CSTUG

ZPRAVODAJ

Československého sdružení uživatelů TEXu

1–4

2019

ISSN 1211-6661 Tištěná verze

ISSN 1213-8185 Online verze

Ročník 29

OBSAH

Zpravodaj Československého sdružení uživatelů TeXu je vydáván v tištěné podobě a distribuován zdarma členům sdružení. Vydaná čísla Zpravodaje v elektronické podobě (PDF) jsou bezodkladně veřejně vystavena na webové adrese `https://www.cstug.cz/`.

Své příspěvky do Zpravodaje můžete zasílat v elektronické podobě, nejlépe jako jeden archivní soubor `(.zip, .arj, .tar.gz)`, na e-mailovou adresu `bulletin@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)

Úvodník staronového předsedy

Petr Sojka

Milé čtenářky a čtenáři, příznivci kvalitní typografie sázecím systémem TEX,

otevíráte první a poslední číslo Zpravodaje v roce 2019, čtyřčíslo s letošními články. Po třech letech mi opět dovolte při příležitosti mého znovuzvolení předsedou našeho zapsaného spolku malé zastavení a zhodnocení stavu CSTUGu a Zpravodaje.

Ohlédnutí zpět

Jelikož jsem předsedal CSTUGu kromě období 1995–2001 již poslední tři volební období (2010–2019), je na místě zhodnotit alespoň uplynulé tři roky. Sdružení finišuje třetí dekádu své existence, a charakter sdružování se ve světě vysokorychlostního internetu a snadné digitální komunikace mění. Mnoho se od mého předání sdružení v roce 2001 kolegům Olšákovi (2001–2004) a Kubenovi (2004–2010) a za dobu mého předchozího předsedování změnilo.

Lokální skupiny uživatelů se spíše stávají profesními organizacemi, které pečují regionálně o dědictví a perspektivu oboru, v našem případě o dědictví a perspektivu digitální typografie sázecím systémem TEX, resp. spíše pdfTEX, který v našem regionu vznikl.

V úvodníku staronového předsedy ve Zpravodaji 2016/1–4 jsem avizoval svůj program udržitelnosti sdružení a novou koncepci vydávání Zpravodaje. Kromě dodržení všech zákonných povinností chodu neziskové organizace se podařilo i tuto koncepci a algoritmus vydávání čísel časopisu dodržet. Dnes již dokonce máme převis nabízených článků a tak doufám, že další číslo Zpravodaje již najdete ve svých schránkách v první půli roku 2020. Konsolidace vydávání našeho Zpravodaje stejně jako členské základny se tedy podařila. Velké díky patří šéfredaktorovi Zpravodaje Honzovi Šustkovi a technickou redakci Zpravodaje skvěle zvládá Vít Novotný. Věkový průměr autorů článků se pozvolna snižuje a vznikají smysluplné projekty hodné podpory, což mě naplňuje dalším optimismem ohledně udržitelnosti sdružení. Podařilo se také přispět drobným dílem k návštěvě Grand Wizarda v Brně.

Valné shromáždění

Letošní valná hromada proběhla již v říjnu, a byla volební. Valné hromadě předcházela přednáška *The Unreasonable Effectiveness of Patterns Generation*, kde jsem se synem zopakoval přednášku z konference TUG 2019, a prezentoval výsledky

o zdokonaleném vývoji a generování nových českých vzorů dělení. Odpovídající článek najdete v tomto čísle Zpravodaje.

Pro volby nového výboru se podařilo získat dostatek kandidujících členů výboru. Členové zvolili nový výbor sdružení a revizory. Po abdikaci Tomáše Hály jako revizora a kooptaci Tomáše Hály jako člena výboru a Pavla Haluzy jako revizora výbor sdružení nyní pracuje ve složení Ján Buša, FEI TU Košice, Jan Paseka, PřF MU Brno, Jiří Demel, FSV ČVUT Praha, Jaromír Kuben, Univerzita obrany Brno, Vít Novotný, FI MU Brno, Michal Hoftich, FF UK Praha, Tomáš Hála, Mendelova univerzita v Brně, Jiří Rybička, MENDELU Brno, Petr Sojka, FI MU Brno, a Zdeněk Wagner, AV ČR Praha. Revizory jsou Pavel Sekanina a Pavel Haluza. Nový výbor sdružení pak ze dvou kandidujících členů zvolil staronového předsedu.

Poděkování

Musím poděkovat předchozímu výboru a všem aktivním členům sdružení za to, že sdružení a jeho tradice byla udržena. Web Zpravodaje byl doplněn o čísla posledních let a články s referencemi byly registrovány u CrossRef, a jsou jim průběžně přidělována DOI pro snadnou a dlouhodobou odkazovatelnost. I nadále byly udržovány diskusní listy a zpřístupňován archiv CTAN zrcadlením do Brna.

Setrvávající aktiva sdružení byla využívána na podporu TEXových projektů doma i ve světě. CSTUG finančně podporoval projekty TEX Gyre, tex4ht, a prezentaci projektu nových vzorů dělení na TUG 2019. Díky placení kolektivního členství CSTUGu v TUGu bylo rozesíláno osm čísel časopisu TUGBoat do center TEXového života v Praze (knihovny MFF UK, FEL ČVUT a MÚ AV), Liberci, Ostravě (OU), Brně (Univerzita obrany), Bratislavě (MFF UK) a Košicích (UJŠ).

Zpravodaj

Od svého založení CSTUG nepřerušeně vydává od roku 1991 primárně pro komunikaci mezi svými členy Zpravodaj CSTUGu. Včasné vydávání se podařilo dodržet, při dodržení poměrně vysoké obsahové a formální kvality. Nová pravidla vydávání avizovaná před třemi lety redakční rada dopracovala, a aktualizovala procesy redakce a pravidla recenzního řízení a otevřenost zapojení odborníků. Jste-li ochotni přiložit ruku k dílu recenzemi, korekturami, či svým know-how, ozvěte se prosím redakci.

Podařilo se domluvit zařazení časopisu do Digitální matematické knihovny DML-CZ na `https://dml.cz`, kde by se měly všechny vydané články objevit během roku 2020. Tím se dále zvýší viditelnost sdružení, dostupnost vydávaných článků a informací o digitální typografii sázecím systémem TEX.

Editorial čísla

TEXovou událostí roku je nepochybně návštěva profesora Knutha v Brně. To se podařilo zařazením provedení české premiéry Knuthova oratoria Fantasia Apocalyptica do oslav 25. výročí založení Fakulty informatiky Masarykovy univerzity, a pozváním prvního čestného doktora FI MU Donalda Ervina Knutha k opětovné návštěvě Brna. O audiovisuálním představení referuje první článek tohoto čísla.

Stalo se již tradicí pořádat setkání uživatelů balíku ConTEXt v ČR. Na setkání vás fotoreportáží z předchozích setkání zve jménem ConTEXt Group Jano Kula, a dalším článkem o sazbě tabulek v ConTEXtu i Tomáš Hála.

Do hlubin vnitřností TEXu se noří Lucie Schaynová a Jan Šustek. Ve svém článku se zabývají vysvětlením způsobů, jakými lze modifikací output rutiny docílit specifických tvarů a efektů na stránce.

Jiří Rybička ve svém článku sdílí své dlouholeté zkušenosti s výukou zpracování textů na brněnské Mendelově univerzitě.

Problematice dělení slov a dechberoucí efektivity technologie vzorů dělení slov se zabývám v článku, který jsem napsal se svým synem. Píšeme o přípravě nového, aktuálního, rozsáhlého a rozděleného seznamu slov pro generování kvalitních vzorů dělení slov. Na příkladu češtiny jsme ověřili možnost technologie vzorů dělení připravit vzory, které nejen nechybují, ale dělí bezchybně celý seznam miliónů slov jazyka, přitom rychlost dělení je obrovská a paměťové nároky jsou minimalistické.

Dalším článkem čísla je zpráva Jana Kuly o verzích makrobalíku ConTEXt. Ukazuje směr, jakým se ubírá společné využití LuaTEXu, METAPOSTu a zmíněného makrobalíku.

Číslo se uzavírá překladem další části seriálu Petera Wilsona o tom, jak vkládat METAPOSTové obrázky do (pdf)LATEXového dokumentu.

Poděkování členům, zejména kolektivním

Největší členské příspěvky pro finanční zabezpečení chodu sdružení tradičně přináší kolektivní členové, za což patří dík i jejich kontaktním osobám. Po vzoru TUGu jmenovitě uvádím s poděkováním jejich seznam:

1. Bílková Hana, Ústav informatiky AV ČR, v. v. i.
2. Bojar Štěpán, Univerzita Karlova v Praze
3. Borková Eva, Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, Ústav matematiky
4. Buša Ján, Technická univerzita v Košiciach
5. Dvorský Jiří, VŠB-TU Ostrava, Fakulta elektrotechniky a informatiky
6. Genčev Marian, VŠB-TU Ostrava
7. Halíř Jiří, Konzervatoř, Brno
8. Jakl Ondřej, Ústav geoniky AV ČR, v. v. i.
9. Jurák Josef, Slezská univerzita v Opavě
10. Kuben Jaromír, Ministerstvo obrany ČR

11. Nemoga Karol, Matematický ústav SAV, Bratislava
12. Písačka Jan, Ústav fyziky plazmatu AV ČR, v. v. i.
13. Plch Roman, Ústav matematiky a statistiky PřF MU
14. Pospíšil Herman, Ústav makromolekulární chemie AV ČR, v. v. i.
15. Rákosník Jiří, Matematický ústav AV ČR, v. v. i.
16. Roupec Petr, Fyzikální ústav AV ČR, v. v. i.
17. Rybička Jiří, Mendelova univerzita v Brně, Ústav informačních technologií
18. Sklenák Vilém, Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky
19. Sojka Petr, Masarykova univerzita Brno, Fakulta informatiky
20. Svoboda Vojtěch, České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská
21. Šustek Jan, Ostravská univerzita
22. Vaníček Petr, Ústav teorie informace a automatizace AV ČR, v. v. i.
23. Veselý Jiří, Univerzita Karlova, Matematicko-fyzikální fakulta
24. Vyhnánek Jiří, České vysoké učení technické v Praze, Fakulta strojní, Centrum počítačových služeb

Kolektivní členové často deponují ve svých knihovnách Zpravodaj sdružení a DVD TEX Live, neboť obojí dostávají v několika kusech, a takto přirozeně sdružují uživatele v centrech vzdělanosti na jednotlivých pracovištích. Pokud víte ve svém okolí o organizaci, která TEX používá, či dokonce na TEXových technologiích staví své podnikání, prosím navrhněte, aby zvážila své členství v CSTUGu.

Prioritou mého předsednictví CSTUGu zůstává zajištění alespoň minimálního chodu sdružení, pravidelné vydávání Zpravodaje a podpora aktivních jednotlivců a projektů naplňujících poslání sdružení.

Aktivity sdružení stojí a padají na ochotných jednotlivcích. Máte-li co sdružení nabídnout, ať už v souvislosti s výše naznačenými činnostmi sdružení, nebo byste chtěli začít novou aktivitu v duchu stanov sdružení, napište prosím na adresu výboru z tiráže!

Budeme vděčni i za pomocnou ruku v získávání nových členů, propagaci sdružení, editaci a redakci tematických webových stránek na webu sdružení, ale i za konstruktivní náměty na další aktivity či připomínky ke stávajícímu chodu sdružení.

Dojmy a fotoreportáž z návštěvy u profesora Knutha v Palo Alto při TUG 2019

Bylo mi potěšením a národním svátkem se zúčastnit konference TUG 2019 v rodišti TEXu: v Palo Alto u campusu Stanfordské univerzity. Skoro dvě desítky účastníků pozval Don Knuth k sobě domů (univerzita svým profesorům půjčuje na campusu na 99 let dům). Do toho svého si Don pořídil vlastní varhany. Na dalších stránkách vidíte z návštěvy u Knutha a z konference krátkou fotoreportáž.

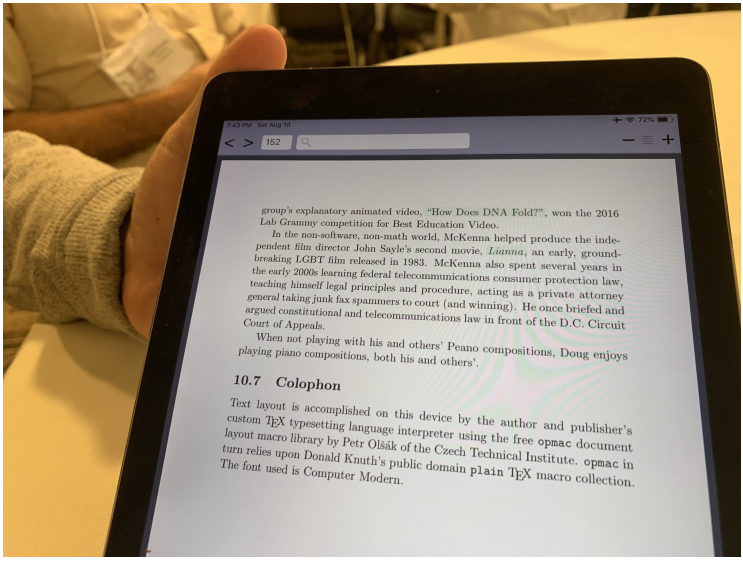
České sdružení uživatelů TEXu jsem zastupoval se svým synem, kdy jsme prezentovali článek, který najdete v tomto čísle. Zástupci CSTUGové komunity byli několikrát zmíněni v poděkováních, kromě prezentace Douga McKenny (viz obrázek 5) to bylo poděkování Zdeňkovi Wagnerovi při prezentaci Shakthi Kannana.

Obrázek 1: Účastníci konference TUG 2019. Najdete na fotce držitele Turingovy ceny, dva prezidenty, dlouholetou editorku TUGboatu, děkana Fakulty informatiky a středoškolského studenta brněnského gymnázia?

Obrázek 2: Don při práci stojí, pracuje na stroji bez Internetu, s relativně starým Linuxovým operačním systémem, a s výhledem na stěnu plnou fotek slavných matematiků a informatiků

Obrázek 3: Boris Veytsman, president TUG si prohlíží jednu z Knuthových pracoven, kam mohou vzpomínkové předměty i Internet

Obrázek 4: Místenka na stole Grand Wizarda na TUG 2019



Obrázek 5: Na konferenci padala jména českých TEXistů v několika poděkováních. Toto je pro Petra Olšáka od Doug McKenna

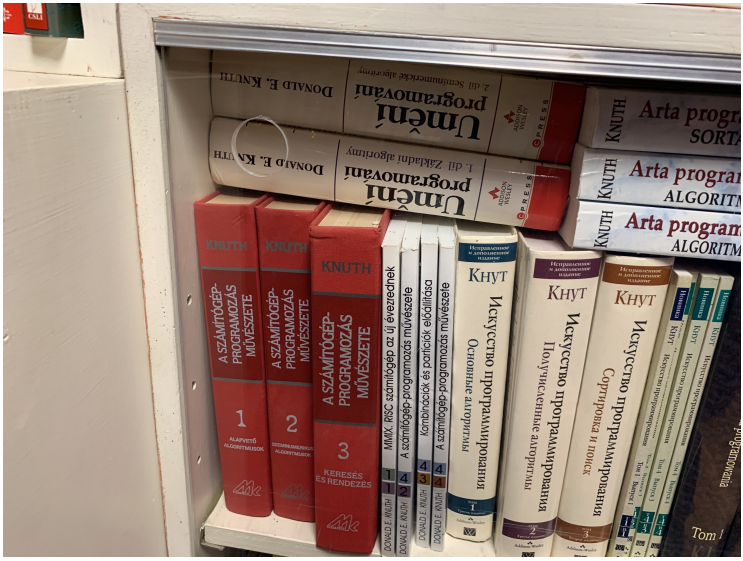
Obrázek 6: Don ve své třetí, varhanní pracovně ukazuje návštěvníkům konference (Petra Sojková, Didier Verna a Frank Mittelbach) svou hudební knihovnu

Obrázek 7: Don šťasten ve své hlavní pracovně s odbornými časopisy a bez Internetu

Obrázek 8: Dračí křivka vlastnoručně navržená, vyrobená a vypálená Donem a Jill Knuthovými. Jen chybička se vloudila – najdete ji?

Obrázek 9: TEXová hostina

Obrázek 10: TUG 2019 participants: DEK, Robin Laakso, Boris Veytsman and Paulo Ney de Souza



Obrázek 11: České vydání Umění programování v Knuthově knihovně

Summary: Introductory Word by Once and Future President

Editorial discusses CSTUG's past three years, possible future direction of the group, membership issues and shaping the organization in the Internet era, together with changes related to the publishing of Zpravodaj CSTUG. The recent visit of the Grand Wizard is also reported with insight.

Go forth and participate in CSTUG to make the bright future of TEX & Friends a reality! *You can!*

Masarykova univerzita, Fakulta informatiky, Botanická 68a, 602 00 Brno
sojka@fi.muni.cz

Fantasia Apocalyptica: Česká premiéra

Dávid Lupták

V piatok 11. októbra 2019 sa v Brne pri príležitosti 25. výročia založenia Fakulty informatiky Masarykovej univerzity uskutočnila česká premiéra multimediálneho oratória Fantasia Apocalyptica autora Donalda Knutha za jeho osobnej účasti. Článok prináša správu z konania tohto koncertu a obsahovú podobu českej verzie brožúry, ktorú sme pre tento koncert pripravili.

Úvod

Donald Ervin Knuth navštívil Brno prvýkrát v roku 1996, kedy mu Masarykova univerzita udelila prvý čestný doktorát na návrh Fakulty informatiky. Po 23 rokoch sa Donald Knuth do Brna vracia, a to pri príležitosti osláv 25. výročia založenia Fakulty informatiky v rámci *Týždňa s držiteľmi Turingovej ceny*. Okrem dvoch „prednášok“ *Otázky a odpovede s Donaldom Knuthom* jeho návšteva vyvrcholila českou premiérou, a len tretím svetovým verejným predstavením, jeho multimediálneho organového diela s názvom Fantasia Apocalyptica. Toto dielo hudobne i vizuálne interpretuje biblickú knihu Zjavenia, poslednú knihu Nového zákona, ktorá je známa tiež ako Apokalypsa.

Obr. 1: Vizuálne spracovanie na troch obrazovkách; zľava: ilustrácie, grécke a anglické preklady textov knihy Zjavenia, notový zápis (autor M. Morávková)

DOI: 10.5300/2019-1-4/11

Správa z akcie

Koncert sa konal v piatok 11. októbra 2019 so začiatkom o 19. hodine v Jezuitskom kostole Nanebovzatia Panny Márie v Brne. Dekan Fakulty informatiky, prof. RNDr. Jiří Zlatuška, CSc., privítal účinkujúcich i zhruba stovku prítomných divákov česko-anglickým príhovorom. Hlavnými účinkujúcimi boli organista Jan Rotrekl, Don Knuth sa ako autor spolupodieľal aj na premietaní textov na prostrednej obrazovke, Vítek Novotný zabezpečoval premietanie ilustrácií na ľavej obrazovke a Tomáš Szaniszlo premietal noty na pravej obrazovke v intenciách hudobnej skladby. V hudobnom doprovode vypomáhal Slavomír Kvasňovský ako hráč na bicie nástroje.

Dĺžka trvania organového koncertu nezávisí len od samotného hudobného diela a jeho kompozície, ale aj od nastavenia organových registrov a spôsobu umeleckého vyjadrenia organistu. Svoj umelecký výkon predniesol mladý Jan Rotrekl vo vyše 95 minútach, pričom i sám autor mal po skončení koncertu slzy v očiach a vyjadril sa, že v tento deň počul svoje dielo v jeho najlepšom prevedení. Zásluhu na tom mal okrem organistu aj organ sv. Edmunda Kampiána, ktorý patrí medzi najkvalitnejšie a najmodernejšie organy v Českej republike.

Koncert organizovala a usporiadala Fakulty informatiky Masarykovej univerzity pri príležitosti 25. výročia jej založenia (viac na `25.fi.muni.cz`).

Príhovor pred koncertom

Dámy a pánové, vážení přátelé,

je mi potěšením, že vás mohu přivítat při téhle speciální příležitosti, kdy nás navštívil stanfordský profesor Donald Knuth při příležitosti 25. výročí založení Fakulty informatiky Masarykovy univerzity. Donald Knuth je velmi významný autor v oblasti informatiky, příkladem je jeho dílo Umění programování. Jeho práce zahrnuje celou řadu oblastí. Petr [Sojka] by vás odkázal na literaturu od Dona, a mimo jiné také, a to se hodí připomenout teď při 30. výročí, je [Don] autorem sázecího sytému TEX, který používají [hlavně] matematici a fyzici, ale zde v Brně byl před 30 lety již používán pro sazbu samizdatu.

Donald Knuth je nejen nositel Turingovy ceny, což je cena, která se přirovnává k Nobelově ceně v informatice, ale co je dál na něm pozoruhodné, je to, že je také nesmírně skromnou a otevřenou osobností, se kterou je skutečně potěšením být ve styku. Tato část osobnosti Donalda Knutha je spojená s jeho rodinnou výchovou, s jeho rodinnou tradicí a tím, nakolik se ve svém životě, vedle té vědecké práce, věnuje i práci ve své luteránské komunitě. Z tohoto [pozadí] pochází také tady toto dílo, které za chvíli uslyšíme a uvidíme, Fantasia Apocalyptica, přepis knihy Zjevení do hudebního jazyka a příslušných ilustrací.

(a) Organ sv. Edmunda Kampiána

(b) Účinkujúci; zľava: Jan Rotrekl, Tomáš Szaniszlo, Donald Knuth a Vít Novotný

Obr. 2: Zábery z koncertu českej premiéry (autor M. Morávková)

Jsem velice rád, že můžeme oslavit s Donem Knuthem výročí naší Fakulty [informatiky Masarykovy univerzity] nejen s Knuthem jako vědcem, ale také jako hudebníkem, a navíc u tohoto mistrovského díla.

Než začneme, tak bych rád poděkoval těm, díky kterým je tady tahle událost skutečností. Donald Knuth jako autor, Jan Rotrekl jako varhaník, Vít Novotný a Tomáš Szaniszlo jako ti, kteří budou spolupracovat při promítaní toho, co uvidíme, Slavomír Kvasňovský [jako hráč] na perkuse, Vojtěch Suchý, rektor tohoto kostela, který nám umožnil uskutečnit zde tento koncert, [a zejména] Jan Martin Bejček z Nadace Campianus, který se zasloužil o to, že zde stojí takovéto, po hudební stránce velkolepé, varhany, které dnes uslyšíme.

Já bych poprosil v obou těchto případech – celá tato produkce byla ze strany Fakulty [informatiky Masarykovy univerzity] koncipovaná jako bezplatná pro účastníky – uvítáme, pokud zde v kasičkách kostele zanecháte, místo toho, aby se vybíralo vstupné, nějaké dary na provoz jak kostela, tak varhan.

[Poděkování si dále zaslouží] Pavel Šiler a AV Media [, kteří jsou] zodpovědní za projekce [po technické stránce], Petr Holubář za audio nahrávku, a [v neposlední řadě] Petr Sojka, který se zasloužil o to, že má poněkud naivní představa, že stačí Dona pozvat s tím, aby se tady odehrála táto skladba, když dáme varhaníkovi noty,

byla skutečně zrealizována. „Skutečně“ [myslím doslova], protože ta má naivní představa, co [všechno] obnáší uspořádaní takovéhle akce, se ukázala jako mnohem naivnější, než by si člověk mohl představit.

Užijte si koncert, děkuji vám za to, že jste přišli.

prof. RNDr. Jiří Zlatuška, CSc.
Děkan Fakulty informatiky Masarykovy univerzity

Brožúra (česká verzia)

Nasledujúci text je rozšírený preklad brožúry zo svetovej premiéry roku 2018 [1].

O varhaníkovi

Jan Rotrekl se narodil v roce 1986. První vzdělání v oboru varhanní hry získal v Praze u Ireny Chřibkové. Ve Francii pokračoval ve studiu u Érica Lebruna na konzervatoři v Saint-Maur-des-Fossés. Od roku 2010 do roku 2014 studoval ve Vídni na Univerzitě hudby a umění ve třídě Martina Haselböcka. Absolvoval několik mistrovských kurzů, např. s Martinem Sanderem, Luigim Tagliavinim a Pierem Damianem Perettim. Kromě České republiky koncertoval na Slovensku, v Německu, Rakousku, Francii, Finsku a Rusku, jako sólista a také s jinými soubory. Jeho repertoár zahrnuje hudební literaturu od renesance do 21. století s důrazem na současnou hudbu. V posledních letech také organizuje koncerty staré hudby v kostele sv. Bartoloměje v Praze, kde působí jako varhaník.

Současně s hudbou studoval Jan Rotrekl chemii. V roce 2012 získal magisterský diplom s vyznamenáním z fyzikální chemie na Vysoké škole chemicko-technologické v Praze a letos obhájil titul Ph.D. v témže oboru na Ústavu chemických procesů AV ČR v Praze.

O nástroji

Jezuitský kostel v Brně byl postaven na přelomu 17. století, ale v roce 1944 byl těžce poškozen, když byly zničeny jeho barokní varhany postavené Thomasem Schwarzem v roce 1743. V poválečných letech byl kostel obnoven a opatřen dočasnými varhanami.

Současné nové varhany z dílny Hermanna Mathise z roku 2014 nesou jméno anglického mučedníka sv. Edmunda Kampiána (1540–1581), který vstoupil k jezuitům v roce 1573 a dokončil noviciát v Brně. Varhany byly tonálně a architektonicky navrženy záměrně v moderním stylu, ale svým designem a proporcemi jsou orientovány na bývalé Schwarzovy varhany. Jejich 48 registrů je rozděleno do tří manuálů a pedálu. Hrací traktura je čistě mechanická, rejstříková traktura je zdvojená – mechanická a elektrická.

Varhany sv. Edmunda Kampiána hrají kromě doprovodu liturgie také důležitou roli v hudebním životě města Brna, a to v sólových koncertech a jako součást velkých orchestrů, jakož i při absolventských koncertech a soutěžích Konzervatoře Brno a Janáčkovy akademie múzických umění. Varhany lze přizpůsobit pro každý koncert přednastavením mnoha kombinací hlasů (například Fantasia Apocalyptica potřebuje kolem tří set takových předvoleb). Varhaník přepíná mezi těmito kombinacemi stisknutím speciálních tlačítek, která jsou umístěna pod klaviaturami nebo nad pedály.

Obr. 3: Ukázka ilustrácií (autor Duane R. Bibby)

O ilustrátorovi

Duane R. Bibby tvoří významné kresby již od té doby, kdy vytvořil nástěnnou malbu kachen na jezeře Tule za pomoci zpětného projektoru jako žák třetí třídy v komunitě farmářů, kde vyrůstal. Některé z jeho prvních významných publikací v časopise *CARtoons* se objevily během jeho tříletého působení v armádě. Ve studiu umění pak pokračoval na Arizonské státní univerzitě a na ArtCenter College of Design v Los Angeles. Později však vysokoškolského studia zanechal, aby se stal nezávislým umělcem, designérem, ilustrátorem a karikaturistou.

