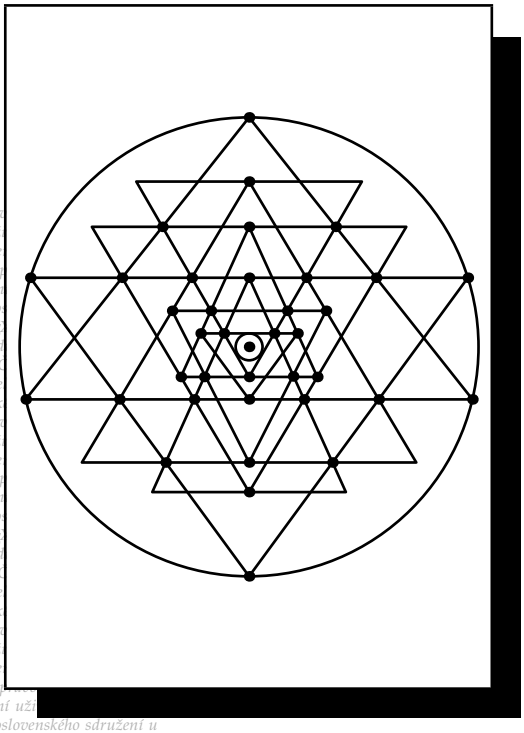
CSTUG



ZPRAVODAJ

Československého sdružení uživatelů TEXu

4

2009

ISSN 1211-6661 Tištěná verze

ISSN 1213-8185 Online verze

Ročník 19

OBSAH

Zpravodaj Československého sdružení uživatelů TEXu je vydáván v tištěné podobě a distribuován zdarma členům sdružení. Po uplynutí dvanácti měsíců od tištěného vydání je poskytován v elektronické podobě (PDF) ve veřejně přístupném archívu dostupném přes `http://www.cstug.cz/`.

Zpravodaj je zařazen do Seznamu recenzovaných neimpaktovaných periodik vydávaných v České republice, viz `http://www.vyzkum.cz/`.

Své příspěvky do Zpravodaje můžete zasílat v elektronické podobě, nejlépe jako jeden archivní soubor (`.zip, .arj, .tar.gz`). Postupujte podle instrukcí, které najdete na stránce `http://bulletin.cstug.cz/`. Pokud nemáte přístup na Internet, můžete zaslat příspěvek na disketě, CD, či DVD na adresu:

Zdeněk Wagner
Vinohradská 114
130 00 Praha 3

`zpravodaj@cstug.cz`

Nezapomeňte přiložit všechny soubory, které dokument načítá (s výjimkou standardních součástí TEX Live), zejména v případě, kdy vás nelze kontaktovat e-mailem.

ISSN 1211-6661 (tištěná verze)
ISSN 1213-8185 (online verze)

Šťastné a veselé . . .

Pavel Stříž

Naše pozdravy všem! Vítejte při četbě nového Zpravodaje!

TEXový svět je čím dál napínavější, že? K tomuto závěru alespoň dochází redakce po zjištění, že tu máme nejen skupinky Plain TEXistů, LATEXistů a ConTEXtistů, ale už i vznikající skupinku LuaTEXistů (LuaTEX je následník pdfTEXu). Vedle toho najdeme skupinky kolem PTEXu (sazba japonštiny), CTEXu (sazba čínštiny), ko.TEXu (sazba korejštiny) a mohli bychom s výčtem pokračovat.

Kdo by si v tom rád udělal pořádek, navíc s historickými poznámkami, tomu doporučujeme Koutek rozhovorů (Interview Corner) na `http://tug.org/interviews/` a zbrusu novou knihu *TEX People* editorů Karla Berryho (předseda TUG) a Davida Waldena (jeden z prarodičů ARPANETu; jeho články se pravidelně objevují v časopisu PracTEX Journal, viz `http://www.tug.org/pracjourn/`).

Takový unikátní TEXový počin si zaslouží ještě pár řádků. V knize nalezneme 50 rozhovorů z let 2004–2009 s lidmi od TEXu, což nám dává jedinečný obrázek o TEXové komunitě ve světě (včetně několikrát zmíněného našince). Zpovídáni jsou lidé od prvních Knuthových spolupracovníků až po lidi zapojené do aktuálně bežících TEXových projektů (XƎTEX, LuaTEX), nebo třeba lidé, kteří „jen“ TEX rádi používají. Většina rozhovorů bude vstřebatelná i lidem v TEXu novějším (většina rozhovorů ostatně není tak moc technická), odborníci si však též přijdou na své, inu, minimálně mohou zavzpomínat. A co víc si přát? Až dočtete tuto knihu, tak vězte, že na dalších rozhovorech se usilovně pracuje.

Co nalezneme v tomto čísle Zpravodaje? Pojďme společně prozkoumat to, co máte před sebou. Samé předvánoční dobroty: české a po dlouhé době i anglické.

Začneme překladem jednoho z rozhovorů ze zmíněné knihy, konkrétně rozhovoru Davida Waldena s Hàn Thế Thành, tvůrcem pdfTEXu, a seskládanými a přeloženými informacemi o LuaTEXu z `http://www.luatex.org/`. To vše z překladatelské dílny Michala Mádra za asistence Pavla Stříže.

Poté Jan Šustek zabrousí do šifrování TEXového souboru.

Nebojácně čtěme dál. Představujeme vám příspěvky v jazyce anglickém. První je o kresbě vnitřní části indického kultovního obrazce Shri Yantra v programu METAPOST Portugalce Luíse Nobre Gonçalvese, Michala Mádra a Pavla Stříže.

Musíme podotknout, že pan Gonçalves trval na tom, abychom byli autoři, že prý jsme zasáhli do článku a zdrojových kódů více než je obvyklé u běžné redakční práce. Na naši obranu píšeme, že recenzní řízení nebylo v prvním kole ve prospěch publikování tohoto článku, a tak jsme se rozhodli v té době vytíženému autorovi pomoci. Co by Češi pro Portugalce neudělali, dodáváme s úsměvy. Bránili jsme

se spoluautorství, jak jsme uměli. Nepomohlo to. Nepublikovat takové problémy s řešeními a myšlenkami by však byl hřích vůči čtenářům Zpravodaje.

Druhý článek je o programu Highlight německého autora André Simonse a také o tom, jak Zpravodaj popostrčil vývoj sazby více programovacích jazyků užitých v jednom zdrojovém souboru. Je toho teď všude plno: HTML+PHP+CSS+..., Lua+TEX, Perl+TEX, Python+TEX a další kombinace. Naváže na články Jiří Demel postřehy z proběhlého květnového ročníku konference TEXperience 2009.

V PF 2010 je užito tří vrstev šifrovaných zdrojových kódů Jana Šustka prohnaných PSTricks, konkrétně balíčkem pst-text. Natočení −20, 0 a +20 stupňů.

Úplný závěr Zpravodaje tvoří připomínka na nové a některé starší a stále zajímavé knihy.

Obrázek na přední straně obálky je upravená verze z článku o kresbě symbolu Shri Yantra. Obálku uzavírá Michal Mádr překladem pozvánky Karla Berryho na konferenci TUG 2010. Autorům a překladateli děkujeme za podklady.

Pánům Petru Aubrechtovi a Davidu Cattovi děkujeme za nezávislé pročtení tohoto čísla Zpravodaje. Petru Březinovi děkujeme za řadu podnětů.

Co se letos Čechoslovákům podařilo? Při troše štěstí se v tomto Zpravodaji (případně příštím) můžeme těšit na DVD TEX Collection 2009. Nebojme se však instalovat TEX přes Internet. TEX Live instalátor bude od roku 2009 s plnou českou a slovenskou podporou. Velké díky za to patří Jánu Bušovi, Jano Kulovi a Michalu Mádrovi. Vše nutné vyřešili přímo s autorem TEX Live, tedy s Norbertem Preiningem z Rakouska.

Za údržbu Příručky TEX Live, CS v1.34, děkujeme Jánu Bušovi, Petru Sojkovi a Michalu Mádrovi, poděkování za jazykovou korekturu patří Karlu Píškovi.

Díky Petru Tesaříkovi se úspěšně podařilo zařadit českou a slovenskou podporu do balíčku babel. Aktivní divis se dá lokálně či globálně vypnout pomocí příkazu `\shorthandoff{-}` nebo `\catcode‘\-=12`. Zpětná aktivace lze zrealizovat buď příkazem `\shorthandon{-}` nebo `\catcode‘\-=13`.

Nadále se díky členské základně daří pomáhat mezinárodním projektům (Latin Modern, TEX Gyre, LuaTEX a MPlib). Proběhly konference DML 2009 a TEXperience 2009. V TUGboatu 1/2009 se objevily články Jana Přichystala (TEXonWeb) a Víta Zýky (Current typesetting position in pdfTEX). Ve druhém vydání „The LATEX Graphics Companion“ jsou na straně 825 citováni Jan Holeček, Petr Sojka a na straně 834 Jiří Zlatuška. V srpnové elektronické verzi „The XETEX Companion“ je mezi korektory zmíněn Karel Píška.

Na „The TEX showcase“, `http://tug.org/texshowcase/`, najdeme Therion Stacho Mudráka a Martina Budaje. Na `http://mitek.webpark.cz/boisik/` nebo CTAN.ORG najdeme novou verzi písma Boisik. Pomoc Zdeňka Wagnera byla i letos vidět na mnoha místech doma i ve světě, minimálně u balíčků xindy, polyglossia a předpřípravy knihovny libXau pro Linux distribuce typu Red Hat.

Je toho určitě víc; tohle se podařilo redakci dohledat a s určitostí ověřit. Gratulace, velké díky všem a hodně zdaru v budoucnu!

Co nás čeká v nejbližší době? Právě běží nebo je těsně po valné hromadě CSTUG, která se koná v sobotu 21. 11. 2009 v Brně na půdě Fakulty informatiky Masarykovy univerzity. Přednášejícími jsou Petr Sojka a Michal Růžička s tématem „TEXové technologie pro digitální matematickou knihovnu“.

Řečníci přislíbili z této zajímavé přednášky, ze které přikládáme abstrakt, připravit článek do některého z dalších Zpravodajů. Je se na co těšit.

Abstrakt přednášky

Úvodem bude představen projekt České digitální matematické knihovny (DML-CZ; více viz `http://dml.cz/`), který shromažďuje a v digitální podobě zpřístupňuje recenzovanou matematickou literaturu vydanou v Česku a na Slovensku. Ve workflow projektu se hojně využívá také typografický systém TEX.

Pro všechny dokumenty zveřejněné v knihovně jsou XEILATEXem automaticky generovány předsádky s doplňujícími informacemi a metadaty. Dále bude předvedeno použití pdfTEXu pro vkládání původních TEXových zdrojových textů matematických výrazů jako textu kopírovaného z výsledného PDF po označení jejich vysázené podoby, a dávkové podepisování PDF dokumentu.

Pro snadné předávání nově vydávaných matematických publikací do DML-CZ byl připraven na TEXu založený redakční systém, jehož vedlejším produktem jsou XML metadata pro potřeby digitální knihovny. Jádrem systému je nástroj Tralics (překladač LATEXu do XML), jehož použití bude demonstrováno.

Co se na vás chystá v příštím kalendářním roce 2010? Přibližně v listopadu 2010 budou volby do výboru. Uvažte své zapojení. Než to rozmyslíte, rád bych poděkoval dalším lidem mimo výbor, a to paní Čurdové (sekretariát), panu Vlčkovi (účetní), panu Sekaninovi (revizor) a panu Hálovi (tiskař a revizor).

V roce 2010 budou znovu otevřeny české a slovenské překlady elektronické knihy *The Not So Short Introduction to LATEX 2ε*. Bude snaha dostat výsledek na CTAN a potažmo i do TEX-Collection.

Čeká nás výročí existence TEXu v pořadí 2^5, spojené s konferencí TUG 2010, jíž se zúčastní i sám D. Knuth. V našich krajích se bude slavit dvacáté výročí založení CSTUG s třetí, předběžně plánovanou konferencí TEXperience.

Před tím vším se však členská základna může těšit na číslo časopisu MAPS s příspěvky z konference EuroTEX 2009, `http://www.ntg.nl/EuroTeX2009/`. Děkujeme Jaromíru Kubenovi, že zakoupení sborníku pro členy zařídil. Co vše bude zahrnuto v tomto sborníkovém čísle, už tuší holandští organizátoři, my jen odhadujeme, že srdce Plain TEXisty, LATEXisty a především ConTEXtisty dost zaplesá. Snad poskočí srdíčko i těm LuaTEXistům… Zdar a sílu!

Do nového roku nejen s hrstkou TEXu! Ale i METAPOSTu!
Za výbor a redakci Pavel Stříž, `zpravodaj@cstug.cz`

Rozhovor s Hàn Thế Thànhem, tvůrcem a správcem pdfTEXu

Dave Walden, Hàn Thế Thành

Abstrakt

Tento článek je překladem rozhovoru, který byl poprvé publikován v „koutku rozhovorů“ na stránkách TEX Users Group, `http://tug.org/interviews/`, 24. července 2008. Tázaný i tazatel dali souhlas k překladu a přetištění.

Klíčová slova: TEX, PDF, Adobe, pdfTEX, TEX2PDF, Masarykova univerzita, mikrotypografie, *hz*-program, TEXbook, Web, Web2C, Kpathsea, programovací jazyk C, plain TEX, LATEX, TUGboat, teTEX, TEX Live, Prolog, TUG, vietnamština, písmo, Computer Modern, METAFONT, Type 1, FontLab, MetaFog, InDesign, FMP, a2ac, vnTEX, TDS, CTAN.ORG.

Hàn Thế Thành je autorem pdfTEXu a i po mnoha letech jej stále udržuje a vyvíjí. Rozhovor s ním vedl Dave Walden, člen výboru TUG.

Dave Walden: *Můžete mi říci něco o sobě, věci nesouvisející s TEXem?*

Hàn Thế Thành: Narodil jsem se v roce 1972 ve Vietnamu, kde jsem žil do roku 1990. Tehdy se mi naskytla příležitost studovat v České republice, tehdy ještě Československu. Studoval jsem na Masarykově univerzitě v Brně, od roku 1991 do roku 2001. Získal jsem titul magistr (Mgr.) a později doktorát (Ph.D.) v oboru informatika. Potom jsem se vrátil do Vietnamu a pracoval na Pedagogické univerzitě v Ho Či Minově městě (známém také jako Saigon). Od roku 2006 žiji s manželkou v německém Bielefeldu, kde manželka studuje.

Dave Walden: *Jakou práci jste na Pedagogické univerzitě dělal?*

Thành: Učil jsem úvod do programování a pracoval jsem jako správce sítě.

Dave Walden: *Můžete říci, kdy a jak jste se poprvé setkal s TEXem?*

Hàn Thế Thành: Během prvních let na Masarykově univerzitě jsem od spolužáků občas o TEXu slýchával – že je to skvělý sázecí systém, mocný, ale také těžký na zvládnutí. Ale sám jsem TEX vůbec nepoužíval až do doby, kdy jsem

*This is a translation of the interview which was first published in the TEX Users Group's Interview Corner, `http://tug.org/interviews/`, on July 24, 2008. Reprinted by permission of the interviewee and interviewer. Translation and corrections by Michal Mádr and Pavel Stříž.

si ve čtvrtém ročníku musel vybrat téma diplomové práce. Témat na výběr bylo hodně, já jsem si zvolil „Automatizované sázecí systémy“ nebo tak nějak. Představa vedoucího práce byla, že přepíšu TEX pomocí programovacího jazyka vyšší úrovně. Později se ukázalo, že by to pro studenta byl příliš obtížný úkol, a tak vedoucí téma změnil na „Sázecí systém TEX a PDF“. Cílem bylo vytvořit to, co pdfTEX dnes skutečně dělá – upravit TEX tak, aby jako svůj výstup produkoval PDF. Nejdříve jsem se ale musel s TEXem vůbec seznámit – do té doby jsem o něm slyšel, ale nikdy jsem ho nepoužil. Tohle se odehrálo v roce 1994.

Dave Walden: *Dalo by se říci, že klíčem k dnešní životaschopnosti TEXu je existence právě programu pdfTEX. Z toho, co jste řekl, se zdá, že jste se k práci na pdfTEXu, který dnes používá každý, dostal náhodou – že vás vedoucí diplomové práce nasměroval, spíš než že byste o tuto oblast měl zájem vy sám. Je to tak?*

Hàn Thế Thành: Ano, takto opravdu začátky pdfTEXu vypadaly. Můj vedoucí, profesor Jiří Zlatuška, byl v té době velmi aktivním uživatelem i programátorem TEXu. Jiří je také fanouškem logického programování. Jeho původním záměrem tedy bylo, abych TEX přepsal pomocí deklarativního jazyka typu Prolog a výsledek použil pro další vývoj. Sotva jsem tehdy tušil, co to bude obnášet. Téma jsem si vybral, protože: 1) logické programování se mi líbilo; 2) podle toho, co jsem o TEXu slyšel, vypadalo téma zajímavě; 3) zajímavější téma k dispozici nebylo.

Po několika měsících experimentování s přepsáním TEXu bylo mně i Jiřímu jasné, že implementace TEXu v Prologu by pro mě byla příliš obtížná. Jednoho dne mě Jiří pozval k sobě do kanceláře, dal mi vytištěnou specifikaci PDF verze 1.0 a řekl, že bych mohl upravit TEX tak, aby produkoval výstup v PDF (později jsem se dozvěděl, že tento nápad vznikl v diskuzích Jiřího s Philem Taylorem a Donaldem Knuthem na Stanfordské univerzitě.). Tento nápad se mi líbil, navíc původní zadání se zdálo nerealistické. Takže jsme náš plán změnili. Začal jsem pročítat specifikaci PDF a zjišťovat, jak přizpůsobit TEX s Knuthovým systémem Web, Web2C, Kpathsea a dalšími. Po několika měsících jsem byl schopen vytvořit PDF soubor „Ahoj, světe!“ z TEXového zdroje – to byl pro nás docela vzrušující okamžik. Ale že dnes bude pdfTEX tak rozšířený, mě tehdy nenapadlo. (A myslím, že ani Jiřího, i když se můžu mýlit.)

TEX jsem se učil obtížně: Začal jsem čtením knihy TEXbook, kterou mi na začátku Jiří dal. Potom jsem začal používat plain TEX, protože jsem slyšel, že LATEX není tak dobrý v případě, že se chcete naučit detaily TEXu – kontrolovat každou část sazby atd. Takže jsem pomocí plain TEXu sázel periodika, diplomové práce kamarádů a další příležitostné materiály. Ale později jsem začal LATEX používat, protože dělat všechno v plainu je docela komplikované. Takže většinou dávám přednost LATEXu a plain TEX používám jen pro pár specifických aplikací, na které se hodí lépe.

Učení se formátu PDF nebylo příliš obtížné, protože specifikace PDF verze 1.0 se vlezla do útlé knížečky. (Kdybych ale musel vycházet z verze 1.3 nebo některé

pozdější, hned bych to vzdal.) Ale učení se Webovým změnovým souborům, Web2C a podobným věcem bylo docela obtížné: musí se udělat příliš mnoho kroků a v případě problémů není lehké najít chybu.

Jiří chtěl, abych se držel přístupu literárního (kultivovaného) programování a všechno dělal pomocí mechanismu změnových souborů. To ale bylo čím dál obtížnější, a tak jsem se rozhodl naprogramovat část věcí v C. Pro rozhodování, co ponechat v Pascalovském Webu a co naprogramovat v C, jsem použil toto kritérium: věci týkající se „back-endu“ se udělají v C, zbytek se udělá ve Webu. Jiřímu se tento přístup moc nelíbil, ale víceméně ho akceptoval (přinejmenším mě nechal ho zrealizovat).

Dave Walden: *Dokončil jste fungující verzi pdfTEXu už v rámci diplomové práce, nebo až později?*

Hàn Thế Thành: Už si nepamatuji, kdy přesně se pdfTEX stal „použitelným“, protože jeho vývoj byl pozvolný a v různých oblastech k němu přispívalo mnoho lidí. Ve chvíli, kdy jsem dokončil diplomovou práci, byl pdfTEX víceméně ve stavu popsaném v článku Petra Sojky: podpora pro začleněné fonty Type 1, virtuální fonty, odkazy, LZW kompresi (později byla místo LZW použita ZIP-komprese). Ale vkládání obrázků ještě podporováno nebylo!

