20, 36   \Brown: 17–19, 21   \Green: 15–17, 36   \Yellow: 17, 19–22
\Black: 15–22, 27, 33, 35–36   \setcmykcolor: 17   \oriBlack: 15, 17, 20, 27   \rectangle: 17–20
\docbytex: 17, 34, 40   \printsec: 17–18, 34, 44   \printsecbelow: 17–18, 34–35
```

K dispozici máme hodnotu `\secnum` a `\subsecnum` a dále můžeme použít test `\ifsavetoc`, kterým se ptáme, zda daný nadpis bude v obsahu. Nebude-li, měli bychom místo `\the\secnum` tisknout `\emptynumber`. V makru `\seclabel` je obsah lejbličky sekce, nebo je makro prázdné. To můžeme využít při tisku v režimu „nahrubo“, například tisknout tyto lejbličky do okrajů. DocBy.T_EX tuto vlastnost implicitně neimplementuje.

```

128: \def\printsec #1{\par
129:   \removeatlastskip\bigskip\medskip
130:   \noindent \makelinks
131:   \rectangle{16pt}{9pt}{25pt}{\Brown\bbf\ifsavetoc\the\secnum\else\emptynumber\fi}%
132:   \kern5pt{\bbf\let\_=\subori #1}\par
133: }
134: \def\printsecbelow {\nobreak\medskip}

```

Makra `\printsubsec` a `\printsubsecbelow` fungují analogicky jako právě zmíněná, ale spolupracují s makrem `\subsec`.

```

136: \def\printsubsec #1{\par
137:   \removeatlastskip\bigskip
138:   \noindent \makelinks
139:   \vbox to0pt{\vss
140:     \rectangle{16pt}{9pt}{25pt}{\Brown\bf
141:       \ifsavetoc\the\secnum.\the\subsecnum\else\emptynumber\fi}\kern-5pt}%
142:   \kern5pt{\bbf\let\_=\subori \let\tt=\btt #1}\par
143: }
144: \def\printsubsecbelow {\nobreak\smallskip}

```

Makro `\printpart` vytiskne nadpis části a dopředu dá velké písmeno. Makro `\printpartbelow` tiskne mezeru pod nadpisem části.

```

146: \def\printpart #1{\par
147:   \removeatlastskip\bigskip\medskip
148:   \noindent {\linkskip=60pt\makelinks}%
149:   \rectangle{16pt}{9pt}{25pt}{}%
150:   \kern-20pt{\Brown\partfont\thepart\Black}\kern10pt{\bbf #1}\par
151: }
152: \def\printpartbelow {\nobreak\bigskip}

```

Makro `\emptynumber`, které se použije při `\savetocfalse`, je implicitně nastaveno na prázdnou hodnotu.

```

154: \def\emptynumber{}

```

4.3 Titul, autor

Makro `\title` $\langle titul \rangle \par$ čte parametr $\langle titul \rangle$ pomocí makra `\secparam`, které se postará o případné ignorování mezery na konci parametru (viz sekci 5.8). Makro `\secparam` uloží parametr $\langle titul \rangle$ do tokenlistu `\sectitle` a spustí interní `\iititle`. Toto makro pracuje ve dvou módech (DVI a PDF). V obou módech `\iititle` uloží $\langle titul \rangle$ do makra `\headtitle` (pokud je toto makro prázdné, tedy neinicizované uživatelem) a pomocí příkazu `\noheadline` potlačí na aktuální stránce tisk záhlaví.

```

158: \def\title{\def\tmpA{title}\futurelet\nextchar\secparam}
159: \ifx\pdfoutput\undefined
160:   \def\iititle {\par
161:     \ifx\headtitle\empty\edef\headtitle{\the\sectitle}\fi
162:     \noheadline
163:     \ifx\projectversion\empty \else
164:       \line{\hss\rmsmall\titversion\projectversion}\nobreak\medskip\fi
165:     \centerline{\bbf \let\_=\subori\the\sectitle}\nobreak\medskip}
166: \else
167:   \def\iititle {\par
168:     \ifx\headtitle\empty\edef\headtitle{\the\sectitle}\fi
169:     \noheadline

```

`\printsubsec`: 18, 35, 44 `\printsubsecbelow`: 18, 35 `\printpart`: 18, 35 `\printpartbelow`: 18, 35
`\emptynumber`: 12, 18, 35 `\title`: 12, 4, 18–19, 44 `\iititle`: 18

```

170: \indent\rlap{\rectangle{25pt}{15pt}{\nwidth}{\Black\let\_=\subori\bbf\the\sectitle}}%
171: \ifx\projectversion\empty \else
172: \hbox to\nwidth{\hss
173: \raise26pt\hbox{\Brown\rmsmall\titversion\projectversion\Black}}\fi
174: \par\nobreak\vskip15pt}
175: \fi

```

Makro `\title` v DVI verzi je prosté `\centerline`, zatímco v PDF verzi tiskne podkladový obdélník šířky `\nwidth`.

Pokud není makro `\projectversion` definováno, nastavíme mu výchozí hodnotu jako prázdné makro:

```

177: \ifx\projectversion\undefined \def\projectversion{}\fi

```

Makro `\author` (`autor`)`\par` je společné v obou módech. Umístí jméno autora tučně a na střed.

```

179: \def\author #1\par{\centerline{\bf #1\unskip}\smallskip}

```

4.4 Hlavičky a patičky

DocBy.T_EX nemění výstupní rutinu `plain`. Využívá tedy klasické nástroje na modifikaci vzhledu, tj. text `\footline` a `\headline`.

Návrh vzhledu stránky nepočítá s pravou a levou stranou, protože dokumentaci většinou čteme na monitoru a když ji tiskneme, tak kdo ví, na čem...

Text `\footline` je nastaven tak, aby byla stránková číslice uprostřed podbarvena případně orámována pomocí `\rectangle`.

```

183: \footline={\hss\rectangle{8pt}{2pt}{25pt}{\tenrm\Black\folio}\hss}

```

Text `\headline` se mění. Implicitně obsahuje jen makro `\normalhead`, ale při použití příkazu `\noheadline` na chvíli změni svůj obsah.

```

185: \headline={\normalhead}
186: \def\normalhead {\savepglink \let\_=\subori
187: \vbox to\opt{\vss \baselineskip=7pt \lineskiplimit=0pt
188: \line{\indent\Black\tenit \firstmark \hss \headtitle\indent}
189: \line{\indent\Yellow\xleaders\headlinebox\hfil\indent\Black}}

```

Makro `\normalhead` uloží stránkový link pomocí `\savepglink` a `\vbox/\hbox` gymnastikou vytvoří potřebné záhlaví. Zleva je tištěn název sekce (`\firstmark`) a zprava konstantní text `\headtitle`.

Makro `\noheadline` nastaví `\headline` přechodně na text, podle kterého se vloží jen stránkový odkaz a provede změna obsahu `\headline` na standardní hodnotu. Operace musíme provádět globálně, protože jsme uvnitř výstupní rutiny.

```

191: \def\noheadline {\global\headline={\savepglink\hfil\global\headline={\normalhead}}

```

Makro `\headtitle` obsahuje text shodný v celém dokumentu tištěný vpravo v záhlaví. Implicitně je makro prázdné, po použití příkazu `\title` obsahuje název dokumentu, pokud si uživatel makro nedefinoval sám.

```

193: \ifx\headtitle\undefined \def\headtitle {}\fi

```

Pomocné makro `\headlinebox` udělá v DVI módu prázdný čtvereček a v PDF módu plný (žlutý) čtvereček. Je použito na řádce 189 pro vytvoření čtverečkové čáry v záhlaví,

```

195: \ifx\pdfoutput\undefined
196: \def\headlinebox{\hbox{\kern2pt\rectangle{4pt}{0pt}{4pt}{\kern2pt}}
197: \else
198: \def\headlinebox{\hbox{\kern2pt\vrule height4pt depth0pt width4pt\kern2pt}}
199: \fi

```

`\projectversion`: 12, 14, 18–19 `\author`: 12, 4, 19 `\footline`: 19 `\headline`: 19, 36
`\normalhead`: 19, 44 `\noheadline`: 18–19 `\headtitle`: 12, 18–19, 40 `\headlinebox`: 19

4.5 Tisk cíle odkazu a odkazů pod čarou

Cíl odkazu vytvořený makry `\dg` nebo `\dl` je potřeba vytisknout výrazně, aby jej čtenář pokud možno rychle našel. Tisk probíhá v makru `\printdg` `{⟨před⟩}{⟨slovo⟩}{⟨za⟩}`, kde `⟨před⟩` je text před slovem a `⟨za⟩` je prázdný parametr nebo obsahuje `()`, pokud tyto závorky uživatel v příkaze `\dg`, `\dl` použil.

Současný návrh DocBy.TEXu tiskne z těchto tří parametrů jen jeden, sice `⟨slovo⟩`. V DVI módu tiskne `⟨slovo⟩` v rámečku a v PDF módu tiskne `⟨slovo⟩` červeně a na pozadí je žlutý obdélník.

```
203: \ifx\pdfoutput\undefined
204:   \def\printdg#1#2#3{\leavevmode\hbox{\kern-.6pt
205:     \vbox{\hrule\hbox{\vrule height8.5pt depth2.5pt \kern.2pt
206:       \tt#2\kern.2pt\vrule}\hrule\kern-2.9pt}\kern-.6pt}}
207: \else
208:   \def\printdg#1#2#3{\leavevmode \setbox0=\hbox{\tt#2}%
209:     \Yellow\rlap{\vrule height8.7pt depth2.7pt width\wd0}%
210:     \printdgininside{#2}{\box0}}
211: \fi
```

docby.tex

Červený text se tiskne pomocným makrem `\printdgininside`, které tiskne jednoduše červeně, pokud ke slovu neexistuje `\api` cíl a tiskne červeně pomocí `\ilink`, jestliže existuje `\api` cíl.

```
213: \def\printdgininside#1#2{\ifnum\pgref[+#1]>-1 {\let\Blue=\Red \ilink[+#1]{#2}}%
214:   \else \Red#2\relax\Black\fi}
```

docby.tex

Údaj pod čáru tiskneme makrem `\printfnote` `{⟨před⟩}{⟨d-slovo⟩}{⟨za⟩}{⟨k-slovo⟩}`, kde parametry `⟨před⟩` a `⟨za⟩` mají stejný význam, jako u makra `\printdg`. Parametr `⟨k-slovo⟩` (krátká verze slova) tiskneme červeně, ostatní parametry černě. Parametr `⟨d-slovo⟩` (dlouhá verze slova) není použit.

K naprogramování tohoto makra využijí makro `\specfootnote` `{⟨text⟩}`, které pošle text do speciální poznámky pod čarou. Dále je potřeba vědět, že `\pgref[+⟨slovo⟩]` vrátí číslo strany, kde je `\api` deklarace `⟨slova⟩` nebo vrátí `-1000`. Toto číslo vložíme do `\apinum` a je-li nezáporné, tak jej uvedeme jako první v seznamu stránek a podtržené. Seznam stránek vytiskneme pomocí `\listofpages{⟨slovo⟩}`. V seznamu bude chybět stránka `\apinum`, protože makro `\listofpages` ji vynechává. Prázdný seznam stránek (při kterém netiskneme dvojtečku ani čárku) poznáme podle toho, že `\box0` má nulovou šířku.

docby.tex

```
216: \def\printfnote #1#2#3#4{%
217:   \specfootnote{\let\Black=\oriBlack \ttsmall #1\Red #4\Black#3\rmsmall
218:     \apinum=\pgref[+#2]\relax
219:     \ifnum\apinum>-1 :~\lower1.4pt\vbox{\hbox{\pglink\apinum}\kern1pt\hrule}\fi
220:     \undef{w:#2}\iftrue \setbox0=\hbox{} \else \dgnum=-1 \setbox0=\hbox{\listofpages{#2}}\fi
221:     \ifdim\wd0=0pt \else
222:       \ifnum\apinum>-1 , \else :~\fi
223:       \unhbox0
224:     \fi}}%
225: }
```

4.6 Tisk údaje v obsahu a v rejstříku

Příkaz `\ptocline` `{⟨číslo⟩}{⟨text⟩}{⟨strana⟩}` se postará o tisk údaje o sekci nebo části do obsahu. Dále příkaz `\ptocsubline` `{⟨číslo⟩}{⟨text⟩}{⟨strana⟩}` vytiskne údaj o subsekci. Jak je patrné, tyto dva příkazy se liší jen o jeden `\indent`:

docby.tex

```
229: \def\ptocline #1#2#3{%
230:   \if^^X#1^^X\advance\partnum by1 \medskip \fi
231:   \line{\rectangle{8pt}{1pt}{25pt}}{%
232:     \if^^X#1^^X\ilink[sec:\thepart]{\bbf \thepart}\else\ilink[sec:#1]{#1}\fi}\kern5pt
233:     {\bf\let\_=\subori #2}\mydotfill\pglink#3}}
234: \def\ptocsubline #1#2#3{%
235:   \line{\indent\rectangle{8pt}{1pt}{25pt}{\ilink[sec:#1]{#1}}}\kern5pt
236:     \let\_=\subori #2\mydotfill\pglink#3}}
237: \def\mydotfill{\leaders\hbox to5pt{\hss.\hss}\hfil}
```

Příkaz `\mydotfill` vytiskne tečky do obsahu tak, aby byly pod sebou zarovnané.

Příkaz `\ptocentry` `⟨typ⟩{⟨slovo⟩}{⟨k-slovo⟩}` vytiskne jednu položku o dokumentovaném slově do

```
\printdg: 20, 31–32   \printdgininside: 20, 32   \printfnote: 20, 31   \ptocline: 20, 38
\ptocsubline: 20, 38 \mydotfill: 20       \ptocentry: 21, 37, 39–40
```


obsahu. Parametr $\langle typ \rangle = +$, pokud je v daném místě `\api` dokumentace, a $\langle typ \rangle = @$, je-li v daném místě `\dg` dokumentace. $\langle k-slovo \rangle$ je prázdné, ale při použití `\dl` je v něm krátká verze slova, zatímco ve $\langle slovo \rangle$ je dlouhá verze slova. Dlouhou verzi odkazujeme, krátkou verzi tiskneme.

docby.tex

```
239: \def\ptocentry#1#2#3{\ifhmode,\hskip 7pt plus 20pt minus 3pt \fi
240:   \noindent \hbox{\ttsmall \if+#1\apitext\fi \ilink[#1#2]{\ifx^^X#3^^X#2\else#3\fi}}%
241:   \nobreak\myldots\pglink\pgref[#1#2]\relax
242: }
243: \def\myldots{\leaders\hbox to5pt{\hss.\hss}\hskip20pt\relax}
```

Kdyby bylo potřeba tisknout text před slovem nebo závorky za slovem, je možné využít kontrolní sekvenci `\csname- $\langle slovo \rangle$ \endcsname` jako v následujícím makru `\printindexentry`.

Makro `\myldots` vytvoří tři tečky, které jsou zarovnány s ostatními tečkami v obsahu.

Makro `\printindexentry` $\langle slovo \rangle$ tiskne údaj o slově do rejstříku. Začíná ve vertikálním módu uvnitř sloupce, vytiskne údaj a pomocí `\par` se musí vrátit do vertikálního módu.

docby.tex

```
245: \def\printindexentry #1{%
246:   \expandafter \expandafter \expandafter \separeright \csname-#1\endcsname\end
247:   \apinum=\pgref[+#1]\relax
248:   \leavevmode\llap{\ttsmall \ifnum\apinum>-1 \apitext\fi\tmpa}%
249:   {\tt \ilink[@#1]{#1}\tmpb}: {\bf\pglink\pgref[@#1]}%
250:   \ifnum\apinum>-1 , $\underline{\pglink\apinum}$\fi
251:   \dgnum=\pgref[@#1]\relax
252:   \undef#w:#1\iftrue \setbox0=\hbox{\} \else \setbox0=\hbox{\it\listofpages{#1}}\fi
253:   \ifdim\wd0=0pt \else, \unhbox0 \fi
254:   \hangindent=2\parindent \hangafter=1 \par
255: }
256: \def\separeright #1\right#2\end{\def\tmpa{#1}\def\tmpb{#2}}
```

Pomocí `\separeright` uloží do `\tmpa` text vlevo od slova a do `\tmpb` text vpravo od slova. Makro `\refdg` tyto údaje uložilo do makra `\csname- $\langle slovo \rangle$ \endcsname` oddělené od sebe značkou `\right`. Pomocí makra `\pgref` $[\langle slovo \rangle]$ získám stránku s `\dg` deklarací slova. Pomocí `\pgref` $[+\langle slovo \rangle]$ získám stránku s `\api` deklarací slova. Tuto stránku (pokud existuje) tisknu podtrženě.

4.7 Tisk zdrojového textu

Makra `\ifirst` a `\inext` přetisknou požadovanou část zdrojového textu. Při řešení návrhu vzhledu tisku nás nyní pouze zajímá, že tato makra založí skupinu, pak zavolají příkaz `\printiabove`, pak pro tisk každého řádku zavolají `\printiline` $\langle číslo \rangle$ $\langle text-řádku \rangle$ a nakonec před ukončením skupiny se spustí `\printibelow`. Právě tato tři makra si nyní naprogramujeme. Budeme rozlišovat mezi DVI a PDF módem.

docby.tex

