Styly pro dokumentaci na KI

Vilém Vychodil

9. května 2014

Abstrakt

Následující text by měl sloužit jako dokumentace k interním dodatečným stylům pro formát LaTeX 2ε používaných k vytváření dokumentace na KI, PřF UP Olomouc. Text je určen jednak pro studenty, kteří chtějí vytvářet dokumentace ke svým ročníkovým a diplomovým pracím, dále je určen všem lidem využívajícím vytvořené hypertextové styly. Dokument po uživateli vyžaduje základní znalosti systému TeX a formátu LaTeX 2ε. První část textu je věnována použitým datovým formátům, při čtení může být bez újmy vynechána. Rovněž závěrečnou část dokumentu věnující se některým aspektům implementace může čtenář vynechat. Autor může být kontaktován prostřednictvím elektronické pošty na adrese <vilem.vychodil@upol.cz>.

1. Používané datové formáty

Společnost *Adobe Systems Incorporated* se zasloužila o vznik několika standardů v oblasti elektronického publikování. Jedná se zejména o jazyk PostScript a přenositelný formát dokumentů – PDF. V 80. letech byl společností Adobe navržen originální obrazový model, na jeho základech byl navržen jazyk PostScript. Jazyk PostScript se rychle rozšířil do grafických aplikací a systémů pro elektronické publikování. V současnosti tvoří de facto standard pro popis stránky. Na počátku 90. let započala společnost Adobe vyvíjet datový formát PDF. Tento formát se PostScriptu v mnohém podobá, ale v mnohém se i liší. Jazyk PostScript a formát PDF jsou pro elektronickou publikaci velmi významné, proto je jim věnován detailnější popis v úvodní kapitole textu. Pokud čtenáře tato problematika nezajímá, může bez újmy přejít na kapitolu 2.

Jazyk *PostScript* je bohatý zásobníkový jazyk vycházející z jazyka FORTH. Samotný jazyk je syntakticky dost jednoduchý, je však vybaven mnoha funkcemi pro zpracovávání grafických dat. Hlavním účelem PostScriptu je popisovat vzhled stránky – obdélníkové oblasti, v níž může být obsažen text, vzory složené z grafických primitiv a dokonce i digitálně vzorkovaná data. PostScript se snaží popisovat stránku nezávisle na výstupním zařízení. Jazyk je plně přizpůsoben na

snadné vytváření složitějších obrazců, disponuje funkcemi pro vykreslování grafických primitiv, operátory měnícími styl vykreslování primitiv a ořezáváním. Jazyk definuje obecný systém souřadnic a disponuje funkcemi pro pohodlné afinní transformace částí stránky. Součástí jazyka PostScript je například i komprese používající Lempel-Ziw-Welchův algoritmus.

PostScript je interpretován programy označovanými zkratkou RIP – *Raster Image Processor*. RIP musí být součástí všech zařízení a software, které chtějí pracovat se stránkami popsanými jazykem PostScript. RIP je například součástí každé tiskárny využívající PostScript jako svůj nativní jazyk, stejně tak jej musejí obsahovat i všechny prohlížeče PostScriptu, například program GhostScript. RIP během interpretace příkazů jazyka PostScript vytváří rastrovou *předlohu stránky.* Rastrová předloha stránky – obdélníková síť bodů udržovaná v paměti je na začátku zpracování vstupního souboru prázdná. Podle interpretovaných příkazů tuto předlohu zaplňuje, respektuje při tom zvolené rozlišení výstupního zařízení. Pokud dojde RIP na příkaz ukončení stránky, dá povel ke zobrazení rastrové předlohy na výstupním zařízení, typicky na tiskárně. Předloha je vyprázdněna a RIP pokračuje další stránkou.

PostScript je jazyk výborně se hodící pro popis statických stránek, například předloh pro tištěné knihy. V praxi bývá PostScript často vytvářen výstupním ovladačem `dvips` typografického systému TEX. Mezi hlavní nedostatky PostScriptu patří z pohledu archivace a použitelnosti v heterogenním síťovém prostředí absence hypertextových odkazů. I když je PostScript v podstatě kompletní programovací jazyk – o tom svědčí například fakt, že existuje i WWW server naprogramovaný v PostScriptu – neumožňuje vytvářet *interaktivní dokumenty.* Navíc interprety jazyka PostScript, ať už hardwarové, či softwarové, nejsou mezi laickou veřejností příliš rozšířeny.

Přenositelný datový formát PDF – *Portable Document Format* začal být vyvíjen kolem roku 1993. Mnohem víc reflektuje moderní potřeby elektronické publikace, než jazyk PostScript. Předurčení datového formátu PDF je totiž od počátku jiné, než v případě jazyka PostScript. Obě dvě technologie ale nejsou „konkurenční“, jejich uplatnění se doplňuje. PDF záměrně neoznačuji jako jazyk, ale pouze jako *datový formát.* Toto označení není bezúčelné. Zjednodušeně řečeno, cílem PDF je umožnit uživateli snadno prohlížet elektronické dokumenty nezávislé na výstupním zařízení a jeho zobrazovacím rozlišení. PDF dokonce používá týž obrazový model jako jazyk PostScript. Jedná se ale o výrazně jednodušší jazyk než je PostScript – tím je usnadněna jeho interpretace. V následujícím výpisu jsou stručně shrnuty hlavní rozdíly proti PostScriptu.

- PDF má *pevně definovánu strukturu,* striktní definice struktury dokumentu umožňuje přistupovat k různým částem dokumentu přímo. V PostScriptu je možný v zásadě jen sekvenční přístup.

- PDF není plnohodnotný programovací jazyk. Neobsahuje základní prvky většiny imperativních programovacích jazyků – například *proměnné, pojmenované procedury* nebo *podmíněné výrazy.* Absence těchto prvků zjednodušuje interpretaci PDF.
- PDF definuje sadu *objektů,* které přímo nesouvisejí s obrazovým modelem, ale mají význam při navigaci v dokumentu. Sem patří například objekty umožňující vytvářet *návěstí* a *odkazy.* Rovněž sem patří například i objekty definující *logickou strukturu dokumentu.*
- Soubor ve formátu PDF může být přeložen do jazyka PostScript. Pro úspěšný překlad v podstatě stačí nahradit zjednodušené sekvence vykreslovacích příkazů formátu PDF příkazy jazyka PostScript. Rovněž je nutné provést do výsledného PostScriptu zařazení patřičných fontů, které jsou popsány ve vstupním PDF souboru.

Formát PDF má řadu výhod. Mezi jeho přednosti patří *nezávislost na výstupním zařízení.* Tento požadavek je velmi podstatný například při sazbě matematických textů nebo textů obsahujících složitější symboliku, která by mohla být v případě jiného formátu prohlížečem deformována nebo zkreslena. PDF disponuje kromě komprese i šifrovacími metodami. Hlavní výhodou PDF proti PostScriptu je sada speciálních objektů, které mimo jiné umožňují navigaci v dokumentu na úrovni kapitol, stránek i řádků textu.

Společnost Adobe spolu s formátem PDF vyvíjí rodinu nástrojů *Acrobat.* Nejdůležitějším z této rodiny je program *Acrobat Reader* – volně dostupný prohlížeč souborů ve formátu PDF disponující i možnostmi tisku. Program Acrobat Reader je masivně rozšířen, v současnosti prakticky všechny firmy zabývající se vývojem hardware a software používají pro vytváření dokumentací a manuálů ke svým produktům právě formát PDF. I z tohoto důvodu je nasazení PDF při vytváření elektronické dokumentace výhodné.

Zpracování dokumentu systémem TEX

TEX je typografický systém vyvinutý kolektivem odborníků soustředěných kolem Donalda E. Knutha. Knuth měl záměr vytvořit vysoce přesný, přenositelný a nezávislý typografický systém. Pro vývoj TEXu byl navržen vlastní programovací jazyk Web. Typografický systém TEX je navržen opravdu velmi ambiciosně. Například při implementaci vnitřní aritmetiky TEXu bylo přísně dbáno na to, aby byly veškeré odchylky ve výpočtech menší než je vlnová délka světla. Tím je dosaženo absolutní nezávislosti na výstupním zařízení. Lze jen stěží očekávat, že se nějakému výstupnímu zařízení podaří této rozlišovací schopnosti dosáhnout. TEX je rovněž unikátní v práci s fonty – umožňuje používat v podstatě libovolný standard fontů díky oddělení metrik fontů od popisu tvaru jednotlivých znaků fontu.

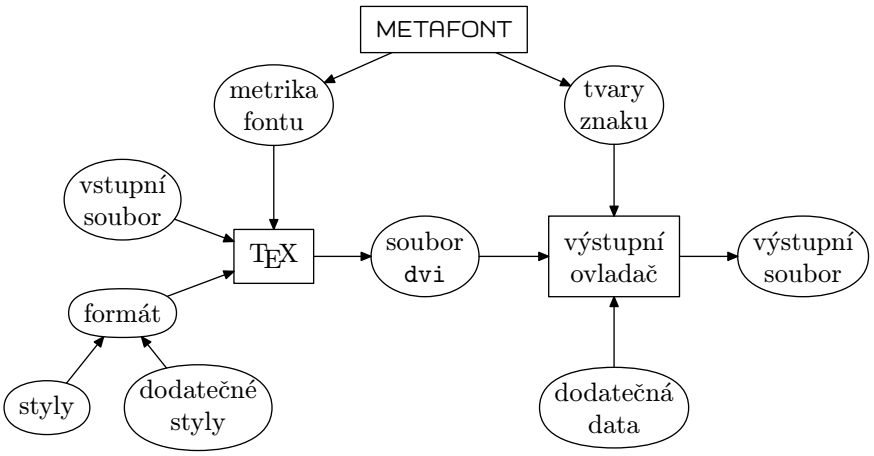
TEX pracuje jako *formátovač*. Zpracovává vstupní soubor skládající se z textu, dodatečných maker a primitiv. Jádrem TEXu je program `virtex`. Název programu `virtex` – *Virgin TEX* plně charakterisuje jeho základní povahu. Program `virtex` interpretuje asi jen 300 základních primitiv. V této podobě téměř nikdo TEX nevyužívá. Sazba jen pomocí základních primitiv by byla velmi náročná. Při zpracování vstupního dokumentu formátovačem se zpravidla využívá jistý *formát*. Formát sestává z definic nových maker usnadňujících práci se systémem. Během své činnosti `virtex` expanduje tato makra na primitivy a potom je interpretuje. Mezi nejznámější formáty patří plainTEX a LATEX. Aby byly formáty rychle dostupné, jsou uchovávány v částečně zpracovaném stavu v souborech s příponou `.fmt`.

Jednotlivé formáty jsou zpravidla vybaveny *styly* – dodatečnými soubory definic, které upravují chování formátu při potřebě sázet odlišné typy dokumentů, třeba články, knihy, dopisy a podobně. Ke stylům mohou být dodávány i *dodatečné styly*. Úkolem dodatečných stylů je většinou rozšířit možnosti stylů. Například formát LATEX obsahuje styly article, book, letter, a další. Dodatečné styly v LATEXu například upravují standardní velikost stránky, nebo přidávají nová makra podporující sazbu speciálních objektů, třeba tabulek. Styly a dodatečné styly jsou ukládány v samostatných zdrojových souborech.