Dave Walden: *Máte na mysli článek „The Joy of TEX2PDF – Acrobatics with an Alternative to DVI Format“ od Petra Sojky, vás a Jiřího, který vyšel v roce 1996 v TUGboatu, ročníku 17, čísle 3?*

Hàn Thế Thành: Ano.

Dave Walden: *Jak se o vašem programu pdfTEX dozvěděli lidé okolo TEXu a jak došlo k tomu, že byl zařazen do všech TEXových distribucí?*

Hàn Thế Thành: Poprvé jsem se do kontaktu s lidmi okolo TEXu dostal poté, co si Jiří ohledně pdfTEXu (tehdy ještě stále nazývaném TEX2PDF) dopisoval se Sebastianem Rahtzem. Sebastiana pdfTEX zaujal, začal si s ním hrát a různě ho podporovat: založil mailing list pdfTEXu, přeložil a testoval pdfTEX na dalších platformách, seznámil s ním další TEXové uživatele atd. Sebastian vývoji pdfTEXu v jeho začátcích hodně pomohl.

Později byl pdfTEX ukázán Donaldu Knuthovi během jeho návštěvy Masarykovy univerzity a Knuth se o něm vyjádřil pochvalně. To pro nás (mě a Jiřího) bylo velmi povzbudivé. Prvním článkem na téma pdfTEX byl ten, který jsem už zmínil, jehož autorem byl Petr Sojka. Mailing list pdfTEXu byl v počátcích velmi užitečným místem pro diskuze o vývoji pdfTEXu. A jednoho dne se na mailing listu objevil Hans Hagen, začal s pdfTEXem experimentovat, upozorňovat na problémy, diskutovat nové nápady a rysy atd., což mělo na pdfTEX také velký vliv. To, že se pdfTEX líbil tak známým a aktivním členům TEXové komunity (Sebastianovi, Hansovi atd.), bylo klíčové pro to, aby se o pdfTEXu vědělo více.

Potom ho Thomas Esser zařadil do teTEXu. A jakmile se něco dostane tam, obvykle to pak bude akceptováno i ostatními TEXovými systémy.

Dave Walden: *Můžete vyjasnit rozdíl (pokud nějaký je) mezi pdfTEXem a microtypografickým rozšířením TEXu, které bylo popsáno ve vaší disertační práci a publikováno v TUGboatu jako článek „Microtypographic extensions to the TEX typesetting system“?*

Hàn Thế Thành: Cílem mé diplomové práce bylo umožnit, aby TEX produkoval výstup v PDF. Když jsem začínal své doktorské studium (Jiří byl opět mým vedoucím), věděli jsme jen, že budeme pracovat na něčem týkajícím se sazby. Během zhruba prvního roku mého doktorského studia jsem stále vyvíjel pdfTEX a zároveň se poohlížel po tématu disertační práce. Pár témat mě napadlo, potom ale Jiří rozhodl, že zůstaneme u mikrotypografických rozšíření. Myslím, že to opět bylo rozumné rozhodnutí.

Dave Walden: *Řekl jste, že jste na Masarykově univerzitě studoval v letech 1991 až 2001. Můžete upřesnit, kdy jste ukončil magisterské studium a kdy začalo vaše doktorské studium?*

Hàn Thế Thành: Magisterské studium jsem dokončil v létě 1996. O pár měsíců později jsem začal své doktorské studium.

Dave Walden: *Můžete blíže popsat váš přístup k studiu mikrotypografie, k tvorbě nutných rozšíření pdfTEXu a k výzkumu komponent nezbytných pro úspěšnou disertační práci?*

Hàn Thế Thành: Necítím se moc kvalifikovaný radit, jak udělat úspěšnou disertační práci, protože se svou vlastní jsem zápasil.

Dave Walden: *Promiňte, nechtěl jsem vám připomínat stresující okamžiky. Zajímalo mě jen, jak jste se naučil to, co jste potřeboval vědět o mikrotypografii.*

Hàn Thế Thành: Už si přesně nepamatuji, jak jsem mikrotypografii studoval – byl to pozvolný proces, asi jako pro většinu lidí. Začal jsem čtením knih a článků a potom jsem hledal další relevantní zdroje informací. Ty nejužitečnější, na které si vzpomínám, byly články Hermanna Zapfa „About microtypograhy and the *hz*-program“ a brožura o *hz*-programu od URW, německé písmolijny. Také jsem hodně experimentoval s InDesignem firmy Adobe, o kterém se tvrdí, že obsahuje integrované moduly *hz*-programu. Je zajímavé, že některé ideje *hz*-programu byly původně inspirovány TEXem samotným.

Dave Walden: *Vím, že než jste dokončil doktorské studium, byl už pdfTEX „zhruba“ funkční. Pokračoval jste v jeho vývoji i po návratu do Vietnamu?*

Hàn Thế Thành: Po návratu do Vietnamu jsem kvůli problémům s přístupem k síti a dalšími věcmi na dlouhou dobu s vývojem pdfTEXu přestal. Později jsem příležitostně našel čas dělat drobná rozšíření, ale už to nebyl aktivní vývoj

jako dříve. Na pdfTEXu samozřejmě pracovali také další lidé. Nejvýznamnější příspěvky v minulých letech udělal Hartmut Henkel, velmi tichý člověk. Jeho opravy pdfTEX výrazně vylepšily v mnoha ohledech: v rychlosti, stabilitě, čistotě a přehlednosti zdrojového kódu, lepší funkcionalitě atd.

Dave Walden: *Zdá se, že se nyní na vývoji pdfTEXu opět podílíte výrazněji.*

Hàn Thế Thành: Ano, od chvíle, kdy jsem se s manželkou přesunul do Německa, jsem s pdfTEXem v intenzivnějším kontaktu. V Německu pracuji z domu, jako konzultant společnosti River Valley Technologies. Dělám podporu síťové administrativy, automatizování editačních procesů a také nasazení pdfTEXu.

Dave Walden: *Předpokládám, že vývoj pdfTEXu funguje v jistém smyslu jako vývoj Open Source Softwaru. A minulý týden jste mi napsal, že musíme interview na pár dní přerušit, protože Karl Berry po vás chce rychlou opravu v pdfTEXu pro připravované vydání TEX Live. Můžete přiblížit, jak je vývoj pdfTEXu organizován a koordinován?*

Hàn Thế Thành: Vývoj pdfTEXu se časem ustálil a vypadá zhruba takto: projekt pdfTEX má zřízenou stránku na `http://sarovar.org/`, kam lidé hlásí chyby, požadavky na novou funkcionalitu a opravy. Pro zájemce o vývoj pdfTEXu existuje také mailing list. A pak je zde vývojářské jádro (Hans Hagen, Taco Hoekwater, Hartmut Henkel, Martin Schröder a já), kde pdfTEXové záležitosti diskutujeme.

Dave Walden: *Jestli se nemýlím, strávil jste hodně času tvorbou podpory pro vietnamštinu u mnoha písem. Jak jste se k tomu dostal a jak postupujete?*

Hàn Thế Thành: Ve chvíli, kdy jsem se TEX učil, jsem ho mimo jiné zamýšlel použít i na sazbu vietnamštiny. V té době už existoval balík nazvaný vcmr od Wernera Lemberga, který poskytoval docela dobrou podporu. Mně se ale nelíbily tvary vietnamských znaků, a tak jsem se rozhodl tyto znaky sám přidat do písem Computer Modern. Byla to pro mě jen zábava, nemám žádnou uměleckou minulost. Učil jsem se z existujících písem, z textů, ke kterým jsem se dostal a z komentářů od zkušených lidí. Vietnamská písmena jsem do písem Computer Modern přidal pomocí METAFONTu. Ke konverzi fontů do formátu Type 1 jsem zkombinoval několik nástrojů: MetaFog, FMP (od YandY), a2ac a své vlastní skripty v Perlu. Stejné nástroje jsem použil pro přidání vietnamských písmen do existujících Type 1 písem, akcenty jsem však nakreslil pomocí FontLabu. Na vnTEXu se podílí další lidé: Werner Lemberg a Vladimir Volovich (podpora pro LATEX) a Reinhard Kotucha (testování a údržba balíku, kompatibilita s TDS a poskytování věcí vyžadovaných od TEX Live a CTAN.ORG, např. soubor README, copyright atd.) Vývoj vnTEXu už nepokračuje, protože už bylo vytvořeno docela velké množství písem. Existuje dokonce vietnamský překlad průzkumu o matematických písmech pro TEX od Stephena Hartkeho, což dokládá, že většina písem zmíněných v průzkumu má verzi pro vietnamštinu.

Dave Walden: *Děkuji, Thànhe, že jste si našel čas na náš rozhovor. Bylo mi ctí být v kontaktu s někým, kdo má tak výraznou zásluhu na trvající oblibě TEXu.*

∼ ∼ ∼

Lidé kolem pdfTEXu: Hàn Thế Thành, Jiří Zlatuška, Phil Taylor, Donald Ervin Knuth, Petr Sojka, Sebastian Rahtz, Hans Hagen, Thomas Esser, Hermann Zapf, Hartmut Henkel, Karl Berry, Taco Hoekwater, Martin Schröder, Werner Lemberg, Vladimir Volovich, Reinhard Kotucha, Stephen Hartke, Pavel Janík, Heiko Oberdiek, Jiří Osoba, Ricardo Balbino Sánchez Cármenes, Robert Schlicht.

∼ ∼ ∼

Závěrečná poznámka: Při kontrole pořízeného rozhovoru Thành upozornil na tyto důležité spolutvůrce pdfTEXu: Pavel Janík přidal podporu pro TIFF (která byla později odstraněna); Heiko Oberdiek přidal podporu pro color stack [uložení aktuální barvy při přechodu na novou stranu PDF]; Jiří Osoba přidal podporu JPG, Ricardo Sánchez Cármenes přidal podporu šifrování (později odstraněnou); Robert Schlicht vytvořil LATEXový balík s podporou pro mikrotypografii; Martin Schröder po mnoho let udržoval pdfTEX. A samozřejmě, pdfTEX je jen rozšířením programu TEX. Kdyby Donald Knuth nenapsal TEX, nevznikl by ani pdfTEX. Pokud jsme na někoho zapomněli, velice se omlouváme!

Z pdfTEXu vychází ambiciózní projekt LuaTEX, který bude postupně představován v dalších článcích Zpravodaje.

Zmíněná a doporučená literatura

[1] *Interview with Hàn Thế Thành.* [Interview s Hàn Thế Thànhem.] [online, vytvořeno 24. 7. 2008] Dostupné z URL: `http://www.tug.org/interviews/interview-files/han-the-thanh.html`

[2] *TEX People: Interviews from the World of TEX.* [Mistři TEXu: Rozhovory s lidmi ze světa TEXu.] Berry, Karl (editor); Walden, David (editor). USA, TEX Users Group, 2009. 312 stran. ISBN 9780982462607. Více informací o knize na URL: `http://tug.org/interviews/book/`

[3] Sojka, Petr; Thế Thành, Hàn; Zlatuška, Jiří. The Joy of TEX2PDF – Acrobatics with an Alternative to DVI Format. [Úspěchy programu TEX2PDF – akrobacie s alternativním výstupem do formátu DVI.] *TUGboat*, vol. 17(**3**): 244–251, 1996. ISSN 0896-3207. Článek je dostupný z URL: `http://www.tug.org/TUGboat/Articles/tb17-3/tb52sojk.pdf`

[4] Thế Thành, Hàn. Experience from a Real-World Application of Micro-Typography with pdfTEX. [Zkušenosti s mikrotypografií v pdfTEXu z reálných úloh.] *The Asian Journal of TEX,* vol. 2(**1**): 1–10, Duben 2008. Proceedings of the Asian TEX Conference 2008, the Korean TEX Society. ISSN 1976-1228. Dostupné z URL: `http://ajt.ktug.kr/assets/2008/5/1/0201thanh.pdf`

[5] Thế Thành, Hàn. Experiences with Micro-Typographic Extensions of pdfTEX in Practice. [Praktické zkušenosti s mikrotypografickými rozšířeními pdfTEXu.] *Die TEXnische Komödie*, vol. 18(**2**): 159–164, 2006. Deutschsprachige Anwendervereinigung TEX e.V. Proceedings EuroTEX2005, Pont-à-Mousson, Francie. ISSN 1434-5897. Dostupné z URL: `http://www.tug.org/TUGboat/Articles/tb27-0/tb85complete.pdf`

[6] Thế Thành, Hàn. Margin Kerning and Font Expansion with pdfTEX. [Optické vyrovnávání okrajů a natahování znaků pomocí pdfTEXu.] *TUGBoat*, vol. 22(**3**): 146–148, 2001. Proceedings of the 2001 Annual Meeting. ISSN 0896--3207. Dostupné na URL: `http://www.tug.org/TUGboat/Articles/tb223/tb72thanh.pdf`

[7] Thế Thành, Hàn. Micro-typographic extensions to the TEX typesetting system. [Mikrotypografická rozšíření v typografickém systému TEX.] Disertační práce, Masarykova university v Brně, 2000. Školitel práce: Jiří Zlatuška. Přetištěno v časopisu *TUGboat*, vol. 21(**4**): 317–434, 2000. ISSN 0896-3207. Dostupné na URL: `http://www.tug.org/TUGboat/Articles/tb21-4/tb69thanh.pdf`

Summary: Interview with Hàn Thế Thành, the Creator and Maintainer of the pdfTEX

Translation of an interview conducted by Dave Walden with Hàn Thế Thành, the creator and maintainer of the pdfTEX program. Discussed were the origins of the program pdfTEX, its early development, introduction to the wider TEX community and the organization of the current development and maintenance.

Key words: TEX, PDF, Adobe, pdfTEX, TEX2PDF, Masaryk University, Microtypography, *hz*-program, TEXbook, Web, Web2C, Kpathsea, the programming language C, plain TEX, LATEX, TUGboat, teTEX, TEX Live, Prolog, TUG, Vietnamese, Typeface, Computer Modern, METAFONT, Type 1, FontLab, MetaFog, InDesign, FMP, a2ac, vnTEX, TDS, CTAN.ORG.

Překlad a dokončení:
Michal Mádr, `m.madr@seznam.cz`
Pavel Stříž, `bulletin@cstug.cz`
CSTUG c/o FEL ČVUT, Technická 2
Praha, CZ-166 27, Czech Republic

Představení LuaTEXu

Michal Mádr, Pavel Stříž

Abstrakt

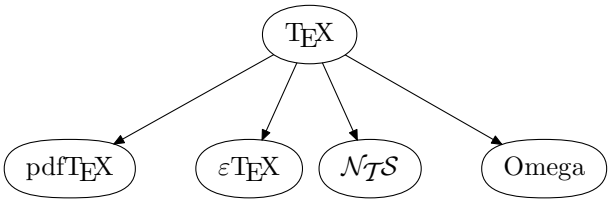
Tento článek ve stručnosti představuje program LuaTEX [čtěte luatech nebo také luáhtech] – ambiciózního následníka programu pdfTEX. Článek se snaží zasadit LuaTEX do kontextu následnických programů TEXu a představit nové možnosti, které tento systém již přináší a měl by v budoucnu přinést.

Většina informací v tomto článku byla čerpána z webové stránky projektu LuaTEX, `http://www.luatex.org/`, s vřelým souhlasem jejich tvůrců.

Klíčová slova: LuaTEX, Lua, pdfTEX, Omega, Aleph, $\mathcal{NTS}$, εTEX, ConTEXt, Oriental TEX, XeTEX, Unicode, OpenType.

LuaTEX v kontextu dění

V roce 1990 v článku [2] v časopisu TUGboat ohlásil Donald Knuth ukončení nového vývoje programu TEX – nadále bude jen opravovat závažné chyby. Kdokoliv bude moci použít zdrojové kódy TEXu k vytvoření systému s přidanou funkcionalitou, ale takovému novému systému musí dát jiné jméno než TEX.



Obrázek 1: Hlavní systémy vycházející z TEXu (rok 2000).

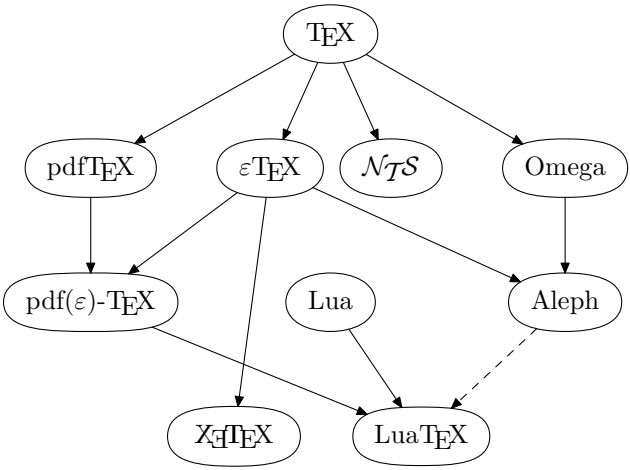
O vytvoření nového TEXového systému se od té doby pokusila řada projektů. Tyto projekty se pokusily TEX vylepšit v různých směrech a s různým úspěchem. Obrázek 1 ukazuje hlavní následnické TEXové projekty na přelomu tisíciletí. Podobně to popsal i Petr Olšák v kapitole 3.8 „Následníci TEXu“ v knize *Typografický systém TEX* [3] z roku 2000.

*CS TUG finančně podpořil vznik LuaTEXu, knihovny MPlib a také vznik plnohodnotných rodin písem Latin Modern a TEX Gyre. Za postřehy k článku děkujeme pánům Vítu Zýkovi, Arthuru Reutenauerovi, Karlu Horákovi a Petru Aubrechtovi.

Jedním z těchto následníků byl systém Omega, který přinesl podporu pro zpracování vícejazyčných textů zapsaných v různých kódováních. Omega umožnila kvalitní sazbu nelatinkových jazyků včetně sazby zprava-doleva. Omega vnitřně kódovala znaky podle Unicode a byla schopná pracovat s fonty obsahujícími přes 65 tisíc znaků. Omega přinesla řadu inovací, např. konfigurovatelný překlad znaků ze vstupního souboru do (svého vnitřního kódování podle) Unicode[1].

Dalšími dvěma následnickými systémy byly produkty projektu $\mathcal{NTS}$ (New Typesetting System), který usiloval o vylepšení a překlopení jádra TEXu do jazyka Java, s využitím téměř dvacetileté zkušenosti s používáním TEXu. Jako vedlejší produkt systému $\mathcal{NTS}$ vznikl εTEX – sada TEXových rozšíření, zejména o podporu pravolevé sazby, zvýšení počtu registrů z 256 na 65 636, usnadnění expanze a mírné zjednodušení trasování.

Poslední z následníků byl aktivitou Hàn Thế Thành, jehož program pdfTEX přinesl výstup ve formátu PDF a podporu mikrotypografických rozšíření a brzy si získal popularitu, která trvá dodnes. Více viz předchozí článek.



Obrázek 2: Hlavní systémy vycházející z TEXu (začátek roku 2010).

Obrázek 2 ukazuje, jak vypadá situace dnes. Projekt $\mathcal{NTS}$, který dále nepokračuje, nedosáhl prakticky použitelných výsledků. εTEXová rozšíření úspěšná byla a jsou použita většinou formátů současných TEXových distribucí. Vývoj systému Omega ani jeho následníka Aleph už také nepokračuje. Aleph oproti Omega přinesl některá vylepšení, např. integraci právě s εTEXem. Systém pdfTEX se

[1] Vstupní text byl předzpracován externím programem a potom teprve načten jádrem TEXu.

v TEXovém světě stal de facto standardem. Po integraci s εTEXem je také označován jako pdf(ε)-TEX. Na scéně se mezitím objevil další významný TEXový následník, XƎTEX, ke kterému se vrátíme na konci článku. Podrobněji o historii TEXu viz článek [4].

