Zpravodaj Československého sdružení uživatelů TeXu

Petr Olšák

DocBy.TeX – dokumentování zdrojových textů TeXem

*Zpravodaj Československého sdružení uživatelů TeXu*, Vol. 18 (2008), No. 3, 130–141

Persistent URL: http://dml.cz/dmlcz/150055

# DocBy.TEX – dokumentování zdrojových textů TEXem

Petr Olšák

DocBy.TEX je makro TEXu, které umožní jednoduše dokumentovat zdrojové texty v programovacích jazycích, například v jazyku C. Obsahuje možnost vkládání vymezených úseků zdrojových textů a díky encTEXu umí automaticky zvýraznit dokumentovaná slova v těchto textech a udělat z nich aktivní odkazy. Automaticky vytváří obsah a rejstřík. K vytvoření cílové dokumentace v PDF formátu je potřeba pouze pdfTEX doplněný encTEXem. Není nutné použít žádné preprocesory.

## 1. Úvod, motivace, zadání

Můj syn Mirek píše ročníkovou práci z programování a je zřejmé, že pokud to bude chtít odevzdat, měl by své zdrojové kódy nějak dokumentovat. Nabídl jsem se mu, že mu pro tyto účely napíšu TEXové udělátko. Snažil jsem se splnit následující cíle

- Formát dokumentace je klikací PDF. Dokumentaci ve formátu html negenerujeme.
- Věc by měla být co nejjednodušší, tj. pokud možno jen použití TEXu. Případné preprocesory by práci zbytečně komplikovaly.
- K napsanému zdrojovému kódu se může psát dokumentace zvlášť do „vedlejšího“ TEXového souboru, v němž budou příkazy na vkládání vymezených úseků zdrojového kódu podle potřeby. Kód bude tedy vložen v době zpracování dokumentace TEXem aktuální. Tj. přesně ten, který načítá kompilátor při překladu programu.
- Zdrojový kód je členěn do modulů. Kompilátor je překládá do objektů `*.o` a ty pak sestavuje do výsledného programu. Jeden modul z pohledu programátora řeší jeden ucelený problém a stojí za to jej dokumentovat jako „samostatnou jednotku“. TEXové soubory s dokumentací tedy respektují strukturu členění na moduly.
- Pomocí encTEXu je možné zařídit, že libovolný výskyt dokumentovaného slova v začleněném zdrojovém kódu se stane automaticky aktivním odkazem na místo, kde je slovo dokumentováno. Dokumentovaným slovem může být funkce, struktura, proměnná nebo cokoli jiného.
- DocBy.TEX sám vytvoří pod čarou na stránce, kde je slovo dokumentováno, seznam všech stránek, kde se slovo vyskytuje napříč celou dokumentací.