Během formátování dokumentu potřebuje `virtex` kromě vstupního dokumentu, formátu a stylů rovněž informace o fontech. Samotné formátování v podstatě spočívá v rozvržení jednotlivých znaků na stránku podle patřičných nastavení velikosti zrcadla a podle nastavení řádkového a stránkového zlomu. V této fázi překladu není nutné manipulovat s fonty na úrovni „vykreslování znaků“, `virtex` používá pouze *metriky fontů*. Metriky fontů obsahují pro každý znak, nebo skupiny znaků z fontu informace o jejích velikosti. Skupiny znaků tvoří zpravidla slitky – ligatury. Metriky dále obsahují tabulky pro podřezávání dvojic znaků – kerningové tabulky. Rozdělení fontů na metriky a vlastní popis tvarů umožňuje TEXu pracovat s libovolnými fonty. Celou situaci vystihuje obrázek 1.

Výstupem formátovače je soubor *nezávislý na výstupním zařízení*. Soubory mají příponu `.dvi` – *Device Independent*. Datový formát DVI je velmi jednoduchý a slouží k popisu stránky. DVI má jen několik základních instrukcí, je zcela oproštěn například od příkazů sloku, nebo od podmíněných výrazů. Zjednodušeně řečeno, v DVI souborech jsou uloženy v podstatě jen polohy obdélníkových rámců. Každý rámec zastupuje polohu jednoho znaku ve výsledném dokumentu.

Soubor ve formátu DVI je dále konvertován *výstupním ovladačem* do některého z používaných formátů elektronických dokumentů, případně do datového formátu vhodného pro tisk. Typicky je používán program `dvips` pro konversi do jazyka PostScript. Výstupní ovladače ke své činnosti potřebují mít informace o tvarech znaků jednotlivých fontů. Jelikož je ale výstupní ovladač oddělen od samotného formátovače, může používat libovolné druhy fontů, od bitmapových



Obrázek 1. Činnost TEXu.

fontů generovaných programem METAFONT až po PostScriptové Type 1 fonty a jiné. Během formátování se mohou do DVI souboru přidávat i informace specifické pro výstupní ovladač. Způsob naložení s těmito informacemí je již plně v kompetenci výstupního ovladače. Typickým případem použití dodatečných informací je třeba vkládání obrázků do dokumentu.

V následujícím textu jsou podrobněji porovnány způsoby generování PDF výstupů pomocí systému TEX. Vytvořené dodatečné styly umožňují vytvářet PDF soubory v součinnosti s výstupním ovladačem `dvipdfm`. Toto řešení je technicky nejjednodušší a jeho návrh je v souladu se samotnou filosofií typografického systému TEX. Vytvořené styly lze samozřejmě použít i k vytvoření PostScriptu výstupním ovladačem `dvips`.

Systém TEX a datový formát PDF

Současnosti existují dva projekty umožňující využít TEX k vytváření PDF dokumentů. Prvním z nich je systém pdfTEX, jehož autorem je Hàn Thế Thành. Tento projekt jde směrem rozšíření současných možností samotného formátovače. Jelikož pdfTEX rozšiřuje formátovač, musí být nejprve přeložen jako samostatný program. Ve většině distribucí TEXu je již pdfTEX ve své binární podobě, větším problémem ale je, že musí mít vytvořeny vlastní formáty. Hlavní úskalí použití pdfTEXu je právě v udržování formátů. Zkušenější čtenář již jistě někdy narazil na distribuci TEXu, ve které se týž zdrojový dokument odlišně zalomil při použití klasického TEXu a pdfTEXu. Rovněž udržování dvou různých stylových souborů není příliš šťastné.

Další možností vytváření PDF souborů pomocí typografického systému TEX je výstupní ovladač `dvipdfm`. Autorem programu je Mark A. Wicks. Jedná se o výstupní ovladač, který z přeloženého DVI souboru generuje PDF. Program

navíc umožňuje interpretovat speciální příkazy v DVI a vkládat podle nich do vytvářeného PDF souboru speciální objekty a příkazy. TEX disponuje primitivem `\special`. Jediným úkolem `\special` je vložit během formátování do výstupního DVI souboru dodatečné informace. Dodatečné informace umožňují výstupnímu ovladači `dvipdfm` určovat logickou strukturu dokumentu, vytvářet odkazy a podobně.

Výsledný DVI soubor lze samozřejmě zpracovat i jiným výstupním ovladačem, například `dvips`. Tento ovladač všechny neznámé dodatečné informace – třeba informace pro `dvipdfm` jednoduše ignoruje. V tomto případě je tedy možné vytvořit pouze jeden styl a využít jej při tvorbě jak PostScriptu, tak PDF. Navíc je zaručeno, že oba dva dokumenty budou vypadat identicky. Jelikož výroba PDF probíhá až na úrovní výstupního ovladače, stačí do systému nainstalovat v podstatě jen jeden program a není třeba generovat další formáty. Program `dvipdfm` je navíc šířen pod licencí GNU GPL zaručující jeho dostupnost.

Třetím způsobem výroby PDF souborů ze zdrojů v TEXu je využití programu *Adobe Distiller*. Když přehlédneme že jde o komerční platformově závislý produkt, proti jeho použití hovoří sama filosofie PDF. Program Distiller vytváří PDF z dokumentů v jazyku PostScript. Při použití s TEXem tedy musí být nejprve vytvořen PostScriptový soubor. V souboru musejí být navíc speciálními makry vytvořeny dodatečné informace například o odkazech. PostScript je ale dost komplikovaný jazyk, výrazně složitější než PDF. Naproti tomu DVI je svou strukturou formátu PDF velmi příbuzné. Jedná se rovněž o velmi jednoduchý datový formát. Z tohoto důvodu je generování PDF souborů přímo z DVI mnohem přímočařejší.

Využití TEXu a PDF při vytváření textů

Tato kapitola stručně popisuje možnosti PDF objektů, které lze využít pomocí programu `dvipdfm`. Pro stručnost jsou uvedeny pouze třídy PDF objektů uplatnitelné při výrobě hypertextové dokumentace. Pro detailní popis možností datového formátu PDF odkazuji čtenáře na jeho specifikaci [2].

K navigaci *na úrovni dokumentu* lze v PDF použít *záložky*. Záložky tvoří stromovou strukturu, každé jméno v záložce může být asociováno s cílovým místem v dokumentu. Pokud PDF dokument obsahuje záložky, jsou prohlížečem zobrazeny na levé straně textu. Záložky se v dokumentu používají obvykle místo obsahu nebo jako jeho doplněk. Jelikož jsou záložky neustále viditelné, může uživatel plynule přecházet mezi kapitolami bez nutnosti přejít k obsahu, který je umístěn zpravidla na začátku dokumentu. Záložky lze během práce skrývat – není nutné mít zobrazeny všechny položky najednou. Záložky nemusejí být použity jen pro vymezení kapitol. Ve vytvořených makrech lze vytvářet záložky i explicitně. Při definici cílového místa záložky lze v PDF určit stranu v dokumentu, polohu cílového místa i zvětšení zobrazeného výřezu.

Z pohledu vytváření odkazů patří mezi nejdůležitější PDF objekty anotace. *Anotace* umožňují sdružit PDF objekty, například poznámky, zvuky, video, odkazy a jiné s akcemi. *Akce* jsou pomocí anotací navázány na své aktivační objekty. Jednou z anotací je i *link* – odkaz. Na odkaz bývá zpravidla navázána akce *goto* – přesun do jiné části dokumentu. Část dokumentu může být definována buďto absolutně, nebo relativně. Části dokumentu je možné i pojmenovat a odkazovat se na ně jejich jmény. Na odkaz může být také navázána akce spouštějící externí program nebo odkazující se na externí dokument pomocí URL. Všechny typy těchto akcí jsou v implementaci stylů využity.

Formát PDF disponuje i *informačními objekty.* Tyto objekty slouží k specifikaci informací o souboru, jeho autorovi a původu. Pro každý dokument lze rovněž definovat sadu charakteristických klíčových slov. Ačkoliv nejsou tyto elementy z pohledu vytváření hypertextových dokumentů životně důležité, nalézají uplatnění například při archivaci většího množství dokumentů a zejména při jejich rychlém prohledávání.

Mezi základní anotace, které nejsou v implementaci obsaženy, patří *textová anotace.* Jedná se o poznámku, která je na straně umístěna v podobě ikony. Tuto ikonu lze rozvinout do podoby poznámky. Hlavní nevýhodou je nejednoznačnost kódové stránky použité v poznámce. Ve specifikaci PDF je uvedeno, že obsah poznámky je zobrazen fontem a kódováním určeným prohlížečem. Toto omezení prakticky znemožňuje presentovat v poznámkách české texty, prohlížeče standardně používají kódovou stránku ISO 8859 1, která neobsahuje českou diakritiku. Textové anotace rovněž nelze tisknout. Mezi další nepodporované objekty patří *formuláře.* Jedná se o podobné formuláře jako jsou v jazycích HTML a XHTML. Na formuláře je možné navázat akci v podobě URL, na nějž se odešlou data z formuláře. Rovněž je možné zpracovávat data z formulářů obslužným JavaScriptem.

2. Dokumentace ke stylům

Při vytváření hypertextových stylů byly v první řadě aktualisovány již existující styly vytvořené na katedře informatiky pro formát LaTeX. Jedná se o styly pro psaní dokumentací k *ročníkovým pracím, diplomové práci* a styl pro vytváření *reportů.* K těmto stylům byly připojeny další dva – styl pro psaní *skript* a dodatečný *matematický styl.*

U všech již existujících stylů bylo dbáno na dodržení jejich dosavadní visuální podoby, rozměru zrcadla a struktury dokumentu. Mírně se však změnila syntaxe jejich maker. Pro všechny styly je k disposici sada okomentovaných ukázkových příkladů demonstrujících jejich použití. Bez újmy je lze použít jako kostru při vytváření nových dokumentů. Dosavadní styly byly při úpravě rovněž aktualisovány. Staré styly byly vyrobeny pro dnes již zastaralý LaTeX 2.09. Nové styly jsou

určeny pro LaTeX 2ε. V neposlední řadě nové styly upravují nevhodné typografické konvence používané standardními styly formátu LaTeX.

Základní požadavky na styly

Při implementaci stylů bylo respektováno několik požadavků zaručujících maximální využitelnost stylů při zachování typografické úrovně a přístupnosti široké veřejnosti uživatelů. V následujícím soupisu jsou uvedeny stěžejní z nich.