```
261: \ifx\pdfoutput\undefined
262:   \def\printiabove{\line{\leaders\specrule\hfill \kern2pt
263:     {\ttsmall \Brown\inputfilename}\kern2pt \specrule width\parindent}\nobreak
264:     \setsmallprinting}
265:   \def\printibelow{\vskip2pt\hrule\medskip}
266:   \def\specrule{\vrule height 2pt depth-1.6pt }
267:   \def\printiline#1#2{\noindent\ttstrut
268:     \hbox to\parindent{\hss#1:\kern.5em}{#2\par}\penalty11 }
269: \else
270:   \def\printiabove{\smallskip \setsmallprinting}
271:   \def\printibelow{\medskip}
272:   \def\printiline #1#2{\noindent
273:     \rlap{\Yellow \ttstrut width\hsize}%
274:     \ifx\isnameprinted\undefined
275:       \rlap{\line{\hss \raise8.5pt
276:         \hbox{\ttsmall \Brown \vrule height5pt width0pt \inputfilename}}}%
277:       \let\isnameprinted=\relax
278:     \fi
279:     \hbox to\parindent{\hss\Brown#1:\Black\kern.5em}{#2\par}\penalty11 }
280: \fi
```

`\myldots`: 21 `\printindexentry`: 21, 39–40 `\separeright`: 21 `\printiabove`: 21, 26
`\printiline`: 21–22, 27–28 `\printibelow`: 21, 27

V DVI módu tiskneme nahoře čáru se jménem souboru pomocí `\leaders` a makra `\specrule`. Dole pak tiskneme jen jednoduchou čáru. V PDF módu nahoře pouze nastavíme `\setsmallprinting` a vložíme malou mezeru. Dole vložíme střední mezeru.

Makro `\printiline` přejde nejprve do horizontálního módu, tam vloží v DVI módu podpěru a dále box s číslem a box s řádkem. Mezi řádky vkládám penaltu 11. V PDF módu se místo podpěry tiskne celý žlutý proužek v `\rlap`. Protože přes první řádek je potřeba vpravo nahoru vytisknout jméno souboru (později než žlutý proužek), je potřeba zjistit, zda tisknu první řádek nebo další řádky. K tomu slouží kontrolní sekvence `\isnameprinted`, která je typicky `\undefined`. Po vytištění jména souboru (řádky 275 a 276) nastavím `\isnameprinted` na `\relax` a tím poznám, že už je práce provedena. Až makro `\ifirst` nebo `\inext` ukončí skupinu, bude zase mít `\isnameprinted` hodnotu `\undefined`.

4.8 Tisk z prostředí `\begtt/\endtt`

Makro `\begtt` založí skupinu a zavolá `\printvabove`. Dále pro každý tištěný řádek volá makro `\printvline` `{<číslo>}{<text-řádku>}` a nakonec zavolá `\printvbelow`. Číslo řádku jsme se rozhodli nevyužít. V DVI verzi kreslíme jen čáry nahoře a dole. V PDF verzi kreslíme žluté čáry nahoře a dole a v každém řádku pomocí `\rlap` kreslíme žluté obdélníky vpravo a vlevo.

docby.tex

```
284: \ifx\pdfoutput\undefined
285:   \def\printvabove{\smallskip\hrule\nobreak\smallskip\setnormalprinting}
286:   \def\printvbelow{\nobreak\smallskip\hrule\smallskip}
287:   \def\printvline#1#2{\hbox{\ttstrut\indent#2}\penalty12 }
288: \else
289:   \def\printvabove{\medskip\Yellow\hrule height2pt \setnormalprinting\nobreak}
290:   \def\printvbelow{\Yellow\hrule height2pt \Black\medskip}
291:   \def\printvline#1#2{\noindent
292:     \rlap{\hbox to\hsize{\Yellow\ttstrut width25pt\hfil
293:       \vrule width25pt\Black}}\hbox{\indent#2}\par\penalty12 }
294: \fi
```

4.9 Vkládání obrázků

Obrázky jsou vkládány nalevo podle odstavcové zarážky. Tato zarážka je dostatečně velká, takže to působí docela dobře. Celkovou šířku prostoru pro obrázek `\figwidth` spočítám jako `\hsize` mínus `\parindent`

docby.tex

```
298: \newdimen\figwidth \figwidth=\hsize \advance\figwidth by-\parindent
```

Makro `\ifig` `<poměr-šířky>_<název>_` v DVI módu vloží `<název>.eps` a využije k tomu makrobalík `epsf.tex`. V PDF módu vloží `<název>.pdf` a využije k tomu pdfT_EXové primitivy `\pdfximage`, `\pdfrefximage`, `\pdflastximage`.

docby.tex

```
300: \ifx\pdfoutput\undefined
301:   \input epsf
302:   \def\ifig #1 #2 {\bigskip\indent
303:     \hbox{\epsfxsize=#1\figwidth\epsfbox{\figdir#2.eps}}\bigskip}
304: \else
305:   \def\ifig #1 #2 {\bigskip\indent
306:     \hbox{\pdfximage width#1\figwidth {\figdir#2.pdf}}%
307:     \pdfrefximage\pdflastximage}\bigskip}
308: \fi
309: \def\figdir{fig/}
```

Makro `\figdir` obsahuje adresář, ze kterého se obrázky loví.

4.10 Výčty

Makra pro výčty jsou natolik jednoduchá, že asi nepotřebují dalšího komentáře. `\begitem`s zahájí prostředí s výčty, `\enditem`s ukončí toto prostředí, `\itemno` čísluje a `\dbtitem` `<značka>_` zahajuje položku, přičemž se uvnitř prostředí převtělí na `\item`.

```
\specrule: 21 \isnameprinted: 21-22 \printvabove: 22, 28 \printvline: 22, 28
\printvbelow: 22, 28 \figwidth: 22 \ifig: 13, 22 \figdir: 13, 22 \begitem: 13, 23
\enditem: 13, 23 \itemno: 13, 23 \dbtitem: 23 \item: 13, 23
```

```

313: \newcount\itemno
314: \def\beginitems{\medskip\begingroup\advance\leftskip by\parindent \let\item=\dbtitem}
315: \def\dbtitem #1 {\par\advance\itemno by1 \noindent\llap{\ifx*#1$\bullet$\else#1\fi\kern3pt}}
316: \def\enditems{\medskip\endgroup}

```

docby.tex

5 Pro otrlé

Zde je dokumentována implementace DocBy.T_EXu. Je zde výpis všech jeho interních maker včetně podrobného komentáře, jak fungují. Asi není rozumné tato makra měnit, ledaže by si chtěl čtenář naprogramovat DocBy.T_EX vlastní.

5.1 Pomocná makra

Makro `\dbtwarning` zprostředkuje tisk varovných hlášek:

```
321: \def\dbtwarning#1{\immediate\write16{DocBy.TeX WARNING: #1.}}
```

docby.tex

Makra `\defsec` `{<text>}`, `\edefsec` `{<text>}` a `\undef` `{<text>}` jsou zkratky za časté operace s `\csname<text>\endcsname`.

```

323: \def\defsec#1{\expandafter\def\csname#1\endcsname}
324: \def\edefsec#1{\expandafter\edef\csname#1\endcsname}
325: \def\undef#1\iftrue\expandafter\ifx\csname#1\endcsname\relax}

```

docby.tex

Makro `\undef` je potřeba použít takto:

```
\undef{<text>}\iftrue <sekvence nedefinovaná> \else <sekvence definovaná> \fi
```

Nutnost použití `\iftrue` se bohatě vyplatí, až budeme `\undef` přeskakovat vnějšími podmínkami typu `\if`.

Definuji makro `\nb` (normální backslash). Toto makro je pak možné používat při vyhledávání textu s tímto znakem. Rovněž definuji aktivní tabelátor a zástupné sekvence `\obr`, `\cbr`, `\percent` a `\inchquote`.

```

327: {\catcode'\^^I=\active \gdef^^I{\space\space\space\space\space\space\space\space}
328: \catcode'\|=0 \catcode'\|=12 \gdef|nb{\}}
329: \bgroup
330: \catcode'\[=1 \catcode'\]=2 \catcode'\{=12 \catcode'\}=12 \catcode'\%=12
331: \gdef\obr{[ ] \gdef\cbr{[ ]} \gdef\percent [%]
332: \egroup
333: \def\inchquote{"}

```

docby.tex

Makro `\softinput` je vysvětleno v T_EXbooku naruby na straně 288, takže bez komentáře.

```

335: \def\softinput #1 {\let\next=\relax \openin\infile=#1
336: \ifeof\infile \dbtwarning{The file #1 does not exist, run me again}
337: \else \closein\infile \def\next{\input #1 }\fi
338: \next}

```

docby.tex

Makro `\setverb` nastaví kategorie všech speciálních znaků na normální. Viz T_EXbook naruby, stranu 28.

```
340: \def\setverb{\def\do##1{\catcode'##1=12}\dospecials}
```

docby.tex

5.2 Inicializace

Ohlásíme se na terminál:

```

344: \immediate\write16{This is DocBy.TeX, version \dbtversion, modes:
345: \ifx\mubyte\undefined NO\fi enc+%
346: \ifx\pdfoutput\undefined DVI\else PDF\fi+%
347: \ifnum\language=0 ENG\else CS\fi}

```

docby.tex

`\dbtwarning`: 23–25, 27, 31, 36–37, 39 `\defsec`: 23, 28, 30, 36–37, 40, 42 `\edefsec`: 23, 30, 37–38, 40, 42 `\undef`: 20–21, 23, 25, 28–30, 32, 36, 40 `\nb`: 8, 12, 15, 29–30, 34–36, 40
`\obr`: 8, 23 `\cbr`: 8, 23 `\percent`: 8, 15, 23 `\inchquote`: 8, 23 `\softinput`: 23
`\setverb`: 23, 26, 28, 44

Makro `\dbtversion` obsahuje verzi DocBy. \TeX u a je definováno na začátku souboru `docby.tex`. Tam je autor DocBy. \TeX u pozmění, pokud přejde na novou verzi.

```
4: \def\dbtversion {May 2014} % version of DocBy.TeX
```

Je-li použit `csplain`, je aktivován UTF-8 vstup pomocí `enc \TeX` . To ale bohužel není kompatibilní s použitím `enc \TeX` DocBy. \TeX em. Je tedy potřeba deaktivovat UTF-8 vstup a české texty napsat například v ISO-8859-2.

```
7: %           - \input utf8off introduced
8: % Mar. 2011: - fnote: prints short words instead long.
9: %           - index: printsno braces
10: %          - \module is able to read name\_of\_modul with underscore
11: % Jan. 2009: - ^^I prints eight spaces, no three.
12: %           - the \part macro introduced
13: % Nov. 2008: - fith replaced by xyz
14: %           - \refns correction
15: %           - space corrections in \namespace, \endnamespace
16: %           - run me twice -> run me again
17: % Oct. 2008: - first version released
```

Inicializujeme `enc \TeX` ový mód:

```
351: \ifx\mubyte\undefined % encTeX ??
352:   \dbtwarning{encTeX is not detected}
353:   \message{ \space The documented words will be not recognized in source code.}
354:   \message{ \space Use pdftex -ini -enc format.ini to make
355:             your format with encTeX support.}
356:   \csname newcount\endcsname \mubytein
357:   \def\enctextable#1#2{}
358:   \def\noactive#1{}
359: \else
360:   \def\enctextable#1#2{%
361:     \def\tmp ##1,##2\end{\ifx^^X##2^^X}%
362:     \expandafter \tmp \owordbuffer ,#1,\end
363:     \expandafter \mubyte \csname.#1\endcsname #1\endmubyte \fi
364:     \expandafter \gdef \csname.#1\endcsname {#2}%
365:   }
366:   \def\noactive#1{\mubyte \emptysec ##0 #1\endmubyte}
367:   \def\emptysec{}
368: \fi
```

Makro `\enctextable` $\langle slovo \rangle$ $\langle \text{tělo-makra} \rangle$ vloží do `enc \TeX` ové tabulky vzor $\langle slovo \rangle$. Jakmile takový vzor `enc \TeX` objeví, zruší jej ze vstupního proudu a promění jej v kontrolní sekvenci `\. $\langle slovo \rangle$` , která expanduje na $\langle \text{tělo-makra} \rangle$. Například makro `\dg` $\langle slovo \rangle$ aktivuje pro `enc \TeX` $\langle slovo \rangle$, takže provede (mimo jiné) `\enctextable` $\langle slovo \rangle$ $\langle \text{sword} \langle slovo \rangle \rangle$, což způsobí, že se $\langle slovo \rangle$ v načítaném zdrojovém kódu promění na `\sword` $\langle slovo \rangle$.

Makro `\enctextable` odmítá uložit do `enc \TeX` ové tabulky slova, která jsou v seznamu „zakázaných“ slov `\owordbuffer`. Tam jsou slova (oddělená z obou stran čárkou), která se nesmějí aktivovat kvůli `\onlyactive`. Pro taková slova provede `\enctextable` jen definici sekvence `\. $\langle slovo \rangle$` .

Makro `\noactive` $\langle text \rangle$ vloží do `enc \TeX` ové tabulky vyhledávaný text, který ve vstupu zůstane a před něj bude vložena sekvence `\emptysec`. Protože `enc \TeX` neumí ze své tabulky zrušit údaj (umí jen přepsat informaci, na co se má vyhledávaný text proměnit), je potřeba texty, které už v `enc \TeX` ové tabulce nepotřebujeme, deaktivovat alespoň pomocí `\noactive`.

Na `\sword` $\langle text \rangle$ se díky `enc \TeX` u proměňují texty, které se mají automaticky stát klikatelnými linky.

```
370: \def\sword#1{\ilink[#1]{#1}\write\reffile{\string\refuseword{#1}{\the\pageno}}}
```

Makro `\onlyactive` $\langle před \rangle$ $\langle slovo \rangle$ $\langle za \rangle$ zakáže vkládat $\langle slovo \rangle$ do `enc \TeX` ové tabulky (vloží je do `\owordbuffer`, ovšem jen za předpokladu, že už tam není), a nechá celý text $\langle před \rangle \langle slovo \rangle \langle za \rangle$ proměnit v `\oword` $\langle \#1 \rangle$ $\langle \#2 \rangle$ $\langle \#3 \rangle$. Dále pomocí `\noactive` deaktivuje $\langle slovo \rangle$ (při čtení `\reffile` totiž pravděpodobně bylo aktivováno). Makro `\oword` $\langle před \rangle$ $\langle slovo \rangle$ $\langle za \rangle$ tiskne normálně $\langle před \rangle$, dále,

```
\dbtversion: 23–24   \enctextable: 24–25, 29, 31–32, 39   \owordbuffer: 24–25
\noactive: 7, 15, 24–25, 29, 32   \emptysec: 24   \sword: 24, 26, 31–32, 39   \onlyactive: 7, 24–25
\oword: 24–25
```

pokud je definováno `\.<sluvo>`, tak je spustí, jinak tiskne normálně `<sluvo>`. Konečně tiskne vždy normálně text `<za>`.