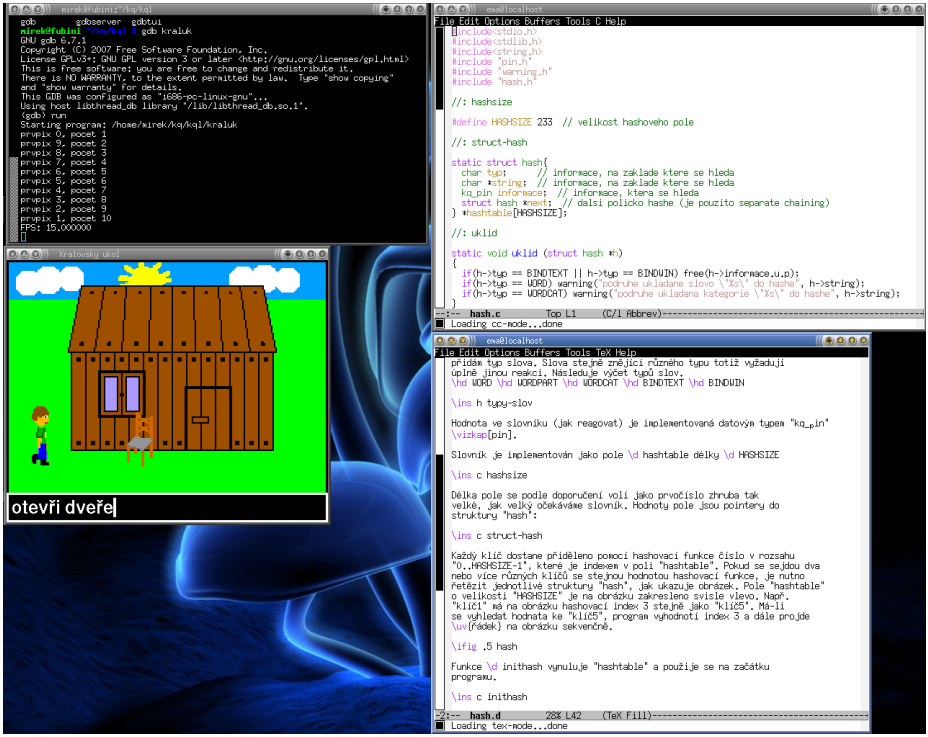
Podobné seznamy stránek vznikají v rejstříku.
- Rejstřík vzniká ze všech dokumentovaných slov zcela automaticky uvnitř TEXu. Tj. bylo potřeba implementovat makrojazykem TEXu abecední třídění.
- Členění na sekce a podsekce a automatické vytvoření obsahu, který je klikací, je samozřejmostí.
- Věc funguje v csplainu, protože já i syn používáme tento formát.

Přiznám se bez mučení, že možnosti současných nástrojů na dokumentování zdrojových kódů jsem moc nestudoval. Je tedy možné, že v některých aspektech znovu vynalézám kolo a že některé vlastnosti, které se běžně při dokumentování používají, chybí. Je to taková „rodinná“ zakázka na míru programu, který syn odevzdává jako ročníkovou práci. Nicméně jsem se snažil nástroj udělat aspoň trochu obecně použitelný a dovolím si jej na konferenci TEXperience 2008 předvést. Posluchači sami posoudí, zda jim to může být k něčemu dobré. Obecnou použitelnost deklaruji například tím, že dokumentaci k DocBy.TEXu (včetně technické dokumentace s rozborem všech maker) píšu v DocBy.TEXu.

Zvažoval jsem aspoň na vteřinu doporučit synovi literární programování, jak jej vytvořil Donald Knuth (tzv. WEB). Ovšem, velmi rychle jsem od toho upustil. Domnívám se, že doba, kdy Knuth vytvářel své dílo, byla poznamenána poněkud jinými počítači, než jaké máme před sebou dnes. Dnes je běžné grafické rozhraní, ve kterém je možné současně otevřít více editorů vedle sebe v různých oknech s různými texty. V jednom třeba může být text určený pro počítač a v druhém text, který je určen pro člověka. Vše je vidět naráz, jak může vidět i čtenář tohoto článku na obrázku 1. Není tedy nutné tyto informace slučovat do jediného souboru, aby byly blízko sebe, a poté je preprocesory (tangle, weave) zase oddělovat. Dnešní programátoři také obvykle přemýšlejí v intencích, jak kód zpracovává kompilátor. Trhání tohoto kódu na úseky (jako v Knuthově WEBu) asi moc nemilují.

Poznamenávám ještě, že jsem Knuthův WEB nezavrhl z neznalosti. Sám jej velmi dobře znám. Vrtal jsem se poměrně do hloubky ve zdrojových kódech TEXu a také jsem napsal jednoduchoučký program vlna v CWEBu. Synův program je ovšem asi stokrát složitější než program vlna, takže rozhodování o CWEBu ani moc dlouho netrvalo. CWEB jsme nepoužili.

Protože syn bude odevzdávat svou ročníkovou práci až v lednu roku 2009, nečekejte nyní hotové dílo. Jeho program je ve stádiu zrodu a DocBy.TEX se také zatím vyvíjí podle požadavků programátora. V době, kdy se snažím aspoň částečně dodržet termín odevzdání tohoto příspěvku (srpen 2008) tedy není DocBy.TEX ve stádiu, kdy bych jej mohl zveřejnit na internetu. Ani ukázek zatím moc není. Syn má naprogramováno v tuto chvíli asi 20 modulů svého programu a bude přidávat ještě mnoho dalších. Dokumentovány DocBy.TEXem má moduly dva. Na obrázcích 2, 3 můžete posoudit, jak vypadá současná verze dokumentace jeho programu. Na obrázku 4 pak vidíte stránku z dokumentace




Obrázek 1: Pracovní plocha s více editory

samotného DocBy.TEXu.