1. Vytvořené styly jsou *dodatečné styly* ke stylu article formátu LaTeX. Formát LaTeX neklade narozdíl od formátu plainTeX tak velké nároky na uživatelovu znalost samotného systému TeX. Makra formátu LaTeX se v podstatě snaží skrýt před uživatelem samotnou úpravu dokumentu, v LaTeXu je prioritní definice struktury dokumentu. V neposlední řadě se jedná asi o nejpoužívanější formát TeXu vůbec.

2. Styly jsou dostatečně *konservativní* – uživatel by jejich použití na své práci neměl poznat. Samozřejmě, že styly obsahují i nová makra, ale pokud je uživatel nepoužije, pracuje v podstatě jako s klasickým LaTeXem. Například odkazy mezi kapitolami pomocí maker `\label` a `\ref` fungují jako doposud, navíc však do dokumentu vkládají i hypertextové odkazy.

3. Styly přidávají do DVI souboru instrukce, kterými jsou pomocí `dvipdfm` vytvářeny speciální PDF objekty. Vkládané instrukce nijak neomezují činnost jiných výstupních ovladačů, například `dvips`. Z jednoho zdroje lze tedy vytvářet přímo PostScriptový výstup i PDF výstup.

4. Všechny styly umožňují uživatelům používat *obecnou sadu maker* pro vytváření hypertextových odkazů mezi částmi dokumentu i vně dokumentu. Pomocí speciálních maker lze definovat návěstí na úrovni jednotlivých řádků dokumentu. Na tato návěstí se lze odkazovat. Tyto odkazy mají v dokumentu tvar vysvíceného textu. Při tisku dokumentu nebo při vytvoření PostScriptového výstupu jsou vidět jako klasický text.

5. Chování stylů lze korigovat pomocí přepínačů předávaných při jejich inicialisaci. Přepínače lze například ovlivňovat zobrazování některých částí z úvodu dokumentu. Jiné argumenty slouží k ovlivňování velikost nadpisů a podobně.

6. Styly rovněž opravují nestandardní typografické konvence vyskytující se implicitně v české formě LaTeXu. Jde zejména o nevhodné nastavení nulové odstavcové zarážky v prvním odstavci kapitoly. V LaTeXu je z pohledu českých dokumentů rovněž nevhodné nastavení číslování kapitol. Oba dva problémy jsou ve všech stylech odstraněny.

7. Styly umožňují vkládat do dokumentů obrázky. Formáty PostScript a PDF ale podporují rozdílné datové formáty. Pokud chce uživatel použít jeden vstupní soubor pro vytváření PostScriptu i PDF, musí používat jen jisté metody jejich výroby. Jednou z možností je použít velmi jednoduché PostScriptové obrázky, nebo obrázky vytvářené v jazyku METAPOST. Další možností je konverse bitmapových obrázků do METAFONTu.

V dalších kapitolách je popsáno základní použití stylů včetně popisu dodatečných maker. Některá makra a prostředí je možné používat jen v některých stylech, jiná jsou k disposici v každém stylu. Popis obou dvou tříd maker je rozdělen do samostatných kapitol.

2.1. Používání stylů

Nyní je k disposici šest dodatečných stylů. Čtyři z nich jsou určeny pro přímé použití. Jeden je určen pro vytváření obecných dokumentů. Poslední styl je *matematický styl* a měl by být používán spolu s některým z ostatních stylů. Matematický styl je specifický, nedefinuje žádná záhlaví stránek a podobně, ale obsahuje pouze definice maker na jednodušší odkazování na číslované vztahy a definice prostředí pro sazbu definice, vět a podobně. Všechny základní styly se liší svým nasazením, to se promítá do jejich vzhledu i vnitřní struktury. V následujícím seznamu jsou shrnuty základní markanty všech stylů.

uproject Styl určený pro dokumentaci k *ročníkovým projektům.* Mezi hlavní rysy stylu patří jeho jednostranný charakter. Styl nelze používat s přepínačem `twoside`. Úvodní strany stylu obsahují informace o autorovi, ročníku, datu a názvu ročníkové práce. Další strana je vyhrazena pro abstrakt. Dále následuje obsah a vlastní text. Při použití stylu uproject by neměly být vytvářeny dodatky, rovněž není možné vytvářet rejstřík pojmů. Na závěr textu by měly být uvedeny reference na použitý materiál.

updiplom Styl určený pro psaní *diplomové práce.* Možnosti stylu jsou již bohatší než v případě stylu uproject. Samostatná strana je věnována místopřísežnému prohlášení. Dále následuje anotace práce a poděkování. Za nimi je umístěn obsah a vlastní text. Za koncem textu následuje závěr v češtině a závěr v angličtině, dále pak reference na použitý materiál. V další části mohou být dodatky a nakonec rejstřík.

upreport Styl určený pro vytváření *reportů* a krátkých dokumentů. Od předchozích dvou stylů se upreport liší především tím, že je určen o pro oboustranný tisk. Stránky dokumentu nejsou číslovány pouze číslicí na spotu stránky, ale každá strana textu má záhlaví s číslem strany a aktuální kapitolou. Za úvodní stranou může být abstrakt spolu s dodatečnými informacemi a kontaktem na autora. Za vlastním textem může být seznam literatury, dodatky a rejstřík.

upbook Styl určený pro vytváření *skript* a delších textů. Svou strukturou je velmi podobný stylu upreport. V úvodní straně je nadpis implicitně sázen versálkami. Záhlaví stránek je podtrženo čarou. Matematické výrazy jsou číslovány v rámci kapitol, nikoliv v rámci celého dokumentu. Některé z těchto vlastností lze upravit vhodnými přepínači, viz další text.

upsimple Všestranný styl určený pro vytváření jakékoliv hypertextové dokumentace. Styl nemá definovánu žádnou pevnou strukturu, uživatel pomocí něj může vytvářet libovolně strukturované dokumenty. Některé prvky běžné v ostatních stylech, například automatické generování záložek, je potřeba ve stylu upsimple nejprve povolit speciálními přepínači.

upmath Matematický styl určený pro použití s předcházejícími styly. Styl definuje makra pro odkazování na číslované vztahy. Dále definuje několik standardních prostředí pro sazbu číslovaných definic, vět, lemmat, důkazů, důsledků, algoritmů a podobně.

Dodatečné styly jsou určeny pro LaTeX 2ε a měly by být použity se standardním stylem article. V případě jejich použití s jiným stylem, například report, není zaručena jejich úplná funkčnost. S největší pravděpodobností by přestalo fungovat generování záložek. Pokud by se uživateli zdálo používání stylu article jako příliš omezující, může si podle dokumentace z kapitoly 3. patřičně styly upravit.

Při použití stylů je nutné inicialisovat formát makry ve tvaru

```
1 \documentclass{article}
2 \usepackage[◁seznam přepínačů▷]{◁název stylu▷}
```

kde ◁*název stylu*▷ je jeden z názvů dodatečných stylů, například upbook. Argument ◁*seznam přepínačů*▷ je i s hranatými závorkami nepovinný. Pokud je uveden, musí obsahovat názvy přepínačů oddělené čárkami. Přepínače slouží k počátečnímu nastavení dodatečného stylu. Kromě zvoleného dodatečného stylu většinou již není nutné vkládat žádný další styl. Dodatečný styl sám importuje makra z jiných stylů. Mezi stěžejní z nich patří například styly czech, epsf a styly pro využití maker $\mathcal{AMS}$-LaTeXu. Výjimku tvoří pouze styl upsimple, který z pochopitelných důvodů neimportuje žádné dodatečné styly.

Pomocí přepínačů může uživatel stylu částečně ovlivňovat jeho chování. Styly jsou vybaveny sadou přepínačů, některé z nich jsou použitelné ve všech stylech, jiné se vážou na konkrétní styl. Použití přepínačů pro jednotlivé styly je přehledně zobrazeno v tabulce 1. Následující popis uvádí jména všech přepínačů a jejich význam.

czech Přepínačem czech se zapínají ve stylu upsimple české typografické konvence. Standardně jsou použity konvence formátu LaTeX 2ε. Pokud je přepínač uveden, dojde ke změně stylu číslování kapitol a všechny úvodní

Tabulka 1. Možnost použití přepínačů ve stylech.

přepínače	vytvořené styly				
	uproject	updiplom	upreport	upbook	upsimple
czech					×
figures	×	×	×	×	
index					×
joinlists	×	×	×	×	
master		×			
nopdf	×	×	×	×	
noseceqn				×	
outlines					×
seceqn			×		
smalltitle				×	
tables	×	×	×	×	

odstavce kapitol budou mít nastavenu odstavcovou zarážku. V ostatních stylech jsou tuto konvence nastaveny automaticky.

`figures` Pokud je přepínač uveden, je za obsahem dokumentu vytištěný i seznam obrázků. Kromě seznamu obrázků může dokument obsahovat i seznam tabulek, viz přepínač `tables`.

`index` Přepínač je určen pro aktivaci rejstříku ve stylu upsimple. Pokud není přepínač uveden, uživatel může používat klasický rejstřík ze systému LaTeX 2_ε, který ovšem není hypertextový. Pokud uživatel nechce rejstřík využívat vůbec, nemusí přepínač uvádět. V ostatních stylech pracujících s rejstříkem nemá tento přepínač smysl.

`joinlists` Je-li přepínač uveden, bude seznam obrázků a tabulek uveden na jedné straně. Implicitně jsou oba seznamy odděleny stránkovým zlomem. Přepínač se uplatňuje pouze v případě, jsou-li současně uvedeny přepínače `figures` a `tables`.

`master` Přepínač je určen pro styl updiplom a zajišťuje vysázení správného označení diplomové práce na titulní straně v případě magisterské práce. Implicitně je vysázeno označení pro bakalářskou práci.

`nopdf` Přepínač potlačující vkládání speciálních příkazů pro výstupní ovladač `dvipdfm`. Pokud uživatel nechce pomocí stylů vytvářet PDF soubory, může v případě potíží s výstupním ovladačem použít tento přepínač.

`noseceqn` Ve stylu upbook jsou standardně číslovány vztahv v matematickém prostředí ve tvaru dvojice $(x.y)$, kde x je číslo kapitoly a y je číslo vztahu v rámci kapitoly. Uvedením přepínače `noseceqn` se zapne jednoduché číslování vztahů v rámci celého dokumentu.

`outlines` Přepínač je určen pro styl upsimple a je-li uveden, jsou v dokumentu automaticky s každou kapitolou generovány i záložky. V opačném případě nejsou záložky generovány automaticky a uživatel je musí vytvářet pouze explicitně. Pro zobrazení záložek je ve stylu upsimple navíc nutné použít makro `\displayoutlines`. V ostatních stylech nemá přepínač `outlines` význam.

`seceqn` Styl upreport je koncipován pro kratší texty než upbook, proto jsou v něm matematické vztahy číslovány v rámci celého dokumentu. Přepínačem `seceqn` lze zapnout číslování v rámci kapitol.

`smalltitle` Ve stylu upbook je standardně první řádek nadpisu sázen versálkami. Pokud si tak uživatel nepřeje, může použít tento přepínač. Druhý řádek není sázen versálkami ani v případě, že `smalltitle` není uveden.