```

372: \def\onlyactive #1#2#3{\enctextable{#1#2#3}{\oword{#1}{#2}{#3}}%
373:   \def\tmp #1,#2,##2\end{\ifx^^X##2^^X}%
374:   \expandafter \tmp \owordbuffer ,#2,\end
375:   \addtext #2,\to\owordbuffer \noactive{#2}\fi}
376: \def\owordbuffer{,}
377: \def\oword#1#2#3{#1\undef{.#2}\iftrue #2\else\cname.#2\endcname\fi #3}

```

docby.tex

Nakonec inicializujeme DVI/PDF mód:

```

379: \ifx\pdfoutput\undefined
380:   \dbtwarning{pdfTeX is not detected}
381:   \message{ \space The document will be without colors and hyperlinks.}
382:   \message{ \space Use pdfTeX engine, it means: pdfetex command, for example. }
383: \else
384:   \pdfoutput=1
385: \fi

```

docby.tex

5.3 Makra `\ifirst`, `\inext`, `\ilabel`

Deklarujeme `\lineno` jako číslo řádku, `\ttlineno` jako číslo řádku pro `\begtt... \endtt` výpisy, `\ifcontinue` pro řízení cyklu a `\infile` je deskriptor souboru otevřeného ke čtení. `\ifskipping` implementuje uživatelské `\skippingfalse` a `\skippingtrue`.

docby.tex

```

389: \newcount\lineno
390: \newcount\ttlineno
391: \newif\ifcontinue
392: \newif\ifskipping \skippingtrue
393: \newread\infile

```

Příkaz `\ifirst` `{<soubor>}{<odkud>}{<kam>}{<jak>}` nejprve pomocí `\readiparamwhy` analyzuje parametr `<jak>`, pak otevře soubor ke čtení primitivem `\openin`. Je-li otevření neúspěšné, vypíše varování, jinak si uloží název souboru do makra `\inputfilename` a analyzuje parametry pomocí `\scaniparam`: `<odkud>` je uloženo do `\tmpa` a `<kam>` do `\tmpb`. Do `\tmpa` a `\tmpb` se uloží počet opakování (z konstruktoru `\count=<num>`). Nakonec se spustí makro `\insinternal` s expandovanými parametry `<odkud>`, `<kam>`. K tomu je použit známý trik s makrem `\act`.

docby.tex

```

395: \def\ifirst #1#2#3#4{\par\readiparamwhy#4..\end
396:   \openin\infile=#1 \global\lineno=0
397:   \ifeof\infile
398:     \dbtwarning {I am not able to open the file "#1" to reading}
399:   \else
400:     \xdef\inputfilename{#1}
401:     \scaniparam #2^^X\tmpa\tmpA \scaniparam #3^^X\tmpb\tmpB
402:     {\let~=\space \def\empty{^^B^^E}\let\end=\relax \uccode'\~='"\uppercase{\let~}"}%
403:     \noswords \xdef\act{\noexpand\insinternal {\tmpa}{\tmpb}}\act
404:   \fi
405: }

```

Příkaz `\inext` `{<odkud>}{<kam>}{<jak>}` pracuje analogicky, jako `\ifirst`, pouze neotevírá soubor, ale pomocí testu na definovanost makra `\inputfilename` kontroluje, zda náhodou nebyl spuštěn příkaz `\inext` bez předchozího `\ifirst`.

docby.tex

```

406: \def\inext #1#2#3{\par\readiparamwhy#3..\end
407:   \ifx\inputfilename\undefined
408:     \dbtwarning {use \string\ifirst\space before using of \string\inext}
409:   \else
410:     \ifeof\infile
411:     \dbtwarning {the file "\inputfilename" is completely read}
412:   \else

```

`\lineno`: 8, 25, 27–28 `\ttlineno`: 25, 28 `\ifcontinue`: 25–27, 37 `\infile`: 8, 23, 25–27
`\ifskipping`: 25–27 `\skippingfalse`: 8, 9, 27 `\skippingtrue`: 8, 25, 27 `\ifirst`: 8, 9, 14–15,
21–22, 25 `\inputfilename`: 21, 25, 27 `\inext`: 8, 9, 14–15, 21–22, 25

```

413: \scaniparam #1^X\tmpa\tmpA \scaniparam #2^X\tmpb\tmpB
414: {\let~=\space \def\empty{^^B^E}\let\end=\relax \uccode'~='\" \uppercase{\let~}%
415: \noswords \xdef\act{\noexpand\insinternal{\tmpa}{\tmpb}}\act
416: \fi\fi
417: }

```

V rámci expanze parametrů chceme, aby zmizely všechny kontrolní sekvence, které nám do textu vložil automaticky $\text{enc}\TeX$. To provede makro `\noswords`.

```

418: \def\noswords{\def\word##1{##1}\def\lword##1{##1}\def\fword##1##2##3{##2}%
419: \let\flword=\fword \def\leftcomment{\def\returntoBlack{\def\linecomment{}}

```

Makro `\readiparamwhy` načte znaky + nebo - z parametru $\langle jak \rangle$ a uloží je do sekvencí `\startline` a `\stopline`.

```

421: \def\readiparamwhy#1#2#3\end{\let\startline=#1\relax\let\stopline=#2\relax}

```

Makro `\scaniparam` $\langle param \rangle^X \langle out \rangle \langle outnum \rangle$ čte $\langle param \rangle$ ve tvaru `\count= $\langle num \rangle$ $\langle text \rangle$` . Do sekvence $\langle out \rangle$ uloží $\langle text \rangle$ a do sekvence $\langle outnum \rangle$ uloží $\langle num \rangle$. Protože konstruktor `\count= $\langle num \rangle$` je nepovinný, dá trochu více práce parametr analyzovat. K tomu slouží i pomocná makra `\scaniparamA`, `\scaniparamB`, `\scaniparamC`. V případě nepřítomnosti `\count= $\langle num \rangle$` je v $\langle outnum \rangle$ jednička.

```

423: \def\scaniparam{\futurelet\nextchar\scaniparamA}
424: \def\scaniparamA{\ifx\nextchar\count \expandafter\scaniparamB
425: \else \def\tmp{\scaniparamB \count=1 } \expandafter\tmp
426: \fi}
427: \def\scaniparamB \count{\afterassignment\scaniparamC\tempnum}
428: \def\scaniparamC #1^X#2#3{\def#2{#1}\edef#3{\the\tempnum}}

```

Hlavní práci při vkládání zdrojového textu do dokumentace dělá makro `\insinternal` s parametry $\langle odkud \rangle$ a $\langle kam \rangle$.

```

430: \def\insinternal #1#2{%
431: \bgroup
432: \printabove % graficke zpracovani zacatku
433: \setverb \catcode'"=12 \catcode'\^^M=9 \catcode'\^^I=\active
434: \mubytein=1 \obeyspaces \continuetrue \tempnum=\tmpA\relax
435: \def\testline##1#1#2\end{\ifx^^Y##2^^Y\else \nocontinue \fi}%
436: \ifx^X#1^X\def\testline##1\end{\nocontinue}\fi
437: \loop % preskakovani radku
438: \ifeof\infile \returninsinternal{Text "#1" not found (\string\count=\the\tempnum)}\fi
439: \readnewline
440: \expandafter \testline \expandafter ^^B\etext ^^E#1\end
441: \ifcontinue \repeat
442: \let\lastline=\empty
443: \continuetrue \tempnum=\tmpB\relax
444: \def\testline##1#2##2\end{\ifx^^Y##2^^Y\else \nocontinue \fi}%
445: \ifx^X#2^X\def\testline##1\end{\nocontinue}\fi
446: \ifx+\startline \printilineA
447: \expandafter \testline \expandafter ^^B\etext ^^E#2\end
448: \ifcontinue\else\returninsinternal{\fi
449: \readnewline
450: \else
451: \readnewline
452: \ifskipping\ifx\text\empty \readnewline \fi\fi
453: \fi
454: \loop % pretisk radku
455: \expandafter \testline \expandafter ^^B\etext ^^E#2\end
456: \ifcontinue
457: \printilineA
458: \ifeof\infile \returninsinternal{\fi
459: \readnewline \repeat

```

`\noswords`: 25–28 `\readiparamwhy`: 25–26 `\startline`: 26–27 `\stopline`: 26–27
`\scaniparam`: 25–26 `\scaniparamA`: 26 `\scaniparamB`: 26 `\scaniparamC`: 26
`\insinternal`: 25–27


```

460:     \ifx+\stopline \printilineA
461:     \ifx\lastline\relax \else \printiline{\lastline}{}\relax\fi
462:     \fi
463:     \global\let\Black=\oriBlack % pokud jsme skončili vypis uvnitr komentare
464:     \printibelow % graficke zpracovani konce
465:     \egroup\gdef\ilabellist{}\Black
466: }

```

Makro `\insinternal` se skládá ze dvou hlavních cyklů. První (na řádcích 437 až 441) čte postupně řádky ze vstupního souboru (makrem `\readnewline`) a uloží je do makra `\etext`. V tomto cyklu hledá výskyt textu `<odkud>` a nic netiskne.

Druhý cyklus na řádku 454 až 459 čte postupně řádky ze vstupního souboru a hledá výskyt textu `<kam>`. V této chvíli tiskne pomocí makra `\printilineA`.

Před prvním cyklem jsou provedeny přípravné práce: nastavení kategorií, fontů, `\mubytein`. Dále je v přípravné fázi definováno makro `\testline` se separátorem `<odkud>`, pomocí něhož budeme testovat přítomnost textu `<odkud>`. Variantní definice makra `\testline` následují pro speciální případ parametru `<odkud>` (viz uživatelská dokumentace v sekci 2.6). Ukončení cyklu je řízeno podmínkou `\ifcontinue`. Příkaz `\nocontinue` provede `\continuefalse`, ovšem ne vždy. Pokud je zadáno `\count>1`, tj. `\tempnum>1`, pak příkaz pouze zaznamená výskyt hledaného textu a sníží `\tempnum` o jedničku.

docby.tex

```

467: \def\nocontinue{\advance\tempnum by-1 \ifnum\tempnum<1 \continuefalse \fi}

```

Před druhým cyklem v makru `\insinternal` jsou provedeny podobné přípravné práce jako před prvním, znovu je definováno makro `\testline`, tentokrát se separátorem `<kam>`. Vyhledávání probíhá podobně, jako když jsme hledali `<odkud>`.

Pomocí `\ifx+\startline` testujeme, zda tisknout výchozí řádek. Pomocí `\ifx+\stopline` testujeme, zda tisknout ukončovací řádek.

Makro `\ilabellist` obsahuje testování přítomnosti lejbliků deklarovaných příkazem `\ilabel`.

Trikoidní je makro `\returninsinternal {<text>}{<možná-fi>}{<ignoruj>}`, které se spustí při dosažení konce čteného souboru. Marko opustí svůj cyklus pomocí parametru `<ignoruj>`, který je separován textem `\printibelow`, takže to přeskočí větší část obsahu makra `\insinternal` až po řádek 464. Abychom správně opustili vnořené podmínky, jsou přečtena v druhém parametru případná `\fi` a v makru použita. První parametr obsahuje varovací hlášku, chceme-li vypsát varování. Chceme-li být zticha, je parametr prázdný.

docby.tex

```

469: \def\returninsinternal #1#2#3\printibelow{%
470:   \ifx^^X#1^^X\else
471:     \dbtwarning{#1 in file \inputfilename}\fi
472:   #2\fi\printibelow
473: }

```

Makro `\readnewline` je naproti tomu jednoduché:

docby.tex

```

474: \def\readnewline {\read\infile to\text \global\advance\lineno by1\relax
475:   {\noswords \xdef\etext{\text}}}

```

Pracujeme s řádkem čteného souboru ve dvou verzích: neexpandovaným `\text` a expandovaným `\etext` při `\noswords`. Tím máme zaručeno, že v `\etext` nejsou kontrolní sekvence vytvořené enc \TeX em (pro test přítomnosti `<odkud>` nebo `<kam>` by tam ty sekvence překážely). Verze s enc \TeX ovými sekvencemi `\text` se použije při tisku.

Makro `\printilineA` musí mít svou inteligenci: nesmí bezhlavě tisknout prázdné řádky, ale ty tiskne až se zpožděním, následuje-li tisk neprázdného řádku. Tím je zaručeno, že se při `\skippingtrue` nevytiskne poslední prázdný řádek. Makro `\lastline` má tři stavy: `\empty` (na začátku), `\relax` (po vytištění řádku), `<číslo-řádku>` (je-li předchozí řádek prázdný).

docby.tex

```

477: \def\printilineA {%
478:   \ifskipping\else \ifx\text\empty \def\text{ }\fi\fi % trik pro pripad \skippingfalse
479:   \ifx\text\empty
480:     \ifx\lastline\empty % nacten prvni prazdny radek
481:     \let\lastline=\relax

```

`\testline`: 26–27 `\nocontinue`: 26–27 `\returninsinternal`: 26–27 `\readnewline`: 26–27
`\text`: 26–28, 37–38 `\etext`: 26–28 `\printilineA`: 26–27 `\lastline`: 26–28

```

482:     \else           % nacten pozdejsi prazdny radek
483:         \ifx\lastline\relax \else \printiline{\lastline}{}\relax\fi
484:         \edef\lastline{\the\lineno}%
485:     \fi
486: \else           % nacten plny radek
487:     \ifx\lastline\empty \let\lastline=\relax \fi
488:     \ifx\lastline\relax \else \printiline{\lastline}{}\relax\fi
489:     \printiline{\the\lineno}{\text}\relax
490:     \let\lastline=\relax
491: \fi \ilabellist
492: }

```

Pro uložení deklarací pomocí `\ilabel` [*lejblik*]{*text*} slouží makro `\ilabellist`, které musíme nastavit nejprve na prázdnou hodnotu.

docby.tex

```

493: \def\ilabellist {}
494: \def\ilabel [#1]#2{\noswords\edef\act{\noexpand\ilabelee{#1}{#2}}\expandafter\act}
495: \def\ilabelee #1#2{\expandafter\def\expandafter\ilabellist\expandafter{%
496:     \ilabellist \expandafter\testilabel\etext\end{#1}{#2}}
497: }

```

Makro `\ilabel` nejprve expanduje své parametry (pomocí `\act`) a zavolá interní `\ilabelee`. Toto makro přidá do `\ilabellist` toto:

```
\expandafter\testilabel\etext\end{<lejblik>}{<text>}
```

Makro `\testilabel` *řádek*\end{<lejblik>}{<text>} si definuje pomocné makro `\tmp` se separátorem *text*, aby zjistilo, zda je *text* uvnitř *řádek*. Pokud se to povede, registruje cíl odkazu pomocí `\labeltext`.

docby.tex

```

498: \def\testilabel#1\end#2#3{%
499:     \def\tmp ##1#3##2\end{\ifx^^Y##2^^Y\else
500:         \undef{d:#2}\iftrue \defsec{d:#2}{\labeltext[#2]{\the\lineno}}\fi\fi}
501:     \tmp^^B#1^^E#3\end
502: }

```

5.4 Příkazy `\begtt`, `\endtt`

Makro `\begtt` a `\endtt` je podrobně popsáno v T_EXbooku naruby na stranách 27 až 30. Makru `\startverb` dodáme kompletní verbatim text separovaný `\endtt`. Tento text je dělený znakem `^M` (kategorie 12) na řádky a koncový řádek obsahuje token `\end`. Makro spustí ve spolupráci s makrem `\runttloop` cyklus a řádky rozebere, každý řádek zvlášť předá makru `\printvline`. Na konci cyklu se provede makro `\endttloop`. To udělá závěrečné činnosti (zavolá `\printvbelow`, ukončí skupinu) a pomocí makra `\scannexttoken` otestuje první následující token. Pokud to není `\par`, není pod `\endtt` prázdný řádek, takže se provede `\noindent`.

docby.tex

```

506: \def\begtt {\bgroup\printvabove
507:     \setverb \catcode'\="=12 \catcode'\^M=12 \obeyspaces
508:     \begthook\relax \startverb}
509: {\catcode'\|=0 \catcode'\^M=12 \catcode'\|=12 %
510: |gdef\startverb^M#1\endtt{\runttloop#1\end^M}%
511: |gdef\runttloop#1^M{\ifx\end#1 |expandafter|endttloop%
512: |else|global|advance|ttlineno by1 %
513: |printvline{|\the|ttlineno}{#1}|relax|expandafter|runttloop|fi}} %
514: \def\endttloop#1{\printvbelow\egroup\futurelet\nextchar\scannexttoken}
515: \long\def\scannexttoken{\ifx\nextchar\par\else\noindent\fi}

```

V numerickém registru `\ttlineno` je číslo řádku průběžně zvětšované v celém dokumentu. Pokud by někdo chtěl toto číslo využít, může jej nulovat například na začátku každé sekce.

```

\ilabel: 9, 27–28   \ilabellist: 27–28   \ilabelee: 28   \testilabel: 28   \begtt: 9, 14, 22, 25, 28
\startverb: 28   \runttloop   \endttloop: 28   \scannexttoken: 28

```

5.5 Jmenné prostory

Každý jmenný prostor si udržuje své `\namespacemacro`, což je makro s jedním parametrem, které příkazem `\namespace{<tělo-makra>}` mimoděk definuje uživatel. Na počátku je `\namespacemacro` prázdné:

```
519: \def\namespacemacro#1{}
```

docby.tex

Ke každému jmennému prostoru budeme chtít přiřadit lejblik. Rozhodl jsem se za lejblik považovat výsledek expanze `\namespacemacro{@!}`. Budu jej nadále značit `<nslejblik>`. Existuje sice určité riziko nejednoznačnosti `<nslejbliku>`, ale předpokládám, že v praxi nenastane.

Každý jmenný prostor už na počátku musí vědět, jaká všechna lokální slova obsahuje, aby jejich výskyt mohl směřovat na místo, kde je deklarace `\dl`, která může být třeba později než výskyt. Jmenný prostor na svém startu musí tedy do encT_EXových tabulek uložit všechna lokální slova a na svém konci vrátit vše pokud možno do původního stavu. Je tedy zřejmé, že není vhodné čekat až na příkaz `\dl`, ale že je třeba využít soubor `\reffile`. V prvním průchodu tedy jmenné prostory nemohou být aktivní.

Po přečtení `\reffile` má každý jmenný prostor k dispozici makro `\ns:<nslejblik>`, které obsahuje seznam všech svých lokálně deklarovaných slov ve formátu

```
\locword{<slovo1>}\locword{<slovo2>}\locword{<slovo3>}...
```

Protože ukládání do encT_EXové tabulky je globální, definujeme v rámci duševní hygieny všechna makra s tím spojená globálně. Proto je prostředí `\namespace... \endnamespace` nezávislé na skupinách T_EXu.

Při startu `\namespace` je třeba definovat `\namespacemacro`. Původní hodnotu `\namespacemacro` uložíme do `\no:<nslejblik>`, abychom se k němu mohli na konci prostředí `\namespace... \endnamespace` vrátit. Dále definujeme makro `\locword` tak, aby uložilo potřebné údaje do encT_EXové tabulky a před tím ještě si uložilo stávající významy předdefinovaných kontrolních sekvencí. Pak se prostě spustí `\ns:<nslejblik>`.