Až DocByTEX dospěje do nějaké rozumně stabilní verze, najdete jej na internetu na `http://www.olsak.net/docbytex.html`.

## 2. Příklad použití DocBy.TEXu

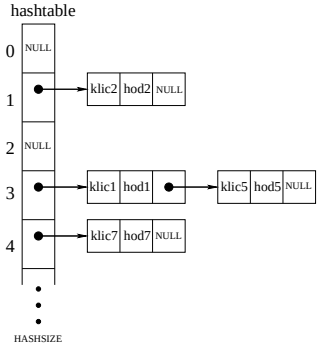
Předpokládá se, že program je členěn na moduly. Každý modul obsahuje skupinu funkcí, která řeší jeden ucelený problém. Předpokládejme teoretický program `lup`, který má tři moduly: main v souboru `main.c`, win v souborech `win.c` a `win.h` a konečně modul base v souborech `base.h` a `base c`. Dokumentace k takovému programu vznikne tak, že autor napíše hlavní TEXový soubor (například `lup.tex`) a z něj bude číst další TEXové soubory, které obsahují dokumentaci k jednotlivým modulům: `main.d`, `win.d` a `base.d`. V `lup.tex` může být nějaký rozverný úvod a dále příkazem „`\module`“ jsou načítány jednotlivé dokumentační soubory k mo-

hash.c
```
10: #define HASHSIZE 233  // velikost hashoveho pole
```

Délka pole se podle doporučení volí jako prvočíslo zhruba tak velké, jak velký očekáváme slovník. Hodnoty pole jsou pointery do struktury `hash`:

hash.c
```
14: static struct hash{
15:   char typ;      // informace, na zaklade ktere se hleda
16:   char *string;  // informace, na zaklade ktere se hleda
17:   kq_pin informace;  // informace, ktera se hleda
18:   struct hash *next;  // dalsi policko hashe (je pouzito separate chaining)
19: } *hashtable[HASHSIZE];
```

Každý klíč dostane přiděleno pomocí hashovací funkce číslo v rozsahu `0..HASHSIZE-1`, které je indexem v poli `hashtable`. Pokud se sejdou dva nebo více různých klíčů se stejnou hodnotou hashovací funkce, je nutno řetězit jednotlivé struktury `hash`, jak ukazuje obrázek. Pole `hashtable` o velikosti `HASHSIZE` je na obrázku zakresleno svisle vlevo. Např. `klíč1` má na obrázku hashovací index 3 stejně jako `klíč5`. Má-li se vyhledat hodnata ke `klíč5`, program vyhodnotí index 3 a dále projde „řádek“ na obrázku sekvenčně.




Funkce `inithash` vynuluje `hashtable` a použije se na začátku programu.

hash.c
```
32: void inithash ()  // Vynuluje hashove pole.
33: {
34:   int i;
35:   for(i=0; i<HASHSIZE; i++){
36:     hashtable[i] = NULL;
37:   }
38: }
```

Funkce `hashfunc` přidělí ke klíči hashovací hodnotu, tj. index do pole `hashtable`. Protože klíč je v našem případě dvojice [typ, slovo], má funkce odpovídající parametry `typ`, `s`.

hash.c
```
42: static int hashfunc (int typ, char *s)  // Spocita cislo ze stringu
43: {
44:   int i, vysledek;
45:
46:   vysledek=typ*5;
47:   for(i=0; (unsigned char)s[i]; i++)
48:     vysledek = (2*vysledek+(unsigned char)s[i])%HASHSIZE;
49:   return vysledek;
50: }
```