`tables` Přepínač má podobný význam jako přepínač `figures`. Pokud je uveden, za obsah a případný seznam obrázků je vytisknut i seznam tabulek. Pokud není uveden přepínač `joinlists`, seznam tabulek je uveden na samostatné straně.

Dokumenty využívající předchozích stylů se překládají stejným způsobem, jako klasické dokumenty psané v LaTeXu. Pokud dokument obsahuje křížové odkazy, je někdy nutné přeložit zdrojový soubor víckrát dokud nedojde k navázání všech odkazů. Bezprostředně pro vytvoření DVI souboru je možné vytvořit z něj PostScript programem `dvips`. Překlad dokumentu může vypadat následovně.

```
cslatex ◁název▷.tex
dvips -t a4 -o ◁název▷.ps ◁název▷.dvi
```

Argument ◁*název*▷ označuje název *vstupního souboru.* Při volání programu `dvips` byla v ukázce použita explicitní definice výstupního média – papír formátu A4. Za argumentem `-o` se uvádí název výstupního souboru. Argument `-o` je vhodné uvádět vždy. V některých operačních systémech by mohlo místo vytvoření výstupního souboru dojít k tisku dokumentu.

Při vytváření výstupního PDF souboru je nutné po vytvoření DVI souboru ještě spustit jeden externí program. Generované záložky na levém okraji dokumentu nesmějí obsahovat diakritiku. Ve specifikaci PDF je u záložek uvedeno, že jejich zobrazení je plně v režii prohlížeče. Prohlížeč si k zobrazení záložek volí font a bohužel i kódovou stránku. V současnosti všechny prohlížeče volí západoevropskou kódovou stránku ISO 8859 1, která neumožňuje zobrazovat české

znaky. Dokud nebude tato patologická situace vyřešena ve specifikaci PDF, je nutné nedostatek vyřešit odstraněním diakritiky ze záložek.

Primitivum `\special` pracuje na úrovni *expand procesoru* systému TEX. Veškeré mě známé mechanismy pro překódování znaků pracují v TEXu až na úrovni *hlavního procesoru,* k překódování záložek je tedy nelze použít. Vytvořil jsem jednoduchý platformově přenositelný externí program `outlines` odstraňující diakritiku ze záložek. Program funguje pro nejčastěji používané kódové stránky ISO 8859-2 a CP 1250. Překlad dokumentu do PDF vypadá následovně.

```
cslatex ◁název▷.tex
outlines ◁název▷.dvi
dvipdfm -p a4 -o ◁název▷.pdf ◁název▷.dvi
```

Argument ◁*název*▷ má význam jako v předchozím případě. Je-li program `outlines` zavolán s DVI souborem, najde v něm všechny speciální příkazy definující začátky záložek a odstraní z jejich popisů diakritiku. Program `outlines` je včetně svých zdrojových kódů součástí distribuce maker.

2.2. Obecně použitelná makra

Tato kapitola popisuje sadu maker, která jsou dostupná ve všech uvedených stylech. Při jejich použití není uživatel nijak omezen. Makra jsou pro přehlednost rozdělena do několika kategorií. Zjednodušeně lze makra rozdělit na *výkonná* a *nevýkonná.* Nevýkonná makra slouží ke vkládání informací o dokumentu, výkonná makra přímo ovlivňují vytváření interaktivních PDF objektů ve výsledném dokumentu.

Informace o dokumentu

`\docinfo{`◁*autor*▷`}{`◁*titul*▷`}`

Toto makro slouží ke vložení dodatečných informací o dokumentu. Dodatečné informace nejsou během prohlížení dokumentu přímo zobrazovány, ale jsou součástí popisu PDF souboru. Informace jsou využívány například vyhledávacími službami nebo při archivaci dokumentů. Ve stylu upsimple může být makro `\docinfo` použito v libovolné části dokumentu. V ostatních stylech by mělo být použito před výskytem makra `\maketitle`, jinak ztrácí svůj efekt.

Argumenty ◁*autor*▷ a ◁*titul*▷ by měly být použity bez diakritiky.

```
3 \docinfo{Jan Novak}{Moje kniha}
```

Interní a externí odkazy

Pro odkazování v rámci dokumentu lze použít i standardní makra LaTeXu. Dosavadní makra neztrácejí svůj význam. Návěstí je definováno makrem `\label`. Na návěstí se lze odkazovat makry `\ref` a `\pageref`. Makro `\ref` vypíše odkaz sekce, příkladu, obrázku či jiného objektu svázaného s daným návěstím. Makro `\pageref` vypisuje číslo strany na níž je objekt vysázen. Odkaz na bibliografii lze provést makrem `\cite`.

U standardních maker se odkazy vytvářejí transparentně. Například uvedením dvojice `\label`, `\ref` se *stejným návěstím,* bude ve výsledném dokumentu vytvořen hypertextový odkaz. Navíc je zachován standardní vzhled odkazu. Pro vytváření zvýrazněných odkazů v rámci dokumentu a vně slouží následující sada maker.

`\hyplabel{`◁*návěstí*▷`}`

Makro slouží k definici hypertextového návěstí na úrovni řádku v dokumentu. Na hypertextové návěstí se lze odkazovat pouze pomocí makra `\emphref`, v žádném případě jej nelze použít při odkazování makry `\ref` a `\pageref`.

Argument ◁*návěstí*▷ určuje jméno návěsti a neměl by kolidovat s žádným existujícím hypertextovým návěstím, ani návěstím definovaným pomocí `\label`.

```
4 \hyplabel{nejaky_pojem}
```

`\emphref{`◁*text*▷`}{`◁*návěstí*▷`}`

Pomocí makra `\emphref` se lze odkazovat na návěstí definovaná buďto makrem `\hyplabel`, nebo makrem `\label`. Ve srovnání s makry `\ref` a `\pageref` je při použití `\emphref` nutné definovat i samotný text, jenž je součástí odkazu. Odkazy vytvářené pomocí `\emphref` jsou velmi podobné odkazům známým z jazyka HTML.

Argument ◁*text*▷ určuje zvýrazněný text, který je součástí hypertextového odkazu – na text je možné kliknout. Argument ◁*návěstí*▷ určuje jméno návěstí definované buďto makrem `\hyplabel`, nebo makrem `\label`.

```
5 Odkaz \emphref{na pojem}{nejaky_pojem}.
```

`\link{`◁*text*▷`}{`◁*adresa*▷`}`

Makro `\link` vytváří obecný odkaz vně dokumentu. Jeho argumenty jsou obdobné jako u makra `\emphref`. Druhý argument ale není lokální návěstí, nýbrž URL adresa. Interpretace URL je plně v režii prohlížeče, proto je vhodné definovat URL adresy v co možná nejúplnějším tvaru. Při definici by rozhodně nemělo chybět označení protokolu a úplné cesty.

Argument ◁*text*▷ určuje zvýrazněný text, který je součástí hypertextového odkazu – na text je možné kliknout. Argument ◁*adresa*▷ určuje cílové URL.

```
6 Odkaz na \link{dokument na síti}{http://www.gnu.org}.
```

`\url{`◁*adresa*▷`}`

Makro \url zjednodušuje používání makra \link v případě, kdy chce autor uvést jen samotné URL bez dalšího komentáře. V dokumentu je URL vysazeno strojopisem, například http://www.gnu.org.

Argument ◁*adresa*▷ určuje cílové URL.

```
7 Například \url{http://www.gnu.org}.
```

`\mail{`◁*adresa*▷`}`

Pomocí makra \mail lze zjednodušit vytváření kontaktu pomocí elektronické pošty. Makro využívá služeb makra \link. Jediným argumentem je adresa ve tvaru *uživatel@hostitel.* V dokumentu je adresa vysazena strojopisem a ohraničena znaky „<“ a „>“, například <vilem.vychodil@upol.cz>.

Argument ◁*adresa*▷ je adresa tvaru *uživatel@hostitel.*

```
8 Moje adresa je \mail{vilem.vychodil@upol.cz}.
```

Vytváření záložek

Záložky jsou v dokumentu generovány automaticky s každou *číslovanou kapitolou.* Rovněž některé významné nečíslované části dokumenty, například *bibliografie* nebo *rejstřík,* jsou přidávány do záložek. Kromě automaticky generovaných záložek může uživatel do dokumentu přidávat i vlastní. K vytváření záložek slouží generické makro \insertoutline.

`\insertoutline{`◁*úroveň*▷`}{`◁*text*▷`}`

Makrem \insertoutline se do dokumentu připojí nová záložka. Jelikož jsou záložky hierarchicky strukturovány, musí být kromě samotného textu záložky uvedena i její hloubka v hierarchii záložek. Makro \insertoutline je potřeba jen zřídka používat přímo. Je využíváno makry pro zahájení sekcí dokumentu a makry pro uživatelské záložky

Číselný argument ◁*úroveň*▷ definuje hloubku záložky. Záložky na nejvyšší úrovni mají číslo 1. Záložky vnořené v záložkách nejvyšší úrovně mají číslo 2 a tak dále. Argument ◁*text*▷ definuje název záložky. Argumentu ◁*text*▷ je před vloženém zpracován pouze expand procesorem TEXu, proto by neměl obsahovat primitivy pracující na vyšších úrovních TEXu.

```
9 \insertoutline{1}{Nadpis záložky}
```

Makro \insertoutline by prakticky nemělo být používáno. Záložky na kapitoly jsou vytvářeny automaticky. Jedním z problémů ale je pouhá expanse argumentu ◁*text*▷ před vložením do výstupu. Například pokud by kapitola ve svém názvu obsahovala symbol TEXvytvořený makrem \TeX, toto makro se expanduje na následující sekvenci.

```
10 T\kern-.1667em\lower.5ex\hbox{E}\kern-.125em{X}
```

Tato sekvence již obsahuje pouze primitivy, které nelze dále expandovat. Celá sekvence znaků by byla uložena do jména záložky. Z tohoto důvodu existuje ve stylech speciální makro \nextoutline umožňující definovat vlastní název pro záložku další kapitoly.

\nextoutline{◁*text*▷}

Makro \nextoutline je nevýkonné, pouze definuje název záložky pro další kapitolu. V praxi se makro používá v případě, kdy nechceme do záložek vložit týž název jako má kapitola. Makro je nutné použít v případě, kdy je v názvu kapitoly použito neexpandovatelné primitivum.

Argument ◁*text*▷ určuje název záložky.

```
11 \nextoutline{Něco o číslu pi}
12 \subsection{Něco o číslu $\boldsymbol{\pi}$}
```

K vytváření uživatelských záložek slouží další sada několika maker. Všechna makra mají stejnou syntaxi, proto je uveden jejich společný popis.

\outline{◁*text*▷}
\suboutline{◁*text*▷}
\subsuboutline{◁*text*▷}
\subsubsuboutline{◁*text*▷}

Makra \outline, \suboutline, \subsuboutline a \subsubsuboutline slouží k definování uživatelských záložek první až čtvrté úrovně. Záložka vytvořená makrem \outline je na úrovni kapitoly, záložka vytvořená makrem \suboutline je na úrovni podkapitoly a tak dále. Záložka směřuje na místo v dokumentu, kde byla definována. Všechna makra jsou vytvořena generickým makrem \insertoutline.