Tým vývojářů pdfTEXu oznámil (podobně jako D. Knuth v roce 1990), že po vydání příští velké verze, 1.50.0, budou vydávány už jen opravy. pdfTEX tedy bude „zakonzervován“ a v implementování nové funkcionality předá štafetu novému programu s názvem LuaTEX. Základní tým vývojářů LuaTEXu je tvořen Taco Hoekwaterem, Hansem Hagenem a Hartmutem Hankelem. Personální propojení s pdfTEXem je těsné – všichni tři byli také součástí základního vývojového týmu pdfTEXu, navíc autor pdfTEXu, Hàn Thế Thành, je jedním ze spolupracovníků LuaTEXu. Součástí projektu LuaTEX bude integrace řady rysů ze systémů Omega/Aleph (kódování podle Unicode, podpora rozsáhlých fontů, konfigurovatelný překlad vstupního textu, sazba v různých směrech), čímž dojde k propojení původních následníků z Obrázku 1 ze strany 191.

Cíle LuaTEXu

LuaTEX si klade velké cíle. Mezi ty hlavní patří přidání podpory pro OpenType[2] fonty, přidání podpory pro rysy implementované v systému Omega/Aleph a přidání podpory pro rozšiřitelnost – tvůrci maker budou mít možnost vstoupit do činností LuaTEXu a změnit jeho chování. V rámci LuaTEXu tedy bude možné dělat změny, které dosud byly možné jen změnou ve zdrojových kódech příslušného TEXového programu, a tím vlastně tvorbou nového TEXového systému.

Do LuaTEXu byl integrován skriptovací jazyk Lua[3]. Konečně tedy bude možno psát složitější algoritmy elegantně s využitím regulárních výrazů, seznamových datových struktur a podobných nástrojů běžných v populárních programovacích jazycích dneška.

Funkce napsané v jazyce Lua jsou rovněž klíčové pro rozšiřitelnost LuaTEXu, kterou jsme zmínili před chvílí. Tvůrci maker mohou pomocí svého kódu, napsaného v jazyce Lua, měnit činnosti vykonávané LuaTEXem[4]: Uživatel vytvoří Lua funkci a specifikuje LuaTEXu, v které fázi zpracování dokumentu má být tato funkce LuaTEXem vyvolána. Seznam fází, ve kterých si uživatel může

[2] Formát fontů vyvinutý firmami Adobe Systems a Microsoft nahrazující starší formáty těchto firem, Type 1, resp. True Type. Kódování znaků fontu je postaveno na kódování Unicode, font může obsahovat až 65 536 znaků. OpenType fonty mohou obsahovat pokročilou typografickou podporu.

[3] Jedná se v poslední době o populární, relativně jednoduchý jazyk. Na stránce `http://www.luatex.org/languages.html` autoři popisují své experimenty s dalšími jazyky a vysvětlují, proč se nakonec rozhodli právě pro Lua, více o jazyku na oficiálních stránkách `http://lua.org/` nebo v knihách zmíněných v závěru tohoto Zpravodaje.

[4] Měnit lze jak sazbu, tak např. vyhledávání souborů nebo tokenizaci vstupních dat.

specifikovat vyvolání své funkce, je popsán v referenčním manuálu LuaTEXu, v kapitole „The callback library“.

Pro každou z těchto fází manuál popisuje, jaká vstupní data budou vyvolávané Lua funkci předána jako její argumenty. Specifikujeme-li např. LuaTEXu, aby naši Lua funkci vyvolal jako `pre_linebreak_filter`[5], naše funkce bude vyvolána vždy předtím, než LuaTEX začne provádět algoritmus řádkového zlomu. Jako vstup funkce dostane seznam nodů příslušného horizontálního seznamu. Může tento seznam modifikovat a tím měnit vstupní data pro následující algoritmy. Konkrétně funkce zaregistrovaná jako `pre_linebreak_filter` může měnit vstup pro následující algoritmus řádkového zlomu.

Následuje seznam cílů deklarovaný autory LuaTEXu pro verzi 1.00:[6]

- Sloučení kódu systému Aleph a pdfTEX tak, aby se LuaTEX v DVI módu choval jako Aleph a v PDF módu jako pdfTEX.
- Zpřístupnění Aleph funkcionality v PDF módu LuaTEXu. Podpora OpenType fontů. Možnost plně kontrolovat všechny aspekty načítání fontů, jejich definice a manipulace s nimi pomocí Lua kódu.
- Zpřístupnění různých vnitřních dat LuaTEXu pomocí Lua callbacků. Úplný přístup k procesu tvorby tokenů. Přístup k seznamům uzlů „v užitečných chvílích“. Např. před startem algoritmu řádkového zlomu.
- Zpřístupnění kontroly různých aspektů tvorby odstavce, jako je dělení slov, tvorba kerningů a ligatur. Dynamické načítání vzorů pro dělení slov.
- Transformování METAPOSTu do knihovny MPlib použité LuaTEXem. Rozšíření této knihovny pro podporu libovolné přesnosti (MegaPost).[7]
- Identifikace grafických objektů a jejich načítání pomocí jazyka Lua.
- Sazba v různých směrech: Odstranění chyb v Aleph a přehodnocení backendu. Víceméně bude použito řešení přítomné v systému Omega.
- Speciální vlastnosti: Pročištění *hz*-optimalizace a vystrkování znaků. Očekávají se různé způsoby optimalizace.
- Podpora pro Unicode v matematickém módu. Více kontroly pro zobrazování. Podpora specifikace OpenType math firmy Microsoft. Nová primitiva pro některé z rysů v současnosti řešených pomocí maker.
- Kontrola procesu tvorby stránky a přístup k relevantním vnitřním datům. Zpřístupnění insertů.
- Přístup k většině možností formátu PDF, mezi které patří anotace, zacházení a manipulace s objekty.

[5] Tj. registrujeme-li si naši funkci jako `pre_linebreak_filter`.

[6] Více viz webová stránka `http://www.luatex.org/roadmap.html`. Naleznete zde seznam cílů a informace o stavu jejich plnění.

[7] Pokročilé LuaTEXové použití, např. runtime generování fontů, zatím nebylo naplánováno.

Časový plán

V této chvíli je projekt zhruba v polovině vývoje verze 1.00, jejíž vydání je plánováno na rok 2012. Už nyní je ale dle autorů LuaTEX dostatečně stabilní ke kontrolovanému použití. LuaTEXová rozhraní se ještě mohou měnit, s výjimkou těch, která jsou zdokumentována jako stabilní. První zdokumentovaná stabilní rozhraní by se měla objevit ve verzi 0.50.

Aktuální verzi LuaTEXu pro řadu operačních systémů je možno získat na stránce projektu `http://www.luatex.org/download.html` nebo např. pomocí balíku *luatex* linuxové distribuce Debian. Konkrétně programy LuaTEX, METAPOST a MPlib, včetně fóra, jsou dostupné na `http://foundry.supelec.fr/gf/project/luatex` a `http://foundry.supelec.fr/gf/project/metapost`.

Po vydání verze 1.00 budou formulovány cíle pro verzi 2.00. Vybíráme některé důležité verze programu:

- 0.00: První demonstrace LuaTEXu na TUG 2005 v čínském Wuhanu.
- 0.10: První hlavní verze představená na konferenci TUG 2007 v San Diegu (USA) demonstrovala první pokus o sazbu arabštiny.
- 0.40: Konec roku 2009: Verze zařazená do TEX Live 2009.
- 0.50: „Na půli cesty.“ První zdokumentovaná stabilní rozhraní.
- 1.00: Plánována na rok 2012.

LuaTEX vs. XETEX

Vraťme se k Obrázku 2 ze strany 192, konkrétně k systému XETEX [čtěte zítech] Jonathana Kewa [čtěte džonathana kjúa]. XETEX je vedle LuaTEXu v současnosti považován za druhého významného následníka TEXu. Rysy obou systémů se překrývají (podpora zpracování vícejazyčných textů a OpenType fontů), ale rozdíl je v implementaci a snadnosti použití koncovými uživateli.

XETEX implementuje nové rysy pomocí knihoven třetích stran (výsledkem je rychlejší vývoj systému), zatímco LuaTEX implementuje tyto záležitosti ve vlastní režii (má tedy lepší kontrolu nad kvalitou sazby). Použití XETEXu (např. pro sazbu neevropských jazyků) je přímočaré, zatímco LuaTEX poskytuje pouze rámec a podporu konečným uživatelům musí zajistit tvůrci maker.

XETEX by měl být představen v některém z dalších čísel Zpravodaje. Na úvod doporučujeme prezentaci, `http://cstug.cz/aktivity/2007/CSTUG-talk.pdf`, a přednášku uskutečněnou v Brně v roce 2007, `http://www.video.muni.cz/public/xetex/xetex.avi`.

Upozorňujeme také na doplněk „The XETEX Companion“, tj. `http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf`.

ConTEXt MkIV

Jak autoři zdůrazňují, LuaTEX sice přináší mnoho nové funkcionality, ale je potřeba vytvořit balíky maker, které tuto funkcionalitu zpřístupní uživatelům. Protože vývojáři LuaTEXu jsou zároveň členy vývojářského týmu ConTEXtu, experimentální verze ConTEXt MkIV je nejspíš prvním systémem maker, který novou funkcionalitu LuaTEXu využívá a intenzivně testuje. V rámci projektu Oriental TEX byla např. do ConTEXtu zabudována pokročilá pravolevá sazba implementovaná pomocí Lua funkcí – demonstrace rozšiřitelnosti LuaTEXu.

TEX Live 2009 a LuaTEX

Ukážeme si, jak lze několika kroky nainstalovat (Pretest) TEX Live 2009, vyzkoušet funkčnost LuaTEXu v Plain TEXu i LATEXu, a navíc rozšířit instalaci o LuaTEX pod Mark IV, tj. v ConTEXtu. Základem zkoumání nám byly webové stránky tvůrců: `http://wiki.contextgarden.net/Running_Mark_IV`.

Microsoft Windows XP

Důvodem primárního zájmu o Microsoft Windows je snaha obejít názvy adresářů a souborů s mezerami. Váháte-li mezi MiKTEXem a TEX Live, tak vězte, že TEX Live obsahuje více balíčků.

Stáhli jsme si `install-tl.zip` z `http://ftp.cstug.cz/pub/tex/tlnet/`. Soubor jsme si rozpakovali. V případě Pretest TEX Live 2009 jsme spustili: `install-tl.bat -repository http://ftp.cstug.cz/pub/tex/tlnet/`.

Pozn. U konečné verze TEX Live vám bude stačit spustit `install-tl.bat`.

Zaškrtli jsme si volbu instalace pro všechny uživatele. Systémová cesta se přidala automaticky. Nainstalovali jsme si Ruby z `http://www.ruby-lang.org/en/downloads/`. Během instalace jsme zaškrtli obě možnosti. Aktivujeme systémovou cestu odhlášením a novým přihlášením, případně restartem počítače. Fungování programů lze ověřit: `tex --help` a `ruby --help`.

Běží nám LuaTEXu pro Plain (`luatex soubor`) i LATEX (`lualatex soubor`). ConTEXt běží na Mark II (`texexec soubor`). ConTEXt zahrnutí Lua skriptu ignoruje, u `\directlua` nahlásí chybu. Příkaz `context` nám zatím neběží.

Upravíme `C:/texlive/2009/texmf/web2c/texmf.cnf` tak, abychom nastavili: `HOMETEXMF = C:/temp`. Bylo možné nastavit jinou cestu, nebo se zahrnutím `$USERNAME`, ale to se celý příběh komplikuje.

Dále v adresáři `C:/texlive/2009/texmf/web2c/` vytvoříme nový soubor `texmfcnf.lua` a zapíšeme do něj: `return { TEXMFCACHE = 'C:/temp' }`.

Kvůli nástroji `ctxtools` (zjištěno z log souboru) je nutné nahrát verzi LuaTEXu 0.43 nebo vyšší. Aktuální verze LuaTEXu je beta-0.40. Nainstalujeme

poslední verzi stažením šesti souborů z `http://minimals.contextgarden.net/current/bin/luatex/mswin/` s uložením do `C:/texlive/2009/bin/win32/`.

Následuje zapsání svaté ConTeXt trojice: `ctxtools --update`; následováno příkazem: `texexec --make --all`; poté: `luatools --generate`. Na závěr se vše aktivuje znovuspuštěním `ctxtools --update`.

Nyní lze aktivovat LuaTeX v ConTeXtu užitím: `texexec --lua soubor` nebo `context soubor`.

Heuréka! Mark IV i všechno ostatní žije!

Mandriva Linux 2009.1-i586

Jestli váháte, zda-li zvolit teTeX nebo TeX Live, tak rozdíl je jen v množství nainstalovaných balíčků. TeX Live vede!

Stáhli jsme si `install-tl-unx.tar.gz` z `http://ftp.cstug.cz/pub/tex/tlnet/` a rozbalili jej. Přepli jsme se na administrátora přes `su` a použili `perl install-tl -repository http://ftp.cstug.cz/pub/tex/tlnet/`. Instalovat jsme začali volbou `I`. Pozn. V případě konečné verze TeX Live 2009 vám bude stačit jen `perl install-tl`.

Existuje několik variant přidání systémové cesty. My jsme zvolili editaci souboru `.bashrc` z home adresáře libovolného účtu, kde jsme přidali: `export PATH=$PATH:/usr/local/texlive/2009/bin/i386-linux:`. Tento příkaz jsme aktivovali u administrátora z příkazové řádky, abychom mohli v instalacích pokračovat. Kontrola úpravy proměnné přes: `echo $PATH`.

Pokud nemáme, doinstalujeme si Ruby.

Poslední verzi LuaTeXu získáme stažením tří souborů z `http://minimals.contextgarden.net/current/bin/luatex/linux/` s uložením do `/usr/local/texlive/2009/bin/i386-linux/`.

Vygenerujme formáty pomocí příkazu: `fmtutil-sys --all`.

Následuje zapsání tří příkazů: `ctxtools --update`; následováno příkazem: `texexec --make --all`; poté: `luatools --generate`.

Toť vše, jsme hotoví!

LuaTeX říká: *Nazdar světe!*

Zkusíme si, zda-li LuaTeX reaguje, navíc dvojnásobně. Zjistíme to tak, když se bez chybové hlášky objeví v PDF souboru věta: „Hello World!“

Ukázka v Plain TeXu:

```
%luatex luaaplain.tex
\directlua{tex.print("Hello \directlua{tex.print('World')}")}!
\bye
```

Pod formátem LaTeX vypadá ukázka takto:

```
%lualatex luaalatex.tex
\documentclass{article}
\begin{document}
\directlua{tex.print("Hello \directlua{tex.print('World')}")}!
\end{document}
```

ConTeXt má navíc své příkazy, zde je trojnásobný výstup:

```
%context luaamarkiv.tex % Nebo luaamarkiv.ctx.
\starttext
\directlua{tex.print("Hello \directlua{tex.print('World')}")}!
\ctxlua{tex.print("Hello \ctxlua{tex.print('World')}")}!
\startluacode
tex.print("Hello World!")
\stopluacode\
\stoptext
```

Pokud jste něžné stvoření, přidejte si prosím před příkaz `\stoptext` následující kód vypůjčený z článku *Practical introduction to METAPOST* Clémenta Hurlina, zdrojové kódy viz spodní část webové stránky francouzského autora `http://www-sop.inria.fr/everest/personnel/Clement.Hurlin/`.

```
\startMPcode
u:=0.05cm; pair p[]; path r[]; path coeur;
p[0] = (0,0);      p[1] = (3u,4u);
p[2] = (1.5u,6u);  p[3] = (0,5u);
p[4] = (-1.5u,6u); p[5] = (-3u,4u);
r[1] = p0{curl 0.7}..tension 2..p1{up}..p2{left}..{down}p3;
r[2] = p3{up}..p4{left}..p5{down}..tension 2..{curl 0.7}p0;
coeur= r[1] .. r[2] .. cycle;
fill coeur withcolor red;
\stopMPcode
```

Rozhovory s autory

V knize TeXových rozhovorů [1] představené v tomto čísle jsou zpovídaní i lidé významně spojení s TeXovými následníky zmíněnými v tomto článku, jako jsou Taco Hoekwater, Hans Hagen (oba LuaTeX i pdfTeX), Hàn Thế Thành (pdfTeX a spolupracovník projektu LuaTeX), Philip Taylor ($\mathcal{NTS}$, εTeX), Yannis Haralambous (Omega), či Jonathan Kew (XeTeX). Detaily viz `luatex --credits`.

Několik užitečných odkazů závěrem

Dokumentační sekce stránek projektu dostupná na `http://www.luatex.org/documentation.html` informuje o LuaTEXových aktualitách. Nalezneme zde referenční příručku (snapshot) a celou řadu přednášek a článků.

Kdo to neví, tak vývojáři LuaTEXu, ConTEXtu a knihovny MPlib sídlí na `http://pragma-ade.com/` a `http://wiki.contextgarden.net/`.

Vážní zájemci a studující by si neměli nechat ujít `http://www.pragma-ade.com/general/manuals/mk.pdf`. Je to až slavnostní dokumentace k příležitosti vzniku LuaTEXu verze 0.50.

Diskuze nalezneme na `http://tug.org/mailman/listinfo/luatex/`.

Vážnější chyby lze ohlásit na `http://tracker.luatex.org/`.

Ke studiu lze dále doporučit webové stránky „The Joy of LuaTEX“ od Yannise Haralambouse, `http://luatex.bluwiki.com/`.

Luigi Scarso má také zajímavé a užitečné webové stránky, viz `http://wiki.contextgarden.net/User:Luigi.scarso#Luatex_examples`.

Jeden z aktuálních archivů s komentovanými zdrojovými kódy nalezneme na `http://foundry.supelec.fr/gf/project/modules/scmsvn/`.

Dříve vzniklý depozitář od tvůrců LuaTEXu objevíme v adresáři `/lua/` na `http://context.aanhet.net/svn/manuals/`.

Ukázky

Pokud už s LuaTEXem či ConTEXtem MkIV experimentujete nebo je dokonce již používáte na aktivní sazbu a chtěli byste čtenářům Zpravodaje představit své zdrojové kódy a podělit se o ně, kontaktujte prosím redakci na emailové adrese: `zpravodaj@cstug.cz`.

Seznam literatury

[1] *TEX People: Interviews from the World of TEX.* [Mistři TEXu: Rozhovory s lidmi ze světa TEXu.] Berry, Karl (editor); Walden, David (editor). USA, TEX Users Group, 2009. ISBN 978-0982462607. Rozhovory jsou přístupné i online na URL: `http://www.tug.org/interviews/`

[2] Knuth, Donald Ervin. The Future of TEX and METAFONT. [Budoucnost TEXu a METAFONTu.] *TUGboat,* vol. 11(**4**): 489–489, 1990. Dostupné z URL: `http://tug.org/TUGboat/Articles/tb11-4/tb30knut.pdf`

[3] Olšák, Petr. *Typografický systém TEX.* [Typesetting System TEX.] 2. vyd. Brno, vydavatelství Konvoj, 2000. Kapitola 3.8: Následníci TEXu. 300 stran. ISBN 80-85615-91-6.

[4] Reutenauer, Arthur. A brief history of TEX, volume II. [Stručná historie TEXu, druhé nahlédnutí.] *TUGboat,* vol. 29(**1**): 68–72, 2007. Článek je dostupný i online na URL: `http://tug.org/TUGboat/Articles/tb29-1/tb91reutenauer.pdf`

Summary: Introduction to LuaTEX

This article introduces the program LuaTEX – an ambitious successor of pdfTEX. The article discusses LuaTEX's position among other TEX follow-ups and new features of this system.

Key words: LuaTEX, Lua, pdfTEX, Omega, Aleph, $\mathcal{NTS}$, εTEX, ConTEXt, Oriental TEX, XeTEX, Unicode, OpenType.