```
521: \def\namespace #1{%
522:   \let\tmp=\namespacemacro
523:   \gdef\namespacemacro##1{#1}%
524:   \global\expandafter\let\csname no:\namespacemacro{@!}\endcsname\tmp
525:   \ewrite{\string\refns{\namespacemacro{@!}}}%
526:   \def\locword##1{%
527:     \global\expandafter\let
528:       \csname\namespacemacro{@!},##1\expandafter\endcsname\csname.##1\endcsname
529:     \enctable{##1}{\lword{##1}}}%
530:   \csname ns:\namespacemacro{@!}\endcsname
531: }
```

docby.tex

Na konci `\endnamespace` znovu definujeme makro `\locword` tentokrát tak, aby vrátilo pozměněným sekvencím původní význam. Pokud původní význam byl „nedefinovaná sekvence“, je potřeba do encT_EXové tabulky vložit aspoň `\nword`, protože zcela odstranit údaj z tabulky nelze. Dále se vrátíme k původní hodnotě `\namespacemacro`, kterou máme uloženu v `\no:<nslejblik>`.

```
532: \def\endnamespace{\if^^X\namespacemacro{@!}^^X}else
533:   \def\locword##1{%
534:     \global\expandafter\let
535:       \csname.##1\expandafter\endcsname\csname\namespacemacro{@!},##1\endcsname
536:     \undef{.##1}\iftrue \noactive{##1}\fi}%
537:   \csname ns:\namespacemacro{@!}\endcsname
538:   \ewrite{\string\refnsend{\namespacemacro{@!}}}%
539:   \global\expandafter\let\expandafter\namespacemacro\csname no:\namespacemacro{@!}\endcsname
540:   \fi
541: }
```

docby.tex

Uvedená makra pracují s užitečnou zkratkou `\ewrite`, která zapíše text do `\reffile` se zpožděním (primitivem `\write`), ale expanzi udělá hned. Přitom neexpanduje `\nb`.

```
542: \def\ewrite#1{ {\let\nb=\relax \edef\act{\write\reffile{#1}}\act}}
```

docby.tex

`\namespacemacro`: 29–30 `\namespace`: 11, 15, 24, 29 `\locword`: 29, 39 `\endnamespace`: 11, 15, 24, 29 `\ewrite`: 29–31, 34–35

EncT_EX od startu jmenného prostoru vkládá tedy místo každého lokálního $\langle slova \rangle$ kontrolní sekenci $\backslash.\langle slova \rangle$, která expanduje na $\backslash\text{word}\{\langle slova \rangle\}$. Jakmile se tedy objeví výskyt lokálního slova, pracuje $\backslash\text{word}$ takto:

```

544: \def\word#1{\genlongword\tmp{#1}\ilink[@\tmp]{#1}%
545:   \write{\string\refuseword{\tmp}\noexpand\the\pageno}}
546: \def\genlongword#1#2{\expandafter\def\expandafter#1\expandafter{\namespacemacro{#2}}}

```

Makro $\backslash\text{genlongword}$ $\langle tmp \rangle\{\langle slova \rangle\}$ vytvoří z krátké verze slova dlouhou verzi slova a uloží ji do $\langle tmp \rangle$. Výskyt $\langle slova \rangle$ dává o sobě vědět v parametru $\backslash\text{ilink}$ i při zápisu do souboru svým dlouhým (jednoznačným) jménem, zatímco krátké jméno se tiskne.

Zbývá zařídit čtení ze souboru $\backslash\text{reffile}$. Makro $\backslash\text{refns}$ $\{\langle nslejblík \rangle\}$ se objeví v souboru v místě začátku jmenného prostoru a $\backslash\text{refnsend}$ $\{\langle nslejblík \rangle\}$ na konci jmenného prostoru. Mezi nimi se vyskytují $\backslash\text{refdg}\{\langle před \rangle\}\{\langle slova \rangle\}\{\langle za \rangle\}\{\langle k-slova \rangle\}$, přičemž si nyní všimáme jen takových výskytů, které mají neprázdné $\langle k-slova \rangle$. Právě tyto výskyty zanesl do $\backslash\text{reffile}$ příkaz $\backslash\text{dl}$.

```

548: \def\refns#1{\edefsec{o:#1}\currns}
549:   \edef\currns{#1}\undef{ns:\currns}\iftrue \defsec{ns:\currns}\fi}
550: \def\refnsend#1{\edef\currns{\csname o:#1\endcsname}}
551: \def\currns{}

```

Makro $\backslash\text{refns}$ si zapamatuje předchozí $\langle nslejblík \rangle$, který je uložen v makru $\backslash\text{currns}$, do sekvence $\backslash\text{o}:\langle nový-nslejblík \rangle$ a definuje pak $\backslash\text{currns}$ jako $\langle nový-nslejblík \rangle$. Připraví také výchozí stav makra $\backslash\text{ns}:\langle nslejblík \rangle$ na prázdnou hodnotu. Makro $\backslash\text{refdg}$ pak postupně plní buffer $\backslash\text{ns}:\langle nslejblík \rangle$ (viz řádky 850 až 853 v definici makra $\backslash\text{refdg}$ v sekci 5.9). Konečně makro $\backslash\text{refnsend}$ vrátí $\backslash\text{currns}$ do stavu, v jakém bylo před vstupem do stávajícího jmenného prostoru.

5.6 $\backslash\text{dg}$ a přátelé

Makra $\backslash\text{dg}$, $\backslash\text{dl}$, $\backslash\text{dgn}$, $\backslash\text{dgh}$, $\backslash\text{dln}$, $\backslash\text{dlh}$ uloží do $\backslash\text{tmpA}$ svůj název, spustí sken parametrů pomocí $\backslash\text{dgparg}$ a nakonec se promění ve svou interní verzi pomocí $\backslash\text{csname}_i\backslash\text{tmpA}\backslash\text{endcsname}$.

```

555: \def\dg{\def\tmpA{dg}\dgparg} \def\dgn{\def\tmpA{dgn}\dgparg} \def\dgh{\def\tmpA{dgh}\dgparg}
556: \def\dl{\def\tmpA{dl}\dgparg} \def\dln{\def\tmpA{dln}\dgparg} \def\dlh{\def\tmpA{dlh}\dgparg}
557:
558: \def\dgparg {\futurelet\nextchar\dgpargA}
559: \def\dgpargA {\ifx\nextchar[\def\tmp{\dparam}\else\def\tmp{\dparam[]}\fi\tmp}

```

Předchozí makra připraví čtení nepovinného parametru. Hlavní práci provede makro $\backslash\text{dparam}$.

```

561: \def\dparam [#1]#2 {%
562:   \def\printbrackets{}%
563:   \ifx^^X#2^^X\nextdparam{#1}\fi
564:   \def\tmpa{#2}\def\tmpb{}%
565:   \varparam,\tmpa, \varparam.\tmpa. \varparam;\tmpa; \varparam:\tmpa:
566:   \expandafter\managebrackets\tmpa()\end
567:   {\let\nb=\relax
568:   \edef\act{\expandafter\noexpand \csname ii\tmpA\endcsname{#1}\tmpa}\printbrackets}%
569:   \expandafter\act
570:   \tmpb \if|\expandafter\ignoretwo\tmpA|\expandafter\maybespace\fi
571: }
572: \def\nextdparam#1#2\maybespace\fi{\fi\dparam[#1 ]}

```

Je-li za ukončovací závorkou \backslash mezera, pak je parametr \backslash #2 prázdný (je separovaný mezerou). V této situaci se makro $\backslash\text{dparam}$ protočí ještě jednou prostřednictvím makra $\backslash\text{nextdparam}$, které sežere obsah zbytku makra $\backslash\text{dparam}$, vloží mezeru dovnitř závorky a spustí $\backslash\text{dparam}$ ještě jednou. Nyní už je možné začít parametr \backslash #2, tj. $\langle slova \rangle$ rozdělit na část před první čárkou, tečkou, středníkem nebo dvojtečkou a za za tímto znakem. Část před bude v $\backslash\text{tmpa}$ a část za (včetně separátoru) bude v $\backslash\text{tmpb}$. Tuto práci vykoná postupné volání makra $\backslash\text{varparam}$:

```

\word: 26, 29–30   \genlongword: 30–31   \refns: 24, 29–30, 38   \refnsend: 29–30, 38
\currns: 30, 39   \dg: 10, 6–7, 9, 11–13, 20–21, 24, 30–32, 38–39   \dl: 10, 11, 13, 20–21, 29–31, 38–39
\dgn: 10, 11, 13, 30   \dgh: 10, 11, 13, 30   \dln: 10, 11, 13, 30   \dlh: 10, 11, 13, 30   \dgparg: 30
\dparam: 30–31   \nextdparam: 30   \varparam: 30–31

```

```

574: \def\varparam#1{\def\tmp ##1##2 {\def\tmpa{##1}\if^^X##2^^X\else
575:   \expandafter\gobblelast\tmpb\end##1##2\fi}%
576:   \expandafter\tmp}
577: \def\gobblelast#1\end##2{\def\tmp##1##2{\def\tmpb{##1##1}}\tmp}

```

docby.tex

Makro `\varparam⟨separ⟩` definuje pomocné makro `\tmp#1⟨separ⟩#2`, kterému je předloženo `⟨slovo⟩⟨separ⟩`. Je-li `#2` prázdné, pak zabral až `⟨separ⟩` na konci, takže uvnitř `⟨slova⟩` není `⟨separ⟩`. Pak v `\tmpa` zůstává `⟨slovo⟩`. Je-li uvnitř `⟨slova⟩` separátor, pak je potřeba doplnit k `\tmpb` zbytek za separátorem včetně tohoto separátoru. V `#2` máme `⟨zbytek⟩⟨separ⟩` a my potřebujeme do `\tmpb` uložit stávající obsah `\tmpb` před kterým předchází `⟨separ⟩⟨zbytek⟩`. Tuto práci udělá `\gobblelast`, kterému je předloženo `⟨obsah-tmpb⟩\end⟨separ⟩⟨zbytek⟩⟨separ⟩`. Makro definuje `\tmp#1⟨separ⟩` a předloží mu `⟨zbytek⟩⟨separ⟩`. Je tedy v `#1` holý `⟨zbytek⟩` a do `\tmpb` se dostává `⟨separ⟩⟨zbytek⟩⟨starý-obsah-tmpb⟩`.

Po rozdělení vyhledání separátoru máme n `\tmpa` jslovo, ovšem může obsahovat na konci `()`. Proto spustíme na řádce 566 makro `\managebrackets`, které se postará o případné oddělení těchto závorek. Pokud se závorky skutečně oddělily od `\tmpa`, zůstávají v `\printbrackets`.

docby.tex

```

579: \def\managebrackets #1()#2\end{\def\tmpa{#1}%
580:   \if|#2|\else\def\printbrackets{}|\fi}

```

Makro `\maybespace` v závěru činnosti makra `\dparam` vytiskne za obsahem `\tmpb` mezeru, ale jen tehdy, když je jméno makra dvoupísmenné (`\dg`, `\dl`) a nenásleduje znak `'`.

docby.tex

```

582: \def\maybespace{\futurelet\tmp\domaybespace}
583: \def\domaybespace{\let\next=\space
584:   \ifx\tmp'\def\next##1{}\fi
585:   \next}

```

Na řádce 568 vytvoří makro `\dparam` z původního příkazu `\dg*` resp. `\dl*` jeho interní verzi `\iidg*` resp. `\iidl*`. Parametry předá expandovány, aby s nimi bylo méně práce. Stačí tedy naprogramovat uvedená interní makra.

Makro `\iidg` vloží do encTeXové tabulky `\sword` (je to mírně nadbytečné, totéž se provede na začátku zpracování při čtení `\reffile` příkazem `\refdg`). Dále makro vytvoří cíl odkazu tvaru `@⟨slovo⟩`, uloží informaci do `\reffile` ve formátu `\refdg{⟨před⟩}{⟨slovo⟩}{⟨za⟩}{}`, vytiskne `⟨slovo⟩` zvýrazněné pomocí `\printdg` a vloží poznámku pod čáru pomocí `\printfnote`.

docby.tex

```

587: \def\iidg #1#2#3{%
588:   \entextable{#2}{\sword{#2}}%
589:   \label [0#2]%
590:   \write\reffile{\string\refdg{#1}{#2}{#3}{}}%
591:   \printdg{#1}{#2}{#3}%
592:   \printfnote{#1}{#2}{#3}{#2}%
593: }

```

Makro `\iidl` nevkládá nic do encTeXové tabulky, vytvoří cíl pomocí `\label[@⟨dlouhé-slovo⟩]`, zapíše info do `\reffile` ve formátu `\refdg{⟨před⟩}{⟨d-slovo⟩}{⟨za⟩}{⟨k-slovo⟩}`, vytiskne `⟨k-slovo⟩` zvýrazněné pomocí `\printdg` a vloží poznámku pomocí `\printfnote{⟨před⟩}{⟨d-slovo⟩}{⟨za⟩}`.

docby.tex

```

594: \def\iidl #1#2#3{%
595:   \genlongword\tmpB{#2}%
596:   \ifx\tmpB\empty \dbtwarning{\string\dl\space#2 outside namespace, ignored}%
597:   \else
598:     \expandafter\label\expandafter [\expandafter @\tmpB]%
599:     \ewrite{\string\refdg{#1}{\tmpB}{#3}{#2}}%
600:     \printdg{#1}{#2}{#3}%
601:     \printfnote{#1}{\tmpB}{#3}{#2}%
602:   \fi
603: }

```

Makra `\iidgh` a `\iidlh` dělají to samé jako jejich non-h protějšky, jen netisknou slovo v místě výskytu. Lokálně tedy předefinujeme, aby `\printdg` nedělalo nic.

```

\gobblelast: 31   \managebrackets: 30–31   \printbrackets: 30–31   \maybespace: 30–31
\iidg: 11, 31–32   \iidl: 11, 31–32   \iidgh: 11, 32   \iidlh: 11, 32

```



```
604: \def\iidgh#1#2#3{\def\printdg##1##2##3{\iidg{#1}{#2}{#3}}
605: \def\iidlh#1#2#3{\def\printdg##1##2##3{\iidl{#1}{#2}{#3}}}
```

docby.tex

Makro `\iidgn` předefinuje makro `\.< slovo >`, které vyrábí encTeX, tak, že výsledkem expanze je `\fword{<před>}{<slovo>}{<za>}` (namísto obvyklého `\sword{<slovo>}`).

docby.tex

```
607: \def\iidgn#1#2#3{\encntextable{#2}{\fword{#1}{#2}{#3}}}
```

Až se `\fword` spustí (při prvním následujícím výskytu `<slova>`), má za úkol provést `\iidgh`, vytisknout `<slovo>` červeně a vrátit `\.< slovo >` do původního stavu.

docby.tex

```
609: \def\fword#1#2#3{\iidgh{#1}{#2}{#3}\printdginside{#2}{#2}}
```

Makro `\iidln` si uloží stávající význam `\.< slovo >` do sekvence `\;< slovo >` a předefinuje makro `\.< slovo >`, které vyrábí encTeX, tak, že výsledkem je `\flword{<před>}{<slovo>}{<za>}`.

docby.tex

```
611: \def\iidln#1#2#3{%
612:   \global\expandafter\let\csname;#2\expandafter\endcsname\csname.#2\endcsname
613:   \encntextable{#2}{\flword{#1}{#2}{#3}}}
```

Makro `\flword` má za úkol provést `\iidlh`, vytisknout `<slovo>` červeně a vrátit význam makra `\.< slovo >` do původního stavu (který je uložen v sekvenci `\;< slovo >`). Byl-li tento původní význam nedefinován, je potřeba potlačit další činnost makra `\.< slovo >` registrováním jako `\nword{<slovo>}`, protože z encTeXové tabulky už záznam nelze odebrat.

docby.tex

```
615: \def\flword#1#2#3{\iidlh{#1}{#2}{#3}\printdginside{#2}{#2}%
616:   \global\expandafter\let\csname.#2\expandafter\endcsname\csname;#2\endcsname
617:   \undef{.#2}\iftrue \noactive{#2}\fi}
```

5.7 Speciální poznámky pod čarou

Poznámky pod čarou jsou řazeny vedle sebe a obsahují jen slova, která mají na stránce své `\dg`. Protože toto řešení je vizuálně nekompatibilní s uživatelskými poznámkami pod čarou, jednoduše je zakážeme:

docby.tex