---

void inithash(): 2  int hashfunc(): 2–3



Obrázek 2: Ukázka z Mirkovy dokumentace, modul hash

```
44:   }
45:
46:   if(*pom){
47:     cas += (*pom)->zbyva; // cas jsem prestrelil
48:     (*pom)->zbyva -= cas; // upravim zbyva v nasledujicim timeru
49:   }
50:
51:   tim->zbyva = cas;
52:   tim->next = *pom;
53:   *pom = tim;
54: }
55:
56: static void vyradtimer (kq_timer *tim)
57: /*************************************/
58: {
59:   kq_timer **pom;
60:
61:   if(tim->pause) pom = &pausedtimers;
62:   else pom = &prvtimer;
63:
64:   for(; *pom != tim; pom = &(*pom)->next)
65:     if(!*pom){
66:       warning("vyradtimer(): timer nenalezen");
67:       return;
68:     }
69:
70:   *pom = tim->next;
71:   if(tim->next || !tim->pause) tim->next->zbyva += tim->zbyva;
72: }
```

Funkce pausetimer práci se seznamy řeší samostatně. Při vyřazení totiž musí spočítat zbývající čas a zařazení k zapausovaným timerům je příliš snadné na to, aby na to byla volána funkce.

## 4 Rejstřík





Obrázek 3: Ukázka z Mirkovy dokumentace, rejstřík

**DocBy.TEX – nástroj na dokumentování zdrojových kódů**

**Petr Olšák**

## Obsah



## 1 Úvod

DocBy.TEX umožňuje jednoduše dokumentovat pomocí TEXu zdrojové kódy programu v jazyce C. Je možné jej použít i k dokumentování něčeho jiného.

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. Nevidím tedy tak hlasitou potřebu tyto informace slučovat do jednoho souboru, jako tomu bylo kdysi.

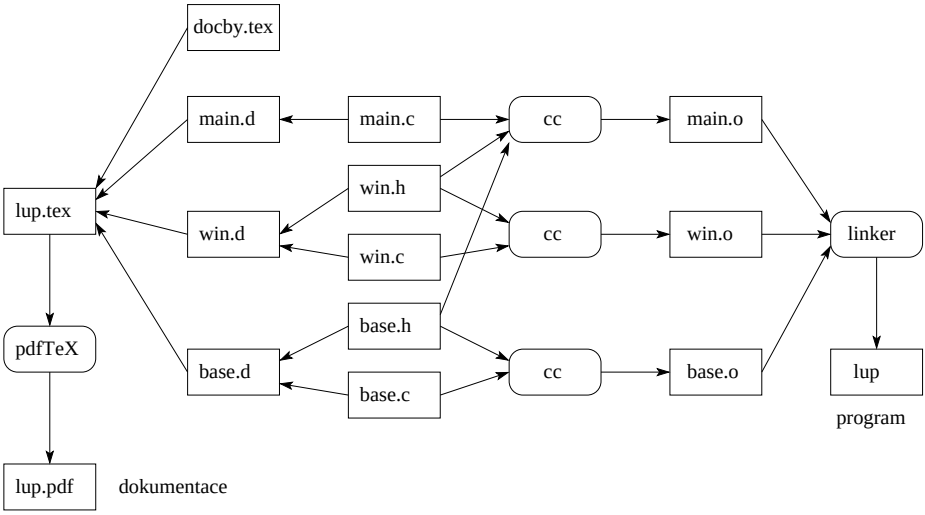
## 2 Dokumentování zdrojových kódů v jazyce C

Předpokládá se, že zdrojové kódy 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.



Obrázek 4: Úvodní strana dokumentace k DocBy.TEXu

dulům. V těchto souborech se autor dokumentace soustředí na problém řešený v daném modulu (něco jako sekce v Knuthově WEBu) a postupně vkládá do dokumentace vymezené úseky kódu ze souborů `*.c` a `*.h` pomocí příkazu „`\ins`“. Dokumentovaná slova vymezuje příkazem „`\dg`“ (dokumentace globálního slova). Vztah mezi jednotlivými soubory při načítání kompilátorem a při načítání TEXem je naznačen na obrázku 5.