Michal Mádr, `m.madr@seznam.cz`
Pavel Stříž, `striz@fame.utb.cz`
ÚSKM FaME UTB ve Zlíně, Mostní 5139
Zlín, CZ-760 01, Czech Republic

Zašifrování zdrojového textu při zachování jeho funkčnosti

Jan Šustek

Abstrakt

Článek ukazuje možnost, jak zašifrovat TEXový soubor tak, aby se navenek choval stejně jako původní soubor. Toho je možné využít například, pokud chceme druhému uživateli poslat balík maker, ale nechceme mu z nějakého důvodu prozradit zdrojový kód těchto maker.

Klíčová slova: Zašifrování souboru, `\uppercase`, Caesarova šifra.

1. Úvod

Při práci v TEXu můžeme stejného výsledku dosáhnout mnoha různými způsoby. Můžeme napsat text bez použití jediného makra. Nebo si naopak práci můžeme zautomatizovat definováním a použitím maker. Tato makra se mohou jmenovat různě a také míra jejich použití může být různá. Makra mohou být napsána čitelně i nečitelně. Zdrojový kód [3] je ukázkou extrémně nečitelně definovaných maker. Po zhlédnutí výstupu čtenář určitě najde mnoho způsobů, jak čitelněji napsat zdrojový text, aby výsledek zůstal stejný.

Tento článek ukazuje možnost, jak jednoduše zašifrovat zdrojový text do nečitelné podoby tak, abychom po přeložení TEXem dostali stejný výstup. Původní (čitelný) zdrojový text budu označovat jako *původní soubor* a výsledný (zašifrovaný) text jako *nový soubor*. Struktura *nového souboru* bude taková, že na prvním řádku budou definována makra, jejichž aplikací na zbývající (zašifrované) řádky se soubor rozšifruje.

Na několika místech budu makra popisovat hlouběji. Čtenář toto může brát jako cvičení z fungování token procesoru a expand procesoru. Proto tato místa budu označovat jako cvičení. Konec cvičení budu označovat symbolem //.

2. Použitá šifra

K zašifrování *původního souboru* je použita Caesarova šifra [4]. Na každém řádku je posun šifry jiný. Pro přesný popis je nutné zavést několik parametrů a označení. V textu budu pracovat pouze s celými čísly a tuto skutečnost nebudu dále zdůrazňovat.

Předpokládejme, že budeme šifrovat pouze znaky s ASCII kódem z intervalu $[K, L]$. Označme symbolem $\diamond$ operaci

$$x \diamond [K, L] := x - \lambda_x (L - K + 1)\,,$$

kde λ_x je celé číslo takové, že výsledek patří do intervalu $[K, L]$. V tomto článku bude vždy $\lambda_x \in \{-1, 0, 1\}$, což zjednoduší použitá makra. Šifra použitá na n-tém řádku bude

$$\varphi_n(c) = \begin{cases} (c + \beta_n) \diamond [K, L] & \text{pro } c \in [K, L]\,, \\ c & \text{jinak.} \end{cases} \tag{1}$$

Je zřejmé, že φ_n je bijekce na množině znaků. Její inverze je

$$\varphi_n^{-1}(c) = \begin{cases} (c - \beta_n) \diamond [K, L] & \text{pro } c \in [K, L]\,, \\ c & \text{jinak.} \end{cases} \tag{2}$$

Posun β_n určíme následovně. Označme $[\alpha, \gamma]$ interval, v němž se může posun nacházet. Dále vezměme parametry $\beta_0, \delta \in [\alpha, \gamma]$. Posun β_n definujme vztahem

$$\beta_{n+1} = (\beta_n + \delta) \diamond [\alpha, \gamma]\,. \tag{3}$$

Použitá šifra tedy závisí na šesti parametrech $K, L, \alpha, \beta_0, \gamma, \delta$. Aby vše fungovalo správně, musí být splněny podmínky[1] $K < L$, $0 < \alpha \leq \gamma \leq L - K$, $\alpha - \delta \leq \beta_0 \leq \gamma$ a $\alpha + \delta \leq \gamma$. Je vhodné, aby čísla δ a $L - K + 1$ byla nesoudělná. Dále je nutné, aby pro všechna $c \in [K, L]$ a pro všechna $\beta_n \in [\alpha, \gamma]$ platilo `\catcode` $\varphi_n(c) \notin \{9, 15\}$, přičemž se jedná o hodnotu `\catcode` platnou v době dešifrování. Také je nutné, aby TEX zapsal znaky $\varphi_n(c)$ do *nového souboru* přímo a ne pomocí dvojité stříšky. S ohledem na [1], sekci 49, je třeba položit $32 \leq K < L \leq 126$.

Hlavní mechanismus pro zašifrování tvoří primitiv `\uppercase`. Hodnota $\varphi_n(c)$ je uložena v registru `\uccode` c. Při práci jsou použity tři čítače:

- `\^^E` je aktuální hodnota β_n;
- `\^^F` je hodnota `\uccode\^^G`;
- `\^^G` je řídicí proměnná cyklu při generování `\uccode`.

V sekci 6 je popsáno, proč jsou tyto čítače pojmenovány tímto způsobem.

Samozřejmě by bylo možné vhodnou úpravou maker typ šifry změnit.

3. Zašifrování souboru

Zašifrování souboru se provede makrem `\zasifruj`, majícím deset parametrů,

`\zasifruj{`*původní soubor*`}{`*nový soubor*`}{`*pomocný soubor*`}`
`{`K`,`L`}{`α`,`β_0`,`γ`,`δ`}{`*poslední příkaz*`}`

[1] Vysvětlení těchto podmínek přenechávám na čtenáři.

Argumenty *pomocný soubor* a *poslední příkaz* budou popsány v sekci 4, o ostatních parametrech už byla řeč.

Makro `\zasifruj` nejdříve uloží hodnoty svých parametrů a nastaví pomocné čítače. Poté otevře používané soubory a makrem `\zapismakra` do *nového souboru* zapíše dešifrovací makra. Nastaví se hodnoty `\uccode` mimo interval $[K, L]$[2] a také počáteční hodnota β_n. Dále je nastaven `\endlinechar` na hodnotu -1, aby se při načítání řádků na konec nepřipojovaly nežádoucí mezery. Makrem `\dospecials` se nastaví kategorie speciálních znaků na hodnotu 12, aby při šifrování nedocházelo k nežádoucím efektům. Makrem `\zpracuj` se spustí šifrování jednotlivých řádků *původního souboru*. V závěru se použité soubory uzavřou.

```
\def\zasifruj#1#2#3#4#5#6{\def\puvodni{#1 }
  \def\novy{#2 }
  \strisky\novyS#2\^^H
  \def\pomocny{#3 }
  \strisky\pomocnyS#3\^^H
  \separujascii#4,
  \separujposun#5,
  \def\posledni{#6}
  \posunsirka\posunmax \advance\posunsirka-\posunmin
    \advance\posunsirka1
  \asciiminI\asciimin \advance\asciiminI-1
  \asciisirka\asciimax \advance\asciisirka-\asciiminI
  \openin\patnact\puvodni
  \immediate\openout\patnact\zasifrovany
  \zapismakra
  {\^^G0
  \loop \ifnum\^^G<255
    \advance\^^G1 \uccode\^^G\^^G
  \repeat
  \^^E\posunzacatek
  \endlinechar-1
  \def\do##1{\catcode`##112}
  \dospecials\zpracuj}
  \closein\patnact \immediate\closeout\patnact}
\def\separujascii#1,#2,{\def\asciimin{#1 }\def\asciimax{#2 }}
\def\separujposun#1,#2,#3,#4,{\def\posunmin{#1}
  \def\posunzacatek{#2 }\def\posunmax{#3 }\def\posunzmena{#4 }}
```

Makro `\zpracuj` načte jeden řádek *původního souboru*. Podle vzorce (3) nastaví β_n a poté podle vzorce (1) změní `\catcode` všech znaků z intervalu $[K, L]$.

[2]Kvůli zjednodušení maker jsou nastaveny hodnoty `\uccode` všech znaků.

Použitím primitivu `\uppercase` se řádek zašifruje a zapíše se do *nového souboru*.[3] Nakonec se makro `\zpracuj` zavolá na následující řádek.

```
\def\zpracuj{\read\patnact to\radek
  \ifeof\patnact\else
    \advance\^^E\posunzmena
    \ifnum\^^E>\posunmax \advance\^^E-\posunsirka\fi
    \^^G\asciiminI
    \loop
      \ifnum\^^G<\asciimax \advance\^^G1
      \^^F\^^G \advance\^^F\^^E
      \ifnum\^^F>\asciimax
        \advance\^^F-\asciisirka
      \fi
      \uccode\^^G\^^F
    \repeat
    \ifx\radek\byestring\else
    \ifx\radek\enddocstring\else
      \expandafter\uppercase\expandafter{\expandafter\immediate
        \expandafter\write\expandafter\patnact\expandafter
        {\radek}}%
    \fi\fi
    \expandafter\zpracuj
  \fi}
```

Cvičení Kdybychom nedefinovali řídicí sekvenci `\patnact` a uvnitř `\uppercase` na řádku 43 psali číslo `15`, nastal by problém s primitivem `\write`. Tokeny $\boxed{1}_{12}\boxed{5}_{12}$ by se změnily na $\boxed{\varphi_n(1)}_{12}\boxed{\varphi_n(5)}_{12}$ a výsledek `\uppercase` by byl

$$\boxed{\texttt{immediate}}\,\boxed{\texttt{write}}\,\boxed{\varphi_n(1)}_{12}\boxed{\varphi_n(5)}_{12}\boxed{\varphi_n(\{)}_1\,\varphi_n^*(\text{expanze }\texttt{\textbackslash radek})\,\boxed{\varphi_n(\})}_2$$

což by způsobilo chybu, protože primitiv `\write` očekává číslo souboru. //

Názvy použitých souborů je nutné do *nového souboru* zapsat. Není však vhodné zapsat tyto názvy přímo. Názvy se zašifrují tak, že se jednotlivé znaky zapíší pomocí dvojité stříšky. Rozšifrování poté provede token procesor. Viz například [2], strana 22. K tomuto účelu slouží makro `\strisky`, jehož použití je

`\strisky\cs` *text*`\^^H`

[3] Pokud je řádek složen z tokenů $\boxed{\backslash}_{12}\boxed{\texttt{b}}_{11}\boxed{\texttt{y}}_{11}\boxed{\texttt{e}}_{11}$ nebo $\boxed{\backslash}_{12}\boxed{\texttt{e}}_{11}\boxed{\texttt{n}}_{11}\boxed{\texttt{d}}_{11}\boxed{\{}_{12}\boxed{\texttt{d}}_{11}\boxed{\texttt{o}}_{11}\boxed{\texttt{c}}_{11}\boxed{\texttt{u}}_{11}\boxed{\texttt{m}}_{11}\boxed{\texttt{e}}_{11}\boxed{\texttt{n}}_{11}\boxed{\texttt{t}}_{11}\boxed{\}}_{12}$, nebude se do *nového souboru* zapisovat. Důvod bude vysvětlen později.

Zašifrovaný *text* se uloží do makra `\cs`. Například po použití

```
\strisky\sifra Ahoj\^^H
```

budou v makru `\sifra` uloženy tokeny

$$\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{4}_{12}\boxed{1}_{12}\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{6}_{12}\boxed{8}_{12}\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{6}_{12}\boxed{f}_{12}\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{6}_{12}\boxed{a}_{12}$$

Spoluautorem makra `\strisky` je Petr Březina.

```
\def\strisky#1{\def#1{}\striskyA#1}
\def\striskyA#1#2{\ifx#2\^^H\else
    \chardef\^^N=‘#2
    \expandafter\striskyB\meaning\^^N
    \edef#1{#1^\empty^\^^O}
    \expandafter\striskyA\expandafter#1\fi}
\def\striskyB#1"#2#3{\lowercase{\def\^^O{#2#3}}}
```

Cvičení Pro získání čísla znaku v šestnáctkové soustavě je použit primitiv `\meaning`. Po expanzi primitivu `\meaning` použitého na token, který je definován primitivem `\chardef` nebo `\mathchardef`, dostaneme tokeny

$$\boxed{\backslash}_{12}\Big(\boxed{m}_{12}\boxed{a}_{12}\boxed{t}_{12}\boxed{h}_{12}\Big)\boxed{c}_{12}\boxed{h}_{12}\boxed{a}_{12}\boxed{r}_{12}\boxed{"}_{12}$$

za nimiž následuje dané číslo v šestnáctkové soustavě reprezentované jako tokeny kategorie 12. Toto je jediná přímá cesta, jak lze v TEXu převést číslo do šestnáctkové soustavy. //

Cvičení Pokud bude text poslaný makru `\strisky` obsahovat znak s kódem menším než 16, bude mít jeho zápis v šestnáctkové soustavě pouze jeden znak. V takovém případě nebude makro `\striskyB` fungovat správně. Řešení tohoto problému přenechávám na čtenáři. //

Cvičení Konverze makrem `\strisky` nebude fungovat také v případě, že bude parametr obsahovat mezeru (kategorie 10).[4] Předpokládejme, že zavoláme

```
\strisky\sifra Z y\^^H
```

Po prvním provedení makra `\striskyA` budou ve čtecí frontě tokeny

$$\boxed{\texttt{striskyA}}\boxed{\texttt{sifra}}\boxed{\sqcup}_{10}\boxed{y}_{11}\boxed{\hat{}\hat{}H}$$

[4] Tento případ nemůže nastat, pokud budeme konvertovat názvy vkládaných souborů. Podle [1], sekce 516, nemůže totiž název vkládaného souboru obsahovat mezeru.

Expand procesor při načítání neseparovaných parametrů přeskakuje tokeny kategorie 10. Proto se makro `\striskyA` zavolá s parametry `#1` = $\boxed{\texttt{sifra}}$ a `#2` = $\boxed{\texttt{y}}_{11}$. V důsledku toho nakonec budou v makru `\sifra` uloženy tokeny

$$\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{5}_{12}\boxed{a}_{12}\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{7}_{12}\boxed{9}_{12}$$

namísto správných tokenů

$$\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{5}_{12}\boxed{a}_{12}\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{2}_{12}\boxed{0}_{12}\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{7}_{12}\boxed{9}_{12}$$ //

Cvičení TEX interpretuje dvojitou stříšku pouze na úrovni token procesoru. Proto se na řádku 53 znaky

$$\hat{}_7\backslash_0 e_{11} m_{11} p_{11} t_{11} y_{11} \hat{}_7\backslash_0\hat{}_7\hat{}_7 0_{11}$$

interpretují jako

$$\boxed{\hat{}}_7\boxed{\texttt{empty}}\ \boxed{\hat{}}_7\boxed{\hat{}\hat{}0}$$

což po expanzi dává

$$\boxed{\hat{}}_7\boxed{\hat{}}_7(\text{expanze } \texttt{\^\^0})$$

V dalším zpracování už se tokeny $\boxed{\hat{}}_7\boxed{\hat{}}_7$ jako dvojitá stříška neinterpretují. //

Nesmíme zapomenout na deklaraci použitých registrů.

```
\newcount\asciiminI \newcount\posunsirka \newcount\asciisirka
\newcount\^^E \newcount\^^F \newcount\^^G \chardef\patnact15
```

4. Rozšifrování souboru

Makra podobná výše uvedeným rozšifrují všechny řádky *nového souboru* s výjimkou prvního řádku do *pomocného souboru*. Rozšifrování se provede podle vzorce (2). Poté se *pomocný soubor* zavře a načte. Protože je v *pomocném souboru* obsažen nezašifrovaný text *původního souboru*, bylo by vhodné *pomocný soubor* smazat. Toto však lze udělat pouze při zapnutém `\write18` s použitím prostředků operačního systému. Protože nemáme zaručeno, že uživatel bude mít zapnuto `\write18`, je třeba si pomoci jinak. *Pomocný soubor* otevřeme pro zápis (tím se jeho obsah vymaže) a ihned zavřeme.

V závěru je třeba zajistit, aby se *nový soubor* nedočetl do konce. K tomu slouží *poslední příkaz*. Pokud šifrujeme soubor, který se vkládá do jiného souboru (například balíček LATEXu), použijeme `endinput`, pokud se jedná o hlavní soubor v plainu, použijeme `bye`.

Je důležité, aby všechna makra z prvního odstavce této sekce byla v *novém souboru* na prvním řádku. O to se postará makro `\zapismakra`. Aby byl *nový soubor* hůře čitelný, jsou použitá makra nazvána `\^^A` až `\^^D` a použité čítače `\^^E` až `\^^G`.

```
\begingroup
  \catcode`\|0 \catcode`(1 \catcode`)2 \catcode`\^12
  \catcode`\\12 |catcode`|{12 |catcode`|}12 |catcode`|#12
  |gdef|byestring(\bye)
  |gdef|enddocstring(\end{document})
  |gdef|zapismakra(|immediate|write|patnact(%
    {\chardef\^^A15%
    \countdef\^^E221\countdef\^^F222\countdef\^^G223%
    \def\^^B{\read\^^Ato\^^C%
      \ifeof\^^A\else%
        \ifnum\^^E=0%
          \^^E|posunzacatek
        \else%
          \advance\^^E|posunzmena
          \ifnum\^^E>|posunmax
            \advance\^^E-|the|posunsirka
          \fi%
          \^^G|the|asciiminI
          \loop%
            \ifnum\^^G<|asciimax \advance\^^G1%
            \^^F\^^G\advance\^^F-\^^E%
            \ifnum\^^F<|asciimin
              \advance\^^F|the|asciisirka
            \fi%
            \uccode\^^G\^^F%
          \repeat%
          \expandafter\uppercase\expandafter%
            {\expandafter\immediate\expandafter%
            \write\expandafter\^^A\expandafter{\^^C}}%
        \fi%
        \expandafter\^^B%
      \fi}%
    \def\^^D{{\def\do##1{\catcode`##112}\endlinechar-1%
      \^^G0%
      \loop \ifnum\^^G<255%
        \advance\^^G1\uccode\^^G\^^G%
      \repeat%
```

```
95        \^^E0\dospecials\^^B}}%
96      \openin\^^A|novyS \immediate\openout\^^A|pomocnyS
97      |space\^^D%
98      \closein\^^A\immediate\closeout\^^A}%
99      \input |pomocnyS
100     \immediate\openout15|pomocnyS \immediate\closeout15%
101     \|posledni))
102 |endgroup
```

Cvičení Při načítání názvu souboru TEX provádí expanzi, dokud nenarazí na mezeru nebo neexpandovatelnou řídicí sekvenci. Kdyby na řádku 97 nebyla mezera vložená makrem `\space`, TEX by při načítání názvu souboru (vloženého makrem `\pomocnyS`) pokračoval expandováním makra `\^^D`. Tam by narazil na token $\boxed{\texttt{\{}}_1$. Protože se tokeny kategorie 1 při načítání názvu souboru interpretují jako běžné znaky, chápal by TEX závorku `{` jako součást názvu souboru. Teprve token $\boxed{\texttt{def}}$ by sloužil jako oddělovač. Proto je nutné na řádku 97 mezeru explicitně vložit. //

Cvičení Kdybychom při šifrování *původního souboru* zašifrovali i tokeny $\boxed{\texttt{\textbackslash}}_{12}\boxed{\texttt{b}}_{11}\boxed{\texttt{y}}_{11}\boxed{\texttt{e}}_{11}$, pak by se TEX ukončil při vložení *pomocného souboru* na řádku 99 a k důležitému řádku 100 už by se nedostal. //

Cvičení Pokud v době dešifrování nebude mít znak `^` kategorii 7, token procesor název souboru nerozšifruje. Například soubor SLOVAK.LDF nastavuje v místě `\begin{document}` znak `^` aktivní. To znamená, že pokud používáme `\usepackage[slovak]{babel}` a *původní soubor* obsahuje `\begin{document}`, pak se názvy souborů vložené na řádku 96 rozšifrují správně, ale název vložený na řádku 100 se nerozšifruje. V důsledku toho se *pomocný soubor* nevymaže. Čtenář určitě najde jednoduché řešení tohoto problému. //

5. Ukázka

Zkopírujme řádky 1–102 do souboru s názvem A.TEX.[5] Poté můžeme libovolný soubor zašifrovat, pokud TEXem přeložíme soubor obsahující jediný řádek

```
\input a \zasifruj... \bye
```

Soubor A.TEX je relativně čitelný. Nyní vytvoříme soubor B.TEX, který bude dělat totéž, ale bude nečitelný. Tento soubor vytvoříme zašifrováním souboru A.TEX, například s nastavením konstant $K = 33$, $L = 126$, $\alpha = 37$, $\beta_0 = 41$,

[5] Soubor je umístěn na adrese `http://www1.osu.cz/~sustek/TeX/zasifruj.tex`.