Ilustroval řadu knih různých žánrů, včetně učebnic pro základní školy, manuálů pro tovární stroje, či botanického časopisu *The Herb Quarterly*. Snad nejznámější jsou jeho poutavé ilustrace, které vytvořil pro knihy *The TEXbook* (1984) a *The METAFONTbook* (1986) Donalda Knutha, po nichž ho začali žádat další počítačoví vědci o ilustrace pro své vlastní práce.

Duane a jeho manželka Jeanette Ahlgren (také umělkyně) žijí ve Fortuně v Kalifornii, kde si rádi hrají se svými domácími mazlíčky a kutí na automobilech.

Obr. 4: Don Knuth hrá na organe vo svojom dome (autor Petr Sojka, leto 2019)

O skladateli

Donald Ervin Knuth je emeritním profesorem na Stanfordově univerzitě, kde se stal profesorem informatiky v roce 1968. Hudba se však stala jeho celoživotním koníčkem.

Jeho otec Ervin Knuth působil celý život jako pedagog a varhaník, již jako mladý muž hrál na Světové výstavě v Chicagu v roce 1934. Don studoval varhany jen krátce, když byl dvanáctiletým studentem klavíru. Varhaníkem se stal „náhodou“ v roce 1965, když náhle onemocněl hudební ředitel jeho kongregace. Ještě v témže roce vstoupil do Amerického spolku varhaníků a do varhan se zamiloval. Sabatické roky mu pak následně umožnily účastnit se kurzů pro pokročilé: v roce 1968 u Mary Krimmelové na Westminster Choir College a v roce 1986 u Scotta Turkingtona v bostonském presbyterním kostele sjednocené církve Krista.

Don a jeho manželka Jill pověřili firmu Abbott a Sieker, aby pro jejich pronajatý dům v areálu Stanfordovy univerzity postavila varhany. Tento 16rejstříkový nástroj, který byl dokončen v roce 1975, je zmíněn nepřímo v Donově knize *Třídění a vyhledávání*, kde jedna z položek v rejstříku zní „Honorář za knihu, použití, 407“.

Don je skladatel samouk. Od studií na střední škole přečetl na toto téma mnoho knih (zejména klasické texty od Pistona, Stainera, Schönberga a Hindemitha), a přehrál stovky klavírních transkripcí nádherných klasických skladeb. S potěšením zjistil, že jeho hlavní životní dílo – psaní knih o počítačovém programování – má s komponováním hudby opravdu hodně společného.

O skladbě

Biblická kniha Zjevení, známá také jako Apokalypsa (zjevení, odhalení), je mystické dílo plné symbolů. Skládá se hlavně ze snu, který byl zaznamenán v prvním století po Kristu apoštolem Janem. Dramatické události v tomto slavném snu vyvolávají celou paletu lidských emocí, protože zdůrazňují zásadní aspekty života, smrti a duchovna.

Na počátku 60. let byl Donald Knuth fascinován způsoby, jakými autor Zjevení zdůraznil mnoho různých čísel (2; 3; 3,5; 4; 7; 12; 24; ...), a dal jim symbolický význam. Knuth brzy začal přemýšlet o možnosti vytvořit uspokojivé hudební dílo, které by začleňovalo čísla ve Zjevení a další mystické symboly v podstatě v jejich původním pořadí. V roce 2011 poznamenal, že takový projekt „může být šílený, ale ‚múza' mě povzbuzuje k tomu, abych se do toho pustil, už více než 40 let. ... Jsem fascinován tím, že tak mnoho umělců a spisovatelů bylo inspirováno [Zjevením], když je tomu nyní už téměř 2 000 let; tak nemohu odolat myšlence, že možná i já bych mohl být veden těmito starodávnými slovy k vytvoření něčeho, co by mohlo nově oslovit lidi 21. století."

Řecký text Zjevení, který obsahuje téměř přesně 10 000 slov, byl základním zdrojem při vytváření hudební kompozice. Knuth v něm identifikoval více než 100 hlavních motivů a každému z nich přiřadil hudební ekvivalent. Tyto motivy je slyšet opakovaně, jak se na sebe vrší, až se nakonec stanou posluchači povědomé.

Některé z těchto motivů jsou melodické, např. ‚Bůh' má téma se třemi notami G, E, a C, některé motivy jsou rytmické, např. dvakrát tečkovaný rytmus představuje ‚královskou hodnost', některé jsou harmonické, např. ‚člověka' představuje tristanovský akord a ‚služebníka' Stravinského akord Petrušky. Další z motivů jsou hudební idiomy: ‚anděl' je vyjádřen arpeggiem, ‚milost' ozdobnou notou. Některé z nich jsou základní hudební prvky: ‚běda' je bluesová stupnice, ‚starci' jsou stupnice chromatická, ‚lev' oktatonická; ‚sladké' a ‚kyselé' durová a mollová. ‚Prorok' je protipohyb, ‚slunce' palindrom, ‚zlato' těsná harmonie. Jiné motivy pocházejí z trigramů I-ťingu: ‚Země' je dolů-dolů-dolů, ‚nebe' nahoru-nahoru-nahoru, ‚oheň' nahoru-dolů-nahoru, ‚voda' dolů-nahoru-dolů. A některé se inspirují přírodou: ‚Beránka' vyjadřuje bečení, ‚koně' imitace ržání.

Některé motivy mohou účinně vyjádřit pouze píšťalové varhany: ‚hvězda' je vyjádřena použitím cimbálové hvězdice (zimbelstern), ‚jméno' pedálovou prodlevou; ‚otevřít' a ‚zavřít' je reprezentováno otevíráním a zavíráním žaluzií, které obklopují píšťaly a změna jejich polohy umožňuje plynule měnit hlasitost zvuku.

Některé motivy v této skladbě se odvolávají na styly velkých skladatelů: ‚pečeť' připomíná Alaina, ‚pravda' Bacha, ‚moc' Beethovena, ‚hlas' Borodina, ‚trůn' Brubecka, ‚oblak' Debussyho, ‚slovo' Francka, ‚chrám' Gershwina, ‚kniha' Hindemitha, ‚uctívání' Messiaena a ‚rouhání' Schoenberga. Významný je i vliv českého velikána Antonína Dvořáka, jehož Symfonie č. 9 se ozývá v motivu Nového Jeruzaléma (verš 21:1).

Protože kniha Zjevení zahrnuje obrovské množství různých událostí a emocí, žádný samostatný styl nemůže adekvátně reprezentovat celý příběh. Proto je Fantasia Apocalyptica eklektickou směsí mnoha stylů: starověká řecká hudba; středověké zpěvy; tradiční britské zvonové hry; barokní kontrapunkt; lidová hudba Blízkého východu; zpěv podle tvarových not; spirituály; kalypso; romantické symfonie, chorály a kombinatorické vzory; atonální hudba; jazz a Broadway; rock a rap. Můžeme rozpoznat i hudbu vyzváněcích tónů mobilních telefonů. Všechny tyto styly utváří koherentní celek díky všestrannosti píšťalových varhan.

Fantasia Apocalyptica vzdává hold desítkám sborových skladeb založených na knize Zjevení. Odlišná zhudebnění stejného textu různými skladateli různých epoch do sebe kupodivu někdy velmi dobře zapadají.

Poděkování ke koncertu

Za myšlenku uskutečnit skladbu Fantasia Apocalyptica v Brně vděčíme Jiřímu Zlatuškovi. Při vlastním provedení skladby spoluúčinkují Don Knuth, Vít Novotný a Tomáš Szaniszlo, kteří zajistí projekci na tři obrazovky synchronně podle varhanního partu, a Slavomír Kvasňovský jako hráč na bicí nástroje.

Poděkování patří také Vojtěchu Suchému, Janu Martinu Bejčkovi, Josefu Gerbrichovi, Dávidovi Luptákovi a Petru Sojkovi za cennou pomoc s uspořádáním koncertu, jakož i Pavlu Šilerovi a firmě AV Media za technické zajištění projekce.

Odkazy

1. *Fantasia Apocalyptica* [online]. Piteå, Sweden: Luleå University of Technology, 2018 [cit. 2019-10-01]. Dostupné z: `https://www-cs-faculty.stanford.edu/~knuth/fant-program.pdf`. Svetová premiéra v Štúdiu Acusticum pri príležitosti osláv 80. narodenín Donalda Knutha.

Summary: Fantasia Apocalyptica: The Czech Première

On Friday, October 11, 2019, the Czech première of the multimedia oratorio Fantasia Apocalyptica by Donald Knuth took place in Brno on the occasion of the 25th anniversary of the foundation of the Faculty of Informatics of Masaryk University with the author's participation. The article presents the event report and the Czech version of the brochure that we prepared for this concert.

Dávid Lupták, dluptak@mail.muni.cz

14. ConTEXt Meeting 2020, Sibřina, 6.–12. 9. 2020

Jano Kula

Od neděle 6. září do soboty 12. září 2020 se bude v české Sibřině konat 14. konference ConTEXt Meeting pořádaná nizozemským sdružením ConTEXt Group.

Klíčová slova: ConTEXt, LuaTEX, MetaPost, LuaMetaTEX

ConTEXt Meeting probíhá každý podzim od roku 2007. I když je v názvu konference ConTEXt, jeho účastníci jsou rovněž vývojáři LuaTEXu, MetaPostu, letos vydaného LuaMetaTEXu nebo správci vzorů dělení slov užívaných i v LATEXu. Nedozvíte se, jak řešit problémy LATEXu. Zato můžete zjistit, co nového se ve světě TEXu děje, s možností tento vývoj ovlivnit. Diskuze proběhlé na minulých konferencích byly podstatnou inspirací pro započetí vývoje LuaMetaTEXu.

Konference je neformální; za třináct let se z účastníků stali přátelé, přesto není uzavřeným klubem. Je otevřenou skupinou, zvědavou, jak je TEX v praxi využíván ve školství, v nakladatelstvích, v akademické i komerční sféře. Noví účastníci jsou vždy vítaní, stejně jako přednášky o využití ConTEXtu, MetaPostu či LuaTEXu. Řešíte-li problém v LATEXu, dozvíte se, jak by byl řešen – jak jinak – ConTEXtem. Příspěvek do programu přednášek není podmínkou účasti.

Zpravidla liché ročníky probíhají v Nizozemsku, kde ConTEXt vznikl a je převážně vyvíjen. Sudé pak v některé z evropských zemí (Slovinsko, Francie, Německo, Česká republika). Belgické město Bassenge na hranicích s Nizozemskem považujeme pro tento účel za Nizozemsko. Nízký počet „sudých“ zemí má jednoduché vysvětlení. Konference dosud třikrát proběhla v České republice, která je tak druhým nejčastějším místem jejího konání.

Po konferencích v Brejlově v letech 2010 a 2013 se v minulém roce „český“ ConTEXt Meeting přesunul do Sibřiny, obce na hranici hlavního města. Máme zde k dispozici krásné prostory statku U Škodů s vlastním barem, stravování v sousední restauraci a podporu místních přátel. Příští ročník se uskuteční opět zde.

Konference je nezisková a i díky každoroční podpoře německého sdružení DANTE se nám daří udržovat nízký konferenční poplatek. V ceně je ubytování, strava, exkurze i náklady spojené s konferencí. V roce 2020 CSTUG podpoří účast jeho členů částkou 50 €, takže se při včasné registraci můžete konference zúčastnit za předpokládaných 300 €. Pro studenty nebo prvoúčastníky bude sleva pravděpodobně ještě vyšší.

Registrace začne 1. 5. 2020, tradičně v době konání konference BachoTEX polského sdružení GUST. Zlevněná včasná registrace bude možná do 30. 6. 2020. Samotná konference pak proběhne od nedělního večera šestého září do sobotního rána září dvanáctého, léta Páně 2020. Přijďte, uvidíte, uvěříte.

Sledujte `https://meeting.contextgarden.net/2020/`.

DOI: 10.5300/2019-1-4/19

Obrázek 1: Přednáška Hanse Hagena, Sibřina (2018)

Obrázek 2: Účastníci konference, Sibřina (2018)

Obrázek 3: Přednáška Arthura Reutenauera, Bassenge (2019)

Obrázek 4: Účastníci konference, Bassenge (2019)

Obrázek 5: Během konference oceňuje ConTEXt Group vybraného člena za přínos k rozvoji ConTEXtu barevným originálem příležitostné ilustrace Duane Bibbyho, Sibřina (2018)

Summary: 14th ConTEXt Meeting 2020

From Sunday, September 6, 2020 to Saturday, September 12, 2020, the 14th ConTEXt Meeting conference organized by the Dutch ConTEXt Group will be held in Sibřina, Czech Republic.

Keywords: ConTEXt, LuaTEX, MetaPost, LuaMetaTEX

Jménem ConTEXt *Group*[1] *Vás na příští* ConTEXt *Meeting srdečně zve*
Jano Kula

[1] Hans Hagen, Taco Hoekwater, Willi Egger, Mojca Miklavec, Arthur Reutenauer, Luigi Scarso, Aditya Mahajan, Pavneet Arora, Wolfgang Schuster, Ulrik Vieth, Thomas Schmitz, Tomáš Hála, Lukáš Procházka, Harald König, Volker Schaa, Alan Braslau a šestatřicet dalších.

Tabulky v ConTEXtu: přístupy, možnosti, algoritmy

Tomáš Hála

Sazba tabulek trvale patří mezi obtížnější prvky při zpracování publikací. Příspěvek přináší přehled možností sazby tabulek v systému ConTEXt. Zabývá se srovnáním dostupných způsobů, starších i současných, konkrétně se jedná o prostředí `table`, `tabulate`, `TABLE`, `xtables`, srovnává jejich možnosti vzájemně i s „konkurenčním“ LATEXem.

Tabulky mohou být i generovány z dat v jiném formátu, velmi často z formátu CSV. Proto se příspěvek dále zabývá i možnými přístupy tzv. databázového zpracování.

Dále budou předvedeny některé jednoduché algoritmy, kterými lze snadno rozšířit stávající možnosti. Algoritmy budou prezentovány v jazyce Lua, jenž je součástí systému ConTEXt MkIV.

Klíčová slova: ConTEXt, LATEX, srovnání, sazba, tabulky, `table`, `tabulate`, `TABLE`, `xtables`, CSV

Úvod

Srovnáme-li nástroje sazby tabulek napříč implementacemi TEXu, uvidíme značně odlišné přístupy. Základní plain nedisponuje specializovanými prostředky pro sazbu tabulek, což ostatně vyplývá i ze skutečnosti, že „bible“ TEXu (Knuth, 1986) se samotnou sazbou tabulek zabývá až v kapitole 22 nazvané Alignment, a to v souvislosti s obecnou problematikou zarovnání sazebního materiálu.

Způsob zápisu v plainu – i přesto, že je ve své podstatě přímočarý a zcela logický – patrně nikdy nebude prohlášen za uživatelsky přívětivý, neboť vyžaduje poněkud hlubší znalost systému a jeho chování.

Požadavek na tvorbu jednoduššího uživatelského prostředí vyústil v 80. letech minulého století vznikem nadstavby LATEX, která pro tabulkově orientovanou sazbu používá dvě základní prostředí – `tabbing` napodobující systém tabelačních zarážek psacího stroje, jež je vhodné pro pořadovou sazbu, a `tabular` pro běžné tabulky.

Od 90. let minulého století je k dispozici další nadstavba – ConTEXt. Jedná se, stejně jako u TEXu samotného, o volně šiřitelný software určený pro sazbu dokumentů s požadavkem vysoké kvality výstupního produktu. Základní vlastnosti verze MkIV této nadstavby byly představeny již dříve (T. Hála, 2013 a 2015).

 doi: 10.5300/2019-1-4/24

ConTEXt: základní tabulky – `table`

Nejstarším způsobem sazby tabulek je prostředí `table`. Ve srovnání s LaTeXem má prostředí `table` nejblíže ke známému prostředí `tabular`. Namísto znaku `&`, který v LaTeXu slouží k oddělování sloupců, zde příkazem `\NC` zahajujeme sazbu nového sloupce (`NC` je zkratka pro new column, nový sloupec).

Definice sloupců

K určení vlastností jednotlivých sloupců jsou k dispozici kódy podobné prostředí tabular, pro sloupec lze však použít i více než jeden kód. Aby nedošlo k chybné interpretaci, svislá čára – v LaTeXu používaná k aktivaci svislých linek – zde slouží pouze k oddělení definic jednotlivých sloupců. Kromě obdobných kódů jako v LaTeXu (`c`, `l`, `r`, `p`) lze zadat i informaci o vyrovnání sloupců (`s0`, `s1`, `s2`, `s3`), kde číslo je faktorem velikosti mezisloupcových mezer. Zde uvedené příklady se vyrovnáním sloupců v tomto prostředí nezabývají.

Ač historicky nejstarším, je prostředí `table` dosud funkčním způsobem sazby tabulek. Četnost jeho užití se však dnes již snižuje, vyspělejší prostředky (`TABLE`, `xtables`) nabízejí pohodlnější správu tabulek i s přístupem k jednotlivým buňkám. Užití prostředí `table` ilustruje následující kód s výstupem na obrázku 1.

```
\starttable[|r|p(0.6\textwidth)|]\HL
\NC Podle úpravy:
       \NC pořadově sázené, otevřené, uzavřené,
           s členícími linkami\,\dots
             \MR
\NC Podle účelu:
       \NC formuláře, bilance, knižní, časopisecké\,\dots
             \MR\HL
\NC Podle nástroje:\NC ~\FR\HL[2]
\NC \LaTeX: \NC tabbing\MR
\NC \NC tabular + halda balíčků\MR\HL
\NC \ConTeXt: \NC tabulate\MR
\NC \NC table, TABLE, xtables:\MR\HL
\NC Lua:\NC neomezené možnosti:\MR\HL[3]
\stoptable
```

Vyrovnávání řádků

Ukázka z obrázku 1 nemá správně vyrovnán první řádek, neboť všechny řádky jsou vyrovnávány příkazem `\MR`, sloužícím jako pro horní, tak pro dolní vyrovnání běžného řádku. Jiné způsoby vyvoláme příkazy `\SR`, rovněž horní i dolní, ale s využitím u oddělovacích řádků, `\FR` pouze pro horní vyrovnání, `\LR` pouze pro dolní vyrovnání a `\NR` bez vyrovnání (Otten a Hagen, 2006).

Podle úpravy:	pořadově sázené, otevřené, uzavřené, s členícími linkami ...
Podle účelu:	formuláře, bilance, knižní, časopisecké ...
Podle nástroje:	
LaTeX:	tabbing
	tabular + halda balíčků
ConTeXt:	tabulate
	table, TABLE, xtables:
Lua:	neomezené možnosti:

Obrázek 1: Tabulka vytvořená prostředím `table`.

V ukázce provedeme dvě změny ve vyrovnání řádků (modře zvýrazněné příkazy), výsledek představuje obrázek 2.

```
\starttable[|r|p(0.6\textwidth)|]
\NC Podle úpravy:
       \NC pořadově sázené, otevřené, uzavřené,
           s členícími linkami\,\dots\FR
\NC Podle účelu:
       \NC formuláře, bilance, knižní, časopisecké\,\dots
              \SR\HL
\NC Podle nástroje:\NC\MR\HL[2]
\NC \LaTeX \NC tabbing\MR
\NC \NC tabular + halda balíčků\MR\HL
\NC \ConTeXt: \NC tabulate\MR
\NC \NC table, TABLE, xtables\MR\HL
\NC Lua:\NC neomezené možnosti\MR\HL[3]
\stoptable
```

Podle úpravy:	pořadově sázené, otevřené, uzavřené, s členícími linkami ...
Podle účelu:	formuláře, bilance, knižní, časopisecké ...
Podle nástroje	
LaTeX	tabbing
	tabular + halda balíčků
ConTeXt:	tabulate
	table, TABLE, xtables
Lua:	neomezené možnosti

Obrázek 2: Tabulka (`table`) se změnou vyrovnání řádků.

Vyznačování sloupců, svislé linky
Pokud chceme, aby v daném místě byl sloupec ohraničen svislou linkou, použijeme namísto příkazu `\NC` příkaz `\VL` (vertical line, svislá linka). Tento způsob se ukazuje jako poněkud nepraktický, pokud sazeč nemá předem rozmyšleno, kde se budou linky nacházet a kde nikoliv.

Vodorovné linky
Předchozí ukázky obsahují dosud nezmíněný příkaz `\HL` vytvářející vodorovné linky. Jeho nepovinný parametr slouží k určení stupně tloušťky čáry. Přípustné hodnoty jsou `1`, `2`, `3`. I přesto, že tento způsob uživatele určitým způsobem omezuje, pro běžnou práci postačuje.

ConTEXt: pořadová sazba – `tabulate`

Prostředí `tabulate` pro pořadovou sazbu vychází z prostředí `table` – s tabulkou pracujeme úplně stejně (způsob definice sloupců, ohraničení sloupců atd.), máme však k dispozici širší repertoár formátovacích povelů a parametrů. Toto prostředí lze využít pro tabelaci. Podrobnější popis uvádí Otten a Hagen (2006), další inspiraci přináší například Mahajan (2007 a 2008).

ConTEXt: Natural Tables – `TABLE`

Natural tables (přirozené tabulky) vznikly o něco později a autor ConTEXtu Hans Hagen se inspiroval jazykem HTML. Z následujícího příkladu a jeho výstupu na obrázku 3 je patrné, jak se jednotlivé značky používají.

```
\bTABLE
\bTR\bTD Česká republika\eTD\bTD CZ \eTD\bTD 10 500 000\eTD\eTR
\bTR\bTD Slovenská republika\eTD\bTD SK \eTD\bTD 5 410 000\eTD\eTR
\bTR\bTD Polská republika\eTD\bTD PL \eTD\bTD 38 500 000\eTD\eTR
\eTABLE
```

Vidíme zde sice náhradu slov `start` a `stop` za `b` a `e` oproti běžným zvyklostem v ConTEXtu, ale mocný příkaz `\setupTABLE`, kterým lze elegantně nastavit řadu vlastností tabulce jako celku i jejím jednotlivým částem, je k dispozici.

Česká republika	CZ	10 500 000
Slovenská republika	SK	5 410 000
Polská republika	PL	38 500 000

Obrázek 3: Výsledný tvar tabulky s implicitním nastavením.

Podle úpravy:	pořadově sázené, otevřené, uzavřené, s členícími linkami ...
Podle účelu:	formuláře, bilance, knižní, časopisecké ...
Podle nástroje:	
LaTeX:	tabbing
	tabular + halda balíčků
ConTeXt:	tabulate
	table, TABLE, xtables
Lua:	neomezené možnosti

```
\setupTABLE[c][2][align=middle]
```

Obrázek 4: Tabulka s centrovaným druhým sloupcem.

Podle úpravy:	pořadově sázené, otevřené, uzavřené, s členícími linkami ...
Podle účelu:	formuláře, bilance, knižní, časopisecké ...
Podle nástroje:	
LaTeX:	tabbing
	tabular + halda balíčků
ConTeXt:	tabulate
	table, TABLE, xtables
Lua:	neomezené možnosti

```
\setupTABLE[r][odd][offset=-2dd]
```

Obrázek 5: Tabulka se záporným vnitřním okrajem.

`TABLE` – nastavení vlastností

Příkaz `\setupTABLE` má velmi pružnou syntaxi. Můžeme jím nastavovat vlastnosti celé tabulky, jednotlivých řádků, sloupců či buněk. Zápis je poměrně pohodlný a především se jedná o koncepční přístup při definici vlastností sazby.

Ukázky na obrázcích 4–7 postupně předvedou vybrané vlastnosti z velmi rozsáhlého repertoáru. Pro lepší pochopení uveďme, že prvním parametrem udáváme, zda definujeme sloupce (`c`) či řádky (`r`), druhým parametrem číslo sloupce nebo řádku, případně můžeme použít označení `first`, `last` a třetí parametr pak obsahuje nastavení vlastností.[1]. Záporná čísla slouží k počítání sloupců či řádky od konce tabulky.

[1] Zde je potřeba zmínit, že hodnoty `left` a `right` u atributu `align` udávají poněkud netradičně umístění praporku, nikoliv zarovnání okraje.

Podle úpravy:	pořadově sázené, otevřené, uzavřené, s členícími linkami ...
Podle účelu:	formuláře, bilance, knižní, časopisecké ...
Podle nástroje:	
LaTeX:	tabbing
	tabular + halda balíčků
ConTeXt:	tabulate
	table, TABLE, xtables
Lua:	neomezené možnosti

```
\setupTABLE[r][1,last][background=color,
                       backgroundcolor=yellow]
```

Obrázek 6: Tabulka s barevným pozadím prvního a posledního řádku.

Podle úpravy:	pořadově sázené, otevřené, uzavřené, s členícími linkami ...
Podle účelu:	formuláře, bilance, knižní, časopisecké ...
Podle nástroje:	
LaTeX:	tabbing
	tabular + halda balíčků
ConTeXt:	tabulate
	table, TABLE, xtables
Lua:	neomezené možnosti

```
\setupTABLE[1][4,6][align=left,
                    background=color,backgroundcolor=yellow]
```

Obrázek 7: Tabulka s barevným pozadím u vybraných buněk.

Speciálním případem je situace, kdy první parametr obsahuje číslo. Tím říkáme, že určujeme vlastnosti konkrétní buňky či buněk. První parametr pak chápeme jako sloupcovou souřadnici, druhý jako řádkovou.

Zarovnání čísel v prostředí `TABLE`

S požadavkem na zarovnání hodnot na desetinnou čárku (nebo tečku či jiný znak) se sazeči setkávají poměrně často. Nejedná se o zcela triviální problém, avšak prostředí `TABLE` nabízí relativně jednoduché řešení, jak ukazují poslední dva příkazy v kódu na obrázku 8. Zbývající příkazy zajišťují vodorovné i svislé linky. Další příklady popisuje například Egger (2003).

Málo:	Hodně
10.000 :	1,0
−10.00 :	10,000000
1000.0000:	−1000,000

```
\setupTABLE[frame=off]
\setupTABLE[column][first][leftframe=on]
\setupTABLE[column][last][rightframe=on]
\setupTABLE[row][first][topframe=on]
\setupTABLE[row][first,last][bottomframe=on]
\setupTABLE[column][1][alignmentcharacter={.},
                       aligncharacter=yes,align=middle]
\setupTABLE[column][2][alignmentcharacter={,},
                       aligncharacter=yes,align=middle]
```

Obrázek 8: Tabulka se čísly zarovnanými podle požadavku.

ConTEXt: Extreme Tables – `xtables`

Jedná se o nejnovější a doporučovaný nástroj, který rozšiřuje možnosti tabulkové sazby o další nástroje (Hagen, 2015). Koncept strukturních značek odpovídá logice prostředí `table`. Rozsah tohoto příspěvku bohužel neumožňuje představit zde všechny zajímavé vlastnosti.