Obrázek 5: Vztahy mezi zdrojovými soubory

Obsah souboru `lup.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
\end
```

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 na konci souboru `lup.tex` třeba napsáno:

```
\module main
\module win
\module base
\doindex
\end
```

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.

Podívejme se nyní podrobněji například do souboru `base.d`, kde je soustředěna dokumentace modulu base. Část tohoto souboru může vypadat třeba takto:

```
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
```

Příkazem „`\ins c dvojice`“ jsme dali najevo, že potřebujeme ze souboru se stejným jménem jako je aktuální název modulu (tj. base) a s příponou `.c` přečíst část kódu, který obsahuje deklaraci zmíněné struktury. Pro vymezení částí z načítaného souboru je potřeba do zdrojového kódu vložit vymezující poznámky tvaru `//: slovo`. Příklad části kódu ze souboru `base.c` vše osvětlí:

```
#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;
}
```

Příkaz „`\ins c slovo`“ tedy načte část kódu od vymezující poznámky tvaru `//: slovo` po výskyt poznámky tvaru `//:` nebo až po konec souboru. Do zdrojového kódu je tedy potřeba vložit vymezující poznámky.

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.

Na obrázku 6 můžete vidět zhruba výsledek zpracování našeho příkladu DocBy.TEXem (první stránka). Níže sice píšu „všimněte si“, „za povšimnutí stojí“ atd., ovšem je to potřeba brát s rezervou. Je možné, že tisk obrázku 6 bude jen černobílý a pravděpodobně dost zmenšený. Omlouvám se tedy preventivně čtenáři, pokud ho nutím, aby si všiml, že něco zmodralo nebo zezelenalo, když asi obrázek 6 mu tuto informaci jednoznačně neposkytne.

Všimněme si, že TEX zapsal čísla řádků přesně podle toho, jak jsou ve zdrojovém kódu. Tj. počítal i přeskakovaný řádek `#include <stdio.h>` i přeskakované prázdné a vymezující řádky. Komentářový řádek s vymezením `//: slovo` není do dokumentace zahrnut a pokud následuje za řádkem s vymezením prázdný řádek, ani ten není do dokumentace zahrnut.

Za povšimnutí stojí použití příkazu „`\dg`“. 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.

Všechny Céčkové komentáře ve vloženém zdrojovém kódu automaticky zezelenaly.

Palcové uvozovky `"..."` vymezují kusy kódu uvnitř odstavce. Text takto uvozený je psán strojopisem a pokud se v něm vyskytují dokumentovaná slova, tato slova automaticky modrají. Text mezi těmito uvozovkami je navíc přepisován ve „verbatim“ módu TEXu, tj. žádné znaky nemají speciální vlastnosti (s výjimkou

**Program lup – dokumentace ke zdrojovým textům**

**Progr a Mátor**

## Obsah



## 1 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.

## 2 Modul base

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

base.c

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

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

base.c

```
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: }
```

## 3 Modul win

Hlavním obsahem tohoto modulu je implementace funkce `uzasne_wokno` která založí okno programu `lup`. Návratová hodnota této funkce obsahuje inicializovanou strukturu `win` tohoto okna.

win.c