$\gamma = 47$ a $\delta = 3$. Protože soubor B.TEX budeme primitivem `\input` vkládat do jiného souboru, ukončíme jeho čtení sekvencí `endinput`. Jako *pomocný soubor* vytvoříme soubor B.TXT.

Vytvořme nový soubor, zapišme do něj

```
\input a \zasifruj{a.tex}{b.tex}{b.txt}{33,126}{37,41,47,3}{endinput} \bye
```

a přeložme tento soubor TEXem. Dostaneme soubor B.TEX, jehož první řádky jsou[6]

```
{\chardef\^^A15\countdef\^^E221\countdef\^^F222\countdef\^^G223\def\^^B{\read\^^Ato\^^C
\ifeof\^^A\else\ifnum\^^E=0\^^E41 \else\advance\^^E3 \ifnum\^^E>47 \advance\^^E-11\fi\^
^G32\loop\ifnum\^^G<126 \advance\^^G1\^^F\^^G\advance\^^F-\^^E\ifnum\^^F<33 \advance\^^
F94\fi\uccode\^^G\^^F\repeat\expandafter\uppercase\expandafter{\expandafter\immediate\e
xpandafter\write\expandafter\^^A\expandafter{\^^C}}\fi\expandafter\^^B\fi}\def\^^D{{\de
f\do##1{\catcode`##112}\endlinechar-1\^^G0\loop \ifnum\^^G<255\advance\^^G1\uccode\^^G\
^^G\repeat\^^E0\dospecials\^^B}}\openin\^^A^^62^^2e^^74^^65^^78\immediate\openout\^^A^^
62^^2e^^74^^78^^74 \^^D\closein\^^A\immediate\closeout\^^A}\input ^^62^^2e^^74^^78^^74\
immediate\openout15^^62^^2e^^74^^78^^74\immediate\closeout15\endinput
*234*H/9=2C8O]O^O_O‘OaObI*234*>CD=2<7IO] K
  -567-K2<@5@G2?JLRa N
  %<=;2<4B%C*48-8?*7BzJY%’’o
  (O12(<;9;/:EGM] I
  +BCA8B:H+?><>2=H"P‘+--u
  #:,7(9<1(:*OOHYQ
  &=/:+<?4:9=?8K]T
  )123)=<@921;6HNaJ
  ,@?CE>C9B;1,@?CE>=1H ,14F1>35,@?CE>C9B;1[,@?CE>=9>
    $),>)6+-$87;=6;1:3)W
  ’,>.44849r’,>.44849 ’,/A,9.O’,>.44849rVZ
  */A177A7@9/*/A177;/F */2D/<13*/A177A7@9/Y*/A177;7<u
  -@A6?:?-A2E?24E-AFG@5?: -:>>65:2E6-@A6?@FE-A2E?24E-K2<@5@G2?J
  %C*92<6*4;*
  G(**qZ
  +;>>? +85=D<+--ti_bb
    #(+=(5*,#%%lV #<**6+,#%%l#%%l
  &</:/+>
  )++p)=<@B;G.O.A28
  ,5>4<9>5381B[_
  $,-.$,7IIWC$+)<+7,-(IIWWXE
  ’/:>;O.4,7>’E;=,.@5H
  *1:=A37<*>/B</1B *7;;327/B3*1:=A3=CB*>/B</1BK
-567-D6A2CF;2D4::R‘[Ra[L-567-2D4::>:?LR‘ N-567-2D4::>2ILRa NN
%-./%<.9*;>398<>7JXSJYSJZSJ[SD%-./%98<>7627DJXF
  (O12(<;?A:F-/-@17GM\ I(O12(<;?A:9-DGM] I(O12(<;?A:F91:-GM^ II
+345+I?A02D9J+A403+?OC=O2C C>+A034:
...
```

Soubor B.TEX můžeme používat stejným způsobem, jako se používá soubor A.TEX. Libovolný soubor můžeme zašifrovat, pokud TEXem přeložíme soubor obsahující jediný řádek

```
\input b \zasifruj... \bye
```

[6] Ve skutečnosti prvních devět řádků výpisu tvoří jediný řádek souboru B.TEX.

6. Možné problémy

Pokud bude *nový soubor* načítán uvnitř jiného souboru, může dojít ke kolizím s makry definovanými před načtením *nového souboru*. Aby se pravděpodobnost těchto kolizí minimalizovala, jsou provedena následující opatření.

- Všechna nová makra jsou definována uvnitř skupiny.[7]
- Nové řídicí sekvence mají „nepravděpodobné“ (a zároveň nečitelné) názvy `\^^A` až `\^^G`.
- Jsou použity registry `\count221` až `\count223`, jejichž pravděpodobnost použití je malá. Tyto registry nejsou deklarovány makrem `\newcount`, ale je přímo použit primitiv `\countdef`.
- Je použito nejvyšší možné číslo souboru 15, které není deklarováno makry `\newread` a `\newwrite`, ale je přímo použit primitiv `\chardef`.

Pro správné rozšifrování je třeba, aby všechny použité primitivy a makra měly svůj původní význam. Pokud je některý z nich před načtením *nového souboru* předefinován, nelze zaručit správné fungování.

Slabým místem použitého postupu je použití *pomocného souboru*. Kdyby nějaký hacker na vhodném místě změnil *nový soubor* tak, aby se *pomocný soubor* nevymazal, mohl by z *pomocného souboru* jednoduše získat *původní soubor*. Otázkou je, zda je možné vše udělat bez použití *pomocného souboru*.

Závěr

V TEXu je možné napsat dokument mnoha různými způsoby. Stejně tak je možné různými způsoby napsat balík maker. Existují způsoby čitelné a způsoby nečitelné. Pokud chceme, aby druhý uživatel používal naše makra, musíme mu poslat zdrojový kód těchto maker. Pokud však nechceme, aby se uživatel k tomuto zdrojovému kódu dostal, je třeba jej zašifrovat a tento zašifrovaný soubor uživateli poslat.

Článek ukazuje jednu z možností, jak toho dosáhnout. Soubor je zašifrován použitím primitivu `\uppercase`. Na prvním řádku zašifrovaného souboru jsou definována makra, která zbytek souboru rozšifrují. Tím je zajištěna původní funkčnost souboru.

Poděkování

Rád bych poděkoval Petru Březinovi, Petře Konečné, Pavlu Střížovi a Janu Štěpničkovi za pročtení článku a cenné připomínky a nápady vedoucí ke zkvalitnění článku.

[7] Pokud je *nový soubor* načítán při nastaveném `\globaldefs=1`, zavedení skupiny nepomůže. Další opatření sníží pravděpodobnost kolize. Největší problém pak mohou způsobit jinak nastavené hodnoty `\uccode`.

Seznam literatury

[1] Knuth, Donald. *TEX: The Program* (Computers and Typesetting, Volume B). Reading, Massachusetts: Addison-Wesley, 1986. ISBN 0-201-13437-3. Dostupné též online ze serveru: `ftp://tug.ctan.org/pub/tex-archive/systems/knuth/dist/tex/tex.web`

[2] Olšák, Petr. *TEXbook naruby*. [TEXbook Inside Out.] 2. vyd. Brno, nakladatelství Konvoj, 2001. ISBN 80-7302-007-6. Dostupné na `ftp://math.feld.cvut.cz/pub/olsak/tbn/tbn.pdf`

[3] Carlisle, David. *obscure.tex*. Dostupné na `http://www.maths.abdn.ac.uk/tex/latex_course/obscure.tex`

[4] Wikipedie: Otevřená encyklopedie: *Caesar Cipher* [online]. [citováno 2. října 2009]. Dostupné na `http://en.wikipedia.org/wiki/Caesar_cipher`

Summary: On Enciphering a Source Code and Preserving Its Functionality

It is possible to write a document in many different ways using TEX. Similarly, one can write a macro package in many different ways. Some ways are readable and some are not. If we want another user to use our macros, we have to send him the source code of those macros. But if we don't want him to see the source code, we have to encipher it and to send the ciphertext only to the user.

This paper shows one possibility of how to achieve this goal. The original file is enciphered using the `\uppercase` primitive. The first line of the ciphered file contains macros for deciphering the other lines. This ensures that the ciphered file has the same functionality as the original file.

Key words: File enciphering, `\uppercase`, Caesar cipher.

Jan Šustek, jan.sustek@osu.cz
Ostravská univerzita, Přírodovědecká fakulta, Katedra matematiky
30. dubna 22, CZ-701 03 Ostrava, Czech Republic

Vykreslení středové části obrazce Shri Yantra pomocí METAPOSTu

Luís Nobre Gonçalves, Michal Mádr, Pavel Stříž

Abstrakt

Tento článek popisuje, jak lze pomocí programu METAPOST jednoduše nakreslit komplikovaný obrazec – středovou část kultovního indického symbolu Shri Yantra. Obrazec se skládá z devíti trojúhelníků rozmístěných uvnitř kruhu. Vztahy mezi trojúhelníky jsou popsány několika pravidly a vytvoření kresby respektující tato pravidla je netriviální. Článek popisuje úspěšný pokus, jak se k správnému nakreslení obrazce přiblížit.

Za pečlivé násobné pročtení rukopisu patří naše poděkování Vítu Zýkovi.

Klíčová slova: METAPOST, Shri Yantra, Yantra, Indie, indické symboly, optimalizace, metoda pokus-omyl, simulace, náboženské symboly, kultovní symboly.

Shri Yantra properties

Throughout history, the Shri Yantra has been drawn in many ways but its most used (plane) form is similar to the one shown in Figure 1 on page 213. These are the properties of the drawing:

- There are nine interlocking triangles inside a circle. Four triangles point upwards and five point downwards.
- The two largest triangles touch the outer circle on all three apexes.
- Except for the two largest triangles, all apexes touch the base of another triangle.
- There are 37 intersections[1] resulting from three lines meeting at a point.[2]
- There are 24 intersections resulting from two lines crossing.
- The figure respects symmetry by reflection on a vertical axis.

When we talk about Shri Yantra in this text, we mean a drawing having the above mentioned properties. In the rest of the article we will present a METAPOST program attempting to draw a Shri Yantra. First, we will generally describe how we went about writing the program. Then, we will list the source code of the program. Finally, we will discuss how to attain a configuration for the program to achieve an acceptable result.

[1] Different requirement on the drawing exists, that specifies 33 such intersections.

[2] The six apexes of the two largest triangles touching the circle are considered to belong to these 37 points.

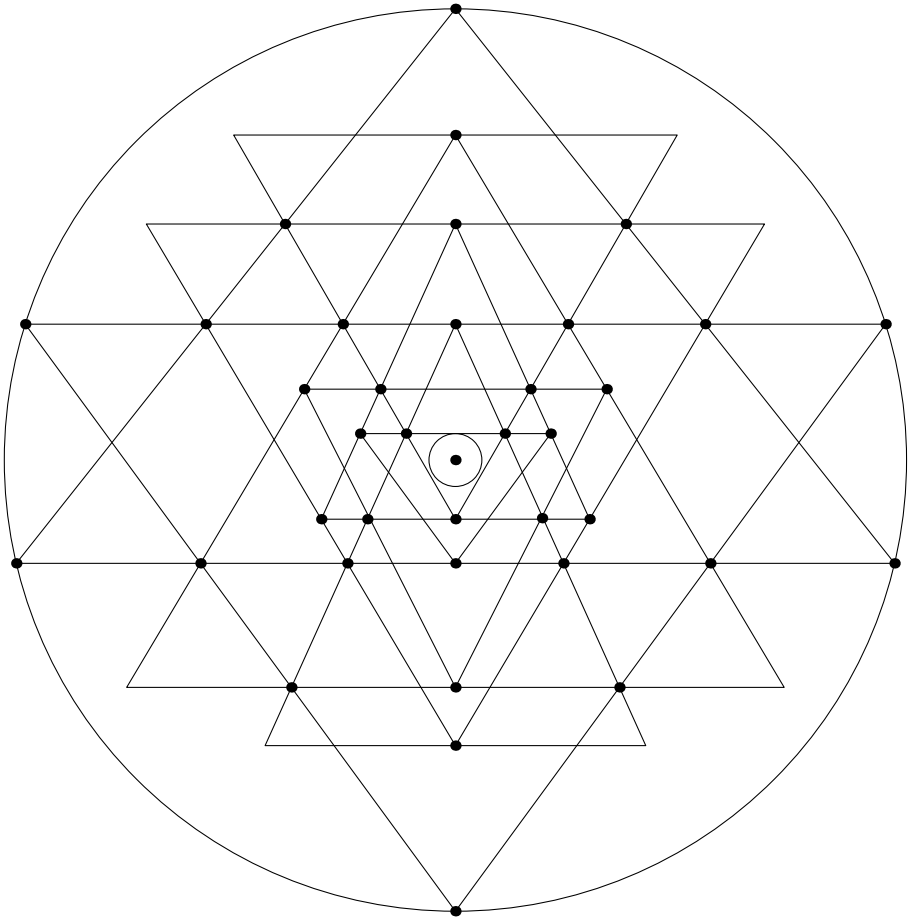


Figure 1: Shri Yantra with the 37 triple intersections marked with black dots. Parameter values (described later) are: A=0.63295, B=0.22896, C=0.13113, D=0.44115, E=0.68517.

How we went about the drawing

We wrote a program in METAPOST that tried to emulate the Shri Yantra drawing from Figure 2 on page 215.[3] The key thing that the program must do is to locate the forty points 1–40.[4] To achieve that, the locations of all the forty points are described in the program using typical METAPOST statements. Figure 2 guided us in formulating the statements. For example, from the figure we could see that the point `z10` lies halfway between the points `z4` and `z5`, thus we described the location of `z10` in the program simply by the statement:

```
z10 = 0.5[z4,z5];
```

Also from Figure 2 we could see that the point `z13` lies at the intersection of the lines `z11--z12` and `z4--z6`, thus we described the location of `z13` as:

```
z13 = whatever[z12,z11];
z13 = whatever[z4,z6];
```

Once three points of any of the triangles (e.g., `z14`, `z15`, and `z16`) had been located, we could finally draw it:

```
draw z14--z15--z16--cycle;
```

But how do we start? Well, there are at least two standard initial steps:

1. Choose a frame of reference.
2. Choose the parametrization.

Let's use a frame of reference with its origin at the center of the circle (of unitary ray), with one horizontal axis pointing to the right (x) and with one vertical axis point up (y). The parametrization is achieved by trial-and-error. Among all possibilities, one chooses arbitrarily the drawing sequence. The coordinates of placements which are not imposed, are parameters.

Of course, I needed to *choose* the location of several points to then be able to start describing other points as shown above. As it turned out, there were only five coordinates of points that needed to be chosen throughout the drawing. Positions of all the rest of the points followed from these five selections and from the properties of the drawing. We denoted the five selected values A–E and created from them five numeric parameters that needed to be passed to our program. Let's describe them:

[3] If you are wondering – yes, such drawings were available even before we wrote our program, see articles [1, 3].

[4] Some of these points are apexes of triangles, other points are just used by the program to locate other apexes.

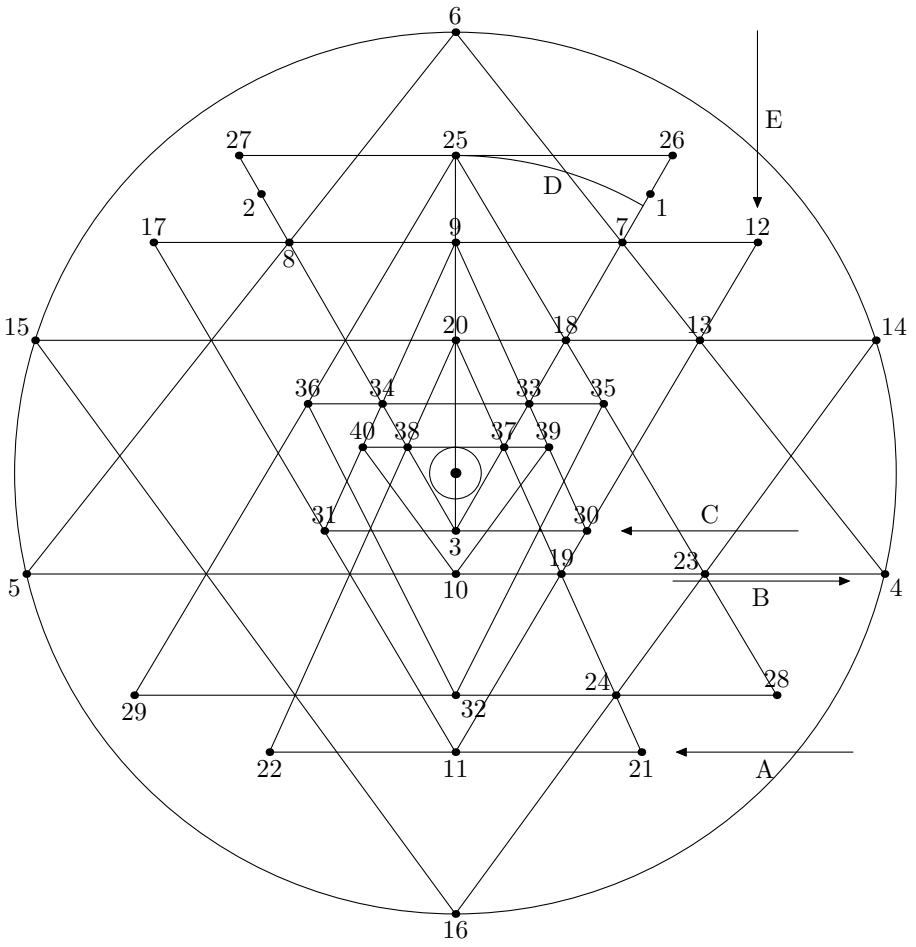


Figure 2: Shri Yantra from Figure 1 on page 213 with points constructed by our program labeled. Arrows highlight points located using the program parameters A–E.

- Parameter A designates the x-coordinate of point 1, parameter D designates the y-coordinate of point 1. Point 1 is a helping point. It is used only to locate other points. Parameter A also designates the (symmetric of the) y-coordinate of point 11, so parameter D is only a measure of the angle between `z3--z25` and `z3--z26`.
- Parameter B designates the y-coordinate of point 4.
- Parameter C designates the y-coordinate of point 3.
- Parameter E designates the x-coordinate of point 12.

Whether the drawing produced by the program will be successful depends on the values of these parameters. We will discuss them after we list the program.