Následující úsek kódu představuje ukázku jednoduché tabulky vytvořené v prostředí xtables (převzato od Hagena, 2015).

```
\startxtable[offset=1cm]
 \startxrow
  \startxcell one \stopxcell
  \startxcell two \stopxcell
 \stopxrow
 \startxrow
  \startxcell alpha \stopxcell
  \startxcell beta \stopxcell
 \stopxrow
\stopxtable
```

Zpracování CSV

Ve všech dosud uvedených příkladech jsme předpokládali, že data jsou součástí zdrojového textu sázeného v jazyce ConTEXt, případně vložena příkazem `\input` z jiného souboru. Často se však lze setkat se situací, kdy máme data ve formátu CSV a nechce se nám tato data ručně ani automatizovaně značkovat.

Formát CSV

CSV (comma separated values) je formát uložení dat v souboru, ve kterém řádky představují záznamy a v rámci řádků jsou jednotlivé hodnoty jednotně oddě-leny čárkou jakožto dohodnutým oddělovačem. Formát CSV není popsán žádnou

normou, existuje k němu však specifikace (Shafranovich, 2016). Dnes se název formátu zobecňuje na všechna podobná data nezávisle na zvoleném oddělovacím znaku. Někteří autoři (Korpela, 2016; Raymond, 2016) však používají jiná označení, například obecné DSV (delimiter separated values) nebo konkrétnější TSV (tabulator separated values).

Modul `database`

Modul database (autor Hans Hagen, soubor `m-database`) je dostupný například v distribuci TEXlive nebo na stránkách ConTEXtu (přispěvatelé CG, 2017). Distribuovaná verze modulu `m-database.mkiv` obsahuje dataci 2010, verze souboru `m-database.lua` nese označení 1.001. Není známo, zda autor modul dále vyvíjí či nikoliv, lze jej však i přes jen velmi stručnou dokumentaci prakticky používat.

Miklavec (2016) uvádí dosud neřešený rozpor se specifikací CSV (Shafranovich, 2016): Podle specifikace totiž může buňka, je-li správně označena uvozovkami, vést přes více řádků. Víceřádkový obsah jedné buňky však není implementován, a proto je pak výsledná tabulka chybně formátována.

Připojení modulu `database`, definice čtení a zobrazení tabulky

Modul `database` připojíme v preambuli, a to obvyklým způsobem:

```
\usemodule[database]
```

Poté uvedeme, jak se mají data správně načíst a jak má být tabulka sázena na výstupu. Obojí definujeme parametry příkazu `\defineseparatedlist`, jehož prvním parametrem pojmenujeme zapsané vlastnosti a zároveň, jak ukážeme dále, vytváříme speciální příkazy, kterými ohraničujeme oblast formátu CSV.

Běžná definice pro konverzi formátu CSV do prostředí `TABLE` může vypadat takto:

```
\defineseparatedlist[CSV]
  [separator=;,
   before=\bTABLE,after=\eTABLE,
   first=\bTR,last=\eTR,
   left=\bTD,right=\eTD]
```

a	b	c	d	e	
a	"b"	"c"	"d"	"	e"
A	B	C	D	E	
A	"B"	"C"	"D"	"	E"

```
a;b;c;d;e
a;"b";"c";"d";";e"
A;B;C;D;E
A;"B";"C";"D";";E"
```

Obrázek 9: Data ve formátu CSV se čtyřmi řádky a pěti sloupci.

Získali jsme výstup (viz obrázek 9), z něhož je patrné, že uvedená definice nebyla dokonalá. Uvozovky, které ohraničují řetězce, nebyly vzaty v potaz. Závadu napravíme doplněním parametru `quotechar`, čímž dostaneme správný výsledek, viz obrázek 10.

```
\defineseparatedlist[CSV]
  [separator=;,   quotechar=",
   before=\bTABLE,after=\eTABLE,
   first=\bTR,last=\eTR,
   left=\bTD,right=\eTD]
```

a	b	c	d	e
a	b	c	d	;e
A	B	C	D	E
A	B	C	D	;E

Obrázek 10: Data ve formátu CSV se čtyřmi řádky a pěti sloupci – oprava.

V definici nového prostředí `CSV` jsme uvedli příkazy známé z prostředí `TABLE`. Tím jsme nyní získali možnost tabulky pohodlně formátovat a graficky upravovat pomocí příkazu `\setupTABLE`.

Dva způsoby psaní příkazu

Jak již bylo zmíněno, příkazem `\defineseparatedlist[CSV][...]` definujeme chování prostředí pro sazbu tabulky dat ve formátu CSV, které se uplatní mezi `\startseparatedlist[CSV]` a `\stopseparatedlist`. Můžeme však také použít stručnější způsob zápisu:

```
\startCSV
a;b;c;d;e
a;"b";"c";"d";";e"
A;B;C;D;E
A;"B";"C";"D";";E"
\stopCSV
```

Konverze LaTeXové tabulky do ConTeXtu

Před více než 15 lety autor tohoto příspěvku sázel v LaTeXu. Jedním z vysázených děl byla učebnice o radioaktivitě (J. Hála, 1999), jejíž autor nyní připravuje nové vydání. Je potřeba s co nejmenší námahou tyto LaTeXové tabulky převést, resp. použít v ConTeXtu.

Takto vypadal původní LaTeXový zdrojový text:

```
\begTAB{Přehled nuklidů transfermiových prvků.}{trfe}{%
\begin{center}\setlength{\tabcolsep}{1mm}\def\arraystretch{1.2}
\let\hp=\hphantom
\small\sf\begin{tabular}{|c|l|l|c|l|} \hline
%~-
 Z
    & známé              & izotop       & T (s)
        & reakce\\
    & izotopy (A)        & s~nejdelším T&
        &       \\[2.5mm]\hline\hline
101 & 248--259           & \(^{258}\)Md & 55 dní
        & \(^{255}\)Es(\(\alpha\),n)\\\hline
102 & 250--259           & \(^{255}\)No & 185\hp{,00000}
        & \(^{244}\)Pu(\(^{16}\)O,~5n)\\\hline
103 & 252--262           & \(^{256}\)Lr & \hp{0}45\hp{,00000}
        & \(^{243}\)Am(\(^{18}\)O,~5n)\\\hline
104 & 253--262           & \(^{261}\)Rf & \hp{0}65\hp{,00000}
        & \(^{248}\)Cm(\(^{18}\)O,~5n)\\\hline
105 & 255--258, 260--263 & \(^{262}\)Db & \hp{0}34\hp{,00000}
        & \(^{249}\)Bk(\(^{18}\)O,~5n)\\\hline
106 & 258--261, 263      & \(^{263}\)Sg & \hp{00}0,9\hp{0000}
        & \(^{249}\)Cf(\(^{18}\)O,~4n)\\\hline
107 & 261, 262, 264      & \(^{262}\)Bh & \hp{00}0,0061\hp{0}
        & \(^{209}\)Bi(\(^{54}\)Cr,~2n)\\\hline
108 & 264, 265, 267, 269 & \(^{269}\)Hs & \hp{0}19,7\hp{0000}
        & produkt \(\alpha\) přeměny \(^{273}\)110\\\hline
109 & 266, 268           & \(^{266}\)Mt & \hp{00}0,0034\hp{0}
        & \(^{209}\)Bi(\(^{59}\)Fe,~n)\\\hline
110 & 269, 271--273      & \(^{269}\)110& \hp{00}0,0027\hp{0}
        & \(^{208}\)Pb(\(^{62}\)Ni,~n)\\\hline
111 & 272                & \(^{272}\)111& \hp{00}0,0015\hp{0}
        & \(^{209}\)Bi(\(^{64}\)Ni,~n)\\\hline
112 & 272                & \(^{272}\)112& \hp{00}0,00028
        & \(^{208}\)Pb(\(^{70}\)Zn,~n)\\\hline
%~+
\end{tabular}\end{center}}{}
\endTAB
```

Na tabulku zapsanou v LaTeXu může určitým způsobem pohlížet také jako na data typu CSV. Pro oddělení sloupců slouží znak `&` a pokud fyzicky tabulku

uspořádáme po řádcích[2], nic nebrání použití modulu `database` pro zpracování dat tabulky.

Nyní si pomocí příkazu `\defineseparatedlist` připravme prostředí pro sazbu LaTeXových tabulek:

```
\defineseparatedlist[LTX][separator=&,
   before=\bTABLE,after=\eTABLE,
   first=\bTR,last=\eTR,left=\bTD,right=\eTD,
   setups=unix]
```

V původním vydání bylo matematické prostředí sázeno dvojicí příkazů `\(` a `\)`, tyto příkazy je totiž možno v LaTeXu předefinovat podle vlastních představ. V ConTeXtu však tato dvojice příkazů není implementována. Abychom nemuseli tyto příkazy ručně nebo jinak nahrazovat za tradiční `$`, vytvoříme si jejich vlastní definice, v nichž vysázíme „dolary“ za pomoci jazyka Lua:

```
\def\({\startluacode context("$") \stopluacode}
\def\){\startluacode context("$") \stopluacode}
```

Zároveň s tím se elegantně zbavíme nežádoucích LaTeXových příkazů, kterými jsou data v tabulkách hojně opatřena:

```
\def\hline{}
\def\\{}
%\begTAB{Přehled nuklidů transfermiových prvků.}{trfe}{%
%\begin{center}\setlength{\tabcolsep}{1mm}\def\arraystretch{1.2}
%\small\sf\begin{tabular}{|c|l|l|c|l|} \hline
%~-
\setupTABLE[column][each][align={low,middle}]
\startLTX
 Z    & známé       & izotop  & T (s)  & reakce\\
      & izotopy (A) & s~nejdelším T&&\\[2.5mm] \hline\hline
101 & 248--259            & \(^{258}\)Md & 55 dní
       &              \(^{255}\)Es(\(\alpha\),n)\\ \hline
102 & 250--259            & \(^{255}\)No & .185\hphantom{,00000}
       &              \(^{244}\)Pu(\(^{16}\)O,~5n)\\ \hline
...
\stopLTX
%~+
%\end{tabular}\end{center}}{}
%\endTAB
```

Výslednou tabulku vysázenou ConTeXtem ukazuje obrázek 11.

Nevyhnuli jsme se ruční editaci, neboť bylo nutné ještě deaktivovat použitá LaTeXová prostředí a naopak aktivovat prostředí ConTeXtové. I přesto, že je možné i toto řešit programově, jeví se ruční editace zatím i při desítkách tabulek jako rychlejší.

[2] V uvedené ukázce kódu původní tabulky máme řádky tabulky rozděleny na více řádků.

Z	známé	izotop	T (s)	reakce
	izotopy (A)	s nejdelším T		[2.5mm]
101	248–259	^{258}Md	55 dní	^{255}Es(α,n)
102	250–259	^{255}No	185	^{244}Pu(^{16}O, 5n)
103	252–262	^{256}Lr	45	^{243}Am(^{18}O, 5n)
104	253–262	^{261}Rf	65	^{248}Cm(^{18}O, 5n)
105	255–258, 260–263	^{262}Db	34	^{249}Bk(^{18}O, 5n)
106	258–261, 263	^{263}Sg	0,9	^{249}Cf(^{18}O, 4n)
107	261, 262, 264	^{262}Bh	0,0061	^{209}Bi(^{54}Cr, 2n)
108	264, 265, 267, 269	^{269}Hs	19,7	produkt α přeměny ^{273}110
109	266, 268	^{266}Mt	0,0034	^{209}Bi(^{59}Fe, n)
110	269, 271–273	^{269}110	0,0027	^{208}Pb(^{62}Ni, n)
111	272	^{272}111	0,0015	^{209}Bi(^{64}Ni, n)
112	272	^{272}112	0,00028	^{208}Pb(^{70}Zn, n)

Obrázek 11: Přehled nuklidů transfermiových prvků po vysázení ConTeXtem bez dalších typografických úprav.

Zpracování dat z externího souboru
Dosavadní ukázky vycházely z předpokladu, že data jsou součástí zdrojového textu. Získat data z externího souboru však lze také:

```
\processseparatedfile[TSV][filename]
```

ScanCSV a HandleCSV

Problematikou zpracování souborů ve formátu CSV, zejména pro vlastní školské potřeby, se zabývá Jaroslav Hajtmar. Na konferencích TeXperience v r. 2012 a ConTeXt Meeting v r. 2013 prezentoval svou knihovnu ScanCSV (Hajtmar, 2012, soubor `t-scancsv.lua`). Dnes však tuto knihovnu sám autor již považuje za zastaralou a nahradil ji novější verzí nazvanou HandleCSV, která se skládá ze dvou souborů `t-handlecsv.lua` a `t-handlecsv-tools.lua`. Na jejím současném vývoji se podílí dále Pablo Rodriguez. Zatím se jedná však spíše o soukromý, méně známý projekt s domovskou stránkou (Rodriguez, 2016), z níž vede odkaz na GitHub, kde zájemce nalezne zdrojový kód a stručnou dokumentaci.

Na jednoduchém příkladu se můžeme podívat, jak lze s knihovnou HandleCSV pracovat. Vstupní data a požadovaná výstupní sestava jsou na obrázku 12. Data zpracujeme následujícím zdrojovým kódem.

```
\usemodule[handlecsv]
\setheader
\opencsvfile{a.csv}
```

```
"Name";"Surname";"Birthdate"
"John";"Smith";10/03/02
"Jane";"Newman";03/03/92
```

```
John Smith was born on 10/03/02.
Jane Newman was born on 03/03/92.
```

Obrázek 12: Vstupní data a požadovaná výstupní sestava.

První položka	
Druhá položka	8.45 EUR
Třetí položka	5.90 EUR
VAT 21%	3.75 EUR
Celkem	21.60 EUR

Obrázek 13: Výsledná tabulka obsahující část faktury.

```
\starttext
  \startbuffer[loop]
    \cA\ \cB\ was born on \cC.\crlf
  \stopbuffer
   \doloop{\getbuffer[loop]%
          \nextrow\ifEOF\exitloop\fi}
\stoptext
```

Tabulkový procesor – modul `spreadsheet`

Modul `spreadsheet` (autor Hans Hagen, soubor `m-spreadsheet`) umožňuje podle očekávání provádět jednoduché výpočty. Je dostupný v distribuci TEXlive i na webových stránkách ConTEXt Garden. Modul je udržován, ale jeho další vývoj neprobíhá. Dokumentace (Hagen, 2016) je aktualizována, ale po jistou dobu nebyla pro drobnou technickou závadu zařazována do distribuce.

Příkazy `\startcell` a `\stopcell` ohraničují prostředí buňky, do níž se nepíše text k sazbě v jazyce TEX, ale v jazyce Lua, v němž jsou příslušné výpočty vyhodnocovány. Jedná se o velmi zajímavé a elegantní řešení, avšak méně pohodlné, pokud sázená tabulka obsahuje větší množství „TEXového“ textu a méně výpočtů, neboť všechny texty je potřeba opatřit uvozovkami. Autor modulu sám doporučuje jeho použití jen pro menší výpočty, například faktury (Hagen, 2016). Jeho příklad v drobné úpravě uvádíme i zde, výsledek přináší obrázek 13.

```
\startspreadsheettable[test][frame=off]
 \startrow
  \startcell[align=flushleft,width=8cm] "První položka" \stopcell
  \startcell[align=flushright,width=3cm] @ "0.2f EUR" 3.50 \stopcell
 \stoprow
 \startrow
  \startcell[align=flushleft] "Druhá položka" \stopcell
  \startcell[align=flushright] @ "0.2f EUR" 8.45 \stopcell
 \stoprow
 \startrow
  \startcell[align=flushleft] "Třetí položka" \stopcell
  \startcell[align=flushright] @ "0.2f EUR" 5.90 \stopcell
 \stoprow
 \startrow[topframe=on]
  \startcell[align=flushleft] "VAT 21\percent" \stopcell
  \startcell[align=flushright] @ "0.2f EUR" 0.21*sum(B) \stopcell
 \stoprow
 \startrow[topframe=on]
  \startcellalign=flushleft] "\bf Celkem" \stopcell
  \startcell[align=flushright] @ "0.2f EUR" sum(B) \stopcell
 \stoprow
\stopspreadsheettable
```

Vlastní řešení

Nyní zde předvedeme speciální řešení zhotovené pro jednu zakázku obsahující ekonomická data, která mají být prezentována jednak tabulkou, jednak grafem.

Zákazník však zpočátku nebyl schopen přesně formulovat požadavky, nebylo jasné, jakou formou se budou data prezentovat, ani jak dlouhá časová řada má být prezentována. K tomu přistoupily i požadavky na snadnou správu, konkrétně obecnou datovou strukturu, s pružným přístupem a indexovatelnou, a také na snadnou editaci zdrojového textu, která musela být jednoduchá.

Pro vstupní data byl navržen tento formát:

```
\tabulka[][title="Pracoviště A",
data=
RS 2517000 2515000 2386000 1954000
PU 4.57 4.62 4.65 4.05
PKL 361 491 392 268
PK 6699 5508 9475 6012
]
```

Původní úvaha spočívala v načítání dat do dvourozměrného pole s obvyklým oddělovačem řádků.[3] Pokud jsou však data předána ke zpracování ve formě atributu v parametrech, jsou součástí zdrojového kódu, předzpracována *input-procesorem*, a tudíž se znak Line Feed změní na mezeru. Pro zjednodušení implementace bylo načítání doplněno o pevně danou konstantu udávající počet sloupců. Toto řešení bylo možno použít proto, že všechny zákazníkovy tabulky měly stejný počet sloupců.

Tabulku vysázíme pomocí ConTeXtového makra propojeného s funkcí napsanou v jazyce Lua:

```
\def\tabulka[#1][#2]{
  \ctxlua{userdata.tabulka('#1','#2')}}
```

Pro vykreslení sloupcového grafu potřebujeme následující definice:

```
\def\grafwidth{0.5}
\def\graf#1#2{%
  \def\koef{\csname#1koef\endcsname}
  \startcolor[#1color]
  \vrule width\grafwidth cc height #2dd\
  \stopcolor
}
```

Následuje zdrojový kód funkce `tabulka` napsaný v jazyce Lua. Protože se jedná o starší projekt, naleznete zde generování tabulky starším způsobem, tj. pomocí prostředí `table`.

```
function userdata.tabulka(keywords, keyvals)
   keyword_options = utilities.parsers.settings_to_array(keywords)
   named_values = utilities.parsers.settings_to_hash(keyvals)
   context.pracoviste(string.unquoted(named_values["title"]))
   local roky = { 2010, 2011, 2012, 2013 }
   roku,kolikroku,delim = #roky,3,','
   t = string.split(named_values["data"]," ")
   context("\\bgroup\\setupbodyfont[8dd]
      \\starttable[s2|p(15cc)|s3w(4cc)c|w(4cc)c|w(4cc)c|]")
   context("\\NC")
   for r=roku-2,roku do
     context("\\NC")
     for i=r,#t-1,roku do -- 1., ., 11. ... údaj
       if t[i]~="x" then
          context("\\graf{"..t[i-r+1].."}{\\compute{"..t[i].."/
           \\csname "..t[i-r+1].."koef\\endcsname}}%")
       end
```

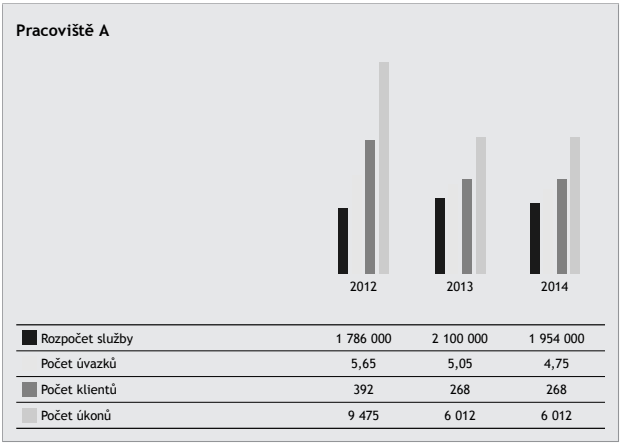
[3] Řešení bylo provozováno v Linuxu, jednalo se o znak LF (Line Feed).

```
    end
  end
  context("\\AR\\NC")
  for i=roku-kolikroku+1,roku do context("\\NC "..roky[i]) end
  context(" \\FR\\HL[2]")
  for i=1,#t-1 do
    s = i % (roku + 1)
    t[i]=string.strip(t[i])
    if s == 1 then
      if utf.len(t[i]) < 6 then
           context("\\NC\\ctverec{"..t[i].."color}\\
             \\csname "..t[i].."\\endcsname")
        else context("\\NC",t[i])
      end
    end
    if s > kolikroku or s == 0 then
      if delim==',' then
       cislo,pocet=string.gsub(t[i],"[.]",",")
       cislo,pocet=string.gsub(cislo,"(%d)(%d%d%d)(%d%d%d)$",
         "%1\\,\\,%2\\,\\,%3")
       cislo,pocet=string.gsub(cislo,"(%d)(%d%d%d)$",
         "%1\\,\\,%2")
      else
       cislo=t[i]
       cislo,pocet=string.gsub(cislo,"(%d)(%d%d%d)(%d%d%d)$",
         "%1,%2,%3")
       cislo,pocet=string.gsub(cislo,"(%d)(%d%d%d)$","%1,%2")
      end
      context("\\NC "..cislo)
    end
    if s == 0 then context("\\FR\\HL") end
  end
  context("[2]\\stoptable\\egroup\\par");
end
```

Výslednou tabulku i se sloupcovým grafem ukazuje obrázek 14.

Projekt CTX-CSV

Další specifické potřeby v oblasti zpracování dat ve formátu CSV vedly autora tohoto příspěvku k zahájení vývoje vlastních nástrojů, zejména s výstupem ve



Rozpočet služby	1 786 000	2 100 000	1 954 000
Počet úvazků	5,65	5,05	4,75
Počet klientů	392	268	268
Počet úkonů	9 475	6 012	6 012

Obrázek 14: Výsledná tabulka i se sloupcovým grafem.

formě grafů, pro což není v ConTeXtu zatím kompaktní a uživatelsky přívětivá podpora. Vzhledem k probíhajícímu vývoji není projekt zatím zveřejněn.

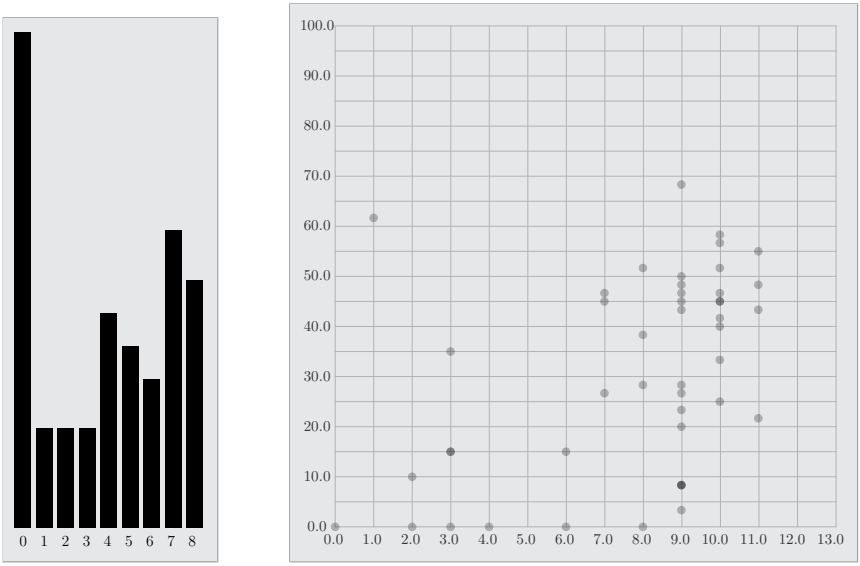
Projekt se skládá ze tří částí – souboru `csv-tools.lua`, obsahujícího vstupně výstupní operace pro zpracování souboru ve formátu CSV, souboru `csv-stat.lua`, zabývajícího se grafickou reprezentací načtených dat, a souboru `t-csv.mkiv`, který slouží jako modul. Grafy jsou vykreslovány MetaPostem, výpočty zajišťují úseky kódu v jazyce Lua.

Základní vstupně-výstupní operace se souborem ve formátu CSV provádějí následující tři makra:

```
\CSVRead[comma2dot][data=\soubor,separator=|]
\CSVWrite[dot2comma][output=pomout.csv,enclosechar="]
\CSVTABLE[dot2comma][]
```

Následuje ukázka části funkce v jazyce Lua, která reprezentuje tabulkové výpočty v načtených datech. Funkce je užita v příkazu `\CSVNewColumn`:

```
function vypocet(t,c)
  keyword_options = utilities.parsers.settings_to_array(keywords)
  named_values = utilities.parsers.settings_to_hash(keyvals)
  ...
  if t[r][c+3]>0 then t[r][c+10]=t[r][c+3] end
  if t[r][c+3]>1 then t[r][c+11]=t[r][c+3] end
  if t[r][c+3]>2 then t[r][c+12]=t[r][c+3] end
  if t[r][c+3]>3 then t[r][c+13]=t[r][c+3] end
  ...
 return t end
```



Obrázek 15: Ukázka vykreslených grafů.

Příkazy sloužící k výpočtům a k vykreslení grafů mohou být například tyto:

```
\CSVNewColumn[][table=1,column=23,expr=vypocet]
\CSVComputeColumn[][table=1,column=23,startrow=2,meze={0,13}]
\CSVDrawDistribution[][table=1,column=23,maxparts=12]
\CSVComputeCorrelation[][table=1,columns={23,18},startrow=2]
    %,meze={0,60}]
\CSVDrawCluster[][table=1,xdraw=19.2,columns={23,18},startrow=2]
    %,groupby=2]
```

Ukázku vykreslených grafů obsahuje obrázek 15.

Závěr

ConTeXt nabízí zajímavé možnosti sazby tabulek. Představené ukázky vycházejí jednak z řešení čistě „TeXových“, tj. ze sady strukturních značek, jednak z možností, jež přináší jazyk Lua.

I přesto, že máme k dispozici vícero možností, základní vlastnosti jsou integrovány do jednoho celku, což zbavuje uživatele nutnosti připojovat různé balíčky, jako je tomu v LaTeXu, z nichž některé nejsou ani vzájemně kompatibilní, jak dokumentuje Talandová (2008).

V oblasti zpracování formátu CSV však různé přístupy různých autorů ukazují zatím spíše na tříštění sil než na hledání cesty pro vývoj univerzálního prostředku.

Odkazy

EGGER, Willi, 2003. *My Way: Use of the natural table environment* [online] [cit. 2019-06-21]. Dostupné z: `http://dl.contextgarden.net/myway/NaturalTables.pdf`.

HAGEN, Hans, 2015. *Extreme Tables:* ConTeXt *MkIV* [online] [cit. 2016-06-21]. Dostupné z: `http://www.pragma-ade.com/general/manuals/xtables-mkiv.pdf`.

HAGEN, Hans. *Simple Spreadsheets:* ConTeXt *MkIV* [online] [cit. 2016-07-20]. Dostupné z: `http://www.pragma-ade.nl/general/manuals/spreadsheets-mkiv.pdf`.