Argument ◁*text*▷ určuje název záložky.

```
13 \section{Kapitola}
14 \suboutline{Záložka v kapitole}
```

2.3. Makra vázaná ke stylům

V této kapitole jsou rozebrána makra definující text zobrazovaný v *úvodních stránkách stylů*. Jelikož se všechny základní styly liší svým nasazením, při definici obsahu úvodních stran využívají různá makra. Další skupinu maker tvoří makra ovlivňující chování hypertextových částí dokumentu, jde vesměs o makra využívaná stylem upsimple. V další části kapitoly nalezne čtenář popis využití *rejstříku pojmů*. S rejstříkem se pracuje podobně jako v klasickém LaTeXu, přibyla ovšem nová makra umožňující měnit jeho vzhled.

Struktura úvodních stran

Do úvodních stran dokumentů lze vkládat data pomocí následujících maker. Všechna makra mají shodnou syntaxi, `\makro{`◁*text*▷`}`, kde ◁*text*▷ je vkládaná textová informace v rozsahu maximálně jednoho odstavce. Níže uvedená makra mají nevýkonný charakter – mohou být uvedena v libovolném pořadí. Všechna makra by ale měla být uvedena před prvním výskytem makra `\maketitle`. Makro `\maketitle` provede vložení úvodních stran do dokumentu.

Součástí úvodních stran je i universitní logo uložené v souboru `uplogo.eps`. Jedná se o obrázek vytvořený jazykem METAPOST. Zdrojové kódy loga včetně jeho inversního barevného provedení jsou rovněž v archivu. Zdrojový kód v jazyku METAPOST byl vytvořen reversním inženýrstvím původního loga vytvořeného komerčním balíkem Corel Draw, původní logo nebylo vhodné pro zpracování programem `dvipdfm`, viz kapitolu 2.4. Následující přehled uvádí jednotlivá makra a jejich význam.

`\about` Makro určené pro styly upreport a upbook sloužící ke vložení informací o autorovi. Vložený text ve vysázen na druhé straně, to jest hned za stranou titulní. Text je umístěn na úpatí strany s je sázen skloněným písmem. V této sekci textu se může objevit kontakt na autora v podobně elektronické pošty, podmínky šíření textu, ale například i krátké poděkování.

`\abstract` Abstrakt je součástí všech stylů, kromě styly updiplom. Ve stylech upreport a upbook je nepovinný, ve stylu uproject musí být přítomen. Účelem abstraktu je stručně popisovat rozebíranou problematiku, svým rozsahem by neměl přesáhnout zhruba 10 řádků. V žádném případě by abstrakt neměl mít víc jak jeden odstavec. Nepřípustné jsou rovněž i jiné vertikální mezery. Abstrakt je vysázen na straně následující za titulní stranou, u stylu uproject je vycentrován na celou stranu, u stylů upreport a upbook je vysázen v záhlaví stránky.

`\author` Jméno autora je vždy uvedeno na první straně. Podle typu dokumentu je zvoleno umístění, řez písma a jeho stupeň. Jméno by mělo být vždy uváděno celé v pořadí *jméno, příjmení.*

`\annotation` Anotace je určena pro styl updiplom a má podobný charakter jako abstrakt v ostatních stylech. Autor by se při jejím psaní měl držet stejných pokynů. Narozdíl od abstraktu je obsah anotace sázen kursívou. Anotace je v dokumentu obsažena až za místopřísežným prohlášením.

`\date` Makro určené pro vkládání data. Není-li makro uvedeno, je do dokumentu vloženo datum jeho překladu systémem TEX. Datum by mělo být uváděno buďto ve tvaru *měsíc, rok,* nebo včetně čísla dne. Datum je součástí titulní strany, v případě stylu updiplom se nachází na straně s místopřísežným prohlášením. Zde by mělo být datum vždy uvedeno v plném tvaru.

`\group` Ve stylu uproject je nutné tímto makrem uvést skupinu. Skupina je vysazena na titulní straně vpravo dole pod jménem autora. Skupina by měla být složena z názvu oboru následovaného ročníkem označeným římskými číslicemi. Například „Informatika, II. ročník“.

`\report` Pokud je makro `\report` uvedeno, způsobí na titulní stránce vysázení čísla technického reportu. Makro je určeno pouze pro styl upreport. Pokud jsou v číslu reportu pomlčky, měly by být důsledně sázeny dvěma znak pro pomlčku, to jest „`--`“.

`\subtitle` Makro `\subtitle` definuje dodatečný nadpis na titulní straně, lze jej použít ve všech stylech. Dodatečný nadpis tvoří druhý řádek nadpisu.

`\thanks` Speciální makro určené pro styl updiplom. Makro slouží pro sazbu poděkování. Poděkování je v dokumentu umístěno hned za anotací na samostatné straně, v rámci strany je umístěno na jejím spodním okraji. Za poděkováním se nachází obsah.

`\title` Makro definuje název dokumentu. Název je vždy součástí úvodní strany, makro `\title` by mělo být vždy uvedeno. Styl sazby názvu, to jest řez písma a stupeň se v rámci stylů liší. Ve stylu upbook je název sázen versálkami. Tuto vlastnost lze potlačit uvedením přepínače `smalltitle`.

`\year` Makro je určeno pro styl updiplom a uživatel jím definuje rok. Rok je uveden na titulní straně vlevo dole.

Tabulka 2. přehledně zobrazuje použitelnost maker v čtyřech základních stylech. Styl upsimple není v tabulce uveden, jelikož v něm není definováno žádné z uvedených maker. Příklady použití maker nalezne uživatel v ukázkových souborech.

Nastavení vzhledu dokumentu

U nekterých typů dokumentu má uživatel možnost alespoň částečně měnit jejich vzhled. Změna vzhledu není příliš vhodná u dokumentů s pevnou strukturou. Naopak obecné styly mohou být použity k různým účelům, je proto nutné mít k disposici makra ovlivňující jejich vzhled. V následující kapitole jsou popsána základní makra pro nastavení vzhledu dokumentu.

`\setcolor{`◁*červená*▷`}{`◁*zelená*▷`}{`◁*modrá*▷`}`

Makro `\setcolor` lze využít ve stylech upreport, upbook a ve stylu upsimple. Pomocí tohoto makra lze jednoduše měnit barvu hypertextových odkazů v dokumentu. Makro má platnost od místa svého uvedení. Teoreticky je tedy možné barvy v dokumentu „střídat“, ale není to příliš doporučené. Ve většině případů je v dokumentu vhodné nechat implicitní barvu.

Tabulka 2. Složení úvodních stran dokumentů.

makra	vytvořené styly			
	uproject	updiplom	upreport	upbook
\about			×	×
\abstract	×		×	×
\author	×	×	×	×
\annotation		×		
\date	×	×	×	×
\group	×			
\report			×	
\subtitle	×	×	×	×
\thanks		×		
\title	×	×	×	×
\year		×		

Argumenty ◁*červená*▷, ◁*zelená*▷ a ◁*modrá*▷ představují hodnoty jednotlivých aditivně skládaných barevných složek. Hodnoty barevných složek jsou prvky intervalu $[0, 1]$. Následující příklad definuje akvamarínovou barvu.

```
15 \setcolor{0.1}{0.4}{0.3}
```

`\defaultcolor`

Makrem `\defaultcolor` uživatel v dokumentu nastaví standardní barvu hypertextových odkazů. Makro lze použít pouze ve stylech podporujících změny barev hypertextových odkazů, viz makro `\setcolor`.

`\bhilight{`◁*červená*▷`}{`◁*zelená*▷`}{`◁*modrá*▷`}`, `\ehilight`

Dvojice maker `\bhilight` a `\ehilight` vymezuje oblast textu, jenž je obarvena stanovenou barvou. Dvojice maker mohou být do sebe libovolně vnořovány, barvy jsou postupně ukládány na zásobník. Obě makra lze použít pouze ve stylech upreport, upbook a upsimple. Při používání barev je vhodné dbát na základní pravidla pracovní hygieny a dobrého vkusu.

Argumenty ◁*červená*▷, ◁*zelená*▷ a ◁*modrá*▷ makra `\bhilight` představují hodnoty intensity jednotlivých barev, viz makro `\setcolor`. Makro `\ehilight` nemá žádné argumenty, pouze zruší poslední aktivovanou barvu.

```
16 \bhilight{0.1}{1}{0.2}Barevný text.\ehilight
```

`\displayoutlines`

Pokud uživatel používá styl upsimple, záložky nejsou implicitně zobrazovány. V případě uvedení makra `\displayoutlines` budou záložky zobrazeny. Makro lze uvést v dokumentu na libovolném místě.

Speciální prostředí

Ve stylu updiplom jsou definována dvě prostředí pro sazbu závěrů v češtině a v angličtině. Prostředí se jmenují `conclusions-cz` a `conclusions-en` a používají se jako standardní prostředí v LaTeXu. To jest, prostředí zahajujeme pomocí `\begin{`◁*název prostředí*▷`}` a ukončujeme pomocí `\end{`◁*název prostředí*▷`}`. V obou závěrech je potlačena odstavcová zarážka a jednotlivé odstavce jsou od sebe odděleny vertikální mezerou. Každé z těchto dvou prostředí automaticky přidává odkaz do obsahu, vytváří hypertextovou záložku a zajišťuje přechod na novou stránku.

Rejstřík pojmů

Součástí formátu LaTeXje i jednoduchý aparát pro vytváření rejstříku pojmů. Rejstřík bývá zpravidla abecedně setříděný, ne jinak je tomu i v tomto případě. Při vytváření rejstříku je využíván externí program `makeindex`, jenž je součástí distribuce LaTeXu. Pokud chce uživatel vytvářet rejstřík, musí při překladu dokumentu spouštět i program `makeindex`. Úkolem programu `makeindex` je zkompletovat seznam odkazů, setřídit jej a vytvořit podle něj nový seznam vhodný pro sazbu.

Standardní prostředí `theindex` zobrazující setříděné fráze bylo v nových stylech reimplementováno. V původním prostředí `theindex` bylo jen velmi těžké měnit počet sloupců rejstříku, rovněž jeho typografická úroveň byla nevyhovující. Rejstřík získal rovněž hypertextovou podobu, odkazy na strany jsou v rejstříku hypertextové. Rejstřík je určen pro styly upbook, upreport a updiplom. V následujícím textu je stručně popsáno používání nových maker.

Pro vytvoření rejstříku je nutné provést následující kroky.