The program in METAPOST

The drawing sequence in METAPOST commands are as follows. Parameter D, as indicated in Figure 2 on page 215, is equivalent to arctan(D/(A+C)) as in this drawing sequence. The full METAPOST program used to search for the correct Shri Yantra configurations may be found at the author's site [2].

```
pA := A*u;
pB := B*u;
pC := -C*u;
pD := D*u;
pE := E*u;
path outercircle;
outercircle = fullcircle scaled (2*u);
draw outercircle;
z1 = (pD,pA);
z2 = (-pD,pA);
z3 = (0,pC);
z4 = ((-u,-pB)--(u,-pB)) intersectionpoint reverse outercircle;
z5 = (-x4,-pB);
z6 = (0,u);
draw z4--z5--z6--cycle;
z7 = (z1--z3) intersectionpoint (z4--z6);
z8 = (-x7,y7);
z9 = 0.5[z7,z8];
z10 = 0.5[z4,z5];
z11 = (0,-pA);
y12 = y7;
x12 = pE;
z13 = whatever[z12,z11];
```

```
z13 = whatever[z4,z6];
z14 = ((-u,y13)--(u,y13)) intersectionpoint outercircle;
z15 = (-x14,y13);
z16 = (0,-u);
draw z14--z15--z16--cycle;
z17 = (-x12,y12);
draw z12--z11--z17--cycle;
z18 = whatever[z1,z3];
z18 = whatever[z14,z15];
z19 = whatever[z11,z12];
z19 = whatever[z4,z5];
z20 = 0.5[z14,z15];
y21 = y11;
z21 = whatever[z20,z19];
z22 = (-x21,y21);
draw z21--z22--z20--cycle;
z23 = whatever[z14,z16];
z23 = whatever[z4,z5];
z24 = whatever[z21,z20];
z24 = whatever[z14,z16];
x25 = 0;
z25 = whatever[z23,z18];
y26 = y25;
z26 = whatever[z3,z7];
z27 = (-x26,y26);
draw z27--z26--z3--cycle;
y28 = y24;
z28 = whatever[z23,z25];
z29 = (-x28,y28);
draw z25--z28--z29--cycle;
z30 = (z3--(u,pC)) intersectionpoint (z13--z11);
z31 = (-x30,y30);
draw z30--z31--z9--cycle;
z32 = 0.5[z28,z29];
z33 = (z3--z1) intersectionpoint (z30--z9);
z34 = (-x33,y33);
z35 = whatever[z25,z28];
z35 = whatever[z33,z34];
z36 = (-x35,y35);
draw z32--z35--z36--cycle;
z37 = (z21--z20) intersectionpoint (z1--z3);
z38 = (-x37,y37);
```

```
z39 = whatever[z37,z38];
z39 = whatever[z30,z9];
z40 = (-x39,y39);
draw z10--z39--z40--cycle;
```

Finding suitable parameter values

As already mentioned, we will not be able to draw Shri Yantra using the listed program above unless we pass suitable values for the five parameters A–E. Figure 3 on page 219 shows what may happen if we pass unsuitable parameter values.

Let's mention how we can obtain suitable values for the parameters. We have seen a drawing using one suitable set of parameter values in Figure 1 on page 213, another one can be seen in Figure 4 on page 220.

One approach could be to measure the *amount of disrespect* in the triple intersections and then minimize it. Knowing that a disrespected triple intersection forms a triangle[5], the *amount of disrespect* may be the length of its sides or its area. We could use any minimization method.

However, another approach worked for us. In fact, it is the least sophisticated method: we randomly generated, e.g., five thousand sets of parameter values, tried them all and registered the set that produced the smallest *amount of disrespect*. With this approach, we needed to take two more things into consideration, though.

First, for some sets of parameter values it may not be possible to reach the end of the drawing sequence, therefore, one must implement defenses against program crashes. The METAPOST interpreter, for instance, crashed when calculating the `intersectionpoint` of two non-intersecting `path`s[6].

Second, using the criterion of the *amount of disrespect* mentioned above, many sets of parameter values may be equally suitable. In order to choose the best one from among them we may also consider some additional criteria:

Amount of nonalignment – All 27 triangle edges could have their global center of mass coincident with the center of the circle.

Amount of inner disharmony – The smallest central triangular area[7] could have all its edges tangent to a concentric circle.

Amount of heterogeneity – the range of distances between consecutive intersections should not be very wide, meaning that consecutive intersections should be very near.

Amount of outness – Triangles should not go out of the outer circle.

[5] In fact, a disrespected triple intersection forms three double intersections.

[6] The solution is to check that `intersectiontimes` does not return $(-1,-1)$.

[7] `z3--z37--z38--cycle`

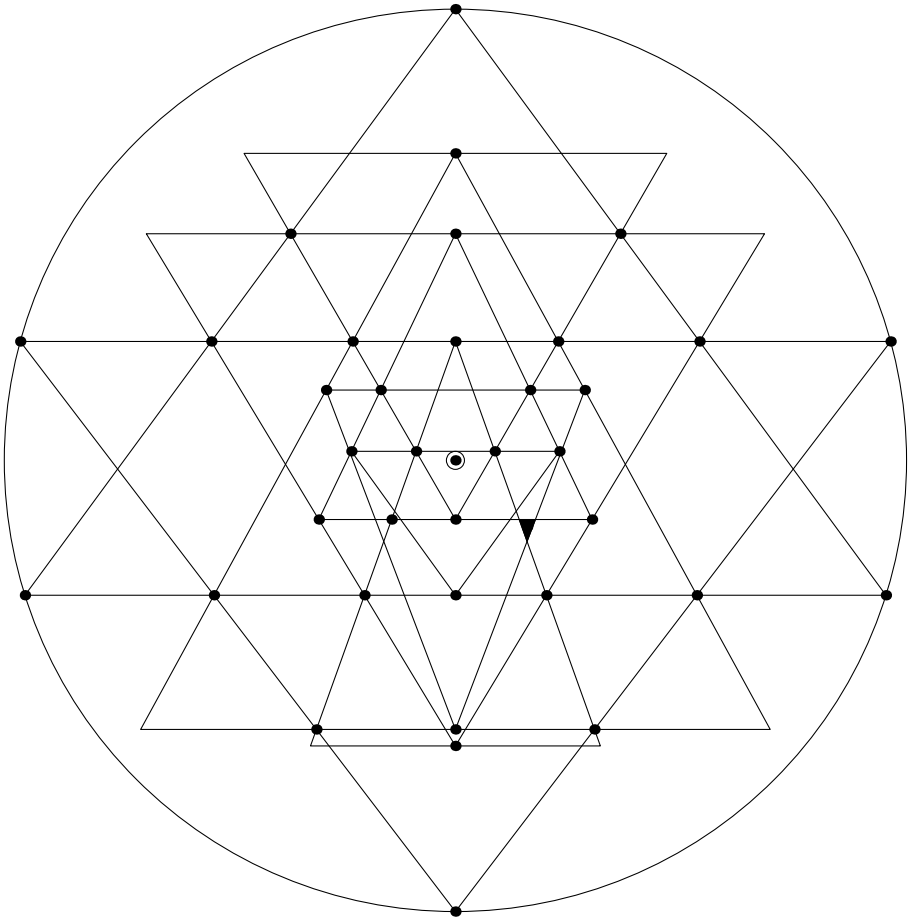


Figure 3: Shri Yantra drawn using unsuitable parameters. One triple intersection is not respected due to a change in parameter B (see the black triangle). A=0.63295, B=0.29896, C=0.13113, D=0.44115, E=0.68517.

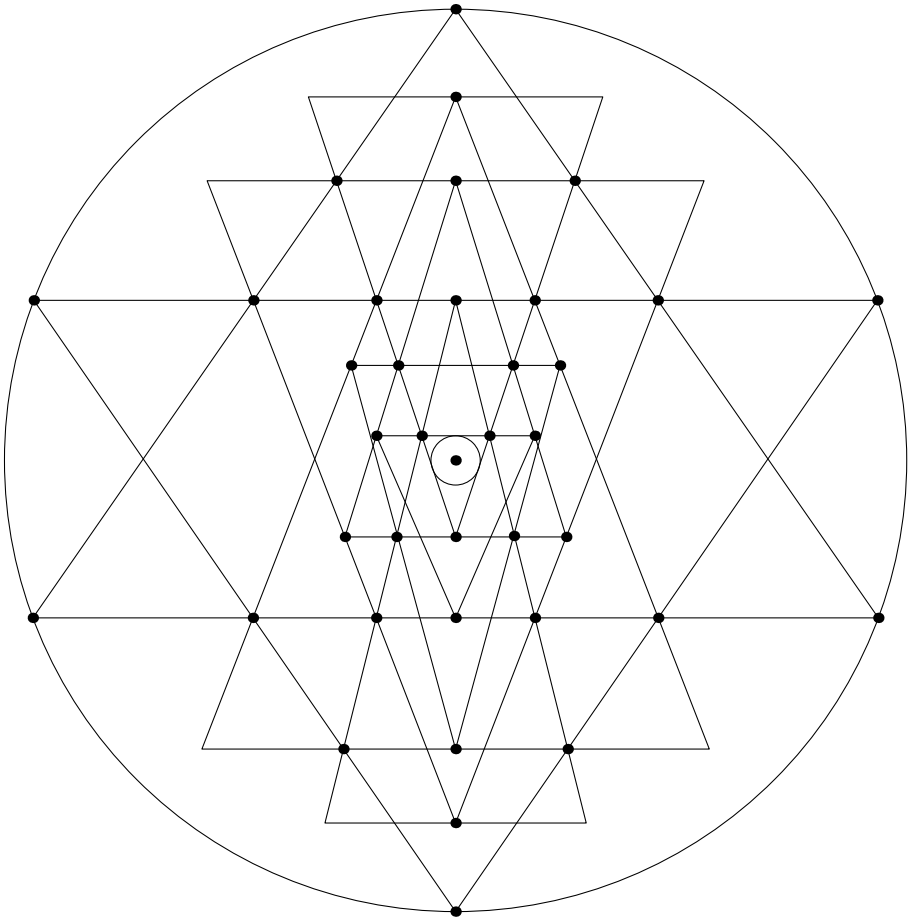


Figure 4: Shri Yantra with a high *amount of heterogenity*.
A=0.80368, B=0.34901, C=0.16962, D=0.32564, E=0.55078.

Conclusion

In this article we have presented a program written in METAPOST that tries to draw the picture of Shri Yantra. The "quality" of the drawing depends on five parameters that we pass to the program. We have described the approach that we used to obtain suitable values for these parameters. We have illustrated two different sets of suitable parameter values.

References

[1] Huet, Gérard. Śrī Yantra Geometry. Elsevier Science B.V.: *Theoretical Computer Science*, vol. 281(**1**–**2**): 609–628, 2002. ISSN 0304-3975. Also available on-line from `http://www.sciencedirect.com/` or may be downloaded from the author's website: `http://yquem.inria.fr/~huet/PUBLIC/Nivat.ps`

[2] Gonçalves, Luís Nobre. The `sriyantraquest.mp` program. [cit. August 1, 2009] The full source-code is available at: `http://matagalatlante.org/nobre/MetaPost/PlainMetaPostProgs/sriyantraquest.mp`. The program listing is also available at: `http://matagalatlante.org/nobre/MetaPost/PlainMetaPostProgs/sriyantra.mp`

[3] Rao, C. S. Śrīyantra – a study of spherical and plane forms. [Shri Yantra – studie kulovitých a rovinných forem.] *Indian Journal of History of Science*, vol. 33(**3**): 203–227, 1998. ISSN 0019-5235.
URL: `http://www.sriyantraresearch.com/References/Rao.pdf`

Summary:
Drawing the Shree Yantra Core with METAPOST

The Shree Yantra Core is a figure composed of nine interlocking isosceles triangles, all inside a circle. This figure is found in some of the oldest hindu temples and contains, in itself, a series of interesting mathematical problems. This article describes an attempt to solve the problem of drawing the Shree Yantra Core with METAPOST.

Key words: METAPOST, Shree Yantra Core, India, Optimization, Trial-and-Error, Simulation, Religion Symbols, Eastern Mysticism.

Luís Nobre Gonçalves, `nobre@cii.fc.ul.pt`
CFMC/UL, Complexo Interdisciplinar da Universidade de Lisboa
Avenida Professor Gama Pinto 2, P-1649-003 Lisboa, Portugal

Michal Mádr, `m.madr@seznam.cz`, Pavel Stříž, `striz@fame.utb.cz`
ÚSKM FaME UTB ve Zlíně, Mostní 5139
Zlín, CZ-760 01, Czech Republic

Program **Highlight** a jeho užití

André Simon

Contents

Abstrakt

Článek představuje program Highlight, `http://andre-simon.de/`, který je určen k formátování zdrojových kódů. Průřezově představuje jeho možnosti a potenciál. Program umí pracovat v grafickém režimu, dávkově a může být načten jako externí knihovna.

Více či méně použitelných alternativ je relativně hodně. Namátkou zmiňme TEX balíčky fancyvrb a listings. Mimo TEX pak:

- GeSHi – Generic Syntax Highlighter, `http://qbnz.com/highlighter/`,
- GNU Source-highlight, `http://www.gnu.org/software/src-highlite/`, či,
- Pygments, `http://pygments.org/`.

V rámci experimentů představuje autor ve svém programu přístup k formátování více jazyků v jednom dokumentu, např. u dokumentů a webových stránek při kombinaci HTML+PHP+CSS+JavaScript, LATEX+Sweave+R, Perl+TEX, Lua+TEX apod.

Klíčová slova: Program Highlight, zvýrazňování syntaxe, formátování zdrojových kódů s více jazyky.

1. Introduction

Syntax highlighting is a must-have feature for a programmer's text editor. The colouring of language elements helps to quickly overview large code portions mingled with comments, or to correct syntax errors before the compilation process is started. Highlight is a utility to preserve code highlighting when the source code is published on the web or in print documents. Apart from adding colours to the output, it contains reformatting features to obtain a consistent code layout.

2. Features

Highlight takes an input file and searches for syntax elements which should be outputted in a distinct format. These elements may be keywords, comments, strings, numbers, escape sequences and directives. Syntax elements are stored in text configuration files (language definitions). The accompanying formatting information is stored in separate configuration files (colour themes). These themes define font faces, language element colours and background colour. Both configuration entities may be customized by the user. Highlight is delivered with 140 language definitions and 40 colour themes.

The following output file types are supported: HTML, XHTML 1.1, RTF, TEX, LATEX, SVG, BBCode, terminal escape sequences and XML. Most of the formats have special options to reduce manual editing of the result.

Long input lines may be wrapped automatically, taking care of function call and statement indentation. C-style languages can be reformatted using several indentation styles (including GNU and K&R variants).

If the output format supports style files, the formatting information is stored in referenced files by default (HTML and SVG: CSS files, TEX and LATEX: `sty` files). The separation of styles and content is more efficient and simplifies design changes of existing documents. Style files may include user defined files to override and enhance highlight's default formatting.

Highlight is distributed as library, command line (CLI) and graphical user interface (GUI). The CLI offers the complete feature set, whereas the GUI includes an instant preview to show the impact of formatting changes.

3. Command line and graphical interfaces

The following examples show how to produce a highlighted C++ file, using an input file called `main.cpp`:

- Generate HTML:

  ```
  highlight -i main.cpp -o main.cpp.html
  highlight < main.cpp > main.cpp.html --syntax cpp
  ```

  You will find the HTML file `main.cpp.html` and `highlight.css` in the working directory. If no input filename is defined by `--input` or given at the prompt, highlight is not able to determine the language type by means of the file extension. Only some scripting languages are determined by the shebang in the first input line. In this case you have to pass Highlight the given language with `--syntax` (this should be the file suffix of the source file in most cases).

- Generate LATEX with embedded style definitions and line numbers:

  ```
  highlight --latex -i main.cpp -o main.cpp.tex
     --include-style --linenumbers
  ```

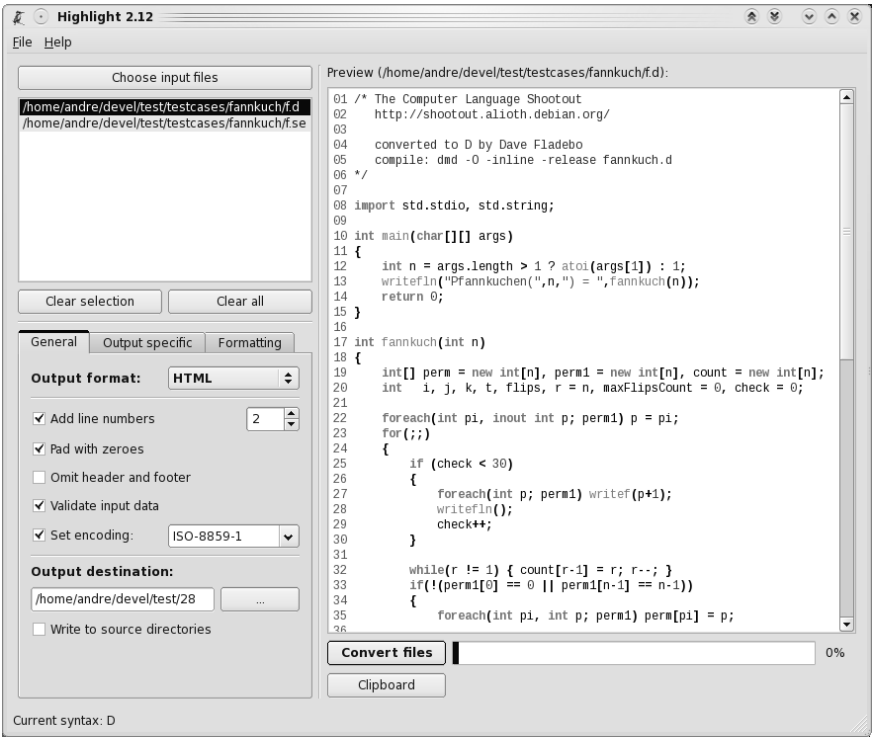
- Generate plain TEX using "ANSI" source formatting style and "print" colour theme:

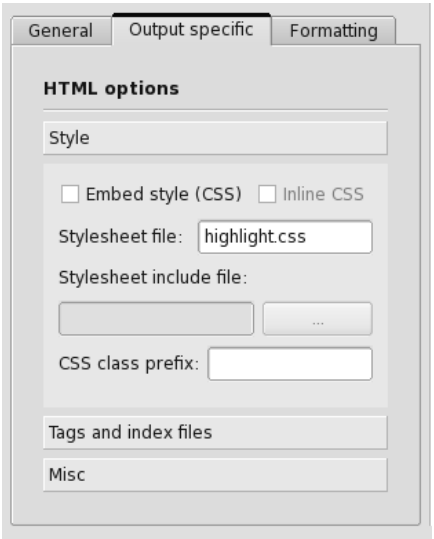
  ```
  highlight --tex -i main.cpp -o main.cpp.tex
     --reformat ansi --style print
  ```

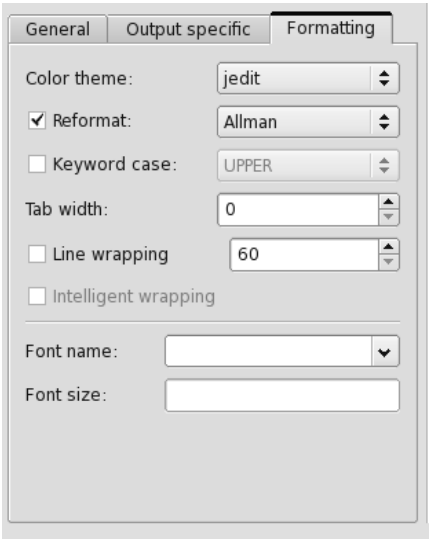
- Define an output directory:

  ```
  highlight -O some/target/dir/ *.cpp *.h
  ```

Screenshots on the next page show the preview window and the option panes.


Highlight 2.12
File Help
Choose input files
/home/andre/devel/test/testcases/fannkuch/f.d
/home/andre/devel/test/testcases/fannkuch/f.se
Clear selection
Clear all
General
Output specific
Formatting
Output format:
HTML
Add line numbers
2
Pad with zeroes
Omit header and footer
Validate input data
Set encoding:
ISO-8859-1
Output destination:
/home/andre/devel/test/28
...
Write to source directories
Preview (/home/andre/devel/test/testcases/fannkuch/f.d):
01 /* The Computer Language Shootout
02 http://shootout.alioth.debian.org/
03
04 converted to D by Dave Fladebo
05 compile: dmd -O -inline -release fannkuch.d
06 */
07
08 import std.stdio, std.string;
09
10 int main(char[][] args)
11 {
12 int n = args.length > 1 ? atoi(args[1]) : 1;
13 writefln("Pfannkuchen(",n,") = ",fannkuch(n));
14 return 0;
15 }
16
17 int fannkuch(int n)
18 {
19 int[] perm = new int[n], perm1 = new int[n], count = new int[n];
20 int i, j, k, t, flips, r = n, maxFlipsCount = 0, check = 0;
21
22 foreach(int pi, inout int p; perm1) p = pi;
23 for(;;)
24 {
25 if (check < 30)
26 {
27 foreach(int p; perm1) writef(p+1);
28 writefln();
29 check++;
30 }
31
32 while(r != 1) { count[r-1] = r; r--; }
33 if(!(perm1[0] == 0 || perm1[n-1] == n-1))
34 {
35 foreach(int pi, int p; perm1) perm[pi] = p;
Convert files
0%
Clipboard
Current syntax: D


General
Output specific
Formatting
HTML options
Style
Embed style (CSS)
Inline CSS
Stylesheet file:
highlight.css
Stylesheet include file:
...
CSS class prefix:
Tags and index files
Misc


General
Output specific
Formatting
Color theme:
jedit
Reformat:
Allman
Keyword case:
UPPER
Tab width:
0
Line wrapping
60
Intelligent wrapping
Font name:
Font size:

4. Output options

The following incomplete list shows the most interesting options applicable to all output formats:

--style-outfile=<file>, --style-infile=<file>
Define style definition filename, and the file to be included in style-outfile.