HAJTMAR, Jaroslav, 2012. ScanCSV – Lua knihovna pro zpracování CSV souborů ConTeXtem a LuaLaTeXem. *Zpravodaj CSTUG*. Roč. 22, č. 2, s. 76–90. ISSN 1211-6661. Dostupné z DOI: `10.5300/2012-2/76`.

HÁLA, Jiří, 1999. *Radioaktivita, ionizující záření, jaderná energie*. Konvoj. ISBN 80-85615-56-8.

HÁLA, Tomáš, 2013. LaTeX, nebo ConTeXt?: První zkušenosti se sazbou ConTeXtem. *Zpravodaj CSTUG*. Roč. 23, č. 1, s. 57–64. ISSN 1211-6661. Dostupné z DOI: `10.5300/2013-1/57`.

HÁLA, Tomáš, 2015. Proč jsem zkusil ConTeXt. In: *Zborník príspevkov medzinárodnej konferencie OSSConf 2015*. Žilinská univerzita v Žilině, s. 37–40. ISBN 978-80-970457-7-7.

KNUTH, Donald Ervin, 1986. *The TEXbook*. Addison-Wesley. ISBN 0-201-13448-9.

KORPELA, Jukka. *Tab Separated Values (TSV): A format for tabular data exchange* [online] [cit. 2016-05-06]. Dostupné z: `https://www.cs.tut.fi/~jkorpela/TSV.html`.

MAHAJAN, Aditya, 2007. ConTeXt basics for users: Table macros. *TUGboat*. Roč. 28, č. 3, s. 372–374.

MAHAJAN, Aditya, 2008. ConTeXt basics for users: Table macros II. *TUGboat*. Roč. 29, č. 1, s. 219–222.

MIKLAVEC, Mojca. *My Way: Creating tables using CSV (comma-separated values)* [online] [cit. 2016-05-12]. Dostupné z: `http://dl.contextgarden.net/myway/csv.pdf`.

OTTEN, Ton a HAGEN, Hans, 2006. Exkurze do ConTeXtu. *Zpravodaj CSTUG*. Roč. 16, č. 2–4, s. 57–224. ISSN 1211-6661. Dostupné z DOI: `10.5300/2006-2-4/1`.

PŘISPĚVATELÉ ConTEXt GARDEN, 2017. ConTEXt *Garden: Modules* [online] [cit. 2017-06-27]. Dostupné z: `http://wiki.contextgarden.net/Modules`.

RAYMOND, Eric Steven. *The Art of Unix Programming: Chapter 5, Textuality: Data File Metaformats* [online] [cit. 2016-05-06]. Dostupné z: `http://www.catb.org/~esr/writings/taoup/html/ch05s02.html`.

RODRIGUEZ, Pablo. *HandleCSV* [online] [cit. 2016-05-06]. Dostupné z: `http://www.handlecsv.tk/`.

SHAFRANOVICH, Y. *Common Format and MIME Type for Comma-Separated Values (CSV) Files* [online] [cit. 2016-05-06]. Dostupné z: `https://tools.ietf.org/html/rfc4180`.

TALANDOVÁ, Petra, 2008. Možnosti tabulkové sazby. *Zpravodaj ČSTUG*. Roč. 18, č. 3, s. 151–160. ISSN 1211-6661. Dostupné z DOI: `10.5300/2008-3/151`.

Summary: Tables in ConTEXt: Ways, Possibilities, Algorithms

In the publication process, the typesetting of tables belongs to the more complicated tasks. This paper reviews old and current ways of typesetting tables in ConTEXt (environments `table`, `tabulate`, `TABLE`, `xtables`), and compares them mutually and with the 'rival' LaTeX.

Tables can be generated from other formats such as the frequently used CSV. Therefore, the paper deals also with database processing.

Finally, some simple algorithms for easy extensions of the available repertoire are presented.

Keywords: ConTEXt, LaTeX, comparison, typesetting, tables, `table`, `tabulate`, `TABLE`, `xtables`, CSV

Mendelova univerzita, Provozně ekonomická fakulta, ústav informatiky, Zemědělská 1, 613 00 Brno, thala@mendelu.cz

Aplikace parametrů řádkového zlomu a output rutiny k formátování sazby v TEXu

Lucie Schaynová, Jan Šustek

V článku projdeme vnitřnosti programu TEX a ukážeme si, jakou cestou se jednotlivé znaky vstupního souboru .tex postupně dostanou až do výstupního souboru .pdf. Zdržíme se u algoritmu řádkového zlomu, který bez debat výrazně předběhl svou dobu. Vhodnou kombinací jeho parametrů lze nastavit nejen zarovnání textu do bloku, na střed nebo na praporek, ale ukážeme si i mnoho dalších možných způsobů zarovnání textu. Na konci této cesty se nachází output rutina, která má za úkol umístit vysázený text na stránku. Ukážeme si, jak lze nastavit různá záhlaví a zápatí a jak lze jednoduše vytvořit hlavičkový papír. Také si ukážeme různé praktické aplikace output rutiny, například k vysázení slajdů pro přípravu prezentací. Přijde řeč i na problematiku zjišťování pozice konkrétního bodu sazby na stránce a využití této informace při kreslení obrázků v METAPOSTu.

Článek vychází z přednášky druhého autora na konferenci OSSconf 2018.

Klíčová slova: TEX, řádkový zlom, output rutina

1 Úvod

TEX je program, který vezme text ze vstupního souboru a vysází jej podle daných instrukcí. V tomto článku si stručně popíšeme, jakou cestou projdou jednotlivé znaky vstupního souboru .tex, než se vytvoří výstupní soubor .pdf. Tuto cestu si lze představit jako montážní linku s desítkou na sebe navazujících nástrojů, kde vstupem prvního nástroje je vstupní soubor, vstupem dalších nástrojů je výstup předchozího nástroje a výstupem posledního nástroje je výsledný soubor .pdf. Z těchto nástrojů se dále v článku budeme podrobněji zabývat řádkovým zlomem a output rutinou.

Každý nástroj začne pracovat okamžitě, když od předchozího nástroje dostane dostatek materiálu, aby mohl začít pracovat (princip nutného minima). Zájemcům o hlubší pochopení problematiky doporučuji [1] nebo [2], zájemcům o maximální pochopení doporučuji během dlouhých zimních večerů [3].

V textu budeme pracovat v PlainTEXu s načteným OPmacem, a to z důvodu, aby lépe vynikly jednotlivé ukázky. Jelikož je ale většina uvedených příkazů obsažena přímo v jádru (pdf)TEXu, budou ukázky po drobných modifikacích fungovat i v jiných nadstavbách (pdf)TEXu.

 doi: 10.5300/2019-1-4/44

1.1 Input procesor

Funkcí input procesoru je načíst jednotlivé řádky textu z konkrétního vstupního souboru, případně z jiného vstupního proudu.[1] Je třeba si uvědomit, že možnosti názvů souborů mohou být v různých operačních systémech různé. Dále mohou být různé i způsoby ukončování řádků v textových souborech. Chování input procesoru tak nutně je systémově závislé. Na druhou stranu je input procesor jediná systémově závislá část TEXu a výstup input procesoru již je systémově nezávislý.

1.2 Token procesor

Token procesor přečte znaky z výstupu input procesoru a vytvoří z nich tzv. tokeny a ty vloží do tzv. čtecí fronty. Token typu řídicí sekvence odpovídá názvu makra, názvu příkazu, názvu registru apod. Pokud se ze znaků nevytvoří token typu řídicí sekvence, vytvoří se token typu dvojice (*znak*, *kategorie*), kde *kategorie* určuje, jak se daný *znak* bude chovat. Zatímco u tokenů typu řídicí sekvence není dopředu známo, jak se budou chovat, a uživatel si je může předefinovat, u tokenů typu dvojice je jeho chování pevně dáno kategorií, která je dále neměnná.

Token procesor tedy identifikuje vše, co se v textu nachází, a tyto informace pošle dalšímu nástroji, expand procesoru.

1.3 Expand procesor

Expand procesor načte jeden token ze čtecí fronty. Pokud se jedná o expandovatelný token (například makro), načte i jeho případné argumenty a token expanduje podle příslušné definice. Vzniklé tokeny vrátí zpět do čtecí fronty. Tento proces se opakuje tak dlouho, dokud expand procesor nenarazí na neexpandovatelný token (například na primitivní příkaz, písmeno nebo konstruktor exponentu). Tento token pak předá hlavnímu procesoru. Na výstupu expand procesoru tak mohou být pouze primitivní příkazy nebo znaky s daným chováním.

1.4 Hlavní procesor

Hlavní procesor příslušné příkazy vykoná. V běžném dokumentu je nejčastějším příkazem vysázení konkrétního znaku. Dalšími častými příkazy jsou definice nových maker, změna fontu nebo práce s registry. V neposlední řadě je třeba zmínit příkaz na ukončení běhu TEXu, který bývá často zmiňován až na posledním řádku.

Znaky se ukládají do tzv. horizontálního seznamu, což si lze představit jako jeden dlouhý řádek textu, který později bude rozlámán na řádky. Do horizontálního seznamu se dále ukládají různé výplňky (horizontální mezery), penalty a další objekty.

[1] Vstup lze načítat z terminálu. Nicméně například v Linuxu lze načítat vstup i z pojmenované roury, protože ta se navenek tváří jako soubor.

1.5 Řádkový zlom

Cílem řádkového zlomu je načíst z horizontálního seznamu celý odstavec textu a ten rozlámat na jednotlivé řádky. Algoritmus najednou najde místa zlomu celého odstavce, aby byl optimální s ohledem na hodnoty mnoha parametrů.

Výsledné řádky textu se ukládají do tzv. vertikálního seznamu, což si lze představit jako jeden vysoký sloupec sazby, který bude později rozlámán na stránky. Do vertikálního seznamu se dále ukládají různé výplňky (vertikální mezery), penalty a další objekty.

1.6 Stránkový zlom

Jakmile je vertikální seznam již dostatečně plný, zavolá se algoritmus stránkového zlomu, který určí, jaká část textu se umístí na aktuální stránce. Protože nalezení optimálního stránkového zlomu vyžaduje obrovskou výpočetní složitost, není TEXový algoritmus natolik propracovaný a je jen pár parametrů, kterými lze stránkový zlom ovlivnit.

1.7 Output rutina

Output rutina se nachází na konci cesty k výstupnímu souboru .pdf. Jejím úkolem je umístění příslušného textu na stránku. Na stránku se umístí ale také například záhlaví, zápatí, poznámky pod čarou nebo plovoucí objekty.

Důležitým úkolem output rutiny je export takto vytvořené stránky do souboru .pdf. Protože až v tuto dobu je přesně známo, na které stránce bude konkrétní text, uloží se nyní také informace o křížových odkazech.

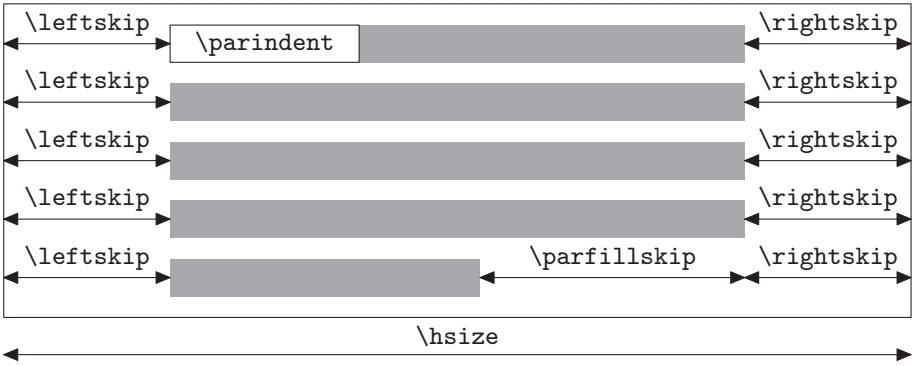
2 Řádkový zlom

Knuthův-Plassův algoritmus řádkového zlomu výrazně předběhl svou dobu. Jeho detaily, matematické odvození a množství příkladů lze najít v šedesátistránkovém článku [4].[2] Algoritmus je závislý na mnoha parametrech:

```
\adjdemerits \doublehyphendemerits \emergencystretch
  \exhyphenpenalty \finalhyphendemerits \fontdimen
\hangafter \hangindent \hsize \hyphenpenalty \leftskip
   \linepenalty \looseness \parfillskip \parindent
    \parshape \pretolerance \rightskip \tolerance
```

V dalším textu si ukážeme vliv některých parametrů na výsledný zlom. Přesný význam každého z nich se lze dočíst v [1] nebo [2].

[2] Nelze opomenout související Liangův algoritmus dělení slov [5], který se při lámání řádků v TEXu využívá.



Obrázek 1: Základní parametry řádkového zlomu

Zarovnání odstavce je určeno registry `\leftskip` a `\rightskip`, mezery těchto velikostí se vkládají na příslušný okraj každého řádku. Jako odstavcovou zarážku TEX vloží `\hbox` šířky `\parindent`. Na konec posledního řádku odstavce se vloží mezera velikosti `\parfillskip`. Situaci zachycuje obrázek 1. Jednotlivá zarovnání budeme v rámečcích na stranách 48–53 demonstrovat na textu z [6].

3 Příkaz `\shipout`

3.1 Minimální příklad

Primitivní příkaz `\shipout` do souboru .pdf okamžitě vloží stránku s vysázeným boxem, který za příkazem `\shipout` následuje. Příkaz `\shipout` neprovede nic dalšího – nevloží záhlaví, nezmění číslo strany a podobně.

Vyzkoušejme si vysázet jednoduchý dokument.

```
1 graf%
2 \shipout\hbox{ahoj}%
3 ika
4 \bye
```

Na řádku 1 hlavní procesor začne plnit horizontální seznam. Na řádku 2 se do souboru .pdf vloží stránka s textem „ahoj“. Na řádku 3 se pokračuje v plnění horizontálního seznamu. Příkaz `\bye` na řádku 4 ukončí horizontální seznam, vyvolá algoritmus řádkového zlomu, algoritmus stránkového zlomu a na závěr pomocí output rutiny do souboru .pdf vloží stránku s textem „grafika“. (Proč se nevysází text „grafika“, se čtenář dočte například v [2] na straně 103.) Výsledný soubor .pdf tak bude mít dvě strany, přičemž stránka ručně vložená pomocí příkazu `\shipout` předchází stránce, na které byl příkaz `\shipout` použit.

`\leftskip=0pt`
`\rightskip=0pt`
`\parindent=20pt`
`\parfillskip=0pt plus 1fil`
	V plainu jsou implicitně registry nastaveny na uvedené hodnoty. Při tomto nastavení je text zarovnaný do bloku, přičemž poslední řádek může být kratší.

Sto roků v šachtě žil, mlčel jsem, sto roků kopal jsem uhlí, za sto let v rameni bezmasém svaly mi v železo ztuhly. Uhelný prach sedl do očí, rubíny ze rtů mi uhly, ze vlasů, z vousů a z obočí visí mi rampouchy uhlí. Chléb s uhlím beru si do práce, z roboty jdu na robotu, při Dunaji strmí paláce z krve mé a z mého potu. Sto roků v kopalně mlčel jsem, kdo mi těch sto roků vrátí? Když jsem pohrozil kladivem, kdekdo se začal mi smáti. Abych měl rozum, šel v kopalnu zas, pro pány robil jak prve: máchl jsem kladivem – teklo vráz na Polské Ostravě krve! Všichni vy na Slezské, všichni vy, dím, nech je vám Petr neb Pavel, mějž prs kryt krunýřem ocelovým, tisícům k útoku zavel, všichni vy na Slezské, všichni vy, dím, hlubokých páni vy dolů: přijde den, z dolů jde plamen a dým, přijde den, zúčtujem spolu!

`\leftskip=0pt`
`\rightskip=0pt plus 1fil`
`\parindent=20pt`
`\parfillskip=0pt`
	Zarovnání na praporek docílíme přidáním nekonečné pružnosti k registru `\rightskip`. Nekonečná pružnost způsobí, že TEX preferuje vytvoření kratšího řádku před případným rozdělením slova. Registr `\parfillskip` můžeme vynulovat.

Sto roků v šachtě žil, mlčel jsem, sto roků kopal jsem uhlí, za sto let v rameni bezmasém svaly mi v železo ztuhly. Uhelný prach sedl do očí, rubíny ze rtů mi uhly, ze vlasů, z vousů a z obočí visí mi rampouchy uhlí. Chléb s uhlím beru si do práce, z roboty jdu na robotu, při Dunaji strmí paláce z krve mé a z mého potu. Sto roků v kopalně mlčel jsem, kdo mi těch sto roků vrátí? Když jsem pohrozil kladivem, kdekdo se začal mi smáti. Abych měl rozum, šel v kopalnu zas, pro pány robil jak prve: máchl jsem kladivem – teklo vráz na Polské Ostravě krve! Všichni vy na Slezské, všichni vy, dím, nech je vám Petr neb Pavel, mějž prs kryt krunýřem ocelovým, tisícům k útoku zavel, všichni vy na Slezské, všichni vy, dím, hlubokých páni vy dolů: přijde den, z dolů jde plamen a dým, přijde den, zúčtujem spolu!

3.2 Vložení strany s obrázkem

S využitím příkazu `\shipout` lze do dokumentu jednoduše vkládat strany jiného dokumentu. Níže použité makro `\margins` uvnitř skupiny lokálně nastaví nulové okraje, aby vložená strana byla přes celou stranu dokumentu.

```
\begingroup
\margins/1 a4 (0,0,0,0)mm
\pdfximage{zadaniVSKP.pdf}
\shipout\hbox{\pdfrefximage\pdflastximage}
\endgroup
```

`\leftskip=0pt plus 1fil`
`\rightskip=0pt plus 1fil`
`\parindent=0pt`
`\parfillskip=0pt`
	Pokud stejnou nekonečnou pružnost přidáme i k `\leftskip`, bude odstavec zarovnaný na střed. Nesmíme ale zapomenout vynulovat `\parindent` a `\parfillskip`.

Sto roků v šachtě žil, mlčel jsem, sto roků kopal jsem uhlí, za sto let v rameni bezmasém svaly mi v železo ztuhly. Uhelný prach sedl do očí, rubíny ze rtů mi uhly, ze vlasů, z vousů a z obočí visí mi rampouchy uhlí. Chléb s uhlím beru si do práce, z roboty jdu na robotu, při Dunaji strmí paláce z krve mé a z mého potu. Sto roků v kopalně mlčel jsem, kdo mi těch sto roků vrátí? Když jsem pohrozil kladivem, kdekdo se začal mi smáti. Abych měl rozum, šel v kopalnu zas, pro pány robil jak prve: máchl jsem kladivem – teklo vráz na Polské Ostravě krve! Všichni vy na Slezské, všichni vy, dím, nech je vám Petr neb Pavel, mějž prs kryt krunýřem ocelovým, tisícům k útoku zavel, všichni vy na Slezské, všichni vy, dím, hlubokých páni vy dolů: přijde den, z dolů jde plamen a dým, přijde den, zúčtujem spolu!

`\leftskip=0pt plus 1fil`
`\rightskip=0pt plus -1fil`
`\parindent=0pt`
`\parfillskip=0pt plus 2fil`
	Zarovnání do bloku s posledním řádkem na střed lze použít při sazbě popisků obrázků. Hodnoty registrů jsou voleny tak, aby jejich pružnost byla nekonečná a aby platilo `\leftskip` + `\rightskip` = `0pt` a zároveň `\leftskip` = `\rightskip`+`\parfillskip`.

Sto roků v šachtě žil, mlčel jsem, sto roků kopal jsem uhlí, za sto let v rameni bezmasém svaly mi v železo ztuhly. Uhelný prach sedl do očí, rubíny ze rtů mi uhly, ze vlasů, z vousů a z obočí visí mi rampouchy uhlí. Chléb s uhlím beru si do práce, z roboty jdu na robotu, při Dunaji strmí paláce z krve mé a z mého potu. Sto roků v kopalně mlčel jsem, kdo mi těch sto roků vrátí? Když jsem pohrozil kladivem, kdekdo se začal mi smáti. Abych měl rozum, šel v kopalnu zas, pro pány robil jak prve: máchl jsem kladivem – teklo vráz na Polské Ostravě krve! Všichni vy na Slezské, všichni vy, dím, nech je vám Petr neb Pavel, mějž prs kryt krunýřem ocelovým, tisícům k útoku zavel, všichni vy na Slezské, všichni vy, dím, hlubokých páni vy dolů: přijde den, z dolů jde plamen a dým, přijde den, zúčtujem spolu!

Strana vložená do souboru .pdf příkazem `\shipout` má rozměry podle nastavení platného v době zavolání příkazu `\shipout`. Příslušný box se umístí s ohledem na nastavený levý a horní okraj strany.

Primitivní příkazy `\pdfximage`, `\pdfrefximage` a `\pdflastximage` ke vkládání obrázků jsou popsány například v [7] v sekci 11.7.

3.3 Vložení speciální strany

Praktické využití může mít vložení velké tabulky samostatně na další stranu dokumentu. V ukázce se tabulka vysází na stránku na šířku. Nesmíme zapomenout na řádku 20 zvýšit číslo strany.

`\leftskip=0pt`
`\rightskip=0pt plus 60pt`
`\parindent=0pt`
`\parfillskip=0pt plus 1fil`
	Pokud má registr `\rightskip` konečnou pružnost, pak budou pružit i mezislovní mezery. TEX bude preferovat delší řádky i za cenu případného rozdělení slov.

Sto roků v šachtě žil, mlčel jsem, sto roků kopal jsem uhlí, za sto let v rameni bezmasém svaly mi v železo ztuhly. Uhelný prach sedl do očí, rubíny ze rtů mi uhly, ze vlasů, z vousů a z obočí visí mi rampouchy uhlí. Chléb s uhlím beru si do práce, z roboty jdu na robotu, při Dunaji strmí paláce z krve mé a z mého potu. Sto roků v kopalně mlčel jsem, kdo mi těch sto roků vrátí? Když jsem pohrozil kladivem, kdekdo se začal mi smáti. Abych měl rozum, šel v kopalnu zas, pro pány robil jak prve: máchl jsem kladivem – teklo vráz na Polské Ostravě krve! Všichni vy na Slezské, všichni vy, dím, nech je vám Petr neb Pavel, mějž prs kryt krunýřem ocelovým, tisícům k útoku zavel, všichni vy na Slezské, všichni vy, dím, hlubokých páni vy dolů: přijde den, z dolů jde plamen a dým, přijde den, zúčtujem spolu!

`\leftskip=0pt`
`\rightskip=0pt`
`\parindent=0pt`
`\parfillskip=0pt`
	Při nulovém nastavení všech registrů bude odstavec zarovnaný do obdélníku. V tomto případě je ale pravděpodobné, že vznikne řádek s extrémně širokými mezislovními mezerami.

Sto roků v šachtě žil, mlčel jsem, sto roků kopal jsem uhlí, za sto let v rameni bezmasém svaly mi v železo ztuhly. Uhelný prach sedl do očí, rubíny ze rtů mi uhly, ze vlasů, z vousů a z obočí visí mi rampouchy uhlí. Chléb s uhlím beru si do práce, z roboty jdu na robotu, při Dunaji strmí paláce z krve mé a z mého potu. Sto roků v kopalně mlčel jsem, kdo mi těch sto roků vrátí? Když jsem pohrozil kladivem, kdekdo se začal mi smáti. Abych měl rozum, šel v kopalnu zas, pro pány robil jak prve: máchl jsem kladivem – teklo vráz na Polské Ostravě krve! Všichni vy na Slezské, všichni vy, dím, nech je vám Petr neb Pavel, mějž prs kryt krunýřem ocelovým, tisícům k útoku zavel, všichni vy na Slezské, všichni vy, dím, hlubokých páni vy dolů: přijde den, z dolů jde plamen a dým, přijde den, zúčtujem spolu!

```
\begingroup
\margins/1 a4l (2,2,2,2)cm
\shipout\vbox{
  \centerline{%
    \table{rcl}
      {ob & rov & ská \cr
       ta & bul & ka \cr}
    }\medskip
  \caption/t Popisek
  \par}
\global\advance\pageno1
\endgroup
```

`\hsize=96mm`
`\leftskip=0pt`
`\rightskip=0pt`
`\parindent=0pt`
`\parfillskip=0pt`
	Pokud potřebujeme odstavec vysázet do obdélníku, ale nezáleží nám na jeho šířce, můžeme se pokusit vyhnout extrémně širokým mezerám změnou šířky sazby `\hsize`.

Sto roků v šachtě žil, mlčel jsem, sto roků kopal jsem uhlí, za sto let v rameni bezmasém svaly mi v železo ztuhly. Uhelný prach sedl do očí, rubíny ze rtů mi uhly, ze vlasů, z vousů a z obočí visí mi rampouchy uhlí. Chléb s uhlím beru si do práce, z roboty jdu na robotu, při Dunaji strmí paláce z krve mé a z mého potu. Sto roků v kopalně mlčel jsem, kdo mi těch sto roků vrátí? Když jsem pohrozil kladivem, kdekdo se začal mi smáti. Abych měl rozum, šel v kopalnu zas, pro pány robil jak prve: máchl jsem kladivem – teklo vráz na Polské Ostravě krve! Všichni vy na Slezské, všichni vy, dím, nech je vám Petr neb Pavel, mějž prs kryt krunýřem ocelovým, tisícům k útoku zavel, všichni vy na Slezské, všichni vy, dím, hlubokých páni vy dolů: přijde den, z dolů jde plamen a dým, přijde den, zúčtujem spolu!

`\leftskip=0pt`
`\rightskip=0pt`
`\parindent=0pt`
`\parfillskip=0pt`
`\emergencystretch100pt`
	Pokud potřebujeme odstavec vysázet do obdélníku a šířka je daná, pak si musíme pohrát s pružností mezislovních mezer. Nastavení registru `\emergencystretch` na kladnou hodnotu způsobí, že TEX nebude vnímat velké mezery jako příliš nevhodné. Bude se tak preferovat řádkový zlom s více řádky s velkými mezerami před zlomem s jedním řádkem s obrovskou mezerou.

Sto roků v šachtě žil, mlčel jsem, sto roků kopal jsem uhlí, za sto let v rameni bezmasém svaly mi v železo ztuhly. Uhelný prach sedl do očí, rubíny ze rtů mi uhly, ze vlasů, z vousů a z obočí visí mi rampouchy uhlí. Chléb s uhlím beru si do práce, z roboty jdu na robotu, při Dunaji strmí paláce z krve mé a z mého potu. Sto roků v kopalně mlčel jsem, kdo mi těch sto roků vrátí? Když jsem pohrozil kladivem, kdekdo se začal mi smáti. Abych měl rozum, šel v kopalnu zas, pro pány robil jak prve: máchl jsem kladivem – teklo vráz na Polské Ostravě krve! Všichni vy na Slezské, všichni vy, dím, nech je vám Petr neb Pavel, mějž prs kryt krunýřem ocelovým, tisícům k útoku zavel, všichni vy na Slezské, všichni vy, dím, hlubokých páni vy dolů: přijde den, z dolů jde plamen a dým, přijde den, zúčtujem spolu!