```
3: win uzasne_wokno ()
4: {
5:   win navrat;
6:   ...
7:   return navrat;
8: }
```

## 4 Modul main

Funkce `main` našeho programu nejprve přečte parametry příkazové řádky, pak v rychlosti spočítá výsledek, přičemž využije funkci `uzasna_funkce`. Nakonec pomocí funkce `uzasne_wokno` zobrazí uživatelsky přívětivým způsobem výsledek.





Obrázek 6: Ukázka ukázkového výstupu

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.

Pozorný čtenář si jistě všiml, že v poznámce pod čarou a v rejstříku je `uzasna_funkce` zmíněna včetně jejího návratového typu a navíc je slovo ukončeno dvojicí závorek `()` a tím je naznačeno, že se jedná o funkci. Je to díky zápisu „`\dg [struct dvojice]uzasna_funkce()`“ v místě dokumentace funkce. Nepovinný parametr s návratovou hodnotou funkce se přepíše před jméno funkce pod čáru i do rejstříku, ale rejstřík při abecedním řazení tento parametr ignoruje.

## 3. Další možnosti

DocBy.TEX umožňuje vyznačit vkládané úseky kódu dalšími způsoby: je možné kontrolovat, zda se vloží nebo nevloží vymezující řádky, nebo jen vymezující řádek na jedné straně, je možné navazovat vkládáním od místa, kde vkládání naposledy skončilo, je možné vložit prototyp funkce, je možné vkládat od libovolně vymezeného místa ve zdrojovém kódu do dalšího vymezeného místa (není tedy nutné vymezovat text jen výše uvedenými poznámkami), je možné vložit jediný řádek s prvním výskytem slova nebo s dalším výskytem slova od místa posledního vložení.

Je možné nastavit jiné vzory, podle kterých zelenají ve vložených souborech komentáře. Implicitně DocBy.TEX považuje za komentář úsek od `//` do konce řádku a úsek mezi `/*...*/`.

V DocBy.TEXu může autor dokumentace založit sekce a subsekce, přitom implicitně příkaz „`\module`“ založí novou sekci. Je možné odkazovat pomocí lejblíků na sekce, subsekce. Je možné vkládat obrázky a odkazovat na ně.

DocByTEX implementuje jmenné prostory, takže je možné dokumentovat slovo lokálně v rámci úseku dokumentace, která vymezuje určitý jmenný prostor. Nastavení jmenných prostorů je poměrně flexibilní a umožňuje na lokálně dokumentované slovo odkazovat globálně při dlouhém výpisu slova včetně specifikace jmenného prostoru.

DocBy.TEX rozlišuje mezi PDF a DVI módem, dále mezi enc a non-enc módem a konečně mezi csplain a plain módem.

Pro zpracování dokumentace je ideální použít pdfTEX s aktivovaným výstupem do PDF (tj. PDF mód) a s encTEXem (tj. enc mód) a s formátem csplain nebo plain (tj. csplain nebo plain mód). V takovém případě fungují barvy, odkazy i automatické tvoření aktivních odkazů ve vkládaném zdrojovém textu.

Není-li aktivován PDF výstup, DocBy.TEX o tom napíše varování na terminál a přejde do DVI módu. V tomto módu nefungují barvy a odkazy nejsou klikací.

Není-li aktivován encTEX, DocBy.TEX přejde do non-enc módu a napíše o tom varování na terminál. V takovém případě dokumentovaná slova ve vkládaném zdrojovém textu automaticky nemodrají a nestávají se klikatelnými odkazy. Ani se jejich výskyty neuvádějí na správných stránkách v rejstříku a v poznámkách pod čarou. Taktéž komentáře automaticky nezelenají.

Při csplain módu generuje DocBy.TEX některé názvy česky. Jedná se o název kapitoly s obsahem, rejstříkem a modulem. V plain módu jsou tyto názvy anglicky. Pracujete-li s jiným jazykem, můžete předefinovat odpovídající makra.

Generování rejstříku i obsahu probíhá v DocBy.TEXu zcela automaticky. Pro vytvoření rejstříku není nutné používat externí program (DocBy.TEX si slova abecedně zatřídí sám). Stačí tedy vložit na požadovaná místa příkazy „`\dotoc`“ a „`\doindex`“. Rejstřík ani obsah nejsou správně vygenerovány po prvním průchodu TEXu. Je potřeba TEXovat dvakrát nebo třikrát. Pro generování obsahu i rejstříku si DocBy.TEX zakládá pomocný soubor s příponou `.ref`, v našem příkladě tedy `lup.ref`.

## Summary: DocBy.TEX – documenting source codes by TEX

DocBy.TEX is a TEX macro software product which gives the possibility to document simply your source codes written in various programming languages, for example written in C. You can include parts of your source code into your documentation. All occurrences of documented words in your included source code are automatically made as active links if encTEX and pdfTEX is in progress. To make PDF output, you need no more than pdfTEX with encTEX. The table of contents and the index are also created automatically. The sorting of the words in the index is implemented at TEX macro level.