- Před začátkem těla dokumentu je nutné uvést makro `\makeindex`. Tímto makrem se vytvoří výstupní soubor ve tvaru ◁*název*▷`.idx`, kde ◁*název*▷ je název překládaného dokumentu.
- Po překladu dokumentu je nutné spustit externí program `makeindex` s argumentem ◁*název*▷`.idx` a provést opětovný překlad. Indexační program vytvoří výstupní soubor ◁*název*▷`.ind`.
- Na místě vložení rejstříku do dokumentu je nutné uvést speciální makro `\printindex`. Rejstřík pojmů by měl být uveden až na poslední straně dokumentu. Upravené makro `\printindex` do dokumentu automaticky vkládá i nadpis pro rejstřík a zavádí pro něj i novou záložku.

Fráze se do rejstříku přidávají pomocí maker `\IN` a `\INEM`. Pokud se v textu nachází důležité fráze a autor ji chce přidat do rejstříku, použije k tomu jedno z těchto dvou maker. Obě makra se přitom liší *vyznačením stránky*. Makro `\IN` se

používá při standardním vyznačení strany, makro \INEM při zvýrazněném vyznačení strany. V dokumentu je může jeden pojem vyskytovat na několika místech a autor jej může zaindexovat několikrát. V této situaci slouží *zvýrazněné vyznačení strany* k orientaci uživateli – mělo by označovat místo v dokumentu, kde je pojem poprvé vysvětlen, nebo definován. Bližší popis obou maker je shrnut dále.

`\IN{`◁*fráze*▷`}`, `\INEM{`◁*fráze*▷`}`

Makra zavádějí do seznamu frází další položku. Makro \IN vytváří záznam s normálním značením strany, makro \INEM vytváří záznam se zvýrazněným značením strany. Argument obou maker má stejný význam, určuje frázi.

Argument ◁*fráze*▷ určuje frázi zahrnutou do rejstříku pojmů. Argument může nabývat několika tvarů. Pokud je uveden jako jednoduché slovo nebo sousloví oddělené mezerami, jedná se o *hlavní frázi.* Pokud se v argumentu vyskytuje více pojmů oddělených znakem vykřičník „!“, pak se jedná o *definici podfráze.* Podfráze jsou sázeny s počátečními odrážkami. Počet odrážek záleží na hloubce vnoření. Pokud je fráze příliš dlouhá, ve výsledném výpisu se řádek zalomí, ale nezačíná již odrážkou – tím se visuálně odlišuje od podfráze. Následující příklad ukazuje definici hlavní fráze a dvou podfrází.

```
17 \IN{fonty}
18 \IN{fonty!proporcionální}
19 \IN{fonty!neproporcionální}
```

Při zpracování vstupního souboru TEXem jsou všechny nalezené fráze zapsány do pomocného souboru ◁*název*▷`.idx`. V tomto souboru se fráze nacházejí ve stejném pořadí, v jakém jsou uvedeny ve zdrojovém dokumentu. Spuštěním programu `makeindex` se vytvoří setříděný seznam frází ◁*název*▷`.ind` využívající prostředí `theindex`. Po tomto kroku je nutné znovu provést překlad dokumentu. Sekvence příkazů pro překlad musí být například následující.

```
cslatex ◁název▷.tex
makeindex ◁název▷.idx
cslatex ◁název▷.tex
outlines ◁název▷.dvi
dvipdfm -p a4 -o ◁název▷.pdf ◁název▷.dvi
```

Vzhled rejstříku je možné částečně měnit. Nová implementace prostředí `theindex` umožňuje nastavit rejstříku *libovolný počet sloupců.* Počet lze změnit vhodnou redefinicí makra \indexcolumns. Implicitně je rejstřík zobrazován do dvou sloupců. Bude-li před výskytem makra `\printindex` uvedena nová definice makra \indexcolumns, rejstřík bude sázen jiným počtem sloupců. Viz následující definici.

```
20 \renewcommand{\indexcolumns}{3}
```

V tomto případě bude rejstřík sázen do tří sloupců. Kromě počtu sloupců lze ovlivňovat i *styl zvýraznění stran* odpovídajících frázím vloženým makrem `\INEM`. Implicitně jsou zvýrazněné strany sázeny italikou. Pokud by uživatel chtěl změnit jejich sazbu, může redefinovat makro `\indexemph`.

```
21 \renewcommand{\indexemph}[1]{{\bfseries #1}}
```

Nová definice musí být uvedena opět před výskytem makra `\printindex`. Narozdíl od makra `\indexcolumns` má makro `\indexemph` jeden argument – tím je vždy číslo zvýrazněné strany. Tvar, váhu, stupeň a *styl písma v rejstříku* lze ovlivnit vhodnou redefinicí makra `\indexfashion`. Obsah makra je vložen před první položku v rejstříku. Všechna makra v něm uvedená mají pouze lokální rozsah platnosti, to jest veškerá nastavení tvaru písma neovlivňují zbytek dokumentu. Příkladem použití budiž následující definice.

```
22 \renewcommand{\indexfashion}{\footnotesize}
```

V tomto případě bude obsah rejstříku sázen písmem o velikosti poznámky pod čarou. Mimo rejstřík však zůstane stupeň písma zachován. Všechna tři předchozí makra jsou využívána v rámci prostředí `theindex` a měla by být redefinována vždy *před uvedením* makra `\printindex`.

2.4. Vytváření a vkládání obrázků

Hlavní nevýhodou generování PostScriptu i PDF z jednoho DVI souboru je obtížná práce s obrázky. Oba datové formáty umožňují využívat rozdílné datové formáty obrázků. Pokud by uživatel používal jen formát obrázků akceptovatelný jedním z obou formátů, nemohl by již využít DVI soubor k vytvoření druhého výstupu. Uživatel by se tím ochudil o jednu z hlavních výhod navrhovaného řešení problému. Oba dva výstupní ovladače ale mohou využívat jednoduché PostScriptové obrázky. Ačkoliv je pojem „jednoduchý“ v tomto kontextu dost subjektivní, jedná se v podstatě o PostScriptové obrázky sestávající ze základních vykreslovacích příkazů, transformačních instrukcí a operátorů měnících styl vykreslování. Jednoduché PostScriptové obrázky dokáže produkovat například matematický balík Maple.

V odborné literatuře se nejčastěji vyskytují *pérovky* – okomentované nákresy a diagramy skládající se z čar a jiných jednoduchých geometrických tvarů, zpravidla doplněné textem. Již méně se v odborných textech vyskytují *autotypie* – obrázky vytvořené fotoreprodukční cestou, to jest rozkladem na síť bodů o různé hustotě, barvě a velikosti. Pro vytváření pérovek je v TEXu určeno několik specialisovaných prostředí a balíků maker. Balíky poskytují různou škálu grafických primitiv s různou mírou uživatelského komfortu. Jednou z nejvhodnějších možností je použít k výrobě pérovek jazyk METAPOST, jelikož dokáže produkovat

dostatečně jednoduchý PostScriptový výstup, zpracovatelný jak výstupním ovladačem `dvips`, tak i ovladačem `dvipdfm`.

Jazyk METAPOST je programovací jazyk odvozený z jazyka METAFONT vyvinutého Donaldem Knuthem, viz [5]. Oproti jazyku METAFONT, který byl primárně vyvinut pro definici fontů a měl tudíž některé nepříjemné rysy – například bitmapový výstup a nemožnost používání barev, je výstupem METAPOSTu soubor v jazyku PostScript. Veškerá syntaxe však zůstává až na malé výjimky shodná s jazykem METAFONT. Síla tohoto programovacího jazyka spočívá především, ale nikoli jen, v možnosti řešit efektivně soustavy lineárních rovnic. To umožňuje velmi snadno programovat v *deklarativním stylu.* Autorem jazyka METAPOST je J. Hobby, viz [3].

Makra pro vytváření obrázků

Jazyk METAPOST je velice komplexní nástroj. Lze pomocí něj především vytvářet velmi přesné obrázky. Díky automatickému řešení soustav lineárních rovnic a silnému makrojazyku není problémem vyvíjet rovněž obecné sady maker sloužící k vykreslování konkrétních tříd obrázků, například automatů, UML diagramů, fuzzy množin a jiných.

Pro zjednodušení kreslení obrázků a zavedení jednotného stylu jejich zpracování byla vytvořena sada maker uložená v souboru `upfigure.mp`. V souboru jsou definována makra pro vytvoření okénkové transformace scény a pro vykreslování a popis os. Pokud bude uživatel používat tato makra, může s obrázkem provádět jednoduše různé typy transformací. Při těchto transformacích nebudou ovlivněny velikosti popisků obrázku. Používání maker není nezbytně nutné, uživatelům začátečníkům ale mohou usnadnit práci. Pro přehlednost uvádím popis nově definovaných maker.

`_mk_transform`

Makro `_mk_transform` načte podle definovaných proměnných fyzickou velikost vykreslovacího okna, respektive uvažovaného prvního kvadrantu. Dále načte informace o logickém souřadném systému a vrací *okénkovou transformaci* ze souřadnic scény do fyzických souřadnic. Pro změnu měřítka obrázku lze změnit jen definici okénkové transformace.

Makro `_mk_transform` nemá žádné argumenty. Informace si bere z lokálních proměnných pevných jmen. Proměnné `dx`, `dy` určují fyzickou velikost prvního kvadrantu. Proměnné `ax`, `bx` určují měřítko x-ové osy prvního kvadrantu. To jest bod $[\texttt{ax}, 0]$ je okénkovou transformací zobrazen na bod $[0, 0]$ a bod $[\texttt{bx}, 0]$ je zobrazen na $[\texttt{dx}, 0]$. Analogicky Proměnné `ay`, `by` určují měřítko y-ové osy. Makro vrací hodnotu typu transformace.

```
23 dx := 4cm; dy := 3cm;
24 ax := 0; bx := 10;
```

```
25 ay := 0; by := 1;
26 transform fuzzy;
27 fuzzy := _mk_transform;
```

`_draw_axis` (◁*fx*▷, ◁*lx*▷, ◁*fy*▷, ◁*ly*▷, ◁*t*▷)
`_draw_arrow_axis` (◁*fx*▷, ◁*lx*▷, ◁*fy*▷, ◁*ly*▷, ◁*t*▷)

Makra `_draw_axis` a `_draw_arrow_axis` slouží k vykreslení os. První z dvojice maker vykresluje osy bez šipek na koncích čar. Druhé makro vykresluje osy včetně zakončujících šipek – šipky jsou umístěny v prvním kvadrantu. Makra by měla být použita těsně za definicí okénkové transformace.

Argumenty ◁*fx*▷, ◁*lx*▷ určují rozpětí, ve kterém bude nakreslena x-ová osa, jedná se o logické souřadnice. Obdobně argumenty ◁*fy*▷, ◁*ly*▷ jsou logické souřadnice určující vykreslení y-ové osy. Argument ◁*t*▷ je okénková transformace získaná makrem `_mk_transform`. Následující příklad ukazuje vykreslení os prvního a druhého kvadrantu.

```
28 _draw_arrow_axis (-bx, bx, ay, by, t);
```

`_x_label` (◁*text*▷, ◁*pos*▷, ◁*t*▷)
`_y_label` (◁*text*▷, ◁*pos*▷, ◁*t*▷)

Tato makra slouží k vytváření popisů os. Makro `_x_label` je určeno k popisu x-ové osy, makro `_y_label` je určeno k popisu y-ové osy. Obě makra mají identické argumenty. Popisy k x-ové ose jsou sázeny pod osou, popisy y-ové osy jsou sázeny na jejím levém okraji. Kromě samotného popisu je na ose vytvořen i malý dílek.