```
621: \let\footnote=\undefined
```

Pro speciální poznámky pod čarou využijí už deklarovaný insert `\footins`. Problém je, jak odhadnout, kolik zabere vertikálního místa v poznámkách jedno slovo, když jich může být vedle sebe více. Dirty trick z T_PXbooku (vkládat inserty ve výšce rovné jistému procentu své šířky) se neujal, neboť zlom často nekonvergoval, ale osciloval. V druhém průchodu poznámky teprve dostávají své seznamy stránek a tyto seznamy se pak mohou dále upřesňovat, což zpětně ovlivní vertikální sazbu. Po její změně se mění seznamy stránek a tak pořád dokola.

Rozhodl jsem se tedy pracovat pouze s průměrným koeficientem poznámek, který budou mít všechny poznámky společný. Tento koeficient získám jako celkový počet řádků poznámek v celém dokumentu dělený počtem poznámek. Každá poznámka pak „překáží“ v hlavním vertikálním seznamu výšku řádku poznámek (10pt) násobenou tímto koeficientem. Stačí tedy nastavit `\count\footins`.

Aby problém určitě konvergoval, bylo nutné fixovat výše uvedený koeficient po druhém průchodu. Kdybych jej každý následující průchod měnil, zase se nedočkáme konvergence. Získat uvedený koeficient hned po prvním průchodu není rozumné, protože v té době poznámky ještě nemají vedle sebe seznamy stránek. Výchozí koeficient pro první a druhý průchod je tedy nastaven na `\count\footins=200` (předpokládám zhruba pět poznámek na řádku).

Pracovat s průměrem místo s každou jednotlivou poznámkou může samozřejmě způsobit, že některé stránky jsou plnější a některé prázdnější. Proto je potřeba mít rezervu ve `\skip\footins` a vertikálně pružit kolem poznámkové čáry.

docby.tex

```
623: \skip\footins=18pt
624: \dimen\footins=\vsize
625: \count\footins=200
```


V registru `\totalfoocount` se bude postupně přičítat jednička za každou poznámku a na konci zpracování tam tedy je celkový počet poznámek. V registru `\totalfoodim` bude na konci zpracování celková výška všech řádků s poznámkami.

docby.tex

```
627: \newcount\totalfoocount
628: \newdimen\totalfoodim
```

Makro `\specfootnote` $\langle text \rangle$ vloží do insertu `\footins` jediný `\hbox{\langle text \rangle}` a připočte jedničku do `\totalfoocount`.

docby.tex

```
630: \def\specfootnote#1{\insert\footins\bgroup
631:   \let\tt=\ttsmall \rmsmall
632:   \floatingpenalty=20000 \setbox0=\hbox{#1}%
633:   \ht0=10pt \dp0=0pt \box0 \egroup
634:   \global\advance\totalfoocount by1
635: }
```

Protože jsem se rozhodl neměnit výstupní rutinu `plainu`, musel jsem se „nabourat“ aspoň do její části na tisk poznámek pod čarou. Je to provedeno předefinováním makra `\footnoterule` výstupní rutiny `plainu`. Separátor `\unvbox\footins` způsobí odstranění stejného textu z output rutiny `plainu`.

docby.tex

```
637: \def\footnoterule \unvbox\footins {
638:   \vskip-12pt \vfil
639:   \moveright\parindent\vbox{\hsize=\width \hrule
640:   \setbox2=\vbox{\unvbox\footins \unskip
641:     \setbox2=\lastbox
642:     \global\setbox4=\hbox{\unhbox2}
643:     \loop \unskip\unskip\unpenalty
644:       \setbox2=\lastbox
645:       \ifhbox2 \global\setbox4=
646:         \hbox{\unhbox2 \penalty-300\hskip15pt plus5pt \unhbox4}
647:       \repeat}
648:   \setbox2=\vbox{\hbox{}} \parskip=0pt
649:   \lineskiplimit=0pt \baselineskip=10pt \raggedright \rightskip=0pt plus7em
650:   \leftskip=0pt \hyphenpenalty=10000 \noindent \Black \unhbox4 }
651:   \global\advance\totalfoodim by\ht2 \unvbox2}
652: }
```

Makro rozebere vertikální seznam insertů `\footins` a poskládá je vedle sebe do boxu 4. Pak nastaví parametry sazby na praporek a vypustí box 4 do horizontálního seznamu (`\noindent`) ukončeném `\endgraf`. Tím jsou ve výstupní rutině poznámky pod čarou vysázeny. Nakonec připočteme `\totalfoodim`.

V závěru zpracování v makru `\bye` (viz řádek 790) zapíšeme do souboru `\reffile` informaci o počtu poznámek $\langle počet \rangle$, o celkové výšce řádků poznámek v dokumentu $\langle výška \rangle$ a přidáme aktuální koeficient příspěvku poznámek do vertikálního seznamu $\langle koeficient \rangle$. Informaci zapisujeme jen tehdy, když je `\indexbuffer` neprázdný, tj. když probíhá aspoň druhý průchod. Kdybychom zapisovali i první průchod, dostali bychom velmi zkreslené informace (poznámky v tu chvíli nemají vedle sebe seznamy stránek). Uvedenou informaci zapsanou v předchozím průchodu přečteme na začátku zpracování makrem `\refcoef` $\langle koeficient \rangle \langle počet \rangle \langle výška \rangle$ a nastavíme podle toho společný koeficient všech poznámek `\count\footins`. Makro změní koeficient z výchozí hodnoty 200 na vypočtenou jen jednou. Při dalších průchodech už zůstává u vypočtené hodnoty. Pomocné makro `\gobblertest` odstraní cifry za desetinou tečkou včetně nápisu `pt`.

docby.tex

```
654: \def\refcoef#1#2#3{%
655:   \ifnum#1=200 % jsme na zacatku tretioho pruchodu
656:     \dimen0=#3 \divide\dimen0 by #2
657:     \multiply \dimen0 by100
658:     \afterassignment\gobblertest \count\footins=\the\dimen0 \end
659:   \else \count\footins=#1
660:   \fi
661:   \message{foot-coef: \the\count\footins}
662: }
663: \def\gobblertest #1\end{}
```

`\totalfoocount`: 33, 37 `\totalfoodim`: 33, 37 `\specfootnote`: 20, 33 `\refcoef`: 33, 37–38
`\gobblertest`: 33, 41

Výstupní rutina `\plainoutput` není změněna. Potřebuji ale uvnitř `\output` potlačit expanzi některých maker, které se objeví v argumentu `\write`. Tato makra jsou tedy uvnitř `\output` nastavena na `\relax`. Aby toto nastavení nezměnilo sazbu záhlaví, je potřeba `\makeheadline` provést před změnou maker a uložit si výsledek do boxu.

docby.tex

```
665: \output={\setbox0=\makeheadline \def\makeheadline{\box0\nointerlineskip}
666: \let~=\relax \let\nb=\relax \let\TeX=\relax \let\docbytex=\relax \let\_=\relax \let\tt=\relax
667: \outputhook \plainoutput }
```

5.8 Sekce, podsekce

Nejprve zapíšeme deklarace `\secnum`, `\subsecnum`, `\sectitle`, `\ifsavetoc`. Poslední deklarace připraví uživatelské `\savetocfalse`.

docby.tex

```
672: \newcount\secnum
673: \newcount\subsecnum
674: \newtoks\sectitle
675: \newif\ifsavetoc \savetoctrue
```

Makra `\sec` a `\subsec` mají možnost nepovinného parametru [*lejblik*], za ním může a nemusí být mezera, kterou musíme ignorovat. Na konci parametru *titul* před `\par` rovněž může a nemusí být mezera, kterou musíme ignorovat. Dá tedy práci parametry správně načíst. Makra si uloží svůj název do `\tmpA` a spustí proces načítání parametrů pomocí `\secparam`.

docby.tex

```
677: \def\sec{\def\tmpA{sec}\futurelet\nextchar\secparam}
678: \def\subsec{\def\tmpA{subsec}\futurelet\nextchar\secparam}
```

Makro `\secparam` se vypořádá s případným nepovinným parametrem [*lejblik*]. Pokud je přítomen, uloží *lejblik* do pomocného makra `\seclabel`, jinak tam je prázdno. Makro `\secparamA` se vypořádává s případnou mezerou za hranatou závorkou] a odstraní ji. Makro `\secparamB` *titul* `\par` načte *titul*, ale ten může mít nežádoucí mezeru zcela na konci. S tím se vypořádá makro `\nolastspace` ve spolupráci s makrem `\setparamC`. Posledně jmenované makro uloží už od nežádoucí mezery ošetřený *titul* do `\sectitle` a spustí `\iisec` resp. `\iisubsec`.

docby.tex

```
680: \def\secparam{\ifx\nextchar[%
681: \def\tmp[#1]{\def\seclabel{##1}\futurelet\nextchar\secparamA}%
682: \expandafter\tmp
683: \else \def\seclabel{}\expandafter\secparamB\fi
684: }
685: \def\secparamA{\expandafter\ifx\space\nextchar
686: \def\tmp{afterassignment\secparamB\let\next= }\expandafter\tmp
687: \else \expandafter\secparamB \fi
688: }
689: \def\secparamB #1\par{\nolastspace #1^^X ^^X\end}
690: \def\nolastspace #1 ^^X#2\end{\ifx^^X#2^^X\secparamC #1\else \secparamC #1^^X\fi}
691: \def\secparamC #1^^X{\sectitle={#1}\csname ii\tmpA\endcsname}
```

Makro `\iisec` nejprve nastaví hodnoty `\secnum` a `\subsecnum`, dále definuje `\makelinks`, kde je připravena tvorba odkazů (to použije makro `\printsec`). Dále zavolá `\printsec` na vytištění názvu sekce. Poté uloží informace do `\reffile` ve tvaru `\reftocline {<secnum>}{<titul>}{<strana>}` Nakonec se provede `\mark{<secnum>}_<titul>` a vloží se závěrečná mezeru pomocí `\printsecbelow`.

docby.tex

```
693: \def\iisec{%
694: \ifsavetoc \global\advance\secnum by1 \global\subsecnum=0 \fi
695: \edef\makelinks{%
696: \ifsavetoc \noexpand\savelink[sec:\the\secnum]\fi
697: \if^^X\seclabel^^X\else \noexpand\labeltext[\seclabel]{\the\secnum}\fi}
698: \expandafter \printsec \expandafter{\the\sectitle}% vlozi horni mezeru, text, nakonec \par
699: \ifsavetoc
700: \write {\string\reftocline{\the\secnum}{\the\sectitle}{\noexpand\the\pageno}}\fi
```

`\secnum`: 18, 34–36 `\subsecnum`: 18, 34–35 `\sectitle`: 18–19, 34–35 `\ifsavetoc`: 18, 34–35
`\savetocfalse`: 12, 18, 39 `\sec`: 12, 15, 17, 34–35, 39, 44 `\subsec`: 12, 18, 34 `\secparam`: 18, 34–35
`\seclabel`: 18, 34–35 `\secparamA`: 34 `\secparamB`: 34 `\nolastspace`: 34 `\setparamC`
`\iisec`: 34–35 `\makelinks`: 17–18, 34–35

```

701: \mark{\ifsavetoc \the\secnum\space \else
702:       \ifx\emptynumber\empty\else\emptynumber\space\fi\fi \the\sectitle}
703: \savetoctrue \printsecbelow
704: }

```

Makro `\iisubsec`, které vytváří podsekcí, pracuje analogicky, jako makro `\iisec`.

docby.tex

```

705: \def\iisubsec {%
706:   \ifsavetoc \global\advance\subsecnum by1 \fi
707:   \edef\makelinks{%
708:     \ifsavetoc \noexpand\savelink[sec:\the\secnum.\the\subsecnum]\fi
709:     \if^^X\seclabel^^X\else \noexpand\labeltext[\seclabel]{\the\secnum.\the\subsecnum}\fi}
710:   \expandafter \printsubsec \expandafter{\the\sectitle}% vlozi horni mezeru, text, nakonec \par
711:   \ifsavetoc \ewrite
712:     {\string\reftocline{\the\secnum.\the\subsecnum}{\the\sectitle}{\noexpand\the\pageno}}\fi
713:   \savetoctrue \printsubsecbelow
714: }

```

Makro `\part` bylo zapracováno dodatečně ve verzi Jan. 2009. Registr `\partnum` uchovává číslo části a makro `\thepart` toto číslo konvertuje na písmeno.

docby.tex

```

718: \newcount\partnum
719: \def\thepart{\ifcase\partnum --\or A\or B\or C\or D\or E\or F\or G\or
720:   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
721:   U\or V\or W\or X\or Y\or Z\else +\the\partnum\fi}

```

Makro `\part` má svou implementaci v makru `\iipart` podobně jako například makro `\sec`.

docby.tex

```

723: \def\part{\def\tmpA{part}\futurelet\nextchar\seccparam}
724: \def\iipart{%
725:   \ifsavetoc \global\advance\partnum by1 \fi
726:   \edef\makelinks{%
727:     \ifsavetoc \noexpand\savelink[sec:\thepart]\fi
728:     \if^^X\seclabel^^X\else \noexpand\labeltext[\seclabel]{\thepart}\fi}
729:   \expandafter \printpart \expandafter{\the\sectitle}% vlozi horni mezeru, text, nakonec \par
730:   \ifsavetoc
731:     \ewrite {\string\reftocline}{\the\sectitle}{\noexpand\the\pageno}}\fi
732:   \savetoctrue \printpartbelow
733: }

```

5.9 Odkazy, reference

Klíkáci odkazy řeší makra `\savelink` [*lejblik*] a `\ilink` [*lejblik*]{*text*}. Makro `\savelink` uloží do sazby cíl odkazu. Cíl odkazu vystrčí do výšky `\linkskip` nad účaří. Makro `\ilink` (čti interní link) je dokumentováno v sekci 2.13. Konečně makro `\savepglink` uloží cíl numerického typu (číslo stránky), který bude využit makrem `\pglink` při odkazech na stránky.

docby.tex

```

737: \ifx\pdfoutput\undefined
738:   \def\savelink[#1]{}
739:   \def\ilink [#1]#2{#2}
740:   \def\savepglink{}
741:   \def\pglink{\afterassignment\dopglink\tempnum=}
742:   \def\dopglink{\the\tempnum}
743:   \def\ulink[#1]#2{#2}
744: \else
745:   \def\savelink[#1]{\ifvmode\nointerlineskip\fi
746:     \vbox to0pt{\def\nb{/}_\vss\pdfdest name{#1} xyz\vskip\linkskip}}
747:   \def\ilink [#1]#2{{\def\nb{/}_\pdfstartlink height 9pt depth 3pt
748:     attr{/Border[0 0 0]} goto name{#1}}\Blue#2\Black\pdfendlink}
749:   \def\savepglink{\ifnum\pageno=1 \pdfdest name{sec::start} xyz\relax\fi % viz \bookmarks
750:     \pdfdest num\pageno fitv\relax}
751:   \def\pglink{\afterassignment\dopglink\tempnum=}
752:   \def\dopglink{\pdfstartlink height 9pt depth 3pt
753:     attr{/Border[0 0 0]} goto num\tempnum\relax\Blue\the\tempnum\Black\pdfendlink}

```

`\iisubsec`: 34–35 `\partnum`: 20, 35, 40 `\thepart`: 18, 20, 35, 40 `\part`: 12, 14, 24, 35
`\iipart`: 35 `\savelink`: 34–36 `\ilink`: 12, 13, 20–21, 24, 30, 35–36 `\linkskip`: 18, 35–36

```

754: \def\ulink[#1]#2{\pdfstartlink height 9pt depth 3pt
755:   user{/Border[0 0 0]/Subtype/Link/A << /Type/Action/S/URI/URI(#1)>>}\relax
756:   \Green{\tt #2}\Black\pdfendlink}
757: \fi
758: \newdimen\linkskip \linkskip=12pt

```

Uvedená makra jsem definoval zvlášť pro DVI výstup (jako prázdná makra) a zvlášť pro PDF výstup. Až zase tvůrci pdf \TeX u změní syntaxi nebo názvy primitivů, bude stačit pozměnit uvedená makra. V makru `\ilink` je přímo řečeno, že se má použít modrá barva pro vytvoření odkazů a že odkaz má být bez rámečku. Pokud to někomu nevyhovuje, může si makro předefinovat.

Trik s předefinováním `\nb` (normální backslash) při tvorbě PDF linků vychází ze zkušenosti, že pokud se v názvu linku objeví backslash, některé PDF prohlížeče si s tím neporadí a chovají se podivně. Je tedy nutné, aby argument příkazů `\savelink` a `\ilink` byl neexpandovaný.

Makro `\savepglink` (definice je v předchozím výpisu) je použito v `\headline` každé stránky, takže vytvoří cíl „nahore“ na každé stránce. Makro `\pglink` $\langle number \rangle$ přečte $\langle number \rangle$ (může být ve tvaru numerického registru i přímo jako číslo) a vytvoří link na stránku s tímto číslem. Číslo samotné je vtištěno modře a dá se na ně kliknout. Ke čtení numerického registru je použit primitiv `\afterassignment` a pomocné makro `\dopglink`.

V souboru `\jobname.ref` se prostřednictvím makra `\labeltext` [$\langle lejblík \rangle$]{ $\langle text \rangle$ } uloží řádek, který obsahuje `\reflabel` { $\langle lejblík \rangle$ }{ $\langle text \rangle$ }{ $\langle strana \rangle$ }. Makrem `\reflabel` tyto údaje přečtu a zapíšu do kontrolních sekvencí `^^X<lejblík>` a `^^Y<lejblík>`. Tyto kontrolní sekvence jsou následně využity v makrech `\numref` a `\pgref`. Za povšimnutí stojí, že pokud je $\langle text \rangle$ prázdný (to jsou například všechny případy dokumentovaných slov), pak kontrolní sekvenci `^^X<lejblík>` vůbec nedefinuji, abych šetřil paměti, kterou má \TeX rezervovanu na kontrolní sekvence.

docby.tex