3.4 Řízený tisk obrázků

Předpokládejme, že máme stovku obrázků uloženou v souborech obrazek1.jpg až obrazek100.jpg. Úkolem bude vytisknout všechny tyto obrázky ve stejné velikosti, v našem případě na stránku A4 na šířku s 5mm okraji.

Zkušenosti autorů ukazují, že u ovladačů tiskáren je možné nastavovat mnoho

`\leftskip=0pt`
`\rightskip=0pt`
`\parindent=0pt`
`\parfillskip=0pt`
`\tolerance500`
	Nastavení velké hodnoty `\tolerance` způsobí, že TEX bude zkoumat i možné řádkové zlomy s většími mezerami a z těchto zlomů najde ten nejlepší.

Sto roků v šachtě žil, mlčel jsem, sto roků kopal jsem uhlí, za sto let v rameni bezmasém svaly mi v železo ztuhly. Uhelný prach sedl do očí, rubíny ze rtů mi uhly, ze vlasů, z vousů a z obočí visí mi rampouchy uhlí. Chléb s uhlím beru si do práce, z roboty jdu na robotu, při Dunaji strmí paláce z krve mé a z mého potu. Sto roků v kopalně mlčel jsem, kdo mi těch sto roků vrátí? Když jsem pohrozil kladivem, kdekdo se začal mi smáti. Abych měl rozum, šel v kopalnu zas, pro pány robil jak prve: máchl jsem kladivem – teklo vráz na Polské Ostravě krve! Všichni vy na Slezské, všichni vy, dím, nech je vám Petr neb Pavel, mějž prs kryt krunýřem ocelovým, tisícům k útoku zavel, všichni vy na Slezské, všichni vy, dím, hlubokých páni vy dolů: přijde den, z dolů jde plamen a dým, přijde den, zúčtujem spolu!

`\leftskip=0pt plus 20pt`
`\rightskip=0pt plus 20pt`
`\parindent=0pt`
`\parfillskip=0pt`
	Při použití ozdobného písma není nutné, aby okraje sazby byly zcela rovné. Pak můžeme zarovnání do obdélníku docílit nastavením malé pružnosti u `\leftskip` a `\rightskip`.

Sto roků trisekci zkoušel jsem, sto roků sekal jsem úhly. Za sto let Galois přišel sem, nálady dobré mi uhly. Rovnice padly mi do očí, kořeny z tělesa uhly. Cardano vzorcem svým útočí, najde ty rozseklé úhly. Kružítko beru si do práce, pravítko beru si taky. Eukleidés radí mi nevzdát se, konstrukcí dělal už mraky. Za sto let kružítka mám už dost, kdy se mi snahy ty vrátí? Od Knutha kamarád Metapost námahu výrazně zkrátí. Abych měl rozum, já sekal jsem zas, novinky zkusil však prve. Zadeha matika úhly vráz namísto sekání urve. Všechny vás nezdolám, všechny ne, vím, úhlové velcí neb malí, s kvantifikátorem existenčním řešení přichází z dáli. Všechny vás nezdolám, všechny ne, vím, všelikých úhly vy stupňů. Angulus rectus, ten já roztřetím a na tři třicítky utnu!

parametrů. Ne vždy je však jednoduché nastavit parametry ovladačů tak, aby byl text nebo obrázek vytištěný přesně v požadovaném rozměru a na požadované pozici. Ukázalo se jednodušším umístit text nebo obrázek přesně v požadovaném rozměru a na požadované pozici do pdf souboru, v němž rozměr stránky odpovídá rozměru stránky v tiskárně. Ovladači tiskárny se pak řekne, aby dokument se vytiskl přímo bez dalších úprav a transformací.

Proto náš úkol vyřešíme vytvořením jednoduchého pdf dokumentu, do nějž v cyklu na jednotlivé stránky vložíme příslušné obrázky.

```
\input opmac
\margins/1 a4l (5,5,5,5)mm
\tmpnum0
```

`\leftskip=1cm`
`\rightskip=1cm`
`\parindent=-1cm`
`\parfillskip=-1cm`
	Při sazbě obsahu nebo rejstříku bývají neprvní řádky odsazeny zleva a neposlední řádky odsazeny zprava.

Sto roků v šachtě žil, mlčel jsem, sto roků kopal jsem uhlí, za sto let v rameni bezmasém svaly mi v železo ztuhly. Uhelný prach sedl do očí, rubíny ze rtů mi uhly, ze vlasů, z vousů a z obočí visí mi rampouchy uhlí. Chléb s uhlím beru si do práce, z roboty jdu na robotu, při Dunaji strmí paláce z krve mé a z mého potu. Sto roků v kopalně mlčel jsem, kdo mi těch sto roků vrátí? Když jsem pohrozil kladivem, kdekdo se začal mi smáti. Abych měl rozum, šel v kopalnu zas, pro pány robil jak prve: máchl jsem kladivem – teklo vráz na Polské Ostravě krve! Všichni vy na Slezské, všichni vy, dím, nech je vám Petr neb Pavel, mějž prs kryt krunýřem ocelovým, tisícům k útoku zavel, všichni vy na Slezské, všichni vy, dím, hlubokých páni vy dolů: přijde den, z dolů jde plamen a dým, přijde den, zúčtujem spolu!

`\parshape12`
`  0mm 96mm 13mm 96mm 18mm 96mm`
`  21mm 96mm 23mm 96mm 24mm 96mm`
`  24mm 96mm 23mm 96mm 21mm 96mm`
`  18mm 96mm 13mm 96mm 0mm 96mm`
`\parindent=0pt`
`\parfillskip=0pt`
	Příkazem `\parshape` lze nastavit libovolný tvar odstavce. První parametr určuje počet nastavovaných řádků a za ním následuje odsazení a šířka každého řádku. I v tomto případě jsou na příslušná místa vloženy mezery z `\parindent`, `\leftskip`, `\rightskip` a `\parfillskip`.

Sto roků v šachtě žil, mlčel jsem, sto roků kopal jsem uhlí, za sto let v rameni bezmasém svaly mi v železo ztuhly. Uhelný prach sedl do očí, rubíny ze rtů mi uhly, ze vlasů, z vousů a z obočí visí mi rampouchy uhlí. Chléb s uhlím beru si do práce, z roboty jdu na robotu, při Dunaji strmí paláce z krve mé a z mého potu. Sto roků v kopalně mlčel jsem, kdo mi těch sto roků vrátí? Když jsem pohrozil kladivem, kdekdo se začal mi smáti. Abych měl rozum, šel v kopalnu zas, pro pány robil jak prve: máchl jsem kladivem – teklo vráz na Polské Ostravě krve! Všichni vy na Slezské, všichni vy, dím, nech je vám Petr neb Pavel, mějž prs kryt krunýřem ocelovým, tisícům k útoku zavel, všichni vy na Slezské, všichni vy, dím, hlubokých páni vy dolů: přijde den, z dolů jde plamen a dým, přijde den, zúčtujem spolu!

```
\loop
  \ifnum\tmpnum<100 \advance\tmpnum1
  \pdfximage width\hsize{obrazek\the\tmpnum.jpg}
  \shipout\hbox{\pdfrefximage\pdflastximage}
\repeat
\end
```

3.5 Stránková montáž

Pomocí příkazu `\shipout` je možné provádět stránkovou montáž, například vysázet dokument jako brožuru. O tomto bylo více pojednáno v článku [8].

4 Output rutina

V kapitole 23 TEXbooku Donald Knuth o output rutině píše:

> *Chapter 22 taught you how to be a TEX master, i.e., a person who can produce complicated tables using* `\halign` *and* `\valign`*; the following material will take you all the way to the rank of Grandmaster, i.e., a person who can design output routines. When you are ready for this rank, you will be pleased to discover that—like alignments—output routines are not really so mysterious as they may seem at first.*

V následujícím textu si na příkladech ukážeme, že i běžný smrtelník si může output rutinu TEXu upravit k obrazu svému.

Output rutina je posloupnost tokenů uložená v token registru `\output`, kterou TEX zpracuje v okamžiku, kdy algoritmus stránkového zlomu přesně určí, jaký vertikální materiál se vysází na aktuální stránce. Algoritmus stránkového zlomu tento materiál uloží do vboxu `\box255` a úkolem output rutiny zpravidla je umístit tento vbox na stránku a vložit stránku do souboru `.pdf`. Proto by output rutina měla obsahovat příkaz `\shipout`.

Nepříjemnou vlastností output rutiny je, že dopředu není známo, ve kterém okamžiku se zavolá. Na základě principu nutného minima se output rutina zavolá až po ukončení algoritmu řádkového zlomu, kdy se do vertikálního seznamu vloží celý odstavec. To bývá většinou v místě, které se poté vysází na následující straně. V případě odstavců delších než jedna strana se může stát, že se output rutina zavolá až v místě, které se poté vysází o několik stran dále.

Jedním důsledkem předchozího je, že TEX interně musí output rutinu zpracovat uvnitř skupiny. Proto je také nastavení platné napříč dokumentem nutné uvnitř output rutiny měnit globálně. To se týká například registru `\pageno`, ve kterém bývá zvykem mít uloženo číslo strany. (V LATEXu se používá registr `\c@page`, k němuž lze přistupovat například přes LATEXová makra `\arabic{page}` nebo `\setcounter{page}`.)

Druhým důsledkem je, že nemá smysl v textu dokumentu registr `\pageno` používat. Pokud bychom jej použili, neměli bychom vůbec jistotu, že by v něm bylo uloženo číslo aktuální strany. (Na řádku 20 jsme k registru `\pageno` mimo output rutinu přistupovali. Nezajímala nás však jeho hodnota, ale pouze jsme jej potřebovali zvýšit o jedničku.)

4.1 Minimální příklad

Vyzkoušejme si vysázet jednoduchý dokument.

```
31 \output{\shipout\box255}
32
33 Hello
34
35 \vfil\break
36
37 World
38
39 \bye
```

Nejprve si na řádku 31 nadefinujeme minimální output rutinu. Ta pouze vezme obsah boxu 255 a vloží jej do pdf souboru. Výsledný dokument tak bude mít dvě stránky, kde na první stránce bude pouze text „Hello“ a na druhé stránce text „World“. Žádné další objekty uvedené v podsekci 1.7 se na stránku nevloží.

4.2 Jednoduchá output rutina

Nyní si vytvoříme jednoduchou output rutinu, která na stránku vysází `\box255` a pod něj vysází číslo strany zarovnané na střed.

```
40 \output{
41   \shipout\vbox{
42     \vbox to\vsize{\unvbox255}
43     \bigskip
44     \centerline{\tenrm\the\pageno}}
45   \global\advance\pageno1}
```

Musíme si uvědomit, že `\box255` může mít pokaždé jinou výšku a output rutina musí zařídit, aby navenek působil jako box stejné výšky. V našem případě bude výška rovna `\vsize`. V opačném případě by na každé stránce bylo číslo strany v jiné výšce, což určitě není žádoucí.

Na řádku 44 je pomocí `\tenrm` nastaven font, v němž bude vysázeno číslo strany. Pokud bychom font nenastavili, mohlo by se stát, že by na každé stránce bylo číslo strany vysázeno jiným fontem, konkrétně fontem, kterým byl aktuální v okamžiku zavolání output rutiny. (Uživatelé LaTeXu použijí jiný přepínač pro změnu fontu včetně velikosti.)

A opět nesmíme zapomenout globálně zvýšit číslo strany.

Výsledek může čtenář vidět na aktuální stránce.

4.3 Záhlaví a zápatí

V plainu jsou pro sazbu záhlaví a zápatí vyhrazeny token registry `\headline` a `\footline`. Do těchto registrů uživatel vloží horizontální materiál, který se má vysázet v řádku se záhlavím a zápatím.

Následující řádky vysázejí číslo strany v záhlaví vždy na vnějším okraji stránky. Na sudých stránkách bude nekonečně pružná mezera `\hfil` napravo od čísla a číslo tak bude vlevo. Na lichých stránkách bude více nekonečně pružná mezera `\hfill` nalevo od čísla a číslo tak bude vpravo. Podobně je v zápatí použita nulová mezera s nekonečnou pružností, která se roztáhne na celý řádek.

```
46 \headline{\ifodd\pageno\hfill\fi \tenrm\the\pageno\hfil}
47 \footline{\hss}
```

Uživatel si může samozřejmě nastavit jiné záhlaví a zápatí. Obsah registrů `\headline` a `\footline` pak v output rutině vysázíme pomocí příkazu `\the`.

```
48 \output{
49   \shipout\vbox{
50     \vbox to0pt{\vss\line{\the\headline}\bigskip}
51     \vbox to\vsize{\unvbox255}
52     \bigskip
53     \line{\the\footline}}
54   \global\advance\pageno1}
```

Na řádku 50 se vysází vertikální box nulové výšky, který neovlivní pozici sazby, přičemž materiál použitý v tomto boxu bude z boxu vyčnívat nahoru.

Výsledek může čtenář vidět na aktuální stránce.

Pokud chce čtenář tuto output rutinu použít v LaTeXu, musí si předefinovat makro `\line`

```
55 \def\line{\hbox to\hsize}
```

protože LaTeX definuje makro `\line` jiným způsobem. Tuto definici lze použít také lokálně, například vložit řádek 55 mezi řádky 48 a 49. Dále si čtenář musí v LaTeXu registry `\headline` a `\footline` nejprve nadeklarovat pomocí `\newtoks\headline` a `\newtoks\footline`. V LaTeXu se implicitně text záhlaví namísto do token registrů `\headline` a `\footline` ukládá do maker `\@oddhead`, `\@evenhead`, `\@oddfoot` a `\@evenfoot`, jejichž existence je před uživatelem utajena a musí k nastavení těchto maker použít jiná makra.

4.4 Čára pod záhlavím

Záhlaví je možné oddělit od textu horizontální čarou. Musíme si však uvědomit, že horizontální čáru je možné použít pouze ve vertikálním módu, přičemž obsah

registru `\headline` se expanduje v horizontálním módu. K tomu lze využít příkaz `\vadjust`, který vloží příslušný vertikální materiál pod aktuálně sázený horizontální box. Při použití output rutiny ze strany 56 můžeme registry `\headline` a `\footline` nastavit následovně.

```
\headline{\ifodd\pageno\hfill\fi \tenrm\the\pageno\hfil
  \vadjust{\vskip3pt\hrule\vskip-3.4pt}}
\footline{\hss}
```

Vertikální mezera `3pt` slouží k oddělení záhlaví od čáry. Čára má implicitní tloušťku `0.4pt`, a proto vertikální mezera `-3.4pt` posune sazbu zpět do původního bodu.

Čtenář si může promyslet, kde by bylo třeba použít příkaz `\vskip` a s jakými parametry, aby se vysázela čára nad zápatím.

Alternativní možností je zakomponovat horizontální čáru přímo do output rutiny. Uživatel pak nemusí používat konstrukci z řádku 57.

```
\output{
  \shipout\vbox{
    \vbox to0pt{\vss\line{\the\headline}
      \vskip3pt\hrule\vskip-3.4pt
      \bigskip}
    \vbox to\vsize{\unvbox255}
    \bigskip
    \line{\the\footline}}
  \global\advance\pageno1}
```

Výsledek může čtenář vidět na aktuální stránce.

4.5 Plovoucí záhlaví

Do záhlaví je možné vložit také text, který je závislý na textu vyskytujícím se na aktuální stránce. Například v knihách se do záhlaví vkládá nadpis aktuální sekce, ve slovnících se do záhlaví vkládá rozsah slov vyskytujících se na aktuální stránce.

Vytvoříme si makro `\sekce`, které vysází nadpis sekce a tento nadpis uloží do makra `\nazevsekce`. Cílem bude, aby poté output rutina mohla použít název sekce v záhlaví. V našem případě bude v záhlaví název sekce umístěn naproti číslu strany.

```
\def\sekce#1{\par\bigskip
  \noindent
  \def\nazevsekce{#1}%
  {\bf#1}\par
  \nobreak\bigskip}
```

```
\headline{\tenrm\strut
  \ifodd\pageno
    \nazevsekce\hss\the\pageno
  \else
    \the\pageno\hss\nazevsekce
  \fi}
```

Takto definované makro však nebude správně fungovat. Očekávali a chtěli bychom, aby se v zápatí vyskytl název sekce, která je na aktuální stránce. Nicméně makro `\nazevsekce` se expanduje až v output rutině a v tom okamžiku v něm bude uložen poslední název sekce, který byl definován před zavoláním output rutiny. Z důvodů popsaných na straně 54 se může jednat o nadpis, který se vyskytne až na následující straně.

Pro tyto účely je v TEXu implementován příkaz `\mark`. Tento příkaz uloží svůj argument do paměti a v output rutině jej expanduje. Konkrétně se v output rutině expanduje

- `\topmark` na argument posledního `\mark` použitého na předchozí straně,
- `\firstmark` na argument prvního `\mark` použitého na aktuální straně,
- `\botmark` na argument posledního `\mark` použitého na aktuální straně.

Pokud tedy budeme chtít, ať se do záhlaví dostane název první sekce na aktuální straně, musíme v makru `\sekce` použít příkaz `\mark` a v záhlaví použít příkaz `\firstmark`.

```
\def\sekce#1{\par\bigskip
  \noindent
  \mark{#1}%
  {\bf#1}\par
  \nobreak\bigskip}
\headline{\tenrm\strut
  \ifodd\pageno
    \firstmark\hss\the\pageno
  \else
    \the\pageno\hss\firstmark
  \fi}
```

Čtenář nechť si sám vyzkouší, na co se budou expandovat příkazy `\topmark`, `\firstmark` a `\botmark` v případě, že na aktuální straně nezačíná žádná sekce, a v případě před nastavením prvního `\mark`.

Použité makro `\strut` vytvoří vertikální podpěru pro text, aby záhlaví bylo na každé straně ve stejné výšce, nezávisle na tom, jestli text záhlaví je celý nad účařím, nebo zda zasahuje pod účaří.

Výsledek může čtenář vidět na aktuální stránce.

4.6 Poznámky pod čarou

Nyní naši output rutinu naučíme pracovat s poznámkami pod čarou. V PlainTEXu se vysázené poznámky pod čarou ukládají do vboxu s číslem `\footins`.[3] Proto do naši output rutiny přidáme test, zdali je tento vbox prázdný. A pokud není, vložíme vertikální mezeru velikosti `\skip\footins`, vhodně umístíme horizontální čáru[4] a pak vysázíme tento vbox.

```
\output{\shipout\vbox{
  \vbox to0pt{\vss\line{\the\headline}
    \vskip3pt
    \hrule
    \vskip-3.4pt
    \bigskip}
  \vbox to \vsize{
    \unvbox255
    \ifvoid\footins\else
      \vskip\skip\footins
      \vskip-2.4pt
      \hrule width1cm
      \vskip2pt
      \unvbox\footins
    \fi
    \vfil}
  \bigskip
  \line{\the\footline}}
  \global\advance\pageno1}
```

Výsledek může čtenář vidět na aktuální stránce.

4.7 Jednoduché slajdy

V této podsekci vytvoříme output rutinu, kterou lze použít při sazbě slajdů. Jako rozměr stránky vezmeme formát A5 na šířku a nastavíme okraje pro text slajdů.

```
\margins/1 a5l (15,15,15,10)mm
```

V output rutině si lokálně nastavíme nulové okraje. Pomocí příkazů `\hrule` a `\vrule` vytvoříme obdélníky dané barvy. Pomocí `\hskip` a `\vskip` musíme zajistit, aby se po nakreslení obdélníku pozice sazby dostala zpět na patřičné místo. Výšku a hloubku obdélníků na řádcích 116 a 126 volíme tak, aby jejich součet odpovídal okrajům nastaveným na řádku 109, přičemž hloubka obdélníků

[3] Přesný mechanismus sazby poznámek pod čarou je uveden v [2] v sekci 6.7.

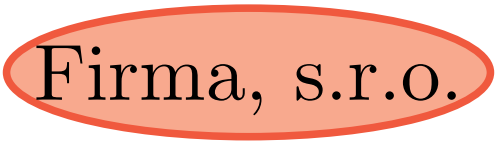
[4] Podle této čáry se poznámky nazývají pod čarou.

určuje vertikální umístění účaří příslušného textu v rámci obdélníku. Samotný `\box255` na slajdu vycentrujeme vertikálně i horizontálně na řádcích 120–123.

```
\output{\margins/1 a5l (0,0,0,0)mm
  \shipout\vbox to\vsize{
    \setcmykcolor{0 0 0.4 0.1}
    \hrule height\vsize width\hsize
    \vskip-\vsize
    \line{\setcmykcolor{0.5 0 0.5 0}%
      \vrule height10mm depth5mm width\hsize
      \hskip-\hsize \hskip15mm
      \typosize[20/]\bf\setcmykcolor{0 0 0 1}%
      \the\headline\hss}
    \vss
    \centerline{\setcmykcolor{0 0 0 1}%
      \vbox{\unvbox255\unskip}}
    \vss
    \line{\setcmykcolor{0.5 0 0.5 0}%
      \hskip\hsize \hskip-10mm
      \vrule height6mm depth4mm width10mm
      \hskip-10mm
      \typosize[12/]\bf\setcmykcolor{0 0 0 1}%
      \hss\the\pageno\hss
      }
    \hrule height0pt
  }
  \global\advance\pageno1
}
\headline{\firstmark}
```

Výsledek může čtenář vidět na aktuální stránce. (Ve skutečnosti byla output rutina drobně upravena, s ohledem na formát časopisu.)

Podobnou myšlenku můžeme použít při sazbě slovníku, kdy na hranu stránky umístíme počáteční písmeno slov na aktuální stránce.



4.8 Hlavičkový papír

Zcela analogicky můžeme vytvořit hlavičkové papíry nějaké firmy. Předpokládejme, že v souboru hlavicka.pdf máme uložen obrázek, který má stejné rozměry jako stránka a který se vloží na pozadí každé stránky dokumentu. Přes tento obrázek se pak bude sázet samotný text.

Přestože se obrázek bude vkládat na každou stránku dokumentu, my jej na řádcích 136–137 vložíme do dokumentu jen jednou a v output rutině budeme pouze vkládat odkaz na tento obrázek uložený v dokumentu. Tímto rozdílným přístupem oproti řádku 8 můžeme výrazně zmenšit velikost výsledného souboru, pokud bude dokument vícestránkový. Pro detaily opět odkazujeme čtenáře například na sekci 11.7 v [7].

Protože v output rutině lokálně na řádku 138 nastavujeme nulové okraje stránky, určuje vertikální mezera na řádku 145 vertikální odstup čísla strany od hrany stránky.

```
136 \pdfximage{hlavicka.pdf}
137 \edef\cislohlavicky{\the\pdflastximage}
138 \output{\margins/1 a5 (0,0,0,0)mm
139   \shipout\vbox to\vsize{
140     \pdfrefximage\cislohlavicky
141     \vskip-\vsize \vskip30mm
142     \centerline{\vbox{\unvbox255}}
143     \vss
144     \line{\hss\tenrm\the\pageno\hss}
145     \vskip20mm
146   }
147   \global\advance\pageno1
148 }
```

Výsledek může čtenář vidět na aktuální stránce.

Závěrem této sekce si uvědomme důležitý význam output rutiny. Autor dokumentu se stará pouze o obsah textu, tedy o `\box255`. Naproti tomu, grafik se *nezávisle* stará o umístění `\box255` a dalších náležitostí na stránku, tedy o output rutinu.

Ulice 256, Město, 655 36 +420 123 456 789 email@firma.cz

5 Pozice sazby

Může se stát, že autor bude potřebovat propojit grafiku s určitými částmi textu. Například když bude v prezentaci chtít zdůraznit, co s čím souvisí. Pro tyto účely může použít mechanismus pdfTEXu[5] na zjištění skutečné pozice konkrétního bodu sazby na stránce. Tento mechanismus úzce souvisí s output rutinou, protože skutečná pozice sazby je známa až v okamžiku vysázení sazby. Nicméně pro jeho použití do output rutiny zasahovat nemusíme.

5.1 Zjištění pozice

Pozici bodu TEX určuje v jednotkách `sp` měřených od levého dolního okraje stránky. Například pozice tohoto bodu je (12685637, 26756928). V bodě, jehož pozici chceme zjistit, použijeme příkaz `\pdfsavepos`. Souřadnice tohoto bodu pak získáme zápisem hodnot registrů `\pdflastxpos`, `\pdflastypos` do nějakého souboru.[6]

Konkrétně můžeme v dokumentu postupovat následovně. Pomocí příkazů

```
149 \newwrite\soubor
150 \openout\soubor pozice.txt
```

vytvoříme a otevřeme soubor `pozice.txt`. V dokumentu pak můžeme psát[7]

```
151 ... pozice tohoto
152 \pdfsavepos
153 \write\soubor{(\the\pdflastxpos, \the\pdflastypos)}%
154 bodu je ...
```

Poté, co zavřeme soubor `pozice.txt` příkazem

```
155 \closeout\soubor
```

bude v tomto souboru uložen řádek

```
156 (12685637, 26756928)
```

Jestliže nám jednotky `sp` nevyhovují, můžeme použít příkaz `\dimexpr`, který načte délkový výraz (v našem případě `12685637sp`), a příkaz `\the`, který hodnotu tohoto výrazu vypíše v jednotkách `pt`. Místo řádku 153 pišme

```
157 \write\soubor{\the\dimexpr\pdflastxpos sp,
158   \the\dimexpr\pdflastypos sp)}%
```

[5] Jedná se až o záležitost pdfTEXu, původní TEX tímto mechanismem vybaven není.

[6] Zápis do souborů TEX provádí až během exportu příslušného boxu do pdf souboru příkazem `\shipout`.

[7] Musíme si dát pozor, ať na řádku 153 nezavlečeme do textu nechtěnou mezeru.

Potom bude v souboru pozice.txt uložen řádek

```
159 (193.56746pt, 408.27832pt)
```

Samozřejmě si řádky 152 a 157–158 můžeme zautomatizovat vhodnými makry.

5.2 Propojení s METAPOSTem

Nyní si popíšeme, jak je možné pozice bodů, zjištěné v předchozí podsekci, předat programu METAPOST, který pak podle nich vytvoří příslušnou grafiku.