Argument ◁*text*▷ je text popisu. Nejčastěji se používá plainTEXovský výraz uzavřený mezi klausule `btex` ⋯ `etex`. Argument ◁*pos*▷ je logická souřadnice na ose. Posledním argumentem je okénkové transformace.

```
29 _y_label (btex $1$ etex, 1, fuzzy);
30 _y_label (btex $\alpha$ etex, 0.8, fuzzy);
31 _x_label (btex $x_6$ etex, 6, fuzzy);
```

Následující příklad může sloužit jako jednoduchý návod pro vytváření obrázků pomocí balíku maker `upfigure.mp`. Aby byly nakreslené obrázky co možná nejpoužitelnější, měly by mít svou vnitřní logickou strukturu. V jednom dokumentu by obrázky měly být rovněž kresleny stejným stylem čar, popisy os by měly mít stejný styl a podobně.

V záhlaví obrázku bývá obvykle definováno měřítko. Do skutečných rozměrů je obrázek transformován pomocí jednoduché okénkové transformace. Hodnoty `dx`, `dy` udávají skutečnou velikost kvadrantu. V následující ukázce jsou nastaveny na 4 cm a 2 cm. Hodnoty `ax`, `bx`, `ay` a `by` udávají logický souřadný systém v prvním kvadrantu. Tento systém by měl být volen tak, aby bylo nakreslení obrázku co nejjednodušší. Například při použití goniometrických funkcí bude vhodné volit rozsah od 0 po 2π v x-ové ose a od 0 po 1 v y-ové ose. Posledním krokem v záhlaví je inicialisace okénkové transformace podle definovaných hodnot.

```
beginfig (1)
  input upfigure.mp;
  dx := 4cm; dy := 3cm;
  ax := 0; bx := 2;
  ay := 0; by := 2;
  transform t;
  t := _mk_transform;
```

Makrem `_draw_axis` se vykreslují osy. Posledním argumentem makra je okénková transformace vytvořená předchozím voláním makra `_mk_transform`. První čtyři argumenty určují rozsah vykreslovaných os. V následující ukázce jsou vykresleny osy prvního kvadrantu s mírným přesahem do čtvrtého kvadrantu. Rozsah os je vyjadřován v logických souřadnicích.

```
  _draw_axis (ax, bx, -0.2by, by, t);
```

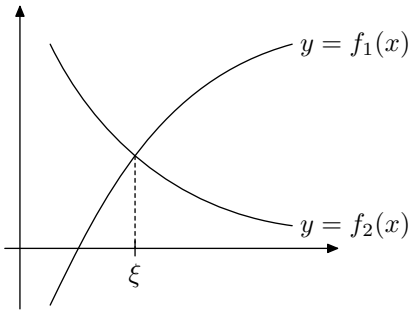
V dalším bloku obrázku by měl být nakreslen vlastní obsah – ovšem bez komentářů, ty by pro přehlednost měly být umístěny až na konci. Další kód ukazuje vytvoření dvou cest representujících protínající se funkce. Pokud nejde o přesně zadané funkce, lze je vytvořit jako jednoduché interpolační křivky se zadanými tečnými vektory v počátku a konci. Pokud se jedná o přesně zadané funkce, měly by být interpolovány přes své funkční hodnoty.

```
  path p, q;
  p := (0.2, -0.5){dir 70} ... {dir 20}(1.8, 1.8);
  q := (0.2, 1.8){dir -70} ... {dir -10}(1.8, 0.2);
  draw p transformed t;
  draw q transformed t;
```

Datový typ `path` má v jazyku METAPOST význam *cesty*. Cesta se může skládat z přímých úseků a křivek. METAPOST disponuje možností vytvářet interpolační i aproximační křivky. V další fázi bude tečkovanou čarou vyznačen průsečík obou cest. Úsečka povede povede z průsečíku do bodu na ose x a bude rovnoběžná s osou y. Průsečík dvou cest může být nalezen například makrem `intersectionpoint`. Hustotu tečkování určuje proměnná `dotted_line` deklarovaná rovněž v balíku maker `upfigure.mp`.

```
  z1 = p intersectionpoint q;
  draw (z1 --- (x1, 0)) transformed t dashed dotted_line;
```

Poslední částí zdrojového kódu obrázku jsou komentáře. Pokud jsou komentáře uprostřed obrázku, lze je vytvářet přímo makrem `label`. Pokud je potřeba popisovat osy, je dobré využít makra `_x_label` a `_y_label`. Makra berou jako argumenty text, parametr a okénkovou transformaci. S komentářem se na ose automaticky vytvoří i značka. Komentování os by mělo být střídmé. Leckdy stačí jen orientační hodnoty – komentovány by měly být zejména důležité hodnoty.



Obrázek 2. Jednoduchý obrázek v METAPOSTu.

```
47  pair pl, ql;
48  pl := point length (p) of p;
49  ql := point length (q) of q;
50  label.rt (btex $y=f_1(x)$ etex, pl transformed t);
51  label.rt (btex $y=f_2(x)$ etex, ql transformed t);
52  _x_label (btex $\xi$ etex, x1, t);
53 endfig;
```

Schéma funkcí odpovídající předešlému zdrojovému kódu naleznete na obrázku 2. Při vytváření obrázku je dobré vyvarovat se použití absolutních souřadnic. Pokud budou v dokumentu uvedeny absolutní souřadnice nebo rozměry, jejich použití znemožní další jednoduché úpravy obrázku. Například při změně měřítka by mohlo dojit k úplnému rozpadu obrázku. Vytváření obrázků v jazyku METAPOST s sebou přináší řadu výhod. Jelikož je jazyk velmi přehledný, obrázky je možné nejen psát ručně, ale snadno generovat jinými programy.

Další nespornou výhodou je možnost vytváření generických maker a maker určených pro speciální typy obrázků. Jejich kreslení se potom výrazným způsobem zjednodušuje. Další nespornou výhodou METAPOSTu je spolupráce se samotných TEXem. Komentáře vytvářené k obrázkům jsou zpracovávány přímo TEXem, takže nenabourávají celkovou typografickou úroveň dokumentu. Tento problém má řada jinak velmi hezky vysazených textů – obrázky v nich jsou vyráběny softwarem bez vazby na TEX. Obrázky potom používají jiné rodiny a řezy písma, což působí na čtenáře většinou rušivě nebo odpudivě.

Vkládání obrázků do dokumentu

Zdrojové kódy obrázků se zapisují do samostatného souboru. V jednom souboru může být uvedeno i několik obrázků. Obrázky jsou vzájemně odděleny klíčovým slovem `beginfig(`◁*číslo*▷`)`, kde číslo unikátně označuje obrázek v rámci jednoho zdrojového souboru. Každý zdrojový soubor je zakončen klíčovým slovem `bye` podobně, jako zdrojový kód pro plainTEX. Má-li být obrázek použit

v dokumentu, musí být nejprve přeložen ze své zdrojové podoby a vložen do cílového dokumentu.

Obrázky jsou ze zdrojové podoby překládány programem `mpost`. Program bere jako argument soubor se zdrojovými kódy obrázků. Při úspěšném překladu jsou v aktuálním adresáři vytvořeny jednotlivé obrázky. Jestliže má vstupní soubor název ve tvaru ◁*název*▷`.mp`, potom budou mít výstupní soubory jména tvaru ◁*název*▷`.`◁*číslo*▷, kde ◁*číslo*▷ je číslo příkladu definované klíčovým slovem `beginfig(`◁*číslo*▷`)`. Tyto soubory jsou v podstatě zjednodušené PostScripty, jejich přímé použití však obecně není možné. V souborech nejsou navázány informace o fontech.

Z tohoto důvodu musí po překladu programem `mpost` následovat překlad vlastního dokumentu a vytvoření výstupu v jazyku PostScript nebo v datovém formátu PDF. Na úrovni DVI souborů obecně nelze obrázky prohlížet. Proces překladu dokumentu se rozšiřuje o další příkaz, viz následující ukázku.

```
mpost ◁obrázky▷.mp
cslatex ◁název▷.tex
outlines ◁název▷.dvi
dvipdfm -p a4 -o ◁název▷.pdf ◁název▷.dvi
```

Jelikož jsou vygenerované obrázky v jazyku PostScript, lze je do dokumentu vkládat jako jiné PostScriptové obrázky, například pomocí makra `\epsfbox` z dodatečného stylu `epsf`. Tento dodatečný styl je automaticky zahrnut i v nových stylech, není jej tedy třeba vkládat makrem `\usepackage`. Obrázky je vhodné umístit do plovoucích prostředí například následujícím způsobem.

```
\begin{figure}
  \centerline{\epsfbox{obrazky.3}}
  \caption{Činnost \TeX u.}
  \label{tex_fig}
\end{figure}
```

Místo makra `\centerline` lze samozřejmě použít jiný způsob vycentrování, například umístění obrázku do prostředí `center`. Na závěr ke vkládání obrázků je dobré připomenout, že obrázky by měly mít svůj popis vždy na spodní straně. U tabulek umístěných v plovoucím prostředí již není tento požadavek striktně kladen, ale v rámci celého dokumentu by měly být popisky umístěny vždy buďto nahoře, nebo vždy dole.

3. Popis implementace stylů

V této kapitole čtenář nalezne stručný popis implementace stěžejních částí vytvořených stylů. Dokumentace není v žádném případě vyčerpávající. Styly jsou

svým obsahem velmi podobné, zaměřil jsem se především na popis implementace hypertextových odkazů. V další části se stručně zmíním o nových typografických konvencích a jejich provedení. Pokročilý uživatel formátu LaTeX by po přečtení této sekce by měl být schopen předložené styly podle svých potřeb dále rozšiřovat.

Hypertextové odkazy

V první řadě byl vyřešen problém vkládání PDF objektů a potlačení vkládání objektů při uvedení přepínače `nopdf`. Hned poté byla vytvořena sada jednoduchých maker umožňujících do dokumentu vkládat PDF objekty, konkrétně *anotace* a *návěstí*. Tyto objekty slouží k vytváření odkazů v rámci dokumentu.

Všechny styly jsou vybaveny přepínačem `nopdf`. Pokud je tento přepínač uveden, do výsledného dokumentu nejsou vkládány žádné PDF objekty. V dokumentu je definováno makro `\@inspdf` následovně.

```
59 \def\@inspdf#1{#1}
```

Je-li uveden přepínač `nopdf`, makro je definováno ve tvaru `\def\@inspdf#1{}`. Každý příkaz vkládající PDF objekty je uveden jako argument makra `\@inspdf`. V případě uvedení přepínače `nopdf` se tím pádem do dokumentu nevloží žádný speciální příkaz. Pokud bude uživatel chtít rozšířit možnosti stylu, měl by důsledně psát všechny nové PDF příkazy jako argumenty makra `\@inspdf` a funkčnost přepínače `nopdf` bude zachována.