```

760: \def\reflabel #1#2#3{%
761:   \undef{^^Y#1}\iftrue
762:   \ifx^^X#2^^X\else\defsec{^^X#1}{#2}\fi
763:   \defsec{^^Y#1}{#3}%
764:   \else
765:   \dbtwarning{The label [#1] is declared twice}%
766:   \fi
767: }
768: \def\numref [#1]{\undef{^^X#1}\iftrue \else \csname^^X#1\endcsname\fi}
769: \def\pgref [#1]{\undef{^^Y#1}\iftrue-1000\else \csname^^Y#1\endcsname\fi}

```

Makro `\labeltext` [$\langle lejblík \rangle$]{ $\langle text \rangle$ }, jak bylo před chvílí řečeno, uloží do souboru potřebné údaje. Jednak zapíše PDF link pomocí makra `\savelink` a dále uloží do souboru `\reffile` potřebné údaje. K tomu je makro `\writelabel` [$\langle lejblík \rangle$]{ $\langle text \rangle$ }, které pracuje se zpožděným `\write` (aby číslo strany bylo správně). V okamžiku načtení parametru $\langle text \rangle$ jej potřebuji expandovat, protože tam obvykle bývá něco jako `\the\secnum`. Pro vyřešení tohoto problému jsem na chvíli prohodil parametry ($\langle lejblík \rangle$ totiž nechci expandovat) a zavedl pomocné makro `\writelabelinternal` { $\langle text \rangle$ }{ $\langle lejblík \rangle$ }. První část, tj. `\writelabel`{ $\langle text \rangle$ } expandují pomocí `\edef`.

docby.tex

```

771: \def\labeltext[#1]#2{\savelink[#1]\writelabel[#1]{#2}}
772: \def\writelabel[#1]#2{\edef\tmp{\noexpand\writelabelinternal{#2}}\tmp{#1}}
773: \def\writelabelinternal#1#2{\write\reffile{\string\reflabel{#2}{#1}{\the\pageno}}}

```

Makro `\label` je už definováno jednoduše jako „prázdný“ `\labeltext`.

docby.tex

```
775: \def\label[#1]{\labeltext[#1]{}}
```

Makro `\cite` [$\langle lejblík \rangle$] vytiskne klikatelný text. Při chybném $\langle lejblíku \rangle$ vytiskne varování na terminál. Makro je dokumentováno v sekci 2.13.

docby.tex

```

777: \def\cite[#1]{\ifnum \pgref[#1]==-1000
778:   \dbtwarning{label [#1] is undeclared}\Red??\Black
779:   \else \edef\tmp{\numref[#1]}%
780:   \ifx\tmp\empty \edef\tmp{\pgref[#1]}\fi
781:   \ilink[#1]{\tmp}%

```

`\savepglink`: 19, 35, 40 `\pglink`: 20–21, 35, 43 `\dopglink`: 35 `\reflabel`: 36–38
`\numref`: 12, 13, 36 `\pgref`: 12, 13, 20–21, 36 `\labeltext`: 13, 28, 34–36 `\writelabel`: 36
`\writelabelinternal`: 36 `\label`: 12, 13, 31, 36–37 `\cite`: 12, 9, 13, 36

```
782: \fi
783: }
```

S odkazy souvisí makro `\api` {*slovo*}, které vloží `\label` [*+slovo*] dá o sobě vědět ještě jednou do `\reffile`.

```
785: \def\api #1{\label[+#1]\write\reffile{\string\refapiword{#1}}}
786: \def\apitext{\succ$}
```

docby.tex

Makro `\apitext` obsahuje text tištěný vedle *slova* do obsahu a rejstříku.

Při činnosti makra `\bye` zapíšeme do souboru `\reffile` údaje pro `\refcoef` (řádek 790) a dále se zabýváme testem konzistence referencí.

docby.tex

```
788: \def\bye{\par\vfill\supereject
789: \ifx\indexbuffer\empty \else % jsme ve druhem a dalsim pruchodu
790: \immediate\write\reffile{\string\refcoef
791: {\the\count\footins}{\the\totalfoocount}{\the\totalfoodim}}
792: \immediate\closeout\reffile
793: \setrefchecking \continuetrue \input \jobname.ref
794: \ifcontinue \indexbuffer \relax \fi
795: \ifcontinue \ifx\text\tocbuffer \else
796: \continuefalse \dbtwarning{toc-references are inconsistent, run me again}\fi
797: \fi
798: \ifcontinue \immediate\write16{DocBy.TeX: OK, all references are consistent.}\fi
799: \fi
800: \end
801: }
```

Test konzistence vypadá následovně: nejprve uzavřeme zápis do souboru `\reffile`, pak pomocí `\setrefchecking` předefinujeme kontrolní sekvence vyskytující se v `\reffile` a soubor znovu načteme. Nyní makra v něm napsaná dělají test a pokud narazí na problém, provedou `\continuefalse`. Můžeme tedy pomocí `\ifcontinue` zjistit, jak test dopadl. Po přečtení souboru je potřeba udělat ještě důkladnou kontrolu všech automatických odkazů. Proč je tato kontrola vyřešena vypuštěním `\indexbuffer` do vstupní fronty bude jasné po prostudování makra `\setrefchecking`.

docby.tex

```
802: \def\setrefchecking{\catcode'\="=12
803: \def\refcoef##1##2##3{
804: \def\reflabel##1##2##3{\def\tmp{##3}\let\next=\relax
805: \expandafter\ifx\cname^~Y##1\endcsname \tmp
806: \ifx^~X##2^~X\else
807: \def\tmp{##2} \expandafter \ifx \cname^~X##1\endcsname \tmp \else
808: \continuefalse
809: \dbtwarning{text references are inconsistent, run me again}
810: \let\next=\endinput
811: \fi\fi
812: \else
813: \continuefalse
814: \dbtwarning{page references are inconsistent, run me again}
815: \let\next=\endinput
816: \fi\next}
817: \def\refuseword##1##2{\expandafter \ifx\cname -##1\endcsname \relax
818: \defsec{-##1}{##2}\else \edefsec{-##1}{\cname -##1\endcsname,##2}\fi}
819: \def\refdg##1##2##3##4{\addtext\ptocentry @{##2}{##4}\to\tocbuffer}
820: \let\text=\tocbuffer \def\tocbuffer{}
821: \def\,##1{\let##1=\relax}\indexbuffer
822: \def\,##1{\edef\tmp{\expandafter\ignoretwo \string ##1}%
823: \expandafter\ifx \cname w:\tmp\endcsname ##1\else
824: \continuefalse
825: \dbtwarning{auto-references are inconsistent, run me again}
826: \expandafter\ignoretorelax \fi}
827: }
828: \def\ignoretorelax #1\relax{}
```

`\api`: 11, 12–13, 20–21, 37–38 `\apitext`: 12, 21, 37 `\bye`: 7, 11, 33, 37–38 `\setrefchecking`: 37

Zde předefinujeme makro `\refcoef`, aby nedělalo nic. Dále nová verze `\reflabel` kontroluje, zda odkaz je na stejné stránce, jako byl a má stejný text. Nové makro `\refuseword` pracuje jako jeho originální protějšek, jen místo sekvencí `w:< slovo >` plní sekvence `-< slovo >`. Tyto sekvence už známe, nyní je využijeme jinak. Šetříme paměti T_EXu, proto nezakládáme sekvence nové. Nejprve je nutné těmto sekvencím nastavit výchozí hodnotu `\relax`, což je provedeno na řádce 804. Pak znovu předefinují sekvenci `\,`, aby provedla test shodnosti sekvence `w:< slovo >` se sekvencí `-< slovo >` a v makru `\bye` na řádce 789 spustím tento test expandováním makra `\indexbuffer\relax`. Když makro najde nekonzistenci, ohlásí chybu a uteče pomocí `\ignoretorelax`. Dále je předefinováno makro `\refdg`, aby pouze zapisovalo do `\tocbuffer`. Ostatní makra z `\reffile` také zapisují do `\tocbuffer`. Stávající verzi `\tocbuffer` uložíme do `\text` a `\tocbuffer` se při načtení `\reffile` vytvoří znovu. Na řádce 795, zda se nezměnil obsah.

5.10 Tvorba obsahu, rejstříku a záložek

Obsah i rejstřík se mohou pomocí `\dotoc` a `\doindex` objevit kdekoli v dokumentu (třeba na začátku, na konci, uprostřed...). Musíme být připraveni je kdykoli vytisknout. Soubor `\reffile` z minulého běhu můžeme otevřít ke čtení jen na začátku, pak jej mažeme a začínáme znova zapisovat. Při čtení ze souboru `\reffile` tedy musíme uložit všechny potřebné informace k sazbě obsahu i rejstříku. Používáme na to makro `\tocbuffer` a `\indexbuffer`. Na začátku tyto „buffery“ vyprázdňujeme. Makro `\addtext <text>\to<buffer>` budeme používat na vkládání `<textu>` do `<bufferu>`, čímž buffery postupně naplníme.

docby.tex

```
832: \def\tocbuffer{}
833: \def\indexbuffer{}
834: \def\addtext #1\to#2{\expandafter\gdef\expandafter#2\expandafter{#2#1}}
```

V souboru `\reffile` se vyskytují tyto příkazy:

```
\reftocline{<číslo>}{<název>}{<strana>} % údaje o sekci a subsekci pro obsah
\refdg{<před>}{<slovo>}{<za>}{<k-slovo>} % údaj o použití \dg, \dl
\refapiword{<slovo>} % údaj o výskytu \api{<slovo>}
\refuseword{<slovo>}{<strana>} % údaj o přímém výskytu < slova >
\reflabel{<lejbĺk>}{<text>}{<strana>} % viz sekci 5.9, odkazy, reference
\refcoef{<koeficient>}{<počet>}{<výška>} % viz sekci 5.7, spec. poznámky
\refns{<nslejbĺk>} % viz sekci 5.5, jmenné prostory
\refnsend{<nslejbĺk>} % viz sekci 5.5, jmenné prostory
```

Při čtení souboru `\reffile` ukládáme potřebné údaje do bufferů. Nejprve se zaměříme na **obsah** a definujeme `\reftocline {<číslo>}{<název>}{<strana>}`.

docby.tex

```
836: \def\reftocline#1#2#3{\def\currb{#1}%
837:   \istocsec#1.\iftrue \def\currsecb{#1}\else \addbookmark\currsecb \fi
838:   \addtext\dotocline{#1}{#2}{#3}\to\tocbuffer}
```

V `\tocbuffer` tedy máme postupně údaje o všech sekcích a podsekcích v za sebou jdoucích sekvencích `\dotocline {<číslo>}{<název>}{<strana>}`. Mezi sekcí a subsekcí rozlišíme jen podle toho, zda parametr `<číslo>` obsahuje tečku. K tomu slouží pomocné makro `\istocsec`.

docby.tex

```
840: \def\dotocline#1#2#3{\par
841:   \istocsec#1.\iftrue \ptocline{#1}{#2}{#3}\else \ptocsubline{#1}{#2}{#3}\fi}
842: \def\istocsec#1.#2\iftrue{\if^X#2^X}
```

Kdybychom spustili makro `\tocbuffer`, dostaneme obsah. Ale ten se neskládá jen z údajů o sekcích a podsekcích. Ještě je potřeba přečíst `\refdg` a `\refapiword`, abychom mohli vkládat do obsahu i údaje o dokumentovaných slovech.

docby.tex

```
844: \def\refdg#1#2#3#4{%
845:   \edefsec{-#2}{#1\noexpand\right\if!#4!#3\fi}
```

```
\ignoretorelax: 37   \addtext: 25, 37–39, 41   \reffile: 24, 29–31, 33–34, 36–38, 40, 42
\reftocline: 34–35, 38, 40   \tocbuffer: 37–40   \dotocline: 38, 40   \istocsec: 38
\refdg: 21, 30–31, 37–40   \refapiword: 37–39
```



```

846: \expandafter\addtext\csname-#2\endcsname,\to\indexbuffer
847: \addbookmark\currb
848: \addtext\ptocentry @{#2}{#4}\to\toctbuffer
849: \ifx^^X#4^^X\encntable{#2}{\sword{#2}} % slovo je z \dg
850: \else \expandafter\def\csname ns:\currns % slovo je z \dl
851: \expandafter\expandafter\expandafter\endcsname
852: \expandafter\expandafter\expandafter
853: {\csname ns:\currns\endcsname \locword{#4}}
854: \fi
855: }
856: \def\refapiword#1{\addbookmark\currb \addtext\ptocentry +{#1}{}\to\toctbuffer}

```

Makro `\refdg` pracuje s parametry `{⟨před⟩}{⟨slovo⟩}{⟨za⟩}{⟨k-slovo⟩}`, kde `⟨před⟩` je text před slovem, `⟨slovo⟩` je dlouhé slovo, `⟨za⟩` obsahuje případné závorky (). Je-li dlouhé slovo rozdílné od krátkého slova (při použití `\dl`), obsahuje `⟨k-slovo⟩` krátké slovo, jinak je tento parametr prázdný. Makro `\refdg` ukládá informace nejen do `\toctbuffer`, ale také do `\indexbuffer`. Rovněž při prázdném `⟨k-slovo⟩` makro ukládá `\sword` do encTeXové tabulky a při neprázdném `⟨k-slovo⟩` makro cosi kutí se jmennými prostory. Nyní je ale naše pozornost věnována tvorbě obsahu. Ten vytvoří makro `\dotoc`.

docby.tex

```

858: \def\dotoc{\bgroup \savetocfalse \sec \tittoc \par \smallskip
859: \leftskip=\parindent \rightskip=\parindent plus .5\hsize
860: \tochook \toctbuffer \par\egroup}

```

Rejstřík je vybudován z bufferu `\indexbuffer`, ve kterém je seznam deklarovaných slov v dokumentu. Každé slovo je v bufferu zapsáno jako kontrolní sekvence (to zabere v paměti T_EXu nejmíň místa) a je odděleno od další sekvence oddělovačem. Před zařazením podle abecedy jsou položky v `\indexbuffer` odděleny čárkami za položkami, po zařazení jsou položky odděleny `\`, před položkami. Takže obsah `\indexbuffer` vypadá zhruba takto:

```

před zařazením: \-⟨slovo1⟩ , \-⟨slovo2⟩ , \-⟨slovo3⟩ , \-⟨slovo4⟩ , ...
po zařazení:    \ , \-⟨slovoA⟩ \ , \-⟨slovoB⟩ \ , \-⟨slovoC⟩ \ , \-⟨slovoD⟩ ...

```

Zde zápis `\-⟨slovo⟩` znamená jednu kontrolní sekvenci. Každá taková kontrolní sekvence je makrem tvaru `⟨před⟩\right⟨za⟩`. To zařídí řádek 845. Rejstřík vytiskneme makrem `\doindex`.

docby.tex

```

862: \def\doindex {\par\penalty0
863: \calculatedimone
864: \ifdim\dimen1<7\baselineskip \vfil\break \fi
865: \sec \titindex \par
866: \ifx\indexbuffer\empty
867: \dbtwarning {index is empty, run me again}
868: \else
869: \message{DocBy.TeX: sorting index...}
870: \sortindex
871: \indexhook
872: \vskip-\baselineskip
873: \begmulti 2 \rightskip=0pt plus5em \parfillskip=0pt plus2em
874: \widowpenalty=0 \clubpenalty=0
875: \let\,\=\doindexentry \indexbuffer \endmulti
876: \fi
877: }

```

Příkaz `\calculatedimone` s následujícím testem `\dimen1` souvisí se sazbou do dvou sloupců, což necháme na sekci 5.13. Makro tedy založí příkazem `\sec` sekci nazvanou `\titindex` a pokud je `\indexbuffer` neprázdný, spustí sazbu rejstříku. Nejprve se příkazem `\sortindex` setřídí `\indexbuffer` podle abecedy (viz sekci 5.11). Pak makro `\doindex` založí dvousloupcovou sazbu (`\begmulti_2`) a oddělovači `\`, přidělí význam `\doindexentry`. Nakonec vypustí `\indexbuffer` do vstupní fronty, takže další práci opakovaně provede makro `\doindexentry` `\-⟨slovo⟩`, které se postará o tisk slova v rejstříku.

docby.tex