Vytvoříme si makro `\bod#1`, které do souboru pozice.txt zapíše pozici aktuálního bodu sazby tak, aby ji METAPOST uložil do proměnné `bod[#1]`.

```
160 \def\bod#1{\pdfsavepos
161   \write\soubor
162     {bod[#1]:=(\the\dimexpr\pdflastxpos sp,
163       \the\dimexpr\pdflastypos sp);}}
```

Po použití

```
164 ... pozice tohoto \bod{1}bodu je ...
```

bude v souboru pozice.txt uložen řádek

```
165 bod[1]:=(193.56746pt, 408.27832pt);
```

Dále si vytvoříme makro `\obr#1`, které v místě svého použití vloží do pdf souboru METAPOSTový obrázek s názvem obr.#1. Makro uloží aktuální pozici sazby do METAPOSTové proměnné[8] `bod[-#1]`, která bude sloužit jako referenční bod pro přesné umístění obrázku. Makro dále zjistí, zdali soubor s názvem obr.#1 existuje,[9] a pokud existuje, pak obrázek vloží na aktuální pozici makrem `\convertMPtoPDF`. Toto makro je definováno v souboru supp-pdf.tex.[10]

```
166 \input supp-pdf
167 \def\obr#1{\quitvmode\bod{-#1}%
168   \openin\testin obr.#1 \ifeof\testin\else
169     \closein\testin \convertMPtoPDF{obr.#1}{1}{1}%
170   \fi}
```

Makro `\obr` můžeme použít na libovolném místě na stránce, na kterou má být příslušný obrázek vložen.

[8] Indexy METAPOSTových proměnných mohou být libovolná čísla, konkrétně tato čísla mohou být i záporná nebo necelá.

[9] Uživatelé nepoužívající OPmac si nejdříve musejí příkazem `\newread\testin` alokovat řídicí sekvenci `\testin` pro načítaní souboru.

[10] Pokud LaTeXoví uživatelé načetli balíček color.sty, mají již soubor supp-pdf.tex interně načtený a řádek 166 mohou vynechat.

Zdrojový soubor METAPOSTových obrázků, v souladu s předchozím odstavcem, nazveme obr.mp. V tomto souboru musíme nejdříve deklarovat pole proměnných bod[] typu pair a načíst náš soubor pozice.txt. Pak vytvoříme jednotlivé obrázky obr.n, jak jsme zvyklí pomocí příkazů beginfig(n) a endfig, přičemž souřadnice bodu sazby, kde bylo použito makro \bod{k}, jsou uloženy v proměnné bod[k]. Na závěr každému obrázku jako jeho okraj nastavíme obdélník s nulovými rozměry umístěný do bodu bod[-n]. Tím zajistíme správné umístění obrázku v místě použití makra \obr{n} a zároveň zajistíme, že obrázek bude mít nulové rozměry a neposune se v tomto místě sazba.

```
pair bod[];
input pozice.txt
beginfig(n)
  ... obrázek obsahující bod[...]
  setbounds currentpicture to bod[-n]--cycle;
endfig;
end
```

Pro vygenerování dokumentu spustíme TEX tolikrát, než se ustálí pozice sazby, což je zpravidla dvakrát. Pak METAPOSTem vygenerujeme obrázky. A nakonec ještě jednou spustíme TEX, čímž obrázky vložíme.

5.3 Propojení s METAPOSTem a output rutinou

Uvedený postup můžeme ještě více zautomatizovat tím, že makro \obr umístíme přímo na nějaké[11] vhodné[12] místo output rutiny. Přitom můžeme využít registr \pageno a vložit obrázek příslušný pro danou stránku.

```
\output{...\obr{\the\pageno}...}
```

V METAPOSTu pak můžeme definovat makra

```
def beginobr(expr n)=
  CisloObr:=n;
  beginfig(n)
enddef;
def endobr=
  setbounds currentpicture to bod[-CisloObr]--cycle;
  endfig;
enddef;
```

[11] Makro \convertMPtoPDF můžeme použít pouze v horizontálním módu. V opačném případě se nejdříve horizontální mód zahájí, čímž se ale posune pozice sazby.

[12] Není vhodné obrázek umísťovat například do zápatí, neboť bychom uvnitř \shipout nejdříve vysázeli \box255 s textem stránky a přes něj bychom překreslili obrázek, čímž bychom zakryli text.

a řádky 173–176 zjednodušit následovně.

```
beginobr(n)
   ... obrázek obsahující bod[...]
endobr;
```

Odkazy

1. KNUTH, Donald Ervin. *Computers & Typesetting, Volume A: The TEXbook.* Addison-Wesley, 1986.
2. OLŠÁK, Petr. *TEXbook naruby.* Konvoj, 2001.
3. KNUTH, Donald Ervin. *Computers & Typesetting, Volume B: TEX: The Program.* Addison-Wesley, 1986.
4. KNUTH, Donald E.; PLASS, Michael F. *Breaking Paragraphs into Lines.* 1981.
5. LIANG, Frank. *Word Hy-phen-a-tion by Com-put-er.* 1983. Disertační práce. Stanford University, Department of Computer Science.
6. BEZRUČ, Petr. *Slezské písně.* Československý spisovatel, 1951.
7. OLŠÁK, Petr. *TEX pro pragmatiky.* CSTUG, 2016.
8. ŠUSTEK, Jan. Načítání souboru s argumenty v TEXu. *Zpravodaj CSTUG.* 2015, roč. 25, č. 1–2, s. 86–94. ISSN 1211-6661. Dostupné z DOI: 10.5300/2015-1-2/86.

Summary: Parameters of the Line Breaking Algorithm and the Output Routine and Their Applications for Typesetting in TEX

In the paper we go through the inner parts of TEX and we show how the particular characters of the input file .tex get to the output file .pdf. We focus on the line breaking algorithm explaining how its parameters affect the paragraph alignment. Then we focus on the output routine showing how to put the typeset text on the page. Finally we mention the way how to find the exact position of a particular point on the page with an application in METAPOST figures.

Keywords: TEX, line break, output routine

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

Výsledky výuky zpracování textů

Jiří Rybička

Výuka zpracování textů je na Provozně ekonomické fakultě MENDELU zavedena formou volitelného předmětu již více než 18 let. Koncepce předmětu se postupem času poněkud změnila z počátečního více technického pojetí k současnému, které se soustřeďuje na obecnější poznatky z typografie a zaměření na odborné, zejména závěrečné práce. Článek se zabývá nástinem analýzy výsledků výuky v tomto předmětu, a to zpracováním vybraných výsledků ze zkouškových písemných prací.

Úvod

Aplikace počítačů v oblasti zpracování textů vykazují již od počátků v 80. a 90. letech minulého století masivní zastoupení napříč všemi obory. Je to pouze projevem skutečnosti, že jde o velmi potřebnou a prakticky použitelnou úlohu a ta je navíc podpořena i rozvojem technického vybavení, které umožnilo téměř ze dne na den zcela vyměnit technologii (psací stroje, horká sazba) za počítačové tiskárny. Tento prudký vývoj však nebyl doprovázen dostatečně masivní osvětou v oblasti dříve nedostupných technologií, jakými bezesporu byla sazba z liter odlévaných z kovu, a to ručním i strojním způsobem. Vznikla doslova propast mezi možnostmi dostupných počítačových programů a technických prvků a mezi znalostmi, které by umožňovaly jejich správné využití.

Tato mezera existuje dodnes a její zmenšování představuje dlouhodobý a náročný proces reprezentovaný existencí mnoha přístupů ve výuce na školách všech stupňů. Je přirozené, že snahou předmětů s touto tématikou je co nejefektivnějším způsobem vybavit studenty základními poznatky. Zůstává ovšem otázkou, do jaké míry se to daří.

Současný stav

Stručný přehled nabízených výukových aktivit

Výuce základů úpravy textu a typografie se věnuje na školách poměrně široká pozornost, i když stále zjevně chybí jakákoliv koordinace a ustálená náplň. Jen pro ilustraci můžeme uvést několik typů takových kurzů:

- Informace o vybraných pravidlech vhodných podle autora pro zpracování závěrečné práce. Texty tohoto typu jsou časté na vysokých školách (např. Raclavský, 2018; Borůvková, 2010).

doi: 10.5300/2019-1-4/66

- Učební texty zaměřené na seznámení s normou ČSN 01 6910 (2014), např. text určený pro střední školu Typografie – základy (2018) nebo text ze základní školy (Šošolík, 2018).
- Komplexněji pojaté texty zabývající se aplikací typografických pravidel v určité oblasti (oboru), například bakalářská práce Cvingráfové (2011) o využití počítačů v chemii.
- Texty o typografii doplněné o návody, jak prezentovaná pravidla realizovat (nejčastěji v MS Word) – například Typografická pravidla (2018).

Nepříjemnou vlastností některých těchto textů je nízká kvalita. Například již zmíněná práce Cvingráfové (2011) nerespektuje typografická pravidla, i když o nich přímo pojednává, podobně i práce Borůvkové (2010) vykazuje značné mezery ve výkladu a samotným vzhledem dává nevhodný příklad zpracování závěrečné práce.

Výuka zpracování textů na Provozně ekonomické fakultě MENDELU

Předmět „Zpracování textů na počítači“ se vyučuje již od akademického roku 1999/2000. Do prvního běhu bylo zapsáno 69 studentů. V posledních letech se předmět vyučuje v zimním i letním semestru a za akademický rok jej odstuduje přes 200 studentů.

Původní náplň byla zaměřena zejména na představení technologie počítačové sazby systémem TEX (přesněji LATEX) a částečně také Adobe InDesign, zatímco typografická pravidla tvořila spíše doplněk a nezbytný základ pro správné formátování dokumentů. Postupem času ovšem toto technické hledisko poněkud ustoupilo a stále větší důraz byl kladen na specifika odborných prací, a to zejména závěrečných prací. To se ukázalo jako poměrně účinný prostředek pro zatraktivnění předmětu i pro studenty, pro něž technická stránka věci neznamenala centrální bod zájmu, ale spíše se chtěli zorientovat při zpracovávání bakalářských a diplomových prací.

Konstelace učebních plánů a možnosti zapisovat volitelné předměty dokonce velmi zvýhodňuje studenty neinformatických oborů, což se odráží v minimálním počtu studentů informatiky, kteří si předmět zapíší. Pro většinu studentů je tedy technologie sazby a přehled možností sázecích systémů nezajímavý. To bylo také příčinou postupné změny ve vedení předmětu. Cvičením stále zůstává technický charakter a řeší se na nich zpracování dokumentů v systému (Xe)LATEX, seznámení s principy práce v systému InDesign bylo zcela opuštěno. Přednášky pak obsahují typografická pravidla a jejich realizaci v programech typu MS Word nebo v jeho open source alternativách.

Rovněž závěrečné ověřování znalostí obsahuje z větší části otázky z oblasti typografických zásad, zatímco technickou stránku zpracování dokumentů reprezentuje jen zápočtová úloha, na níž mají studenti prokázat základní orientaci v nejběžnějších rekvizitách systému (Xe)LATEX.

Pohled fakulty
Tento stav je víceméně v souladu s potřebami fakulty. Z úrovně závěrečných prací je patrný posun oproti dřívějšímu stavu. Za „normální“ se nyní považuje práce, jejíž typografické a technické zpracování respektuje většinu běžných typografických pravidel, zatímco práce formátované laicky a napěchované nejrůznějšími chybami a technickými nedostatky jsou již skutečnou výjimkou. Přispívá k tomu i existence šablony s předdefinovanými styly, šablona ovšem zdaleka nevyřeší všechno a bez aktivního přístupu studentů by kvalitní dokumenty nevznikaly.

Kvalita závěrečných prací je sledována jako jedna z priorit fakulty. Určitě se v této oblasti dá ještě mnoho věcí zlepšovat. V závěrečných pracích se i dnes vyskytují formální chyby, a to i u studentů, kteří předmět Zpracování textů na počítači absolvovali. Vzniká tedy logická otázka, nakolik jsou typografická pravidla studenty přijímána a osvojována. Prvotním záměrem výzkumu tedy bylo zjistit, jaká je úspěšnost studentů, a to nikoliv sumárně v celé škále poznatků předmětu, ale u zcela specifických a dílčích pravidel.

Materiál a metody

Sledované prvky byly rozděleny do skupin s podobným charakterem. Například mezerování je skupina jevů, kde se sleduje správné uvádění mezer u interpunkce, mezi číslem a jednotkou atd. U zkouškové práce byly sledovány pouze úlohy, v nichž jako zadání byl uveden text se záměrnými chybami a úkolem studenta bylo tyto chyby nalézt a vyznačit.

Pro analýzu byly v první fázi určeny pouze tři skupiny prvků. První skupinou byly jevy týkající se uvádění pomlčky a spojovníku. Ve druhé skupině bylo sledováno mezerování a poslední skupina obsahuje pravidla jazyková. Chyby náležející do poslední skupiny nejsou obsaženy ve výukovém obsahu předmětu, jde například o záměny písmen nebo nesprávně zapsané slovo (např. „ůkol“). Proto má tato skupina charakter kontrolní – předpokládáme, že chyby tohoto typu jsou studenti bez jakéhokoliv zaváhání schopni rozpoznat, otázkou pouze je, jak jsou pozorní.

Ze zkouškových prací byly vybrány dvě úlohy s podobně dlouhým zadáním s přibližně stejným počtem chyb a byly zaznamenány všechny jednotlivé případy odhalení či neodhalení příslušné chyby.

Pro ilustraci uvádíme text první ze dvou vybraných úloh (obr. 1).

Základní charakteristikou, která byla sledována, byla úspěšnost odhalení jednotlivých chyb. Úspěšnost byla stanovena jako poměr počtu odhalených chyb a všech chyb v dané skupině. Každá skupina měla 3 výskyty chyby, výjimkou byla skupina pomlček u druhé úlohy, kde byly pouze dva výskyty. Pro zjištění, zda úspěšnost odhalení chyby jedné skupiny nějak souvisí s odhalením chyby jiné skupiny, byly vypočteny korelační koeficienty porovnávající úspěšnosti skupin.

Návod k použití: Před rozbalením si připravte misku s 5dl vody, do níž vlijte dvě lžičky 5 % roztoku octa. Udržujte teplotu kolem 20 °C. Rozbalte velký sáček-označený modře-a nasypte do misky. Za stlého míchááání vsypte malého obsah sáčku (čereného). Po dokonalém rozpuštění nechte 10-15 minut ustát. Firma H&R zaručuje účinnost roztoku na papíry FK-11 po dobu min.5 měsíců.

Obrázek 1: Zadání úlohy

Tabulka 1: Úspěšnosti odhalení chyb v jednotlivých skupinách

Skupina	**Úspěšnost** [%]
Pomlčky	50,5
Mezerování	69,8
Jazyk	75,1

Výsledky

Bylo analyzováno celkem $N = 139$ prací. Úspěšnosti jednotlivých skupin uvádí tab. 1.

Pro doplnění uvádíme ještě absolutní a relativní počty prací, u nichž bylo dosaženo 100% úspěšnosti v některé skupině:

Počet prací, které vykazovaly 100% úspěšnost ve všech třech kategoriích, byl 14 (tj. 10 %). Naopak zcela chybné práce (s nulovou celkovou úspěšností) byly jen 4 (2,9 %).

Korelační koeficienty vypočtené mezi vektory úspěšností vždy dvou skupin uvádí tab. 2.

Diskuse

Jak je patrné z výsledků, pomlčky a jejich správný zápis dělá studentům největší potíže z uvedených tří sledovaných skupin chyb. Lze konstatovat, že polovina

Tabulka 2: Stoprocentní úspěšnosti

Skupina	**Počet prací**	**Relativně** [%]
Pomlčky	28	20,1
Mezerování	64	46,0
Jazyk	70	50,4

Tabulka 3: Korelace mezi úspěšnostmi skupin

Dvojice	**Korelační koeficient**
Pomlčky–Mezerování	0,274
Pomlčky–Jazyk	0,205
Mezerování–Jazyk	0,298

Tabulka 4: Souhrn rozhodnutí o závislosti jevů

Korelace	r	T	$\alpha = 0{,}01$	$\alpha = 0{,}05$
Pomlčky–Mezerování	0,274	3,347	ano	ano
Pomlčky–Jazyk	0,205	2,452	ne	ano
Mezerování–Jazyk	0,298	3,654	ano	ano

chybně zapsaných případů zůstala neodhalena a koresponduje to také s četností chyb v pomlčkách v závěrečných pracích. Na podporu tohoto tvrzení sice neexistuje bohužel žádný dostatečně rozsáhlý výzkum, ale odpovídá to zkušenostem nejen autora, ale i kolegů sledujících kvalitu závěrečných prací u státních zkoušek.

O něco lépe dopadla skupina chyb sledujících správné mezerování. S téměř 70% úspěšností jsou studenti schopni tyto chyby detekovat.

V kontrolní skupině jazykových chyb dosáhla úspěšnost tří čtvrtin. Jde o chyby, které jsou buď systematicky vyučovány již od prvního stupně základní školy, nebo o překlepy, jejichž nesprávnost je zcela zřejmá. To, že se úspěšnost v této kategorii neblíží 100 %, lze tedy přičítat jiným faktorům než neznalosti daného problému. Tyto faktory pravděpodobně působí ve všech kategoriích podobným způsobem.

Z toho je tedy patrné, že liší-li se úspěšnost ve skupině mezerování pouze o zhruba pět procentních bodů od skupiny jazykové, je znalost chyb v mezerování již osvojena relativně velmi dobře.

Korelace mezi jednotlivými skupinami otestujeme testovým kritériem se Studentovým rozdělením: $T = r \cdot \sqrt{\frac{N-2}{1-r^2}}$. V tabulce 4 jsou shrnuty výsledky – v posledních dvou sloupcích je hodnotou „ano“ indikována závislost, hodnotou „ne“ pak nezávislost jevů.

Na hladině významnosti $\alpha = 0{,}01$ je tabelovaná hodnota $t_{1-\alpha/2}(N-2)$ rovna 2,576 a na hladině významnosti $\alpha = 0{,}05$ pak 1,96.

Lze tedy úspěšnost odhalení chyby jedné skupiny považovat za slabě až středně závislou na odhalení chyby ve skupině jiné.

Můžeme si nyní položit otázku, proč chyby v psaní pomlček vykazují v četnosti znatelný rozdíl oproti chybám v mezerování. Jedním z faktorů může být, že způsob psaní pomlček (spojovníků) je relativně novým prvkem, který je spojen až s počítačovou technologií sazby, zatímco mezerování je v základních obrysech v podstatě beze změny převzato už z technologie psacích strojů. Jde o celkové

povědomí, které se utváří dlouhá léta a je i v dnešní době závislé na dřívějších technologiích (psací stroje). Uvážíme-li, že studenti jsou masově ovlivňováni texty pořizovanými laickou komunitou (do níž bohužel musíme započítat i značnou část učitelů všech stupňů škol), je jejich citlivost na tyto chyby značně otupělá. O spojovníku a pomlčce není například ani zmínka ve výuce pravopisu českého jazyka. Oproti tomu i laický uživatel textového procesoru nebo učitel na základní škole tuší, že například za interpunkcí se mezera píše, zatímco před ní ne, většinou tedy pozitivně ovlivňuje čtenáře správným zápisem, i když širší povědomí o typografických a jazykových pravidlech nemá.

Druhou podstatnou otázkou je, zda kvalita výuky v předmětu Zpracování textů na počítači je dostatečná na to, aby studenti byli na závěr schopni reagovat na chyby tohoto druhu. Je samozřejmě potřebné vzít v úvahu i skutečnost, že předmět má daleko širší záběr, může se tedy snadno stát, že některé prvky zaniknou mezi ostatními. Autor se však spíše přikládní k názoru, že padesátiprocentní úspěšnost odhalení chyby v zápisu pomlčky je malá a o nadstandardní kvalitě výuky či přípravy studentů to nesvědčí.

Tyto odpovědi by si samozřejmě zasloužily potvrzení či vyvrácení podrobnějším výzkumem. Bylo by rovněž vhodné porovnat výstupní znalosti s počátečním stavem, a zjistit tak přidanou hodnotu ve studiu daného předmětu. Jistě by také byla zajímavá i analýza časové řady – za 18 let výuky je k dispozici poměrně značné množství dat.

Závěr

Výsledky získané z prvotní analýzy mohou být interpretovány jako stimul pro zvětšení úsilí ve výuce zpracování textů. Podobnou analýzu by bylo vhodné provést pro více skupin sledovaných jevů – východiskem by přitom mohla být i provedená analýza formální kvality závěrečných prací odevzdávaných na fakultě. Pozornost by si rovněž zasloužil výzkum časové řady, pro nějž existuje v této chvíli poměrně značné množství podkladových dat.

Všechny uvedené možnosti by měly vyústit ve zvýšení kvality výuky v předmětu Zpracování textů počítačem, a tím i ve zvýšení kvality závěrečných prací.

Odkazy

1. BORŮVKOVÁ, Jana. *Jak napsat bakalářskou práci*. Brno: FRRMS MENDELU, 2010.
2. CVINGRÁFOVÁ, Eliška. *Počítače v chemii: Příloha bakalářské práce* [online]. Brno, 2011 [cit. 2018-05-12]. Dostupné z: `http://profesornovotny.webnode.cz/_files/200000061-50a70530ea/Z%C3%A1klady%20typografie.pdf`. Masarykova univerzita, Pedagogická fakulta, Katedra chemie.

3. *ČSN 01 6910: Úprava dokumentů zpracovaných textovými procesory.* Praha, 2014. Technická norma. Ústav pro normalizaci a měření.
4. RACLAVSKÝ, Jiří. *Nejdůležitější pravidla úpravy odborného textu* [online] [cit. 2018-05-15]. Dostupné z: `http://www.phil.muni.cz/fil/vyuka/uprava_odb_textu.html`.
5. ŠOŠOLÍK, Ivan. *Typografická pravidla* [online] [cit. 2018-05-15]. Dostupné z: `http://www.zsjablunka.cz/html/vyuka/informat/text/typograf.pps` Prezentace ve formátu PPS.
6. *Typografická pravidla* [online]. 2014 [cit. 2018-05-22]. Dostupné z: `http://www.spskarvina.cz/www/images/stories/files/2011/skola/2011/typ_pravidla.docx` Dokument formátu DOCX.
7. *Typografie – základy* [online] [cit. 2018-05-15]. Dostupné z: `http://www.jardaz.cz/articles.php?article_id=66`.

Summary: The Results of Teaching Text Processing

Teaching of text processing is introduced at the Faculty of Business and Economics of Mendel University in Brno as an optional subject for more than 18 years. The concept of the subject has over time somewhat changed from the initial more technical concept to the current one focused on more general knowledge of typography and on technical texts, especially the final works. The article deals with the outline of the analysis of the learning outcomes in this subject, by processing selected exam results work.

Jiří Rybička
rybicka@mendelu.cz

The Unreasonable Effectiveness of Pattern Generation

Petr Sojka, Ondřej Sojka

Languages are constantly evolving, and so are their hyphenation rules and needs. The effectiveness and utility of TEX's hyphenation have been proven by its usage in almost all typesetting systems in use today. The current Czech hyphenation patterns were generated in 1995, and no hyphenated word database was freely available.

We have developed a new Czech word database and have used the `patgen` program to generate new effective Czech hyphenation patterns efficiently and evaluated their generalization qualities. We have achieved full coverage on the training dataset of 3,000,000 words, and developed a validation procedure of new patterns for Czech based on the testing database of 105,000 words approved by the Czech Academy of Science linguists.

Our pattern generation case study exemplifies a practical solution to the widespread dictionary problem. The study has proven the versatility, effectiveness, and extensibility of Liang's approach to hyphenation developed for TEX. The unreasonable effectiveness of the pattern technology has led to applications that are and will be used, even more widely now, nearly 40 years after its inception.

Keywords: `patgen`, hyphenation patterns, unreasonable effectiveness, Czech

> ... the best approach appears to be to embrace the complexity of the domain and address it by harnessing the power of data: if other humans engage in the tasks and generate large amounts of unlabeled, noisy data, new algorithms can be used to build high-quality models from the data. (Peter Norwig, [1])

Introduction

In their famous essays, Wigner [2], Hamming [3] and Norwig [1] consider mathematical and data-driven approaches to be miraculously, unreasonably effective. One of the very first mathematically founded approaches that harnessed the power of data was Franklin Liang's language-independent solution for TEX's hyphenation algorithm [4], and his program `patgen` for a generation of hyphenation patterns from a word list.

Dictionary problem The task at hand was a *dictionary problem.* A dictionary is a database of records; in each record, we distinguish the key part (the word) and

DOI: 10.5300/2019-1-4/73

the data part (its division). Given an already hyphenated word list of a language, a set of *patterns* is magically generated. Hyphenation patterns are much smaller than the original word list and typically encode almost all hyphenation points in the input list without mistakes. Liang's pattern approach thus could be viewed as an efficient lossy, ideally lossless, *compression* of the hyphenated dictionary with a compression ratio of several orders of magnitude.

It has been proven [5, chapter 2] that the optimization problem of exact lossless pattern minimization is non-polynomial by reduction to the minimum set cover problem.

Generated patterns have minimal length, e.g., shortest context possible, which results in their *generalization* properties. Patterns could hyphenate words not seen during learning: yet another miracle of the generated patterns.

Pattern preparation In the 36 years of `patgen` use, there have been hundreds of hyphenation patterns created, either by hand or *generated* by the program `patgen`, or by the combination of both methods [6]. The advantage of *pattern generation* is that one can fine-tune pattern qualities for specific usage. Having an open-source and maintained word list adds another layer of flexibility and usability to the deployment of patterns. This approach is already set up for German variants and spellings [7] and was an inspiration for doing the same for the Czech language.

In this paper, we report on the development of the new Czech word list with a free license and complementary sets of hyphenation patterns. We describe the iterative process of initial word list preparation, word form collection, estimation of pattern generation parameters, and novel applications of the technology.

> Hyphenation is neither anarchy nor the sole province of pedants and pedagogues. Used in moderation, it can make a printed page more visually pleasing. If used indiscriminately, it can have the opposite effect, either putting the reader off or causing unnecessary distraction. (Major Keary)

Initial word list preparation

As a rule of thumb, the development of a large new hyphenated word list starts with a small dataset. The experience and outputs from this initial phase, e.g., hyphenation patterns, are then applied to the larger and larger lists.