Do první kategorie PDF maker patří makra pro práci s barvami. Makro `\@pdfsetcolor` má dva argumenty, prvním a nich je *barva,* druhým argumentem je sekvence příkazů. Makro slouží k vysazení text se změněnou barvou popředí. Barva se v PDF definuje jako pole, to jest ve hranatých závorkách. PDF akceptuje množství barevných schémat. V implementaci jsem použil RGB, to jest aditivní skládání barev ze tří základních – červené, zelené a modré. Pro zvýraznění odkazů je ve stylu předdefinována pevná barva, viz makro `\@@pdflinkcolor`.

```
60 \def\@pdfsetcolor#1#2{\@inspdf{\special{pdf:bc #1}}%
61   #2\@inspdf{\special{pdf:ec}}}
62 \def\@@pdflinkcolor{[0.2 0.3 0.6]}
```

Speciálními instrukce `pdf:bc`, `pdf:ec` se vymezuje počátek a konec vybarveného bloku. Jedná se o instrukce pro výstupní ovladač `dvipdfm`. Barvy jsou v PDF skládány na zásobník, instrukce `pdf:bc` přidá barvu na vrchol zásobníku, příkaz `pdf:ec` odebere barvu z vrcholu zásobníku.

Cílové místo v dokumentu je definováno makrem `\@pdfdest`. Makro má jediný argument a to název cílového místa. Uvedením makra je zadané jméno svázáno s aktuálním místem v dokumentu. Při definici cílového místa je rovněž uvedeno i zvětšení výřezu. V následujícím kódu jsou k určení místa odkazu použity speciální proměnné `@thispage` a `@ypos`. Dereference proměnných je prováděna výstupním ovladačem `dvipdfm`.

```
\def\@pdfdest#1{\@inspdf{%
  \special{pdf:dest (#1) [ @thispage /FitH @ypos ]}}}
```

To jest při kliknutí na odkaz bude stránka zvětšena – šířka strany bude zabírat celé okno prohlížeče, navíc bude výřez posunut k aktuálnímu místu odkazu. Jelikož jsou v PDF hodnoty datového typu *řetězec* uzavírány do kulatých závorek, je uzavřen i argument `#1` makra `\@pdfdest`. K vytváření obecných odkazů uvnitř dokumentu slouží makro `\@pdflink`. Makro má dva argumenty. Prvním z nich je sekvence příkazů, druhým je odkaz. Sekvence příkazů slouží k vysazení obsahu odkazu.

```
\def\@pdflink#1#2{\leavevmode\@inspdf{%
  \special{pdf:bann << /Type /Annot /Subtype /Link
    /Border [ 0 0 0 ] /A << /S /GoTo /D (#2) >> >>}}%
  {\@pdfsetcolor{\@@pdflinkcolor}{#1}}%
    \@inspdf{\special{pdf: eann}}\relax}
```

V předcházejícím kódu maker je nejprve vložen *začátek anotace.* V této deklaraci je uvedeno o jakou anotaci se jedná. Anotace má nastavenu barvu okraje na barvu pozadí, to jest kolem odkazů v dokumentu není vytvářen „rámeček“. Do dokumentu je dále vložen *zvýrazněný text* příslušný prvnímu argumentu. Na posledním místě je v makru celá *anotace ukončena.* Text v odkazu je zvýrazněn pomocí `\@pdfsetcolor`.

Pomocí výše popsaných obecných maker již není problém transformovat stávající makra LaTeXu do hypertextové podoby. Pro ukázku vezměme například makra `\label` a `\ref`. Nejprve je obsah starých maker, to jest *posloupnost tokenů,* navázán na nová jména `\@oldlabel`, `\@oldref`.

```
\let\@oldlabel=\label
\let\@oldref=\ref
```

Poté jsou na původní jména maker navázána makra nová – tato makra zajišťují vytváření odkazů. Ve svém těle nová makra používají makra původní, jenž byla uchována jako sekvence tokenů. Tento přístup má několik výhod. Uživatel maker vlastně ani nepozná, že byla makra změněna – jejich funkčnost se nemění. Řešení je navíc obecné. Pokud se změní vnitřní implementace původních maker, nové dodatečné styly to nijak neovlivní.

```
\def\label#1{\@pdfdest{lnk#1}\@oldlabel{#1}}
\def\ref#1{\@pdflink{\@oldref{#1}}{lnk#1}}
```

Nové makro `\label` definuje nejprve hypertextové návěstí, hned potom definuje i návěstí v LaTeXu – obě návěstí mají shodné jméno. Při odkazu na návěstí je vytvořen hypertextový odkaz, přitom zvýrazněná část odkazu je definována původním makrem `\ref`. Obdobným způsobem jsou definována i ostatní makra.

Pro vytváření externích odkazů slouží makro \@pdfuri. Argumenty jsou stejné jako v případě makra \@pdflink. Makro se pouze liší v definici typu anotace. Anotace URI – *Uniform Resource Identifier* slouží k zadávání unifikovaných identifikátorů, například URL adresy. Viz kód makra.

```
\def\@pdfuri#1#2{\leavevmode\@inspdf{%
  \special{pdf:bann << /Type /Annot /Subtype /Link
    /Border [ 0 0 0 ] /A << /S /URI /URI (#2) >> >>}}%
  {\@pdfsetcolor{\@@pdflinkcolor}{#1}}%
    \@inspdf{\special{pdf: eann}}\relax}
```

Díky makru \@pdfuri lze snadno implementovat makra \link a \mail. Jejich kód zde uvádět nebudu, najdete jej ve zdrojových souborech dodatečných stylů

Poslední kategorií odkazů v rámci dokumentu jsou záložky. Záložky nepatří pod objekty typu anotace, jde o samostatný PDF objekt. V kapitole 2.2. bylo popsáno generické makro \insertoutline sloužící k vytváření záložek. Makro je implementováno velmi jednoduše, do dokumentu vkládá speciální instrukci svazující aktuální stránku se záložkou.

```
\def\insertoutline#1#2{%
  \@inspdf{\special{pdf:outline #1
    << /Title (#2) /Dest [ @thispage /FitH @ypos ] >>}}}
```

Základním požadavkem na záložky bylo jejich automatické vkládání při uvedení začátku kapitoly. Ve stylu article jsou veškeré kapitoly vytvářeny sadou dost neprůhledných obecných maker, hlavní z nich je makro \@startsection. To se mi však pro úpravu nezdálo dostatečně přehledné. Místo něj záložky vkládám v redefinovaném makru \@sect. Při vkládání musí být zohledněn také problém názvů záložek. Je-li v názvu kapitoly použito neexpandovatelné primitivum, není vhodné jej v tomto tvaru vkládat do názvu záložky. Nejprve uveďme kód.

```
\def\nextoutline#1{\gdef\@@nextoutlinelabel{#1}}
\let\@oldsect=\@sect
\def\@sect#1#2#3#4#5#6[#7]#8{%
  \ifx\@@nextoutlinelabel\undefined\insertoutline{#2}{#7}%
  \else\insertoutline{#2}{\@@nextoutlinelabel}%
  \let\@@nextoutlinelabel=\undefined\fi%
  \@oldsect{#1}{#2}{#3}{#4}{#5}{#6}[#7]{#8}}
```

Při uvedení \nextoutline je definováno globální makro. Pokud začne další kapitola, je nejprve testován obsah tohoto globálního makra. Pokud bylo definováno, bude v záložce místo aktuálního názvu kapitoly uveden název vložený makrem \nextoutline. Původní obsah makra \@sect byl uschován podobným způsobem jako u předcházejících maker. Nakonec je název vložený makrem \nextoutline opět „zapomenut“, aby neovlivnil název další kapitoly. Na závěr je v každém dodatečném stylu vložen jeden speciální řádek.

```
\special{\@inspdf{pdf:docview << /PageMode /UseOutlines >>}}
```

Účelem tohoto příkazu je zapnout v dokumentu zobrazování záložek.

Pokud by uživatel chtěl využívat současné dodatečné styly například se stylem report, musel by si adekvátně upravit makro `\chapter`. V opačném případě by začátky kapitol nebyly vkládány do záložek.

Nové typografické konvence

Současnému systému LaTeX 2$_\varepsilon$ je vytýkáno několik nedostatků. Z pohledu zvyklostí při české sazbě je to především nevhodné číslování kapitol a umísťování odstavcových zarážek. Oba dva problémy jsou v nových dodatečných stylech vyřešeny.

Ve standardním LaTeXu je v prvním odstavci kapitoly vypuštěna odstavcová zarážka, to jest všechny řádky prvního odstavce kapitoly začínají na levém okraji. Každý další odstavec začíná zarážkou velikosti 20 monotypových bodů. V českých dokumentech je obvyklé uvádět odstavcovou zarážku bez rozdílu u každého odstavce, nebo ji neuvádět vůbec. Redefinicí makra `\@afterheading` lze dosáhnout požadovaného efektu.

```
\def\@afterheading{}
```

V angloamerické literatuře bývají kapitoly obvykle číslovány bez tečky na konci. V české sazbě je zvykem tečky na konci čísel kapitol uvádět. Stejně tak i obrázky a tabulky by měly být číslovány tímto stylem. Změna této notace si vyžádala největší zásahy do maker, protože ji nebylo možné jednoduše vyřešit pouze na jednom místě. Jednak musela být redefinována makra zobrazující čísla kapitol, podkapitol a podobně, musela být také změněna i makra pro vytváření popisů obrázků a podobně. Na závěr kapitoly uvedu jen některá z maker měnící styl vypisování kapitol, pro další detaily viz zdrojové kódy maker.

```
\renewcommand\thepart       {\@Roman\c@part.}
\renewcommand\thesection    {\@arabic\c@section.}
\renewcommand\thesubsection{\thesection\@arabic\c@subsection.}
```

Ostatní části maker, jako například úvodní stránky, již nepotřebují další komentář, jsou vytvořeny průhledným způsobem pouze pomocí základních maker LaTeXu. Zdrojové kódy maker jsou přehledně strukturovány a komentovány, uživateli by jejich další rozšiřování nemělo činit problémy.

Reference

[1] Adobe Systems Inc. *PostScript Language Reference.*
ISBN 0–201–37922–8, 3rd Edition, Addison Wesley, 1999.

[2] Adobe Systems Inc. *Portable Document Format Reference.*
ISBN 0–201–61588–6, 2nd Edition, Addison Wesley, 2000.

[3] Hobby, J. D. *A User's Manual for METAPOST.*
AT&T Bell Laboratories, Murray Hill, NJ 07974.

[4] Knuth, D. E. *The TEXbook.* Volumes A, B of *Computers & Typesetting.*
Addison Wesley, Reading, Massachusetts 1986.

[5] Knuth, D. E. *The METAFONTbook.* Volume C of *Computers & Typesetting.*
Addison Wesley, Reading, Massachusetts 1986.

Rejstřík