```

878: \def\doindexentry #1{%
879: \edef\tmp{\expandafter\ignoretwo \string #1}%
880: \expandafter \remakebackslash \tmp\end
881: \expandafter \printindexentry \expandafter {\tmp}%

```

`\dotoc`: 7, 4, 14, 38–39 `\indexbuffer`: 33, 37–39, 41–42 `\doindex`: 7, 4, 13–14, 38–39, 44
`\doindexentry`: 39–40

```

882: }
883: \def\remakebackslash#1#2\end{\if#1\nb \def\tmp{\nb#2}\fi}
884: \def\ignoretwo #1#2{}

```

Makro `\doindexentry` pomocí `\ignoretwo` odstraní z kontrolní sekvence `\-< slovo >` úvodní dva znaky `\-`, takže v `\tmp` zůstane `< slovo >`. Pokud `< slovo >` začíná backslashem, uděláme z něj makrem `\remakebackslash` sekvenci `\nb`, neboť přímý backslash není uložen v PDF odkazech (zlobí některé PDF prohlížeče, viz sekci 5.9). Nakonec se vytiskne položka v rejstříku už známým makrem `\printindexentry`.

Při tvorbě **strukturovaných záložek** je potřeba vědět, kolik má každý uzel potomků. Tento údaj je počítaný při čtení `\reffile` voláním makra `\addbookmark <uzel >` (viz makra `\reftocline` a `\refdg`). Parametr `<uzel >` může být číslo sekce, nebo dvojčíslí `<sekce >.<podsekce >`. V makru `\currb` je `<uzel >`, ke kterému je potřeba přičítat potomka a `\currsecb` je případný nadřazený `<uzel >` sekce. Makro `\addbookmark` připočte jedničku k hodnotě makra `\bk:<uzel >`.

docby.tex

```

886: \def\addbookmark#1{\undef{bk:#1}\iftrue\defsec{bk:#1}{1}%
887:   \else \tempnum=\csname bk:#1\endcsname\relax
888:   \advance\tempnum by1
889:   \edefsec{bk:#1}{\the\tempnum}
890:   \fi}
891: \def\currb{} % vychazi hodnota <uzel> pro jistotu

```

Makro `\bookmarks` založí skupinu, předefinuje `\dotocline` a `\ptocentry` (tj. makra obsažená v `\tocbuffer`) vloží první záložku s názvem dokumentu a spustí `\tocbuffer`.

docby.tex

```

893: \def\bookmarks{\ifx\pdfoutput\undefined \else
894:   \bgroup
895:   \def\dotocline##1##2##3{%
896:     \undef{bk:##1}\iftrue \tempnum=0 \else \tempnum=\csname bk:##1\endcsname\relax\fi
897:     \if^X##1^X\advance\partnum by1
898:     \setoutline[sec:\thepart]{##2}{\opartname\space\thepart: }%
899:     \else \setoutline[sec:##1]{##2}{\fi}
900:     \def\ptocentry##1##2##3{\edef\tmpb{\ifx^X##3^X##2\else##3\fi}%
901:     \tempnum=0 \setoutline[##1##2]{\tmpb}{}}%
902:     \def\nb{\string\}\def\TeX{TeX}\def\docbytex{DocBy.TEX}\def\_{_}\def\tt{\tt}\def~{ }%
903:     \def\unskip{}\bookmarkshook
904:     \ifx\headtile\empty \else
905:     \tempnum=0 \setoutline[sec::start]{...\headtile\empty...}\fi % viz \saveglink
906:     \tocbuffer
907:   \egroup \fi
908: }

```

Makro `\setoutline [<lejblík >] { <text > } { <prefix > }` vytvoří záložku `<prefix > <text >` a prolinkuje ji s cílem označeným `<lejblík >`. V `\tempnum` musí být uložen počet potomků záložky.

docby.tex

```

909: \def\setoutline[#1]#2#3{{\def\nb{/_}\xdef\tmp{#1}}%
910:   \def\tmpa{\pdfoutline goto name{\tmp} count -\tempnum}%
911:   \cnvbookmark{\tmpa{#3\nobraces#2\end}}}%
912: }
913: \def\cnvbookmark#1{#1} % zadna konverze
914: \def\nobraces#1{#1\nobrA}
915: \def\nobrA#1{\ifx\end#1\empty\else\nobraces#1\fi}

```

V tomto makru je použito konverzní makro `\cnvbookmark`, které je implicitně neaktivní. Uživatel může například nastavit `\let\cnvbookmark=\lowercase` a nechat konvertovat pomocí `\lccode` znak č na c, znak ž na z, atd. Nastavení `\lccode` musí mít v `\bookmarkshook`.

Dále je text před vložením do záložky podroben konverzi `\nobraces`, která ve spolupráci s makrem `\nobrA` sundá případné závorky `{}`. Takže, pokud máme třeba `{\tt\text}_{\v\TeX}`, po konverzi dostáváme `text_{\v\TeX}`.

`\ignoretwo`: 30, 37, 39–40, 42 `\remakebackslash`: 39–40 `\addbookmark`: 38–40 `\currb`: 38–40
`\currsecb`: 38 `\bookmarks`: 7, 35, 40 `\setoutline`: 40 `\cnvbookmark`: 14, 40 `\nobraces`: 40
`\nobrA`: 40

5.11 Abecední řazení rejstříku

Tuto práci provede makro `\sortindex`. Původně bylo v DocBy.TEXu implementováno algoritmem bubblesort, což vyšlo na šest řádků makrokódu (prezentováno na tutoriálu TEXperience 2008), ale pro větší rejstříky to bylo pomalé. Např. pro rejstřík tohoto dokumentu to vygenerovalo 52 tisíc dotazů na porovnání a trvalo to asi dvě vteřiny. Můj syn Mírek byl pozorný posluchač tutoriálu, takže nabyté znalosti okamžitě využil a přepsal třídící makro na mergesort. Ten na stejném velkém rejstříku generuje 1600 dotazů na porovnání, tedy třicetkrát méně. Cena za to je skutečnost, že makro už nemá jen šest řádků, ale je mírně komplikovanější. Od možnosti použít quicksort jsme upustili, protože implementace by vyžadovala vyšší paměťové nároky na inputstack TEXu.

Nejprve deklarujeme podmínku pro výsledek srovnání dvou položek `\ifAleB` a vytvoříme pomocná makra `\nullbuf`, `\return` a `\fif`. Pomocné makro `\return` ve spolupráci se zakrytým `\fi` uvnitř `\fif` budeme používat pro únik z košatých hluboce vnořených podmínek typu `\if. . \else. . \fi`. Jak uvidíte, makro pracuje na úrovni expandprocesoru a nebude potřeba psát žádné `\expandafter`.

docby.tex

```
919: \newif\ifAleB
920: \def\nullbuf{\def\indexbuffer{}}
921: \def\return#1#2\fi\relax{#1} \def\fif{\fi}
```

Makro `\sortindex` vypustí do vstupní fronty celý `\indexbuffer`, přimaluje k němu `\end, \end`, pronuluje `\indexbuffer` a spustí `\mergesort`.

docby.tex

```
923: \def\sortindex{\expandafter\nullbuf
924:   \expandafter\mergesort\indexbuffer\end,\end
925: }
```

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

Na počátku jsou skupiny jednoprvkové (`\indexbuffer` odděluje každou položku čárkou). Makro `\mergesort` v tomto případě projde seznam a vytvoří seznam zatříděných dvoupoložkových skupin, uložený zpětně v `\indexbuffer`. V dalším průchodu znovu vyvrhne `\indexbuffer` do vstupní fronty, vyprázdní ho a startuje znovu. Nyní vznikají čtyřpoložkové zatříděné skupiny. Pak osmipoložkové atd. V závěru (na řádce 936) je první skupina celá setříděná a druhá obsahuje `\end`, tj. všechny položky jsou už setříděné v první skupině, takže stačí ji uložit do `\indexbuffer` a ukončit činnost. Pomocí `\gobblerest` odstraníme druhé `\end` ze vstupního proudu.

docby.tex

```
926: \def\mergesort #1#2,#3{%
927:   \ifx,#1 % prazdna-skupina,neco, (#2=neco #3=pokracovani)
928:     \addtext#2,\to\indexbuffer % dvojice skupin vyresena
929:     \return{\fif\mergesort#3}% % \mergesort pokracovani
930:   \fi
931:   \ifx,#3 % neco,prazna-skupina, (#1#2=neco #3=,)
932:     \addtext#1#2,\to\indexbuffer % dvojice skupin vyresena
933:     \return{\fif\mergesort}% % \mergesort dalsi
934:   \fi
935:   \ifx\end#3 % neco,konec (#1#2=neco)
936:     \ifx\empty\indexbuffer % neco=kompletni setrideny seznam
937:       \edef\indexbuffer{\napercarky#1#2\end}% % vlozim \, mezi polozky
938:       \return{\fif\gobblerest}% % koncim
939:     \else % neco=posledni skupina nebo \end
940:       \return{\fif\if \expandafter\nullbuf % spojim \indexbuffer+neco a cele znova
941:         \expandafter\mergesort\indexbuffer#1#2,#3}%
942:     \fi\fi % zatriduji: p1+neco1,p2+neco2, (#1#2=p1+neco1 #3=p2)
943:     \isAleB #1#3\ifAleB % p1<p2
944:     \addtext#1\to\indexbuffer % p1 do bufferu
945:     \return{\fif\mergesort#2,#3}% % \mergesort neco1,p2+neco2,
946:   \else % p1>p2
947:     \addtext#3\to\indexbuffer % p2 do bufferu
```

```
\ifAleB: 41   \nullbuf: 41   \return: 41–42   \fif: 41–42   \sortindex: 39, 41
\mergesort: 41–42
```

```

948:     \return{\fif\mergesort#1#2,}%           % \mergesort p1+neco1,neco2,
949:     \fi
950:     \relax % zarazka, na ktere se zastavi \return
951: }

```

Jádro `\mergesort` vidíme na řádcích 943 až 948. 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 zatrižděního 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 927 až 933 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 zářku `\end`, `\end` a je tedy potřeba vystartovat další průchod.

Vlastní srovnání dvou položek dělá makro `\isAleB` $\langle polA \rangle \langle polB \rangle$. Položky jsou tvaru kontrolní sekvence `\-(slovoA)` a `\-(slovoB)`. Makro konvertuje své parametry pomocí `\string` na řadu znaků a expanduje na `\testAleB` $\langle slovoA \rangle \relax \langle slovoB \rangle \relax$. Navíc je na tento test aplikováno `\lowercase`, neboť nerozlišujeme při řazení mezi velkými a malými písmeny.

```

952: \def\isAleB #1#2{%
953:     \edef\tmp{\expandafter\ignoretwo\string#1&0\relax\expandafter\ignoretwo\string#2&1\relax}%
954:     \lowercase \expandafter {\expandafter \testAleB \tmp}%
955: }

```

Makro `\testAleB` $\langle slovoA \rangle \relax \langle slovoB \rangle \relax$ zjistí, zda je $\langle slovoA \rangle$ menší než $\langle slovoB \rangle$. Makro volá samo sebe, pokud jsou první porovnávané znaky stejné. Rekurze určitě skončí, neboť na řádce 953 jsou k porovnávaným slovům připojeny různé ocasy.

```

956: \def\testAleB #1#2\relax #3#4\relax {%
957:     \ifx #1#3\testAleB #2\relax #4\relax \else
958:         \ifnum '#1<'#3 \AleBtrue \else \AleBfalse \fi
959:     \fi
960: }

```

Makro `\napercarky` vloží mezi položky do `\indexbuffer` separátory `\,`. To se provede uvnitř `\edef` `\indexbuffer` na řádce 937.

```

961: \def\napercarky#1{\ifx#1\end \else
962:     \noexpand\,\noexpand#1\expandafter\napercarky
963:     \fi
964: }

```

5.12 Transformace seznamu stránek

Každý výskyt $\langle slova \rangle$ uloží do `\reffile` údaj `\refuseword` $\{\langle slova \rangle\}\{\langle strana \rangle\}$, který přečteme na začátku zpracování v dalším průchodu:

```

969: \def\refuseword#1#2{%
970:     \expandafter \ifx\csname w:#1\endcsname \relax
971:         \defsec{w:#1}{#2}
972:     \else
973:         \edefsec{w:#1}{\csname w:#1\endcsname,#2}
974:     \fi
975: }

```

V sekvenci `\w:\langle slova \rangle` tedy máme seznam stránek s výskyty $\langle slova \rangle$, stránky jsou odděleny čárkami. Seznam může vypadat třeba takto:

```
2,5,5,10,11,12,12,13,13,13,27
```

Cílem je takovýto seznam stránek vytisknout ve formátu `2,□5,□10--13,□27`, tj. odstranit duplicitu a nahradit souvisle jdoucí řadu stránek zápisem ve tvaru $\langle od \rangle -- \langle do \rangle$. Tuto práci dělá makro `\listofpages` $\{\langle slova \rangle\}$, které předhodí makru `\transf` expandovaný seznam stránek ukončený `,0,`.

```

\isAleB: 41–42     \testAleB: 42     \napercarky: 41–42     \refuseword: 24, 30, 37–38, 42
\listofpages: 20–21, 43

```

```

976: \def\listofpages#1{%
977:   \expandafter\expandafter\expandafter \transf\csmame w:#1\endcsmame,0,%
978: }

```

docby.tex

Makro `\transf` vyloučí ze seznamu stránek ty, které jsou rovny `\dgnum` a `\apinum`. Nechceme totiž, aby se v seznamu opakovala hlavní stránka *<slava>* a podtržená stránka. Tyto stránky jsou vytištěny už dříve. Deklarujeme uvedené registry:

```

980: \newcount\apinum
981: \newcount\dgnum
982: \newcount\tempnum
983: \newif\ifdash
984: \newif\iffirst

```

docby.tex

Kromě toho jsme deklarovali pomocné `\tempnum` (aktuálně zpracovávaná stránka), `\ifdash` (zda zpracováváme souvislou skupinu stránek a vytiskli jsme --) a `\iffirst` (zda vkládáme první číslo, tj. není nutné vložit čárku).

Makro `\transf <seznam-stránek>,0`, spustí cyklus pomocí `\cykltransf`.

```

986: \def\transf{\dashfalse \firsttrue \tempnum=-100 \bgroup \cykltransf}
987:
988: \def\cykltransf #1,{\ifnum #1=\apinum \else \ifnum #1=\dgnum \else
989:   \ifnum #1=0 \let\cykltransf=\egroup
990:   \ifdash \pglink\the\tempnum\relax \fi
991:   \else
992:     \ifnum #1=\tempnum % cislo se opakuje, nedelam nic
993:     \else
994:       \advance\tempnum by 1
995:       \ifnum #1=\tempnum % cislo navazuje
996:       \ifdash \else
997:         --\dashtrue
998:       \fi
999:       \else % cislo nenavazuje
1000:       \ifdash
1001:         \advance\tempnum by-1
1002:         \pglink\the\tempnum \relax\dashfalse, \pglink#1\relax
1003:       \else
1004:         \carka \pglink#1\relax
1005:       \fi\fi\fi\fi
1006:       \tempnum=#1 \fi\fi \cykltransf
1007: }
1008: \def\carka{\iffirst \firstfalse \else, \fi}

```

docby.tex

Makro `\cykltransf` je takový malý stavový automat. Věřím, že mu čtenář porozumí bez dalšího komentáře.

5.13 Více sloupců

Sazba do více sloupců je kompletně převzata z T_EXbooku naruby, strany 244–246.