Bootstrapping idea As word lists of a well-established language are sizeable and manual creation of a huge hyphenated word list is tedious work, we used the bootstrapping technique. We illustrate the process of initial word list preparation in the diagram in Figure 1 on the facing page. We have obtained a hyphenated

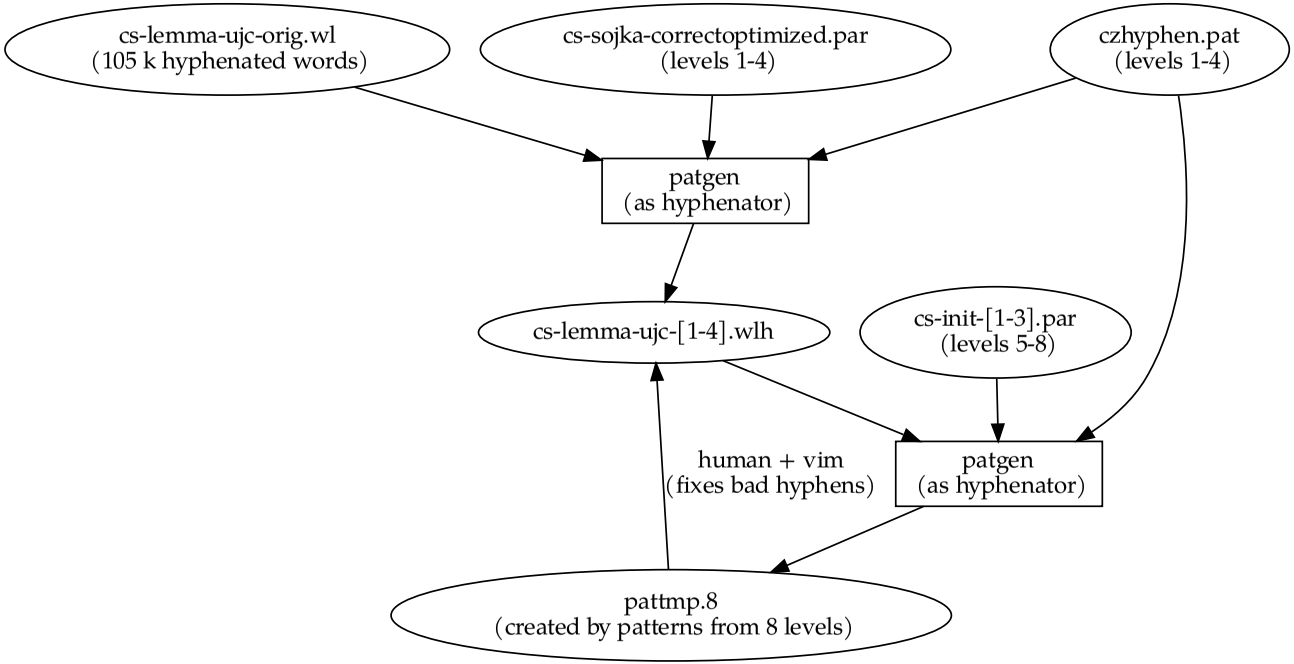


Figure 1: Life cycle of initial word list preparation, illustrated with the development of 105 k Czech consistently hyphenated words. `czhyphen.pat` represents the original Czech hyphenation patterns from [8] and `cs-sojka-correctoptimized.par` are correct optimized `patgen` parameters from the same paper. `cs-init-[1-3].par` are custom parameters that trade off bad hyphens (which have to be manually checked) for missed hyphens. Information on which hyphenations `patgen` missed, and where it wrongly inserted a hyphen is sourced from `pattmp`.

word list with 105,244 words from the Czech Academy of Sciences, Institute of the Czech Language (ÚJČ). Upon closer inspection, we discovered many problems with the data, probably stemming from the fact that it has been crafted by multiple linguists over many years. The few hyphenation rules [9] that are in the Czech language are not applied consistently. The borderline cases were typically between syllabic (ro-zum) and etymological variants (roz-um) of hyphenation, or the way of handling words borrowed from German or English into Czech. There are sporadic examples of words, where correct syllabification depends on the semantics of the word: *narval* and *oblít* are two examples of them in Czech. These are preferably not to be hyphenated, to stay on the safe side.

It is impractical to try to manually find inconsistencies and systemic errors, even in a relatively short word list like this. We slightly modified and extended the process suggested in [10, page 242]: We used `patgen` and the current Czech patterns to hyphenate the word list and manually checked only the 25,813 words, where the proposed hyphenation points differed from the official (were bad or missed), creating a new word list `cs-lemma-ujc-1.wlh` [11] in the process.

However, we are erroneous humans making mistakes. To find these mistakes, we have used `patgen` to generate the four additional levels of hyphenation patterns on top of the current patterns from the checked word list. We have also adjusted the parameters (see `cs-init-[1-3].par` [11]) used for generation of the four additional levels to trade off bad hyphens (which have to be manually checked) for missed ones. We have then used these patterns, with eight levels in total, to hyphenate the checked word list and manually rechecked the wrongly hyphenated points (dots in `patgen` output), with missed hyphenation points (implicitly marked as the hyphen sign in hyphenated word list). We have repeated this process three times, iterating on `cs-lemma-ujc-[2-4].wlh`. Word list number four is used for the generation of bootstrapping patterns and final pattern validation.

Word list preparation and design

Any live language continually changes, and Czech is no exception. Many new Czech words now come from other languages, mostly from English. It presents a challenge for the patterns; they must not only correctly hyphenate Czech words according to Czech syllabic boundaries, but foreign words must be hyphenated correctly too, according to their new Czech syllabic pronunciation [12]. To have the patterns keep up with language evolution, we must maintain not only the patterns but also a hyphenation word list. In this section, we detail how we have built such a word list.

csTenTen corpus We have first obtained a word list with frequencies, generated from the Czech Web Corpus of TenTen family (csTenTen) [13]. We then

filtered this word list to include only words that appear more than ten times in two crawls [14] made in years 2012 and 2017. We ended up with a word list containing 922,216 words, a non-negligible fraction of which are misspellings and jargon.

Word list cleanup We have then cleaned this word list by using the Czech morphological analyzer `majka` [15] to remove all words not known to it. We removed 370,291 typos, misspellings, and similar atypical lexemes, and we kept only 551,925 frequently occurring valid words in the dataset.

Word list expansion The morphological analyzer `majka` [15] also allows us to expand words into all their inflected forms. We chose not to use the expansion feature of `majka`, because the word list would grow to 3,779,379 (almost a fourfold increase) and csTenTen already contains most of the commonly used types of inflections. It would also distort which hyphenation `patgen` gives the most weight to. We tried supplying logarithms of word frequencies from csTenTen to the word list, so more weight could be given to patterns that cover the most common words. It did not significantly improve validation scores in our case, as one can see in Table 3 on page 81. We think that this is partly because `patgen` is limited to one digit of frequency per word and partly because the validation score (computed from error rate on `ujc` word list) does not capture real-world usage.

We expanded the word list with `majka` by adding 54,569 lemmas (base forms) that were present in the word list, but not in their base form. It increased the word list size to 606,494 words.

We list the statistics of word lists used in pattern generation in Table 1.

Table 1: Czech word list shortcut names and statistics

shortcut	word list description	count
`ujc`	checked word list for validation	105,244
`all`	all frequent word forms from web known to `majka` plus all lemmas known to `majka`	606,494
`allflex`	previous plus all word forms generated by majka	2,100,581
`allflexjargon`	previous plus all non-standard and jargon word forms	3,779,379
`biggest`	tokens that are present in the csTenTen more than 10 times	3,918,054

Maintenance The German *wortliste* [7] project served as inspiration for our open word list format, detailed in the `README.md` [11].

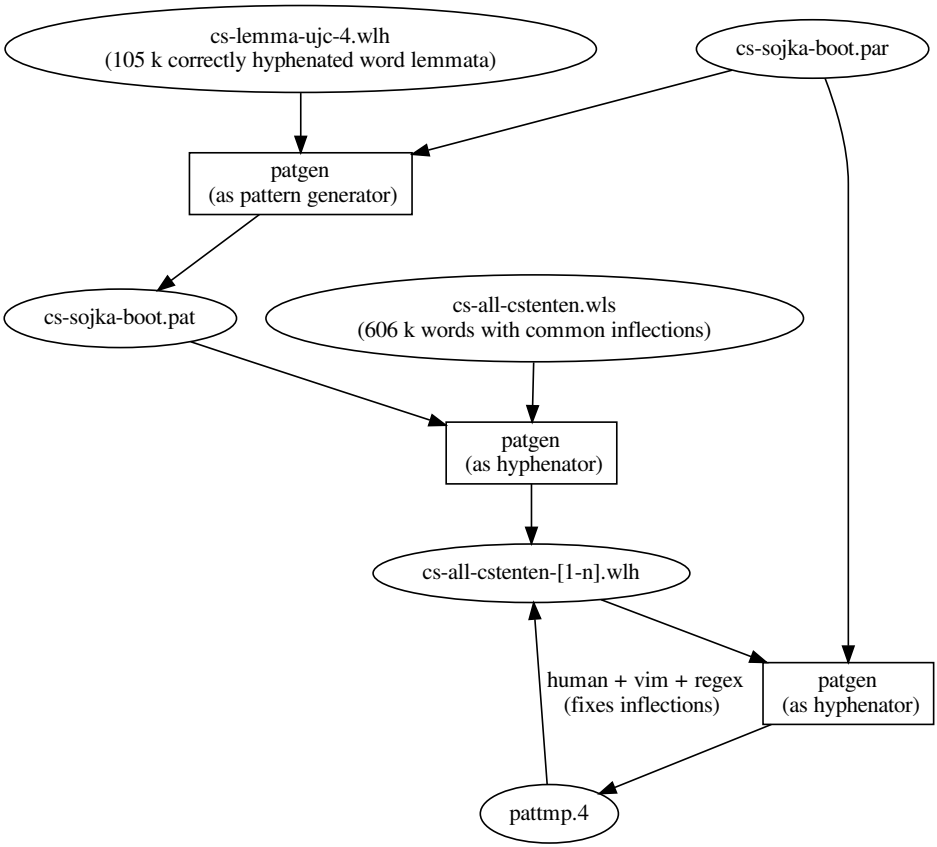


Figure 2: How we bootstrapped hyphenation of the big word list by training patterns (`cs-sojka-boot.pat`) on the small word list and applying them to the big one. `cs-sojka-boot.par` are `patgen` parameters that are designed to generate many patterns but still retain their generalization properties. `pattmp` highlights which hyphenation points in the source file the new pattern level missed, which were correctly covered and where they wrongly put a hyphen.

One must regard the hyphen as a blemish to be avoided wherever possible. (Winston Churchill)

Bootstrapping — iterative development of hyphens in the big word list

It would be tedious to hyphenate such a big word list by hand manually, so we train patterns on a small list and apply them to the big word list, as illustrated in Figure 2 on the preceding page. Then, we train patterns on the (now hyphenated) big word list and have `patgen` show what it would have hyphenated differently. With this approach, we cherry-pick inconsistencies in the word list.

Since the big word list contains not only lemmas of words, but also characteristic inflections, we use regular expressions to add hyphens around them and fix inconsistencies. We keep iterating on this, as shown in Figure 2 on the facing page, until the patterns, generated with `cs-init-[1-3].par` [11], achieve nearly perfect coverage.

The resulting patterns hyphenate according to the standard Czech hyphenation rule: hyphenation is allowed whenever it does not change the pronunciation of the word. Thanks to the effectiveness of pattern generation, this works not only for Czech words but also for foreign (Latin, French, German, English) ones.

Hyphens, like cats, are capable of arousing tenderness or shudders. (Pamela Frankau)

Pattern generation

The last Czech hyphenation patterns were generated in 1995 [8], and are in use not only in TEX but also in other widespread typesetting systems. For conservative users, there is no strong incentive for change, because the error rate is relatively low (the first version of the validation set measured an error rate around 4%), and coverage is relatively high (the first version of the validation set measured around 7% missed hyphenation points).

Pattern generation from 3,000,000 words does not take hours as it did two decades ago, but seconds, even on commodity hardware, which allows for rapid development of "home-made" patterns.

In each training pass, `patgen` generates all possible substrings of every word in its dictionary with length higher than or equal to the parameter `pat_start` but lower than `pat_finish`. Then, for each generated pattern, it counts how many hyphenation points would be correctly covered by the pattern and how many hyphenation points would the patterns wrongly identify. Statistics of Good, Bad and Missed represent the count of points, where the patterns correctly identified

a hyphen, where the patterns expected a hyphen but there was none, and the count of hyphenation points that the patterns did not identify.

We have developed a Python wrapper for `patgen` that we use in Jupyter notebooks. It allows rapid iteration, and easy sharing of results — see Table 2 and `demo.ipynb` [11].

Had Liang in 1983 had the same ease of changing `patgen` parameters, running it, and seeing the results in 60 seconds, he would have inevitably generated higher than 89% coverage while staying within the limit of 5,000 patterns [4, page 37].

Table 2: Outputs from running `patgen` in our Jupyter notebook with two different parameter sets. The first parameter set is from the German Trennmuster project [7] and generates 7,291 patterns, 40 kB. The second one from [8] generates shorter and smaller patterns — 4,774 patterns, 25 kB.

Level	Patterns	Good	Bad	Missed	Lengths
1	750	1,683,529	525,670	0	1 5
2	3,178	1,628,874	38	54,655	2 6
3	2,548	1,683,528	9,931	1	3 7
4	1,382	1,683,287	0	242	4 8
5	92	1,683,528	0	1	5 9
6	0	1,683,528	0	1	6 10
7	1	1,683,529	0	0	7 11

Level	Patterns	Good	Bad	Missed	Lengths
1	1,608	1,655,968	131,481	27,561	1 3
2	1,562	1,651,840	2,533	31,689	1 3
3	2,102	1,683,528	2,584	1	2 5
4	166	1,683,135	6	394	2 5

It has also become common to use a validation dataset to ensure generalization abilities. Our usage of a validation dataset has proven useful. Table 3 shows that if we were to use the *correct optimized* parameters from [8] that have been in use for Czech, we would overfit on the training dataset and perform *worse* than their *size optimized* counterparts. The validation word list has to be carefully checked with linguists from UJČ for consistency to minimize the generalization error. Most of the current errors stem from foreign words used in Czech texts.

When the validation word list is added to training, then patterns could be developed to serve as a lossless compression of word list dataset and thus maximize

Table 3: Effectiveness and effectivity of pattern generation on Czech word lists. Comparison of validation scores of patterns trained on various word list and parameter combinations. Percentages sum up to more than 100%, because they are calculated by dividing the count of Good, Bad or Missed hyphens by the total word count.

Word list	Params	Good %	Bad %	Missed %	Size	Patterns	Time (s)
`all`	correctopt [8]	99.76	2.94	0.24	30 kB	5,593	58.13
	sizeopt [8]	98.95	2.80	1.05	19 kB	3,816	59.46
	german [7]	99.74	2.21	0.26	51 kB	8,991	201.9
weighted `all`	correctopt [8]	99.76	2.94	0.24	30 kB	5,590	59.23
	sizeopt [8]	98.95	2.80	1.05	20 kB	3,821	58.74
	german [7]	99.74	2.21	0.26	51 kB	8,978	207.35
`allflex`	correctopt [8]	99.46	4.02	0.54	28 kB	5,387	212.55
	sizeopt [8]	99.26	3.72	0.74	29 kB	5,537	212.59
	german [7]	99.42	3.35	0.58	49 kB	8,663	1,035.16
`allflexjargon`	correctopt [8]	99.47	4.08	0.53	29 kB	5,612	365.96
	sizeopt [8]	99.31	3.78	0.69	31 kB	5,938	369.92
	german [7]	99.43	3.36	0.57	53 kB	9,308	1,786.4

the effectiveness of the pattern technology. See results with parameter set *german* for an example of almost lossless compression.

> Life is the hyphen between matter and spirit. (Augustus William Hare)

The unreasonable effectiveness

We were able to solve the dictionary problem for Czech hyphenation effectively.

Space effectiveness From 3,000,000+ hyphenated words stored in approximately 30,000,000 bytes we have produced patterns of size 30,000 bytes, achieving roughly 1000× space *lossless* compression.

Time effectiveness Using the trie data structure for patterns makes the time complexity of accessing the record related to the word, e.g. hyphenation point, very low *constant* time. The constant is related to the depth of the pattern trie data structure, e.g. 5 or 6 in the case of Czech. If the entire pattern trie resides in RAM, the time for finding the patterns for a word is on the scale of tens, at most hundreds, of processor instructions. Word hyphenation throughput is then about 1,000,000 words per second on a modern CPU.

Optimality Even though finding exact space and time-optimal solutions is not feasible, finding an approximate solution close to optimum is possible. Heuristics and insight expressed above, together with interactive fine-tuning of `patgen` parameter options, allows for rapid pattern development.

Automation A close-to-optimal solution to the dictionary problem could be useful not only for Czech hyphenation, but for all other languages [6, 16], and more generally, for other instances of the dictionary problem. Heuristics for thresholding `patgen` pattern generation parameters could be based on a statistical analysis of large input datasets. It could allow the deployment of presented approaches on a much broader problem set and scale. We believe that parameters could be approximated *automatically* from the statistics of the input data.

Pattern generation — in Wigner's words — "has proved accurate beyond all reasonable expectations". Let us paraphrase another one of his quotes:

> The miracle of the appropriateness of the language of ~~mathematics~~ *patterns* for the formulation of the laws of ~~physics~~ *data* is a wonderful gift which we neither understand nor deserve. We should be grateful for it and hope that it will remain valid in future research and that it will extend, for better or for worse, to our pleasure, even though perhaps also to our bafflement, to wide branches of learning.

"We should stop acting as if our goal is to author extremely elegant theories, and instead embrace complexity and make use of the best ally we have: the unreasonable effectiveness of data." (Peter Norvig, [1])

Conclusion

We have developed a flexible open language–independent system [11] for hyphenation pattern generation. We have demonstrated the effectiveness of this system by updating the old Czech hyphenation patterns [8] and achieving record accuracy. We have also applied recent data and computer science advancements, such as the usage of interactive Jupyter notebooks and a validation dataset to prevent overfitting, to the more than three decades old problem of pattern generation.

Future work

Word lists for other languages The logical next steps will be applying developed techniques to different languages: to Slovak and virtually all others that do not yet have word list hyphenation patterns based on a word list, but for which a word list — either in Sketch Engine or elsewhere — is available.

Stratification Pattern generation could be further sped up by several techniques, such as the stratification of word lists on the level of input, or on the level of counting pro and con examples for including a new pattern or not.

Pattern-encoded spellchecker We have a big dictionary of frequent spelling errors from the csTenTen word list. Nothing prevents us from encoding these into specific patterns or pattern layers with extra levels and using that information during typesetting, e.g. to typeset misspelled words with a red underline in LuaTEX. LuaTEX allows dynamic pattern loading and Lua programming that enables the implementation of this feature, which people are used to having in editors.

Word segmentations Recent progress in machine-learned natural language processing and machine translation builds on subword representations and various types of semantically coherent sentence or word segmentations. As tokenization and segmentation are at the beginning of every natural language processing pipeline, there is a demand for effective and efficient universal segmentation [17]. New neural machine translation systems are capable of open-vocabulary translation by representing rare and unseen words as a sequence of subword units [18, Table 1]. Segmentation is crucial, especially for compositional languages like German, where there are many compounds (mostly out of vocabulary words),

and for morphologically rich languages like Hebrew [19] or Arabic, that need to be segmented, represented, and translated.

Pattern-based learnable key memories Solutions to variations of the dictionary problem are a hot topic of leading-edge research to design memory data architectures like those used in machine learning of language [20]. Pattern-based memory network architectures could speed up language data access in huge neural networks considerably.

Multilingual hyphenation patterns Given that there are close languages with syllabic-based rules like Czech and Slovak, generating patterns from merged word lists is straightforward. It would save energy on low-resource devices like e-book readers by having them load fewer patterns at a time.

Acknowledgments The authors thank the CSTUG and the TEX Users Group who financially supported the presentation of the project at TUG 2019. We owe our gratitude also to Vít Suchomel of Lexical Computing for word lists from Sketch Engine, to Pavel Šmerk, Frank Liang and Don Knuth for `majka`, `patgen` and TEX, respectively. Thanks go to Vít Novotný and Pavel Šmerk for valuable comments to the paper. Thanks TUG for the permission to reprint the paper from *TUGboat*, with minor extensions, and Don Knuth for discussion after the presentation of the paper at TUG 2019.

References

1. PEREIRA, Fernando; NORVIG, Peter; HALEVY, Alon. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*. 2009, vol. 24, no. 02, s. 8–12. ISSN 1541-1672. Dostupné z DOI: `10.1109/MIS.2009.36`.
2. WIGNER, Eugene P. The Unreasonable Effectiveness of Mathematics in the Natural Sciences. Richard Courant Lecture in Mathematical Sciences delivered at New York University, May 11, 1959. *Communications on Pure and Applied Mathematics*. 1960, vol. 13, no. 1, s. 1–14. Dostupné z DOI: `10.1002/cpa.3160130102`.
3. HAMMING, R. W. The Unreasonable Effectiveness of Mathematics. *The American Mathematical Monthly*. 1980, vol. 87, no. 2, s. 81–90. ISSN 00029890, 19300972. ISSN 00029890, 19300972. Dostupné také z: `https://www.jstor.org/stable/2321982`.
4. LIANG, Franklin M. *Word Hy-phen-a-tion by Com-put-er*. 1983. Dostupné také z: `https://tug.org/docs/liang/`. Disertační práce. Stanford University.

5. SOJKA, Petr. *Competing Patterns in Language Engineering and Computer Typesetting*. 2005. Disertační práce. Faculty of Informatics.
6. REUTENAUER, Arthur; MIKLAVEC, Mojca. *TEX hyphenation patterns* [online]. TUG [cit. 2019-11-14]. Dostupné z: `https://tug.org/tex-hyphen/`.
7. LEMBERG, Werner. *A database of German words with hyphenation information*. Dostupné také z: `https://repo.or.cz/wortliste.git`.
8. SOJKA, Petr; ŠEVEČEK, Pavel. Hyphenation in TEX — Quo Vadis? *TUGboat*. 1995, vol. 16, no. 3, s. 280–289.
9. *Internetová jazyková příručka (Internet Language Reference Book)* [online]. Institute of Czech language, Czech Academy of Sciences [cit. 2019-07-18]. Dostupné z: `http://prirucka.ujc.cas.cz/?id=135`.
10. SOJKA, Petr. Hyphenation on Demand. *TUGboat*. 1999, vol. 20, no. 3, s. 241–247. `https://tug.org/TUGboat/tb20-3/tb64sojka.pdf`.
11. SOJKA, Ondřej; SOJKA, Petr. *cshyphen repository*. Dostupné také z: `https://github.com/tensojka/cshyphen`.
12. SOJKA, Petr. Notes on Compound Word Hyphenation in TEX. *TUGboat*. 1995, vol. 16, no. 3, s. 290–297.
13. JAKUBÍČEK, Milos; KILGARRIFF, Adam; KOVÁŘ, Vojtěch; RYCHLÝ, Pavel; SUCHOMEL, Vít. The TenTen Corpus Family. In: *Proc. of 7th International Corpus Linguistics Conference (CL)*. Lancaster, 2013, s. 125–127.
14. SUCHOMEL, Vít; POMIKÁLEK, Jan. Efficient Web Crawling for Large Text Corpora. In: KILGARRIFF, Adam; SHAROFF, Serge (eds.). *Proc. of the seventh Web as Corpus Workshop (WAC)*. Lyon, 2012, s. 39–43. Dostupné také z: `https://sigwac.org.uk/raw-attachment/wiki/WAC7/wac7-proc.pdf`.
15. ŠMERK, Pavel. Fast Morphological Analysis of Czech. In: SOJKA, Petr; HORÁK, Aleš (eds.). *Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2009*. Karlova Studánka, Czech Republic: Masaryk University, 2009, s. 13–16. ISBN 978-80-210-5048-8. Dostupné také z: `http://nlp.fi.muni.cz/raslan/2009/`.
16. SCANNELL, Kevin Patrick. Hyphenation patterns for minority languages. *TUGboat*. 2003, vol. 24, no. 2, s. 236–239.
17. SHAO, Yan; HARDMEIER, Christian; NIVRE, Joakim. Universal Word Segmentation: Implementation and Interpretation. *Transactions of the Association for Computational Linguistics*. 2018, vol. 6, s. 421–435. Dostupné z DOI: `10.1162/tacl_a_00033`.
18. SENNRICH, Rico; HADDOW, Barry; BIRCH, Alexandra. Neural Machine Translation of Rare Words with Subword Units. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers). Berlin, Germany: Association for Computational Linguistics, 2016, s. 1715–1725. Dostupné z DOI: 10.18653/v1/P16-1162.
19. ZELDES, Amir. A Characterwise Windowed Approach to Hebrew Morphological Segmentation. In: *Proc. of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Brussels, Belgium: Association for Computational Linguistics, 2018, s. 101–110. Dostupné z DOI: 10.18653/v1/W18-5811.
20. LAMPLE, Guillaume; SABLAYROLLES, Alexandre; RANZATO, Marc'Aurelio; DENOYER, Ludovic; JÉGOU, Hervé. *Large Memory Layers with Product Keys* [online]. 2019 [cit. 2019-07-18]. Dostupné z arXiv: 1907.05242 [cs.CL].

Nepochopitelná efektivita generování vzorů dělení slov

Jazyky se vyvíjí a spolu s nimi i jejich potřeby a pravidla dělení slov. Mechanismus vzorů dělení slov v TEXu převzala většina dnešních sazebních systémů, což prokazuje jeho efektivitu a užitečnost. Současné vzory dělení slov pro češtinu ale vznikly v roce 1995, kdy ještě neexistovala žádná volně šiřitelná databáze slov.

Vyvinuli jsme novou českou databázi slov, použili jsme program `patgen` k vygenerování nových efektivních vzorů dělení slov pro češtinu a vyhodnotili jsme jejich generalizační schopnosti. Na trénovací datové sadě 3 milionů slov jsme dosáhli plného pokrytí. Dále jsme vyvinuli postup pro validaci nových vzorů dělení slov pro češtinu s využitím databáze 105 tisíc slov schválených lingvisty Akademie věd České republiky.

Naše případová studie generování vzorů dělení slov představuje praktické řešení častého slovníkového problému. Studie dokazuje pružnost, efektivitu a rozšiřitelnost Liangova přístupu k dělení slov vyvinutého pro TEX. Nepochopitelná efektivita mechanismu vzorů dělení slov dala vzniknout aplikacím, které ho využívají i téměř 40 let po jeho vzniku.

Klíčová slova: `patgen`, vzory dělení slov, nepochopitelná efektivita, čeština

Petr Sojka, Faculty of Informatics, Masaryk University, Brno, Czech Republic and CSTUG, sojka@fi.muni.cz
Ondřej Sojka c/o CSTUG, Nejedlého 1, 638 00 Brno, Czech Republic, ondrej.sojka@gmail.com

ConTEXt Marks

Jano Kula

Přibližně deset let se můžete setkat s označením makrojazyka ConTEXt MkII a MkIV. V článku vysvětlíme, co toto označení znamená a jaké rozdíly označuje.

Klíčová slova: ConTEXt MkI, MkII, MkVI, MkIV, MkIX, MkXI, MkXL, MkLX, LuaMetaTEX

Úvod

ConTEXt, makrojazyk systému TEX (analogicky makrojazyku LATEX), vznikl začátkem devadesátých let 20. století. Se začátkem vývoje LuaTEXu kolem roku 2007 bylo při psaní maker pro ConTEXt potřebné rozlišovat větev, která užívala pdfTEX a XeTEX, od větve, která užívala LuaTEX.

Příkazy se pro bežné uživatele nemění (výjimky se najdou a nové příkazy v MkIV přibyly), ale vnitřní mechanizmy jsou, často radikálně, přepracované. V každém případě je dobré vědět, která varianta je použita, například při reportování chyb nebo při žádostech o radu.

Mark One

Číslo jedna si zpětně vysloužila dnes již neexistující varianta ConTEXtu s vnitřním rozhraním v holandštině.