--fragment
Omit header and footer of the output to embed it in existing documents.

--reformat=<style>
Reformat output in given style, where <style> is one of allman, banner, gnu, java, k&r, linux, stroustrup, or whitesmith. This feature is available for C-style syntax.

--include-style
Include style definition in each output file.

--wrap, --wrap-simple, --line-length=<num>
Wrap long lines with or without taking care of function call indentation. Set line length before wrapping.

--font=<font>, --font-size=<size>
Set font face and size (specific to output format).

--linenumbers, --line-number-start=<cnt>,
--line-number-length=<num>
Print line numbers, set line numbering start and length incl. left padding.

--style=<style name> Set highlighting colour style.

--replace-tabs=<num> Replace tabs by given number of spaces.

--encoding=<enc>
Set output encoding which matches input file encoding; omit encoding information if <enc> is "NONE".

--kw-case=<upper|lower|capitalize>
Output all keywords in given case if language is not case sensitive.

--start-nested=<language>
Define nested language which starts input without opening delimiter.

4.1. HTML/XHTML options

--anchors Attach anchors to line numbers (HTML only).

--anchor-prefix=<prefix> Set anchor name prefix.

--anchor-filename Use input file name as anchor name.

--print-index Output index file with links to all output files.

--ordered-list Output lines as ordered list items (assumes `--linenumbers`).

--class-name=<name> Set CSS class name to avoid name clashes.

--inline-css
Output CSS information within each tag (generates verbose output).

--mark-line='n[=txt]; m'
Mark given lines n..m and add optional help texts as tooltips.
--enclose-pre Enclose fragmented output in pre tags (assumes `--fragment`).
--ctags-file[=<file>] Read ctags file to include meta information as tooltips (default file value: "tags").

4.2. LaTeX options

--babel Disable babel package shorthands.
--replace-quotes Replace double quotes by \dq.
--pretty-symbols Add character boxes to improve appearance of brackets and other symbols.

The resulting LaTeX articles require the color package. If the output is UTF-8 encoded, the ucs package is needed. If the input lines are wrapped, the marvosym package is included to represent inserted line breaks by the right torque symbol.

The formatting information is stored as commands with a "hl" prefix. These commands are used to enclose recognized syntax tokens in the output. Whitespace is escaped as "\ " to assure correct indentation of code blocks. The `--babel` and `--replace-quotes` options can be applied to gain compatiblity with babel or other packages. The `--pretty-symbols` option adds boxes for special characters like parantheses, @, # etc. These boxes improve the appearance of the characters by fitting their dimensions to the surrounding text.

Plain TeX documents include macros to apply formatting to recognized syntax tokens. The macro naming scheme corresponds to the LaTeX output. A "special" command is used to set the background colour, which may not be supported by all `dvi` drivers. Page numbering is disabled.

4.3. RTF options

--page-size=<size>
Set page size, where <size> is one of a3, a4, a5, b4, b5, b6, or letter.
--char-styles Include character stylesheets.

4.4. SVG options

--height=<height>, --width=<width>
Set image height and width (units allowed).

5. Configuration

5.1. File format

All Highlight configuration files are stored as plain ASCII text files, using the convention *$ParamName=ParamValue*.

Parameter names are not case sensitive. The value may be a single character, a list of words or a regular expression. Lists may be split in multiple lines. Comments start with # as the first character of a line.

5.2. Language definitions

A language definition describes all elements of a programming language which will be highlighted by different colours and font types.

File format:

```
# Regular expression to describe valid number tokens.
# Default value is set in the parser.
$DIGIT=regex(<RE>)

# Regular expression to describe valid identifier tokens.
# Default value is set in the parser.
$IDENTIFIER=regex(<RE>)

# List of keywords or regular expressions.
# <group> is the name of the keyword group.
# The group must be defined in the applied colour theme to provide a matching
# highlighting style.
$KEYWORDS(<group>)=regex(<RE> <, GROUP-NUM>) | <List>

# List of String delimiters.
$STRINGDELIMITERS=<List>

# List of string delimiters which are not equal (open != close).
$STRING_UNEQUAL=<open close>

# List of escape characters in Strings (ie. "\") or regular expression.
$ESCCHAR=<List> | regex(<RE>)

# Escape characters may appear outside of strings.
$ALLOWEXTESCAPE=<true|false>

# Prefix which disables highlighting of escape characters within a string.
$RAWSTRINGPREFIX=<character>

# Delimiters of multi line comments.
# Delimiter comment_close may be emitted if $ALLOWNESTEDCOMMENTS is false.
$ML_COMMENT=<comment_begin comment_close>

# List of strings which start single line comments.
$SL_COMMENT=<List> | regex(<RE>)

# Prefix of preprocessor directive lines.
$DIRECTIVE=<prefix> | regex(<RE>)

# Character which continues a compiler directive after a line break.
$CONTINUATIONSYMBOL=<symbol>

# Source code may be reformatted (only C-style languages!).
$REFORMATTING=<true | false>

# Symbols (brackets or operators).
$SYMBOLS=<List>
```

```
# Multiple line comments may be nested.
$ALLOWNESTEDCOMMENTS=<true | false>

# Programming language is case sensitive.
$IGNORECASE=<true | false>

# Include another language definition stored in the same data directory.
$INCLUDE=<language definition>

# Define the opening and closing delimiter expressions of the embedded language.
# There may be multiple entries for the same language.
$NESTED(language)=regex(<RE>) regex(<RE>)
```

5.3. Regular expressions

Highlight supports Perl-style regular expressions, which are evaluated for each input line. They are defined as *regex(<RE> <, GROUP-NUM>)*, where RE is the regex string, and GROUP-NUM is an optional parameter which defines the matching group number. If the regex contains multiple grouping parantheses, GROUP-NUM is the number of the group, whose match should be returned as a highlighted element (count number from left to right). The capturing states of the groups are irrelevant for counting. Legal values: 0 <= GROUP-NUM <= highest group index. A GROUP-NUM of 0 describes the complete match. If GROUP-NUM is undefined, the group match with the highest number will be returned.

Regex Examples:

```
$KEYWORDS(kwa)=regex([A-Z]\w+)
```

Highlight identifiers beginning with a capital letter.

```
1. $KEYWORDS(kwc)=regex(\$\{(\w+)\}), or
2. $KEYWORDS(kwc)=regex(\$\{(\w+)\}, 1)
```

Highlight variable names like $name. Only the name is highlighted as a keyword (\w+). The grouping feature is used to achieve this effect. If no capturing group index is defined, the right-most group's match (highest capturing index) is returned, see Example 1 above.

```
$KEYWORDS(kwd)=regex((\w+)\s*\()
```

Highlight method names. Note that grouping is used again.

5.4. Nested syntax configuration

Language definitions may contain nested language delimiters to change the syntax information while parsing a file. This is useful to highlight mixed code files like HTML with PHP instructions. It is also applicable to literate programming, inputs like LuaTEX and Sweave files. Currently the following code combinations are configured:

- Pascal + Assembly.
- HTML + PHP + CSS + JavaScript.
- LaTeX + Perl + Lua + R.

If the file starts with the nested syntax omitting the open delimiter, the `--start-nested` option will load the given syntax, see LuaTeX output examples.

5.4.1. Pascal definition file

```
$DESCRIPTION=Pascal
$KEYWORDS(kwa)= absolute abstract and array as asm assembler
   automated begin [...]
$KEYWORDS(kwb)=boolean char integer pointer real text true
   false cardinal [...]
$KEYWORDS(kwc)=if else then downto do for repeat while
   to until with
$KEYWORDS(kwd)=regex((\w+)\s*\()
$STRINGDELIMITERS=" '
$SL_COMMENT=//
$ML_COMMENT={ } (* *)
$IGNORECASE=true
$SYMBOLS= ( ) [ ] , ; : & | < > !  = / * %  + - @ . ^
$ESCCHAR=regex(\#\$\p{XDigit}{2}|\#\d{,3})
$DIGIT=regex((?:0x|0X|\$)[0-9a-fA-F]+|\d*[\.\_]?\d+(?:[eE]
   [\-\+]\d+)?[lLuUbfdm]*)
$ALLOWEXTESCAPE=true
$NESTED(asm)=regex(asm) regex(end;)
```

5.4.2. HTML definition file

```
$DESCRIPTION=HTML
   # [...]
$NESTED(php)=regex(\<\?php) regex(\?\>)
$NESTED(jsp)=regex(\<\%[@!\=]?) regex(.*(%>).*)
$NESTED(css)=regex(\<style\s+type\=\"text\/css\"\>)regex(\<\/style\>)
$NESTED(js)=regex(\<script\s+language\=\"JavaScript\"(?:\s+
   type\=\"text\/javascript\")?\>)regex(\<\/script\>)
```

5.4.3. LaTeX definition file

```
$DESCRIPTION=TeX and LaTeX
   # [...]
# PerlTeX delimiters
# Issue: } is also a Perl symbol
$NESTED(pl)=regex(\\perl(?:re)?newcommand\{\\\w+\}\[.*\]\{)
   regex((?<!\{)\})
```

```
$NESTED(pl)=regex(\\perl(?:re)?newenvironment\{\w+\}\[.*\]\{)
   regex((?<!\{)\})

# LuaTeX delimiters
# Issue: } is also a Lua symbol
$NESTED(lua)=regex(\\directlua.*?\{) regex((?<!\{)\})

# Sweave delimiters
$NESTED(r)=regex(^\<\<.*\>\>\=) regex(@)
```

5.4.4. Example: LaTeX, Sweave and R

Formatted example by Professor Leisch follows, see Example 3 (`http://www.stat.uni-muenchen.de/~leisch/Sweave/example-3.Snw`).

documentclass[a4paper]{article}
begin{document}
<<**echo=false,results=hide**>>=
library(**lattice**)
library(**xtable**)
data(cats, package=*"MASS"*)
@
section*{The Cats Data}
Consider the **texttt**{cats} regression example from Venables \& Ripley (1997). The data frame contains measurements of heart and body weight of **Sexpr**{nrow(cats)} cats (**Sexpr**{sum(cats$Sex=="F")} female, **Sexpr**{sum(cats$Sex=="M")} male).
A linear regression model of heart weight by sex and gender can be fitted in R using the command
<<>>=
lm1 = **lm**(Hwt∼Bwt*Sex, **data**=cats)
lm1
@
Tests for significance of the coefficients are shown in Table∼**ref**{tab:coef}, a scatter plot including the regression lines is shown in Figure∼**ref**{fig:cats}.
SweaveOpts{echo=false}
<<**results=tex**>>=
xtable(lm1, caption=*"Linear regression model for cats data."* , label=*"tab:coef"*)
@
begin{figure}
  centering
<<**fig=TRUE,width=12,height=6**>>=

```
lset(col.whitebg())
print(xyplot(Hwt~Bwt|Sex, data=cats, type=c("p", "r")))
@
  \caption{The cats data from package MASS.}
  \label{fig:cats}
\end{figure}
\begin{center}
\end{center}
\end{document}
```

5.4.5. Example: First LuaTEX output

Formatted source code from the [NTG-context][Dev-luatex] mailing list, see `http://markmail.org/message/h3d6ju5c5umah57h`.

```
(* LuaTeX test *)
\def\sortedsequence#1#2{\directlua {
      do
          local sep = "\luaescapestring{#1}"
          local str = "\luaescapestring{#2}"
          local tab = {}
          for s in string.gmatch(str,"([^"..sep.."]+)") do
              tab[\string ##tab+1] = s
          end
          table.sort(tab)
          tex.sprint(table.concat(tab,sep))
      end
}}
\expandafter\ifx\csname directlua\endcsname \relax \else
\directlua 0 {tex.enableprimitives('',tex.extraprimitives ())}
\fi
(* End *)
```

5.4.6. Example: Second LuaTEX output

A LuaTEX output example follows, for the full source code you may like to check `http://www.andre-simon.de/dokuwiki/doku.php?id=luatex` and the website at `http://lua-users.org/wiki/MakingLuaLikePhp`.

```
% This LuaTeX file starts with Lua code: to highlight it properly use this cmd:
% highlight --latex --start-nested=inc_luatex input.tex
function explode(str,divider)
 local result = {}
```

```
 % Function body omitted for brevity.
 return result
end

function implode(t,div)
 return table.concat(t,div)
end

% #1 = continuation (called with the sorted list as its only
% parameter), #2 = delimiter, #3 = <delimiter>-separated list
\def\sort#1#2#3{%
 \directlua0 {%
  % We convert the string into a table, delimited by a
  % (unexpanded, escaped) delimiter:
  local t = explode("\luaescapestring{\unexpanded{#3}}",
       "\luaescapestring{\unexpanded{#2}}")
  % We sort the table in-place:
  table.sort(t)
  % We output the table as a string.
  tex.print("\luaescapestring{\unexpanded{#1}}{" ..
   implode(t," ") .. "}")
 }
}

% Output tokens without expansion.
\def\typeout#1{\message{\unexpanded{[#1]}}}

\sort{\typeout}{ }{%
This file demonstrates LuaTeX.}
\bye
```

5.5. Colour themes

Colour themes contain the formatting information of the language elements which are described in language definitions.

The themes have to be stored as `*.style` files in the themes directory.

File format:

```
<Colour> = #RRGGBB
RR GG BB describe the red/green/blue hex-values which define the colour.
Value range: 00 (none) - FF (full)

<Format> = <bold> <italic> <underline>
Bold, italic and underline are optional attributes and may be combined.
```

```
# Colour of unspecified text
$DEFAULTCOLOUR=<Colour>

# Background colour
$BGCOLOUR=<Colour>

# Font size
$FONTSIZE=<number>

# Formatting of keywords, which belong to the corresponding keyword group
$KW-GROUP(<group>)=<Colour> <Format>

# Formatting of numbers
$NUMBER=<Colour> <Format>

# Formatting of escape characters
$ESCAPECHAR=<Colour> <Format>

# Formatting of strings
$STRING=<Colour> <Format>

# Formatting of comments
$COMMENT=<Colour> <Format>

# Formatting of single line comm. (optional, equals to $COMMENT if omitted)
$SL-COMMENT=<Colour> <Format>

# Formatting of compiler directives
$DIRECTIVE=<Colour> <Format>

# Formatting of strings within compiler directives
$STRING-DIRECTIVE=<Colour> <Format>

# Formatting of symbols (optional, equals to $DEFAULTCOLOUR if omitted)
$SYMBOL=<Colour> <Format>

# Formatting of line numbers
$LINE=<Colour> <Format>

# Background colour of marked lines
$MARK-LINE=<Colour>
```

Example of Darkblue:

```
$DEFAULTCOLOUR=#ffffff
$BGCOLOUR=#000040
$FONTSIZE=10
$KW-GROUP(kwa)=#e2e825
$KW-GROUP(kwb)=#60ff60
$KW-GROUP(kwc)=#26e0e7
$KW-GROUP(kwd)=#e825e2 bold
$NUMBER=#42cad9
$ESCAPECHAR=#a1a1ff
$STRING=#ffa0a0
$STRING-DIRECTIVE=#ffa0a0
$COMMENT=#80a0ff
$SL-COMMENT=#ff7f9f
$DIRECTIVE=#ff80ff
$LINE=#e5d28e
$SYMBOL=#bababa
$MARK-LINE=#006600
```

5.6. Keyword groups

You may define custom keyword groups and corresponding highlighting styles. This is useful if you want to highlight functions of a third party library, macros, constants etc. You define a new group in two steps:

- Define a group in your language definition:
  $KEYWORDS(group)
  The group attribute is the name of the new keyword group. You may use the same group name for multiple entries.
- Add a corresponding highlighting style in your colour theme:
  $KW-GROUP(group) = #RRGGBB <bold> <italic> <underline>

Note that every group name which is listed in a language definition should be defined in the used colour theme. The keyword groups "kwa"–"kwd" are predefined in all packaged colour themes.

```
# Some language definition...
$KEYWORDS(kwa)=for repeat while [...]
$KEYWORDS(debug)=ASSERT DEBUG
$ML_COMMENT=/* */
   # [...]

# Some colour theme...
$KW-GROUP(kwa)=#ddbb00 bold
$KW-GROUP(debug)=#ff0000 bold
$COMMENT=#978345 italic
   # [...]
```

5.7. File type mapping

Several programming languages make use of multiple file types to separate interface definitions from implementation modules, or there exists several file endings for historical reasons. Unix scripts usually contain the path of the script interpreter in the first comment line, which is called a shebang. The file type extensions and shebang patterns are mapped to language definitions using the `filetypes.conf` file:

```
# List of extensions
$ext(ada)=adb ads a gnad
$ext(ampl)=dat run
$ext(amtrix)=s4 s4t s4h hnd t4
$ext(asm)=a51 29k 68s 68x x68
   # [...]

# Input file recognition
# Highlight matches the first input line with the listed expressions.
# Format: $shebang(language) = <regex>
```

```
$shebang(sh)=^#!(\/usr)?(\/local)?\/bin\/(bash|t?csh|[akz]?sh)
$shebang(pl)=^#!(\/usr)?(\/local)?\/bin\/perl
$shebang(py)=^#!(\/usr)?(\/local)?\/bin\/python
$shebang(awk)=^#!(\/usr)?(\/local)?\/bin\/[gn]?awk
```

All file extensions listed in `filetypes.conf` may be used with the `--syntax` command line option.

6. Project milestones

The initial goal of Highlight was to provide a HTML generator which makes use of Cascading Style Sheets (CSS) instead of antiquated and clumsy font tags. Since a highlighted code includes a lot of formatted elements, CSS makes a big difference in terms of output file size and maintenance effort. Hence the utility was called "src2css", which stressed its purpose but was soon renamed to the more memorable "highlight".

Back at the beginning of the development in 2002, none of the available highlighting tools had configurable syntax and colour theme definitions: Either the syntax elements or the accompanying colour settings were hardcoded. To allow a flexible configuration without coding, Highlight's syntax descriptions were separated from colour themes, and all configuration files were stored as plain text format.

The initial HTML output generator code was soon generalized to provide more output formats: LaTeX, Plain TeX, RTF, terminal escape sequences, XML and BBCode followed. According to user feedback, LaTeX and TeX became the most demanded features besides HTML.

The Artistic Style library was included in 2003 to provide syntax reformatting and indentation of C-style languages (C, C++, C#, Java). This feature helped to achieve a consistent code layout, especially useful for print or presentations.

In 2005, regular expressions were added to syntax files which enabled the definition of complex patterns. This feature rendered hardcoded special cases within the parser unnecessary. A side effect of the regex integration was that the parser core had undergone very few changes since 2006, which reduced the testing effort of basic language parsing.

The first Highlight releases were tarballs with a command line interface. Later the GUI was developed using wxWidgets, and Highlight was distributed with both CLI and GUI builds. Later in 2009, wxWidgets was replaced by the Qt GUI toolkit.

While presenting this article in summer 2009 Pavel Stříž, of the Zpravodaj journal, thought about the possibility of letting Highlight parse input with nested source code, i.e. HTML+PHP, Lua+TeX, Perl+TeX, Python+TeX, R+LaTeX (Sweave), and related tools to assist in literate programming. However this was

not supported and nested language support was implemented in version 2.12 to complete the article.