```

1012: \newdimen\colsep \colsep=\parindent % horiz. mezera mezi sloupci
1013: \def\roundtolines #1{%% zaokrouhli na celé násobky vel. řádku
1014:   \divide #1 by\baselineskip \multiply #1 by\baselineskip}
1015: \def\corrsize #1{%% #1 := #1 + \splittopskip - \topskip
1016:   \advance #1 by \splittopskip \advance #1 by-\topskip}
1017:
1018: \def\begmulti #1 {\par\bigskip\penalty0 \def\Ncols{#1}
1019:   \splittopskip=\baselineskip
1020:   \setbox0=\vbox\bgroup\penalty0
1021:   \advance\hsize by\colsep
1022:   \divide\hsize by\Ncols \advance\hsize by-\colsep}
1023: \def\endmulti{\vfil\egroup \setbox1=\vsplit0 to0pt
1024:   \calculatedimone
1025:   \ifdim \dimen1<2\baselineskip

```

docby.tex

`\dgnum`: 20–21, 43 `\apinum`: 20–21, 43 `\tempnum`: 26–27, 35, 40, 43–44 `\ifdash`: 43
`\iffirst`: 43 `\transf`: 42–43 `\cykltransf`: 43

```

1026: \vfil\break \dimen1=\vsize \corrsize{\dimen1} \fi
1027: \dimen0=\Ncols\baselineskip \advance\dimen0 by-\baselineskip
1028: \advance\dimen0 by \ht0 \divide\dimen0 by\Ncols
1029: \roundtolines{\dimen0}%
1030: \ifdim \dimen0>\dimen1 \splitpart
1031: \else \makecolumns{\dimen0} \fi
1032: \ifvoid0 \else \errmessage{ztracený text ve sloupcích?} \fi
1033: \bigskip}
1034: \def\makecolumns#1{\setbox1=\hbox{}\tempnum=0
1035: \loop \ifnum\Ncols>\tempnum
1036: \setbox1=\hbox{\unhbox1 \vsplit0 to#1 \hss}
1037: \advance\tempnum by1
1038: \repeat
1039: \hbox{}\nobreak\vskip-\splittopskip \nointerlineskip
1040: \line{\unhbox1\unskip}}
1041: \def\splitpart{\roundtolines{\dimen1}
1042: \makecolumns{\dimen1} \advance\dimen0 by-\dimen1
1043: \vskip Opt plus 1fil minus\baselineskip \break
1044: \dimen1=\vsize \corrsize{\dimen1}
1045: \ifvoid0 \else
1046: \ifdim \dimen0>\dimen1 \splitpart
1047: \else \makecolumns{\dimen0} \fi \fi}% TBN

```

Zde navíc řešíme problém, že na začátku přepnutí do dvou sloupců pomocí `\begmulti` 2 si makro na řádce 1025 zkontroluje, zda není blízko dna stránky a v takovém případě zahájí dva sloupce až na nové stránce. Ovšem vypadá hloupě, pokud se kvůli tomu ulomí nadpis „Rejstřík“ od jeho obsahu. Je tedy potřeba provést podobné měření stránky už před tiskem nadpisu „Rejstřík“. K tomu slouží makro `\calculatedimone` spuštěné na řádce 863 v makru `\doindex`.

docby.tex

```

1048: \def\calculatedimone{%
1049: \ifdim\pagegoal=\maxdimen \dimen1=\vsize \corrsize{\dimen1}
1050: \else \dimen1=\pagegoal \advance\dimen1 by-\pagetotal \fi}

```

5.14 Závěrečná nastavení, kategorie

Kategorie je rozumné nastavit až na konci souboru `docby.tex`. Na rozdíl od `plainu` přidáváme aktivní kategorii pro znak palce a nastavujeme podtržítka na obyčejnou kategorii, protože se ve zdrojových kódech programů často používá a vůbec ne ve významu matematického indexu. Kdyby mu tento význam zůstal, byly by jen potíže.

Podtržítka je další problém. Skoro vždy chceme, aby se ve vnitřních makrech chovalo jako normální podtržítka, ale když tiskneme text s podtržítkem fontem, který na dané pozici podtržítka nemá (to je Knuthův odkaz), pak by se to mělo udělat plainovským makrem `_`, Toto makro tedy schováme do `\subori` a pak ho probudíme k životu jen v okamžiku tisku v makrech `\printsec`, `\printsubsec`, `\title` a `\normalhead`. Uživatel tedy může napsat `\sec_lojmoje_funkce` a v makrech se bude `_` jako obyčejné podtržítka, zatímco při tisku v nadpise se použije `\subori`.

docby.tex

```

1055: \catcode'\_ =12
1056: \let\subori=\_ \def\_{\_}
1057: \everymath={\catcode'\_ =8 } \everydisplay={\catcode'\_ =8 }

```

Nastavením `\everymath` a `\everydisplay` zaručíme matematikům stále možnost používat podtržítka ve významu indexu.

Aktivní palec spustí lokální verbatim prostředí uvnitř odstavce:

docby.tex

```

1063: \catcode'\"=\active
1064: \let\activeqq="
1065: \def"{\leavevmode\hbox\bgroup\mubytein=1\let\leftcomment=\empty
1066: \let\returntoBlack=\empty \let\linecomment=\empty \let"=\egroup
1067: \def\par{\errmessage{\string\par\space inside \string"... \string"}}%
1068: \setverb\tt\quotehook
1069: }

```


Makro `\langleactive` nastaví aktivní kategorii pro znak je menší (\langle), takže bude možné zapisovat $\langle\text{text}\rangle$ a vytiskne se $\langle\text{text}\rangle$.

docby.tex

```
1071: \def\langleactive{\uccode'\~='\

```

V tomto dokumentu jsem `\langleactive` použil v makru `\quotehook`, protože nechat znak \langle aktivní všude nedělalo dobrotu.

6 Rejstřík

Kontrolní sekvence označené šipkou (\succ) jsou uživatelskými příkazy. Ostatní kontrolní sekvence jsou v DocBy. \TeX u interní. Tučně je označena strana, kde je slovo dokumentováno, pak následuje seznam stran s výskyty slova. Uživatelské příkazy mají v seznamu stránek podtržené číslo, což je stránka, kde je příkaz vyložen na uživatelské úrovni.

<code>\addbookmark:</code> 40 , 38–39	\succ <code>\dln:</code> 30 , <u>10</u> , 11, 13
<code>\addtext:</code> 38 , 25, 37, 39, 41	<code>\docbytex:</code> 17 , 34, 40
\succ <code>\api:</code> 37 , <u>11</u> , 12–13, 20–21, 38	<code>\docsuffix:</code> 15 , 13
<code>\apinum:</code> 43 , 20–21	\succ <code>\doindex:</code> 39 , <u>7</u> , 4, 13–14, 38, 44
\succ <code>\apitext:</code> 37 , <u>12</u> , 21	<code>\doindexentry:</code> 39 , 40
\succ <code>\author:</code> 19 , <u>12</u> , 4	<code>\dopglink:</code> 36 , 35
<code>\bbbf:</code> 16 , 18–20	\succ <code>\dotoc:</code> 39 , <u>7</u> , 4, 14, 38
<code>\bbf:</code> 16 , 18	<code>\dotocline:</code> 38 , 40
\succ <code>\begitems:</code> 22 , <u>13</u> , 23	<code>\dparam:</code> 30 , 31
<code>\begmulti:</code> 44 , 39, 43	<code>\defsec:</code> 23 , 30, 37–38, 40, 42
\succ <code>\begtt:</code> 28 , <u>9</u> , 14, 22, 25	\succ <code>\emptynumber:</code> 18 , <u>12</u> , 35
<code>\begthhook:</code> 14 , 28	<code>\emptysec:</code> 24
<code>\Black:</code> 17 , 15–16, 18–22, 27, 33, 35–36	<code>\enctextable:</code> 24 , 25, 29, 31–32, 39
<code>\Blue:</code> 17 , 15, 20, 35	\succ <code>\enditems:</code> 22 , <u>13</u> , 23
\succ <code>\bookmarks:</code> 40 , <u>7</u> , 35	\succ <code>\endnamespace:</code> 29 , <u>11</u> , 15, 24
<code>\bookmarkshook:</code> 14 , 40	<code>\endttloop:</code> 28
<code>\Brown:</code> 17 , 18–19, 21	<code>\etext:</code> 27 , 26, 28
<code>\btt:</code> 16 , 18	<code>\ewrite:</code> 29 , 30–31, 34–35
\succ <code>\bye:</code> 37 , <u>7</u> , 11, 33, 38	<code>\fif:</code> 41 , 42
<code>\calculatedimone:</code> 44 , 39, 43	\succ <code>\figdir:</code> 22 , <u>13</u>
\succ <code>\cbrace:</code> 23 , <u>8</u>	<code>\figwidth:</code> 22
\succ <code>\cite:</code> 36 , <u>12</u> , 9, 13	<code>\flword:</code> 32 , 26
<code>\cnvbookmark:</code> 40 , 14	<code>\footline:</code> 19
<code>\currb:</code> 40 , 38–39	<code>\fword:</code> 32 , 26
<code>\currns:</code> 30 , 39	<code>\genlongword:</code> 30 , 31
<code>\currsecb:</code> 40 , 38	<code>\gobblelast:</code> 31
<code>\cykltransf:</code> 43	<code>\gobblersrest:</code> 33 , 41
<code>\dbtitem:</code> 22 , 23	<code>\Green:</code> 17 , 15–16, 36
<code>\dbtversion:</code> 24 , 23	<code>\headline:</code> 19 , 36
<code>\dbtwarning:</code> 23 , 24–25, 27, 31, 36–37, 39	<code>\headlinebox:</code> 19
<code>\defsec:</code> 23 , 28, 30, 36–37, 40, 42	\succ <code>\headtitle:</code> 19 , <u>12</u> , 18, 40
\succ <code>\dg:</code> 30 , <u>10</u> , 6–7, 9, 11–13, 20–21, 24, 31–32, 38–39	<code>\hsize:</code> 16 , 21–22, 33, 39, 43
\succ <code>\dgh:</code> 30 , <u>10</u> , 11, 13	<code>\ifAleB:</code> 41
\succ <code>\dgn:</code> 30 , <u>10</u> , 11, 13	<code>\ifcontinue:</code> 25 , 26–27, 37
<code>\dgnum:</code> 43 , 20–21	<code>\ifdash:</code> 43
<code>\dgpar:</code> 30	<code>\iffirst:</code> 43
\succ <code>\dl:</code> 30 , <u>10</u> , 11, 13, 20–21, 29, 31, 38–39	\succ <code>\ifig:</code> 22 , <u>13</u>
\succ <code>\dlh:</code> 30 , <u>10</u> , 11, 13	\succ <code>\ifirst:</code> 25 , <u>8</u> , 9, 14–15, 21–22

`\langleactive:` 14, 45

- `\ignoretorelax`: 38, 37
- `\ignoretwo`: 40, 30, 37, 39, 42
- `\iidg`: 31, 11, 32
- `\iidgh`: 31, 11, 32
- `\iidgn`: 32, 11
- `\iidl`: 31, 11, 32
- `\iidlh`: 31, 11, 32
- `\iidln`: 32, 11
- `\iipart`: 35
- `\iisec`: 34, 35
- `\iisubsec`: 35, 34
- `\iititle`: 18
- `\ilabel`: 28, 9, 27
- `\ilabelee`: 28
- `\ilabellist`: 28, 27
- `\ilink`: 35, 12, 13, 20–21, 24, 30, 36
- `\inchquote`: 23, 8
- `\indexbuffer`: 39, 33, 37–38, 41–42
- `\indexhook`: 14, 39
- `\inext`: 25, 8, 9, 14–15, 21–22
- `\infile`: 25, 8, 23, 26–27
- `\inputfilename`: 25, 21, 27
- `\ins`: 15, 5, 6, 8, 13
- `\insinternal`: 26, 25, 27
- `\isAleB`: 42, 41
- `\isnameprinted`: 22, 21
- `\istocsec`: 38
- `\item`: 22, 13, 23
- `\itemno`: 22, 13, 23
- `\itsmall`: 16
- `\label`: 36, 12, 13, 31, 37
- `\labeltext`: 36, 13, 28, 34–35
- `\langleactive`: 45, 14
- `\lastline`: 27, 26, 28
- `\leftcomment`: 15, 26, 44
- `\linecomment`: 15, 26, 44
- `\lineno`: 25, 8, 27–28
- `\linkskip`: 35, 18, 36
- `\listofpages`: 42, 20–21, 43
- `\locword`: 29, 39
- `\lword`: 30, 26, 29
- `\makelinks`: 34, 17–18, 35
- `\managebrackets`: 31, 30
- `\maybespace`: 31, 30
- `\mergesort`: 41, 42
- `\module`: 15, 4, 5, 10–13, 24
- `\modulename`: 15
- `\mydotfill`: 20
- `\myldots`: 21
- `\namespace`: 29, 11, 15, 24
- `\namespacemacro`: 29, 30
- `\napercarky`: 42, 41
- `\nb`: 23, 8, 12, 15, 29–30, 34–36, 40
- `\nextdparam`: 30
- `\noactive`: 24, 7, 15, 25, 29, 32
- `\nobrA`: 40
- `\nobraces`: 40
- `\nocontinue`: 27, 26
- `\noheadline`: 19, 18
- `\nolastspace`: 34
- `\normalhead`: 19, 44
- `\noswords`: 26, 25, 27–28
- `\nullbuf`: 41
- `\numref`: 36, 12, 13
- `\nwidth`: 16, 19, 33
- `\obrace`: 23, 8
- `\onlyactive`: 24, 7, 25
- `\opartname`: 14, 40
- `\oriBlack`: 17, 15, 20, 27
- `\outpuhook`: 14, 34
- `\oword`: 24, 25
- `\owordbuffer`: 24, 25
- `\parindent`: 16, 21–23, 33, 39, 43
- `\part`: 35, 12, 14, 24
- `\partfont`: 16, 18
- `\partnum`: 35, 20, 40
- `\percent`: 23, 8, 15
- `\pglink`: 36, 20–21, 35, 43
- `\pgref`: 36, 12, 13, 20–21
- `\printbrackets`: 31, 30
- `\printdg`: 20, 31–32
- `\printdginside`: 20, 32
- `\printfnote`: 20, 31
- `\printiabove`: 21, 26
- `\printibelow`: 21, 27
- `\printiline`: 21, 22, 27–28
- `\printilineA`: 27, 26
- `\printindexentry`: 21, 39–40
- `\printpart`: 18, 35
- `\printpartbelow`: 18, 35
- `\printsec`: 17, 18, 34, 44
- `\printsecbelow`: 17, 18, 34–35
- `\printsibsec`: 18, 35, 44
- `\printsibsecbelow`: 18, 35
- `\printvabove`: 22, 28
- `\printvbelow`: 22, 28
- `\printvline`: 22, 28
- `\projectversion`: 19, 12, 14, 18
- `\ptocentry`: 20, 21, 37, 39–40
- `\ptocline`: 20, 38
- `\ptocsubline`: 20, 38
- `\quotehook`: 14, 44–45
- `\readiparamwhy`: 26, 25
- `\readnewline`: 27, 26
- `\rectangle`: 17, 18–20
- `\Red`: 17, 20, 36
- `\refapiword`: 38, 37, 39
- `\refcoef`: 33, 37–38
- `\refdg`: 38, 21, 30–31, 37, 39–40
- `\reffile`: 38, 24, 29–31, 33–34, 36–37, 40, 42
- `\reflabel`: 36, 37–38

`\refns`: 30, 24, 29, 38
`\refnsend`: 30, 29, 38
`\reftocline`: 38, 34–35, 40
`\refuseword`: 42, 24, 30, 37–38
`\remakebackslash`: 40, 39
`\return`: 41, 42
`\returnninsinternal`: 27, 26
`\returntoBlack`: 15, 26, 44
`\rightcomment`: 15
`\rmsmall`: 16, 18–20, 33
`\runttloop`: 28
`\savelink`: 35, 34, 36
`\savepglink`: 36, 19, 35, 40
`\savetocfalse`: 34, 12, 18, 39
`\scaniparam`: 26, 25
`\scaniparamA`: 26
`\scaniparamB`: 26
`\scaniparamC`: 26
`\scannexttoken`: 28
`\sec`: 34, 12, 15, 17, 35, 39, 44
`\seclabel`: 34, 18, 35
`\secnum`: 34, 18, 35–36
`\secparam`: 34, 18, 35
`\secparamA`: 34
`\secparamB`: 34
`\sectitle`: 34, 18–19, 35
`\separeright`: 21
`\setcmykcolor`: 17
`\setlinecomment`: 15, 16
`\setlrcoment`: 15, 16
`\setnormalprinting`: 16, 22
`\setoutline`: 40
`\setparamC`: 34
`\setrefchecking`: 37
`\setsmallprinting`: 16, 14, 21–22
`\setverb`: 23, 26, 28, 44
`\skippingfalse`: 25, 8, 9, 27
`\skippingtrue`: 25, 8, 27
`\softinput`: 23
`\sortindex`: 41, 39
`\specfootnote`: 33, 20
`\specrule`: 22, 21
`\startline`: 26, 27
`\startverb`: 28
`\stoptline`: 26, 27
`\subori`: 44, 18–20
`\subsec`: 34, 12, 18
`\subsecnum`: 34, 18, 35
`\sword`: 24, 26, 31–32, 39
`\tempnum`: 43, 26–27, 35, 40, 44
`\testAleB`: 42
`\testilabel`: 28
`\testline`: 27, 26
`\text`: 27, 26, 28, 37–38
`\thepart`: 35, 18, 20, 40
`\titindex`: 14, 39
`\title`: 18, 12, 4, 19, 44
`\titmodule`: 14, 15
`\tittoc`: 14, 39
`\titversion`: 14, 18–19
`\tocbuffer`: 38, 37, 39–40
`\tochook`: 14, 39
`\totalfoocount`: 33, 37
`\totalfoodim`: 33, 37
`\transf`: 43, 42
`\ttlineno`: 25, 28
`\ttsmall`: 16, 20–21, 33
`\ttstrut`: 16, 21–22
`\undef`: 23, 20–21, 25, 28–30, 32, 36, 40
`\varparam`: 30, 31
`\vsize`: 16, 32, 44
`\writelabel`: 36
`\writelabelinternal`: 36
`\Yellow`: 17, 19–22
`struct dvojice`: 5, 6
`struct dvojice uzasna_funkce()`: 5