7. Parser implementation

The *CodeGenerator* is the abstract base class of all output code generators. It contains the parser logic and offers the highlighter interface for clients. The inheriting generator classes have to implement several virtual methods to pass format specific information to the parser. First of all, the parser needs to know the document structure divided into header, body and footer. The code highlighting is performed with formatting tags. These opening and closing tags have to be defined before the parsing takes place. Each generator has to take care of character replacements: input characters have to be mapped to valid escape sequences (example: '>' becomes '>' in HTML).

The highlighting of syntax elements (tokens) is managed using states. A state is implemented as a method which outputs the current token in the corresponding format and decides on the next token of how to proceed. The state is finished when an ending delimiter token is recognized or if the input stream stops.

Starting in a root state, the parser branches to one of the following states:

- keyword
- number
- multi line comment
- single line comment
- string
- directive
- escape character
- symbol
- nested code begin
- nested code end
- end of line
- end of file
- whitespace

The *root state* outputs all text sections which are not recognized as tokens with default formatting. The *keyword state* handles reserved indentifiers or other tokens defined by regular expressions. This is the only state which results in multiple output formatting tags: Each keyword group comprises of its own formatting information. The *number state* is responsible for outputting numbers. The *comment states* handle single line and block comments, taking care of embedded comments. The other token-related states behave as their names suggest.

The *nested code states* process code sections of other programming languages enclosed in the main (host) language. When a nested section starts, the parser loads the assigned language definition to replace the host language syntax and stores the name of the previously loaded language. The ending delimiter expression (which is defined in the host language definition) is added to the new syntax information in order to recognize the end of the section. When the end of the section is reached, the syntax information of the host language is reloaded.

The remaining *whitespace states* have to be handled by every state, as they may reoccur. They are used to handle line numbering, the correct completion of opened tags and the parsing termination after the file ends.

Each state may branch to another state if appropriate: The string state may call the escape character state. The separation in state methods helps to control the parsing process because each highlighting element can be handled in an isolated and small code section.

8. Embedding Highlight

The Highlight parser is encapsulated in a static C++ library by default (a shared library makefile target is also available). All highlight functionality is obtained from the CodeGenerator base class, see the interface `codegenerator.h`. A CodeGenerator factory method returns an instance equivalent to the given output format, i.e. a LaTeXGenerator, which will transform a given input string or file into the chosen output format.

Apart from using the C++ library, you might prefer the SWIG tool which allows the integration of Highlight in over 10 programming languages. The Highlight distribution includes the SWIG interface file and makefiles for Python and Perl module compilation.

The following Perl code demonstrates the SWIG interface:

```
use highlight;
# get a generator instance (for HTML output)
my $gen = highlightc::CodeGenerator_getInstance($highlightc::HTML);
# initialize the generator with colour theme and syntax
$gen->initTheme("/usr/share/highlight/themes/kwrite.style");
$gen->loadLanguage("/usr/share/highlight/langDefs/c.lang");
# set some parameters
$gen->setIncludeStyle(1);
$gen->setEncoding("ISO-8859-1");
# get output string
my $output=$gen->generateString("int main(int argc, char **argv) {\n".
                                " HighlightApp app;\n".
                                " return app.run(argc, argv);\n".
                                "}\n");
print $output;
# clear the instance
highlightc::CodeGenerator_deleteInstance($gen);
```

9. Plugins and editor integration

Since the Highlight CLI supports IO redirection using the `--syntax` option, it may be easily invoked within shell scripts. The package includes plugins for popular web software (DokuWiki, MovableType, Serendipity and WordPress). To quickly integrate Highlight in other projects a sample invocation code is provided such as PHP, Perl and Python classes.

The tutorial at `http://www.lyx.org/` shows how to invoke Highlight in a convenient manner from within LyX, see `http://wiki.lyx.org/Examples/IncludeExternalProgramListing`.

The SVG output option is used in the Inkscape plugin available at `http://xico.freeshell.org/`, which includes code snippets within vector graphics.

The Highlight documentation Wiki [1] also includes instructions to integrate the CLI interface in CodeBlocks, `http://www.codeblocks.org/`, and the ancient DevC++ programme, `http://www.bloodshed.net/devcpp.html`. A native plugin for Notepad++, `http://notepad-plus.sourceforge.net/`, is available in the Download section [1].

References

[1] Simon, André. Highlight: code & syntax highlighting. [Program Highlight: zvýrazňování zdrojových kódů a syntaxe.] [on-line, cit. September 19, 2009] The programme and its DokuWiki are available from the author's websites: Homepage of the programme: `http://www.andre-simon.de/` Wiki documentation: `http://wiki.andre-simon.de/`

Summary: The Highlight Programme: Code & Syntax Highlighting

The article presents features and options of the Highlight programme which is capable of highlighting source codes of different programming languages. The experimental approach of nested syntax configuration is tested on files with HTML+CSS+PHP+JavaScript, Perl+TeX, LaTeX+Sweave+R and Lua+TeX.

Key words: The Highlight programme, Syntax highlighting, Source code formatting, Nested syntax configuration.

André Simon, as@andre-simon.de
Eurener Str. 53C, Trier, DE-54294 Germany

Pár postřehů z TEXperience 2009

Jiří Demel

Bylo to druhé setkání českých a slovenských TEXistů pod názvem TEXperience. Konalo se v poněkud netradičním termínu 21. až 24. května 2009. Od toho předešlého TEXperience 2008 (říjen 2008) neuplynul ani rok. Mnohé bylo „jako vloni“. Nemohu si pomoci, budu srovnávat.

- Setkání se konalo opět na Valašsku, tentokrát v jeho jižní části, v rekreačním středisku Královec na stejnojmenném kopci nedaleko Valašských Klobouk.
- Pořadateli byli i tentokrát CSTUG a ÚSKM FaME UTB ve Zlíně.
- Hlavními hybateli konference byli opět Pavel Stříž, Pavel Stříž, ještě jednou Pavel Stříž, rodina Pavla Stříže, ještě jednou rodina Pavla Stříže a Zdeněk Wagner. Všichni se zhostili svého úkolu opět mile a perfektně.
- Přivítání bylo – jak se na Valašsko sluší a patří – no ano, opět slivovicí. Kromě toho každý účastník dostal tašku se spoustou konferenčních materiálů. Součástí materiálů byla i letos knížka TEXem sázená. A jako vloni, i letos byl autorem Zdeněk Wagner, ale nebyly to pohádky, nýbrž sci-fi příběh nazvaný *Chromozom 46,YB*.
- Provozovatelé rekreačního střediska, mladí manželé, byli milí a vstřícní. Ubytování i domácí stravu lze jen pochválit.
- Ta knížka tentokrát nebyla jen tak, tentokrát to byla knížka, která zvítězila v soutěži *Kniha pro TEXperience 2009*. Dlužno upřesnit, že tato knížka byla jediným účastníkem soutěže. Příležitost přeje připraveným a Zdeněk Wagner zřejmě byl jediným připraveným. (Podezřívám jej, že vůbec nikdy nespí, že má v šupleti zásobu nevydaných knížek a při vhodných příležitostech je vytahuje.)
- Odborný program se konal v lehce kuriózním prostředí takzvané rehabilitační budovy. To je takový domek, kde je sauna, vířivka, několik masážních stolů a uprostřed toho všeho je větší místnost, nazývaná hrdě tu tělocvičnou, tu společenskou místností. V našem případě byla místnost pomocí promítacího plátna, projektoru, počítače, wifi routeru a spousty židlí proměněna v místnost konferenční. Milou součástí vybavení byla zásoba rozličného pití (ovšem již jen nealkoholického), které bylo účastníkům volně k disposici.
- Program konference nebudu komentovat, psané verze příspěvků byly v minulém a jsou také v tomto čísle Zpravodaje. Příspěvek Pavla Stříže byl (značně rozšířený) otištěn ještě v čísle dřívějším.
- Součástí konference byly dva výlety: autobusem do Vizovic a pěšky do Valašských Klobouk. Ve Vizovicích jsme navštívili zámek, zamlsali jsme si

v zámecké čokoládovně, ovšem korunou výletu byla exkurze ve světoznámé likérce Rudolf Jelínek, včetně ochutnávky a nákupu v podnikové prodejně. Před prodejnou stojí dřevěná socha Jožina z bažin, rodáka z Vizovic, v nadživotní velikosti.

- Druhý výlet byl pěší. Sestupem z kopce jsme dobyli město Valašské Klobouky, vyslechli jsme výklad Miroslavy Dolejšové o historii města (viz samostatný článek v předchozím Zpravodaji) a navštívili jsme dvě místní muzea. V městském muzeu zaujaly četné pravěké nálezy a výstava černobílých fotografií z Valašska 60. a 70. let Jitky Kovandové *Zastavený čas*. Ve druhém muzeu, v tzv. Červeném domě, je národopisná expozice, kde hlavní atrakcí byl funkční tkalcovský stav. Jan Štěpnička, jeden z účastníků konference, zde pod vedením průvodce utkal asi centimetr plátna. Je obdivuhodné, jak ta soustava prkýnek a provázků dokáže fungovat.
- Další atrakcí ve Valašských Kloboukách byla kyselica (zelná polévka), na kterou jsme zašli do místního motorestu. Nakonec nás Miroslava Dolejšová zavedla k městským rybníkům na úpatí Královce, kde nás opustila s tím, že dál už na Královec trefíme sami. Samozřejmě, jen co odešla, okamžitě jsme zabloudili. Naštěstí instinkt, že musíme do kopce, nezklamal. Kolem krásné valašské zvoničky jsme vystoupili na horu Královec (655 m), vylezli jsme na rozhlednu a kolem několika obrovitých mraveništ jsme šťastně trefili na večeři a večerní program.
- Třetí atrakcí byla přednáška amatérského astronoma Jiřího Vašátka o Sluneční soustavě, která skončila uprostřed noci, spíše nad ránem, pozorováním hvězd s dalekohledem i bez něj. Nebe bylo bezmračné, vzduch čistý, vedli jsme řeči, viděli jsme několik meteoritů, někteří si stihli něco přát.
- Nejpůvabnějším výrokem letošní konference byl bezelstný dotaz Karla Píšky: „A k čemu se vlastně používá ten Word?“
- Proslýchalo se, že příští ročník TEXperience by snad mohl být někde poblíž Plzně, snad v období září-říjen 2010.

Jiří Demel, demel@fsv.cvut.cz

~ ~ ~

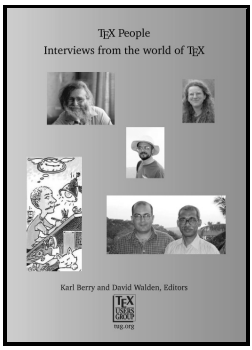
Výbor CSTUG a redakce Zpravodaje Vám přeje

Nové a staronové knihy

Michal Mádr, Pavel Stříž

TEX People

Interviews from the World of TEX
Karl Berry, David Walden (editoři)



Vydání: první.
Počet stran: 312.
Vazba: brožovaná.
Rok vydání: 2009.
Nakladatel: TEX Users Group.
ISBN-10: 0982462603.
ISBN-13: 978-0982462607.
Více o knize: `http://tug.org/store/texpeople`

Tato Kniha je unikátním nahlédnutím do TEXové komunity a do souvislostí mezi různými TEXovými systémy. Jedná se o zajímavé čtení pro zasvěcené i laiky.

Knihu tvoří padesát rozhovorů s lidmi zapojenými do nejrůznějších TEXových aktivit. Otázky pokládali David Walden a Karl Berry v letech 2004 až 2009. Rozhovory byly poprvé publikovány v Koutku rozhovorů (Interview Corner) na stránkách TUG, viz `http://www.tug.org/interviews/`.

Výběr zpovídaných je široký, zastoupeni jsou např. klíčoví tvůrci LATEXu, pdfTEXu, LuaTEXu, ConTEXtu, XETEXu nebo systému Omega. Dále lidé spojení s TUG, TUGboat, správci TEXových distribucí, autoři známých TEXových knih, editoři, ilustrátoři, TEXoví uživatelé a učitelé, tvůrci LATEXových balíků, tvůrci fontů, designéři knih, ... Škoda jen, že chybí rejstřík osob.

Rozhovory začínají otázkou na neTEXovou minulost zpovídaných. V TEXové části rozhovorů se mluví např. o tom, jak se kdo k TEXu dostal, jak a k čemu ho používá, co si myslí o jeho přednostech a nedostatcích, o budoucnosti jak programu TEX, tak TEXové komunity, ... Jak editoři píší v předmluvě této knihy, tři významné TEXové postavy zpovídány nebyly, protože obsáhlé rozhovory s nimi už vyšly. Jedná se o Donalda Knutha, Hermanna Zapfa a Leslieho Lamporta. Odkazy na tyto rozhovory jsou uvedeny v Koutku rozhovorů.

O technických aspektech vzniku knihy viz článek „TEX People: The TUG interviews project and book“ v časopisu TUGboat, vol. 30(**2**): 196–202, 2009.

Pro zajímavost uvádíme: ve dvou článcích tohoto čísla Zpravodaje, rozhovoru s Hàn Thế Thành (zejména o pdfTEXu) a představení LuaTEXu, bylo zmíněno dohromady devět osob (většina z nich lidé, kteří se podíleli na následnických programech TEXu), jejichž rozhovory lze v knize nalézt.

The Evolution of Lua

Roberto Ierusalimschy, Luiz Henrique de Fiqueiredo, Waldemar Celes

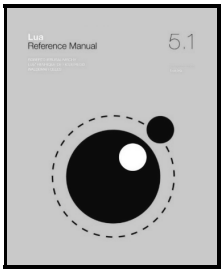
Autoři přináší informace o vzniku a vývoji jazyka Lua. Popisují, jak se z jednoduchého konfiguračního jazyka vyvinul univerzální, široce využívaný jazyk podporující rozšiřitelnou semantiku, anonymní funkce, plný lexikální scoping, koncovou rekurzi a koprogramy. Článek je stáhnutelný z webových stránek autorů: `http://www.tecgraf.puc-rio.br/~lhf/ftp/doc/hopl.pdf`.

Vyzkoušet si Lua můžete v Live verzi na `http://www.lua.org/demo.html`.

Za pozornost též stojí `http://lua-users.org/wiki/TutorialDirectory`.

Lua 5.1 Reference Manual

Roberto Ierusalimschy, Luiz Henrique de Fiqueiredo, Waldemar Celes

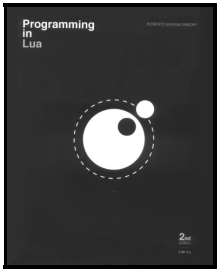


Vydání: první.
Počet stran: 112.
Vazba: brožovaná.
Rok vydání: 2006.
Nakladatel: Lua.org.
ISBN-10: 85-903798-3-3.
ISBN-13: 978-85-903798-3-6.
Informace od nakladatele: `http://lua.org/manual/5.1/`

Tento manuál je oficiální definicí jazyka Lua 5.1. Pokrývá syntaxi a semantiku, kompletní C API a standardní knihovny. Tato tištěná verze obsahuje plný text verze elektronické, dostupné na `http://www.lua.org/manual/`.

Programming in Lua

Roberto Ierusalimschy



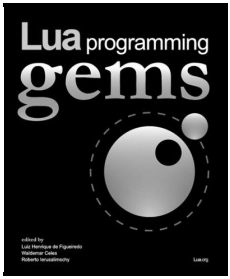
Vydání: druhé.
Počet stran: 328.
Vazba: brožovaná.
Rok vydání: 2006.
Nakladatel: Lua.org.
ISBN-10: 85-903798-2-5.
ISBN-13: 978-85-903798-2-9.
Online verze prvního vydání: `http://www.lua.org/pil/`
Informace od nakladatele: `http://www.inf.puc-rio.br/~roberto/pil2/`

Tato kniha, napsaná hlavním architektem jazyka Lua, je oficiální zdrojem informací o tomto jazyku. Kniha poskytuje slušný základ pro kohokoliv, kdo chce

jazyk Lua používat. Kniha pokrývá všechny aspekty jazyka Lua 5 – od základů až po API pro jazyk C. Autor na mnoha příkladech ukazuje, jak využít výhod tohoto jazyka. Kniha se obrací na čtenáře s určitou zkušeností s programováním, ale žádná předchozí zkušenost s jazykem Lua nebo jiným skriptovacím jazykem se u čtenářů nepředpokládá.

Lua Programming Gems

Roberto Ierusalimschy, Luiz Henrique de Fiqueiredo, Waldemar Celes



Vydání: první.
Počet stran: 368.
Vazba: brožovaná.
Rok vydání: 2008.
Nakladatel: Lua.org.
ISBN-10: 85-903798-4-1.
ISBN-13: 978-85-903798-4-3.
Informace od nakladatele: `http://www.lua.org/gems/`

Tato kniha je kolekcí článků shromažďujících zkušenosti s programováním v jazyce Lua. Články jsou více než přehledem tipů – autoři dávají nahlédnout do různých aspektů programování v Lua, základních i pokročilých. Záběr článků je široký, mimo jiné je zmíněno programování her, programovací techniky, algoritmy a datové struktury a techniky designu.

Sedmero dračích srdcí

Zdeněk Wagner

Vydání: první.
Počet stran: 128.
Vazba: vázaná.
Rok vydání: 2008.
Nakladatel: Martin Stříž.
ISBN-13: 978-80-87106-11-2.
Informace od nakladatele:
`http://icebearsoft.euweb.cz/sedmero.dracich.srdci`

Představujeme vám knihu pohádek Zdeňka Wagnera vysázenou v TEXu, která byla pokřtěna na TEXperience 2008 a příjemně doplnila odborný ráz prvního ročníku našich společných setkání.

Zpravodaj Československého sdružení uživatelů TEXu
ISSN 1211-6661 (tištěná verze), ISSN 1213-8185 (online verze)

Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Obálka:	Antonín Strejc
Ilustrace na obálce:	Luís Nobre Gonçalves, Michal Mádr, Pavel Stříž
Počet výtisků:	500
Uzávěrka:	5. 11. 2009
Odpovědný redaktor:	Zdeněk Wagner
Redakční rada:	Pavel Stříž, David Catto a výbor CSTUGu
Tisk a distribuce:	KONVOJ, spol. s r. o., Berkova 22, 612 00 Brno, tel. +420 549 240 233
Adresa:	CSTUG, c/o FEL ČVUT, Technická 2, 166 27 Praha 6
Tel:	+420 224 353 611
Fax:	+420 233 332 938
Email:	cstug@cstug.cz

Zřízené poštovní aliasy sdružení CSTUG:

`bulletin@cstug.cz`, `zpravodaj@cstug.cz`
korespondence ohledně Zpravodaje sdružení

`board@cstug.cz`
korespondence členům výboru

`cstug@cstug.cz`, `president@cstug.cz`
korespondence předsedovi sdružení

`gacstug@cstug.cz`
grantová agentura CSTUGu

`secretary@cstug.cz`, `orders@cstug.cz`
korespondence administrativní síle sdružení, objednávky CD a DVD

`cstug-members@cstug.cz`
korespondence členům sdružení

`cstug-faq@cstug.cz`
řešené otázky s odpověďmi navrhované k zařazení do dokumentu CSFAQ

`bookorders@cstug.cz`
objednávky tištěné TEXové literatury na dobírku

ftp server sdružení:
`ftp://ftp.cstug.cz`

www server sdružení:
`http://www.cstug.cz`

Pozvánka od Karla Berryho na TUG 2010: TEX sláví 2^5

Nadcházející 31. ročník konference TUG 2010 se bude konat v San Franciscu v USA, od 28. do 30. června 2010.

Protože se TEX v roce 2010 dožívá třiceti dvou let, našeho setkání se mimořádně zúčastní sám „velký čaroděj“. Kromě profesora Knutha by se měli zúčastnit další původní členové Stanfordského TEX projektu, včetně Davida Fuchse, Johna Hobbyho, Michaela Plasse, Orena Pataschnika a Toma Rokickiho.

Na poslední den budou připraveny různé aktivity, včetně banketu k poctě projektu Stanfordského TEXu a k oslavě narozenin TEXu. Informace pro přípravu příspěvku, registrační formuláře a další detaily budou zveřejněny na

`http://tug.org/tug2010/`