Mark Two (MkII, mkii)

Varianta Mark II vznikla spolu s variantou Mark IV v době, kdy bylo potřebné rozdělit vývoj na ConTEXt, který využívá pdfTEX nebo XeTEX (MkII), a ConTEXt, který využívá LuaTEX (MkIV).

V současnosti se ve variantě MkII už jenom opravují závažné chyby, vývoj byl ukončen. V rámci revize webu `wiki.contexgarden.net` budou odstraněné i návody pro MkII, protože pro nový projekt uživatel použije stabilní LuaTEX 1.10, a tedy ConTEXt MkIV.

Mark Four (MkIV, mkiv)

Mark IV pracuje s programem LuaTEX a využívá možnosti jazyka Lua (tabulky Lua, nativní kódování UTF-8) v různých oblastech (podpora Opentype fontů,

DOI: 10.5300/2019-1-4/87

komunikace s Metapostem, knihovnami, apod.). Uživatelské rozhraní ConTeXtu se v zásadě nemění. Vnitřně byla makra výrazaně přepracovaná: Lua přebírá komunikaci s okolím (např. font loader), data jsou uložena v tabulkách Lua (data fontů, vstupní strom XML, struktura pomocných `*.tua` souborů, ...), atd. Kód těchto nízkoúrovňových funkcí se nachází v souborech `*.mkiv` a jim odpovídajících pomocných souborech `*.lua`.

V ConTeXtu MkIV může koncový uživatel stále používat příkazy, které najde v stručném manuálu ConTeXt, *An Excursion*, přeloženého i do češtiny. Najdou se ovšem i výjimky, a proto, začínáme-li s ConTeXtem, použijeme aktualizovanou verzi ConTeXt *Mark IV, An Excursion*.

Některé příkazy byly rozšířeny o nové parametry a vznikly i příkazy nové, které v Mark II nenajdete. Buď jsou to příkazy, které využívají Lua (`\startluacode`, `\stopluacode`, `\ctxlua`, ...), nebo příkazy ze zásadně přepracovaných oblastí, např. definice rodin písma a jejich OpenType vlastností (features).

Mark Six (MkVI, mkvi)

MkVI je drobným rozšířením MkIV, které přidává jediné vylepšení: umožňuje psát makra s pojmenovanými parametry. Jinak se v ničem neliší od MkIV.

V prvním řádku souboru musí být uvedeno, že se jedná o soubor užívající variantu MkVI (`% macros=mkvi`). Druhou možností je soubor pojmenovat s koncovkou `*.mkvi`, což je obvyklé v distribuci ConTeXt.

Uvedeme fiktivní makro pro zapisování poznámek s možností uvedení *Data* a *Kategorie*.

```
% macros=mkvi
\def\Poznamka[#parametry]#obsah%
   {\getparameters[Poznamky][Datum=ASAP,Kategorie=,#parametry]
    % vysadíme text, kde už máme definované příkazy, s prefixem
    % Poznamky
    {\bf\PoznamkyDatum} {\it\PoznamkyKategorie} {#obsah}\par}

\starttext
\Poznamka[]{Zařídit dovolenou}
\Poznamka[Kategorie=CSTUG]{Dopsat článek}
\Poznamka[Datum=28. 10. 2019,Kategorie=CSTUG]{Odeslat článek}
\Poznamka[Datum=10. 11. 2019,Kategorie=CSTUG]{Přečíst recenzi}
\stoptext
```

ASAP Zařídit dovolenou
ASAP *CSTUG* Dopsat článek
28. 10. 2019 *CSTUG* Odeslat článek
10. 11. 2019 *CSTUG* Přečíst recenzi

Na rozdíl od LaTeXu jsou v tomto případě hranaté závorky součástí makra, není to nepovinný parametr. Příkaz `\def` je zde primitivem `\def`, jak jej známe z *TEXbook*u. V definici makra můžeme uvést výchozí hodnoty (parametr *Kategorie*). `Poznamky` v příkazu `\getparameters[Poznamky][...]` je libovolný prefix, který při použití bude prefixem každého pojmenovaného parametru (`\PoznamkyDatum`, `\PoznamkyKategorie`). Počet pojmenovaných parametrů není omezen.

Mark Nine (MkIX, mkix)

Verze MkIX, další rozšíření MkIV, umožňuje používat procesní instrukce (`<?lua ...?>`) ve zdrojovém souboru.

```
% macros=mkix
\starttext
\bTABLE
        <?lua for i=1,20 do ?>
    \bTR
        <?lua for j=1,5 do ?>
    \bTD
        cell (<?lua inject(i) ?>,<?lua inject(j)?>)
    \eTD
        <?lua end ?>
    \eTR
        <?lua end ?>
\eTABLE
\stoptext
```

Na rozdíl od vykonání příkazu jazykem Lua během sazby (`\ctxlua...`) se tyto procesní instrukce vykonávají prepocesorem již při načítání souboru a vytváří tak vstupní soubor.

Mark Eleven (MkXI, mkxi)

Mark MkXI je kombinací verze s pojmenovanými parametry (MkVI) a verze s procesními instukcemi (MkIX).

Mark Fourty (MkXL, mkxl) & Mark Sixty (MkLX, mklx)

Po deseti letech vývoje LuaTeX, který měl být v maximální možné míře zpětně kompatibilní s pdfTeXem i původním TeXem (obsahuje stejné primitivní příkazy a přidává jich několik navíc), dosáhl kýženého stavu. Na LuaTeXu jsou

závislá makra ConTEXtu i mnoho nových balíčků LaTEXu a při dalším vývoji bylo potřebné brát ohled na zpětnou kompatibilitu. To znemožňovalo odvážnější experimenty.

Proto začal Hans Hagen v roce 2018 s vývojem nového programu LuaMetaTEX (LMTX, neboli LuaTEX 2.0). LMTX je spojením **T**eXu, **M**etafontu a jazyka **L**ua 5.4. **X** může znamenat XML, ale také e**x**periment, jak je libo. Používá ho pouze makrojazyk ConTEXt. Soubory specifické pro LMTX mají koncovky `*.mkxl` a `*.mklx` a jsou v nich makra, která využívají nové primitivní příkazy LuaMetaTEXu.

MkXL je analogií MkIV a MkLX analogií MkVI (umožňuje psát makra s pojmenovanými parametry).

Závěr

Kdo hodlá nasadit ConTEXt v produkčním prostředí, použije verzi MkIV (MkVI, MkIX), která pracuje nad dlouhodobě stabilním LuaTEXem 1.10.

Kdo rád testuje nebo experimentuje, stáhne si a zkompiluje LuaMetaTEX. Kompilace je jednoduchá a distribuce obsahuje všechny potřebné soubory. Možná dříve, než vyjde tento článek, budou kompilované binární soubory součástí minimální (beta) distribuce ConTEXtu pro nejpoužívanější platformy.

Verze makrojazyka ConTEXt shrnujeme v Tabulce 1. Že jsme některá čísla vynechali nebo přeskočili rovnou na verzi XL a LX? Názvy se musí také dobře vyslovovat a musí i dobře vypadat, v číslovaní logiku nehledejte.

Mimochodem: v distribuci ConTEXt se používá analogická konvence pojmenování souborů i pro Metapost (`*.mpii`, `*.mpiv`, `*.mplx`, `*.mpxl`).

Tabulka 1: Verze makrojazyka ConTEXt

Program	ConTEXt	Rozšíření
pdfTEX	Mark I	
pdfTEX, XeTEX	Mark II	
XeTEX	Mark III	
LuaTEX	Mark IV	
LuaTEX	Mark VI	pojmenované parametry
LuaTEX	Mark IX	procesní instrukce
LuaTEX	Mark XI	pojmenované parametry a procesní instrukce
LuaMetaTEX	Mark XL	
LuaMetaTEX	Mark LX	pojmenované parametry

Články

HAGEN, Hans, 2016a. *LMX Templates* [online] [cit. 2019-12-02]. Dostupné z: http://www.pragma-ade.nl/general/manuals/templates-mkiv.pdf.

HAGEN, Hans, 2016b. MkII–MkIV. *TUGboat* [online]. Roč. 27, č. 2, s. 219–227 [cit. 2019-12-02]. ISSN 0896-3207. Dostupné z: https://www.tug.org/TUGboat/tb27-2/tb87hagen-context.pdf.

HAGEN, Hans, 2019a. ConTeXt *LMTX: Following Up* [online] [cit. 2019-12-02]. Dostupné z: http://www.pragma-ade.nl/general/manuals/followingup.pdf.

HAGEN, Hans, 2019b. ConTeXt LMTX. *TUGboat* [online]. Roč. 40, č. 1, s. 34–37 [cit. 2019-12-02]. ISSN 0896-3207. Dostupné z: https://www.tug.org/TUGboat/tb40-1/tb124hagen-lmtx.pdf.

Manuály

OTTEN, Ton, 2017. ConTeXt *MarkIV: An Excursion* [online] [cit. 2019-12-02]. Dostupné z: http://www.pragma-ade.nl/general/manuals/ma-cb-en.pdf.

OTTEN, Ton; HAGEN, Hans, 1999. ConTeXt*: An Excursion* [online] [cit. 2019-12-02]. Dostupné z: http://www.pragma-ade.nl/general/manuals/mp-cb-en.pdf.

Software

2019. *ConTeXt Standalone* [online] [cit. 2019-12-02]. Dostupné z: https://wiki.contextgarden.net/ConTeXt_Standalone.

2019. *Installing* ConTeXt *LMTX* [online] [cit. 2019-12-02]. Dostupné z: http://www.pragma-ade.nl/install.htm.

Summary: ConTeXt Marks

For approximately ten years now, we recognize the separation of the ConTeXt format into ConTeXt MkII and ConTeXt MkIV. In this article, I will explain the naming and the differences between ConTeXt formats.

Keywords: ConTeXt MkI, MkII, MkVI, MkIV, MkIX, MkXI, MkXL, MkLX, LuaMetaTeX

Jano Kula

Mělo by to fungovat VIII – Kresby

Peter Wilson

V článku jsou diskutovány některé otázky vkládání METAPOSTových obrázků do (pdf)LaTeXového dokumentu. Dále se článek zaměřuje na kreslení obrazců zvaných spidrony a na manipulace s nimi.

Klíčová slova: LaTeX, METAPOST, spidron

> Calm was the day and through the trembling air
> Sweet-breathing Zephyrus did softly play—
> A gentle spirit, that lightly did delay
> Hot Titan's beams, which then did glister fair.
>
> *Prothalamion*
> Edmund Spenser

Cílem tohoto seriálu je ukázat čtenáři krátké kousky kódu, které mohou vyřešit některé z jeho problémů. Doufám, že situaci ještě více nezkomplikuji v důsledku mých chyb. Opravy, poznámky a návrhy na změny budou vždy vítány.

Tento díl se týká (La)TeXu pouze okrajově.

> Nothing in India is identifiable, the mere asking of a question causes it to disappear or to merge into something else.
>
> *A Passage to India*
> E. M. Forster

1. Repríza

V reakci na předchozí díl [2] mi profesor Klaus Lagally ukázal další způsob, jakým lze odstranit přebytečný znak na konci příkazu. V článku jsem ukazoval krátký kód, který fungoval podobným způsobem jako LaTeXová makra s hvězdičkou, přičemž namísto hvězdičky jsem používal otazník. Problémem bylo zjistit, zda se za názvem makra vyskytuje otazník, a v závislosti na tom něco vykonat. Dále bylo třeba případný otazník na vstupu odstranit. V článku jsem ukázal následující řešení.

Z anglického originálu *Glisterings* [1] přeložil Jan Šustek.

doi: 10.5300/2019-1-4/92

```
\makeatletter
\def\otaznik{%
  \@ifnextchar ?{\@otaznikS}{\@otaznikBez}}
\def\@otaznikS#1#2#3{S~otazníkem (#2) a (#3).}
\def\@otaznikBez#1#2{Bez otazníku (#1) a (#2).}
\makeatother
```

Profesor Lagally navrhoval k odstranění otazníku namísto řádku 4 psát

```
\def\@otaznikS?#1#2{S~otazníkem (#1) a (#2).}
```

V obou případech po použití

```
\otaznik?{první}{druhý}
\otaznik{první}{druhý}
```

dostaneme výsledek
S otazníkem (první) a (druhý).
Bez otazníku (první) a (druhý).

> Child! do not throw this book about!
> Refrain from the unholy pleasure
> Of cutting all the pictures out!
> Preserve it as your chiefest treasure!
>
> *A Bad Child's Book of Beasts*
> Hillaire Belloc

2. METAPOST a pdfLaTeX

Program METAPOST slouží k vytváření PostScriptových obrázků. Tyto obrázky můžeme jednoduše vložit do LaTeXového dokumentu s výstupem do formátu dvi. Naproti tomu pdfLaTeX obecně neumí zpracovat PostScriptové soubory. Naštěstí umí zpracovat ty PostScriptové soubory, které byly vytvořeny METAPOSTem, a proto METAPOSTové obrázky mohou být přímo vloženy do dokumentu vytvořeného pdfLaTeXem. Toto však není tak přímočaré, jak by to mohlo být.

Mějme soubor s názvem obr.mp, který obsahuje řekněme tři obrázky. METAPOST vytvoří tři soubory obr.1, obr.2 a obr.3, pro každý obrázek jeden soubor. LaTeXový balíček graphicx však očekává,[1] že PostScriptové soubory vytvořené

[1] Autor článku očekává, že pro vkládání obrázku používáme makro `\includegraphics` balíčku graphicx. Další možností použití METAPOSTu je přeložení souboru obr.mp programem mptopdf namísto programu mpost, čímž se vytvoří soubory obr-1.pdf, obr-2.pdf a obr-3.pdf. Alternativní způsob vkládání METAPOSTových obrázků je pomocí makra `\convertMPtoPDF`, jak je vidět na stránce 63 tohoto čísla Zpravodaje, nebo přímé vložení výše zmíněných pdf souborů pomocí příkazů `\pdfximage`, `\pdfrefximage` a `\pdflastximage` popsané v témže článku. (pozn. překl.)

METAPOSTem mají příponu .mps. Balíček můžeme následujícím způsobem nastavit tak, že se k souborům s neznámou příponou bude chovat, jako by měly příponu .mps.

```
\DeclareGraphicsRule{*}{mps}{*}{}
```

LaTeX (na rozdíl od pdfLaTeXu), nebo přesněji programy jako `dvips` nebo `xdvi`, umí zpracovat zapouzdřené PostScriptové soubory (s příponou .eps). Balíček graphicx můžeme nastavit podobným způsobem.

```
\usepackage{ifpdf}
\ifpdf
  \usepackage{graphicx}
  \DeclareGraphicsRule{*}{mps}{*}{}
\else
  \usepackage{graphicx}
  \DeclareGraphicsRule{*}{eps}{*}{}
\fi
```

Pokud chceme, aby obrázek METAPOSTu mohl být použitý v LaTeXu, potom na začátku METAPOSTového souboru napíšeme

```
prologues := 1;
```

To zařídí, že METAPOST vytvoří zapouzdřené PostScriptové soubory. Zdá se, že bez problémů můžeme nastavit stejné `prologues` také pro použití v pdfLaTeXu.[2]

> A mathematician, like a painter or a poet, is a maker of patterns. If his patterns are more permanent than theirs, it is because they are made with ideas.
>
> *A Mathematician's Apology*
> G. H. Hardy

3. Spidrony

Jednoho dne jsem si slepě procházel *Science News*, když jsem si všiml krátkého článečku Ivarse Petersona o spidronech [3]. Zájemci si mohou vygooglit „spidron" pro více obrázků a informací o spidronech.

[2] Nastavení hodnoty `prologues` souvisí s vkládáním fontů do výsledného souboru s obrázkem. Aktuální verze METAPOSTu umožňuje nastavit různé hodnoty `prologues` pro různá použití. Pro podrobnosti vizte manuál k METAPOSTu. Překladatel během své 15leté práce v METAPOSTu při vkládání tisícovek obrázků makrem `\convertMPtoPDF` do pdf(La)TeXu zatím hodnotu `prologues` nenastavoval a zatím nenarazil na problém. Proto si nemyslí, že by se měli začínající LaTeXisté nastavováním hodnoty `prologues` nějak trápit. (pozn. překl.)

Spidrony objevil a pojmenoval maďarský návrhář a grafik Dániel Erdély, když si pohrával se šestiúhelníky. Spidrony jsou tvořeny stále menšími navazujícími rovnostrannými a rovnoramennými trojúhelníky.

Uvědomil jsem si, že spidrony můžu nakreslit v METAPOSTu a po několika pokusech jsem napsal následující METAPOSTová makra pro kreslení spidronů.

```
%% soubor semispid.mp
% MetaPostové makro k nakreslení semispidronu
% semispid(střed, vrchol, iterace, barva1, barva2, směr)
def semispid(suffix $$, $)%
            (expr iter, shadea, shadeb, clock) =
  if clock: hxa := -60; else: hxa := 60; fi
  pair v[];
  path phex[];
  v0 := z$$;
  v1 := z$;
  % vnější šestiúhelník
  for i := 2 upto 6:
    v[i] := v1 rotatedaround(v0,(i-1)*hxa);
  endfor
  z$a = v1; z$b = v2; z$c = v3;
  z$d = v4; z$e = v5; z$f = v6;
  phex0 := v1--v2--v3--v4--v5--v6--cycle;
  if showverts:
    dotlabels.lft($a,$e,$f);
    dotlabels.rt($b,$c,$d);
  fi
  if showlines:
    draw v1--v3--v5--cycle;
    draw v2--v4--v6--cycle;
  fi
  % konstrukce trojúhelníků
  for n:= 1 upto iter:
    k := 10(n-1);
    j := 10n;
    v[1+j] := (v[1+k]--v[3+k])
              intersectionpoint
              (v[2+k]--v[6+k]);
    for i := (2+j) upto (6+j):
      v[i] := v[1+j]
              rotatedaround
             (v0, (i-1-j)*hxa);
    endfor
```

```
    if showlines:
      draw v[1+j]--v[3+j]--v[5+j]--cycle;
      draw v[2+j]--v[4+j]--v[6+j]--cycle;
    fi
    phex[n] := v[1+j]--v[1+k]--v[2+k]--cycle;
    phex[n+1] := v[1+j]--v[2+j]--v[2+k]--cycle;
    fill phex[n] withcolor shadea;
    fill phex[n+1] withcolor shadeb;
    if showcells:
      draw phex[n]; draw phex[n+1];
    fi
    if showedges:
      draw v[1+k]--v[1+j];
      draw v[2+k]--v[2+j];
    fi
  endfor
  if showedges: draw v[1+j]--v[2+j]; fi
  if showhex: draw phex0; fi
enddef;
```

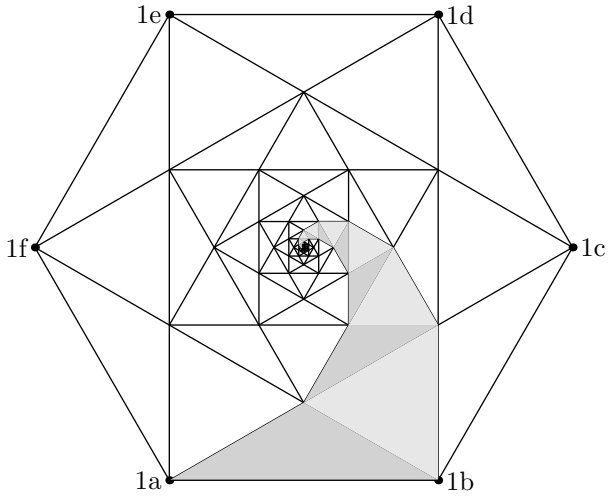
Jak název napovídá, makro `semispid` slouží k nakreslení poloviny spidronu – tu Erdély pojmenoval semispidron. Konstruovaný semispidron je umístěn uvnitř šestiúhelníku. Makru jsou předány pozice středu tohoto šestiúhelníku a jednoho z jeho vrcholů. Další argumenty určují počet trojúhelníků a barvy použité pro jednotlivé trojúhelníky. Makro používá booleovské proměnné definované jinde a podle nich zobrazuje různé části konstrukce obrázků.

K nakreslení obrázku 1 jsem použil následující program.

```
%% soubor glstr9.mp
% obrázky spidronů
prologues := 1;
input semispid
%%% deklarace a počáteční nastavení booleovských proměnných
% zobrazení základního šestiúhelníku
boolean showhex; showhex := false;
% popisky vrcholů
boolean showverts; showverts := false;
% nakreslení pomocných čar
boolean showlines; showlines := false;
% nakreslení obrysů trojúhelníků
boolean showcells; showcells := false;
% otáčení po směru ručiček (ano = true)
boolean rh; rh := false;
```

Obrázek 1: Spidron

```
% nakreslení obrysu spidronu
boolean showedges; showedges := false;
% nastavení barev
color light,dark;
light := 0.1[white,black];
dark := 0.2[white,black];
beginfig(1); % spidron
  u := 1in;  % základní jednotka
showhex := false;
showverts := false;
showlines := false;
showcells := false;
rh := false;
showedges := false;
% střed a počáteční vrchol
z0 = (0,0);
z1 = (x0-2u,y0) rotatedaround(z0,60);
semispid(0, 1, 9, dark, light, rh);
y0-y1a = y1a-y10; x10=x0;
z11 = z1b;
semispid(10, 11, 9, light, dark, rh);
endfig;
%% další obrázky od řádku 118
end
```



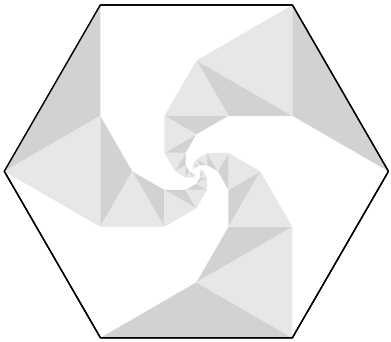
Obrázek 2: Detaily konstrukce spidronu

Detaily konstrukce semispidronu jsou ukázány na obrázku 2. Makro `semispid` vygeneruje vrcholy šestiúhelníku a označí je `a` až `f`, kde `a` je zadaný vrchol, který je argumentem makra. Šestiúhelník je spojnicemi svých vrcholů rozdělen na trojúhelníky a na menší šestiúhelník. Podobně je tento šestiúhelník rozdělen na menší a takto pokračujeme tak dlouho, dokud vzniklý šestiúhelník není dostatečně malý. Vybarvené šestiúhelníky tvoří semispidron začínající na straně `ab` a končící blízko středu původního šestiúhelníku. Druhá polovina spidronu vznikne otočením kolem středu strany `ab` o 180 stupňů a přehozením barev.

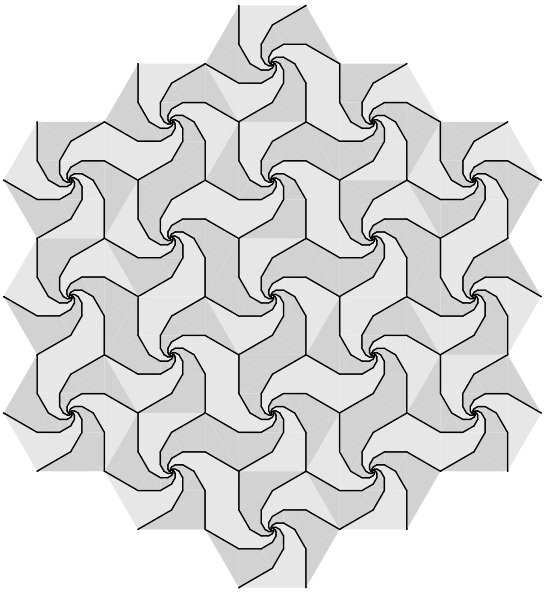
Spidrony jsou obrazce, s nimiž je možné pokrýt rovinu. Návod na toto pokrytí je na obrázku 3, kde jsou tři semispidrony vepsané do šestiúhelníku. Prázdná místa je možné vyplnit dalšími třemi semispidrony. A celou rovinu je možné pokrýt šestiúhelníky. Odtud již vyplývá, že je rovinu možné pokrýt spidrony. Pokud vhodně budeme měnit barvy spidronů, dostaneme zajímavé vzory, jako například na obrázku 4. Mnohem více způsobů pokrytí roviny najdete v knize [4] – spidrony tam však nenajdete, protože v době vydání knihy ješně nebyly známy.

Napojením dvou semispidronů může vzniknout ještě jiný obrazec. Ve spidronu je jeden semispidron oproti druhému otočený. Pokud semispidrony napojíme proti sobě zrcadlově, dostaneme obrazec, který Erdély pojmenoval hornflake[3]. Můžete

[3]Překladatelovi se nepodařilo dohledat původní Erdélyovo pojmenování tohoto útvaru, proto použil překlad názvu uvedený v [1]. Dle jeho názoru se pouhým převzetím překladu názvu dopustíme menší nepřesnosti než doslovným dvojnásobným překladem z neznámého původního názvu. (pozn. překl.)

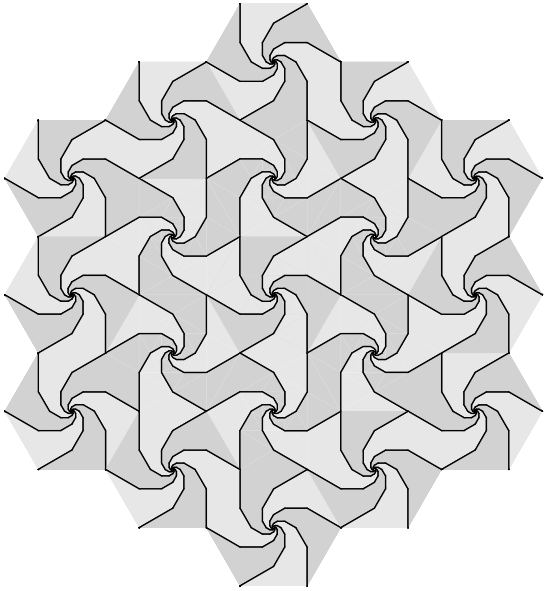


Obrázek 3: Tři semispidrony v šestiúhelníku



Obrázek 4: Pokrytí roviny – pouze spidrony

Obrázek 5: Hornflake



Obrázek 6: Pokrytí roviny – hornflaky a spidrony

jej vidět na obrázku 5. Na rozdíl od spidronů nejsou hornflaky obrazce, s nimiž je možné pokrýt rovinu. Když je však vhodně zkombinujeme se spidrony, rovinu můžeme pokrýt, jak je vidět na obrázku 6.

Ve svém článku Peterson píše:

> Erdély měl na papíře namalované šestiúhelníky, rozmístěné, jako by to byly kachličky v koupelně. Pak začal papír přehýbat nahoru a dolů na hranách jednotlivých spidronů, přičemž ve středu šestiúhelníků nechával malé dírky. Takto pomačkal celý papír, až z obyčejných šestiúhelníků udělal dramatický trojrozměrný reliéf.

Ukazuje se, že ze spidronových vzorů je možné vytvořit nové trojrozměrné útvary podobné krystalům se spirálovitě uspořádanými stěnami.

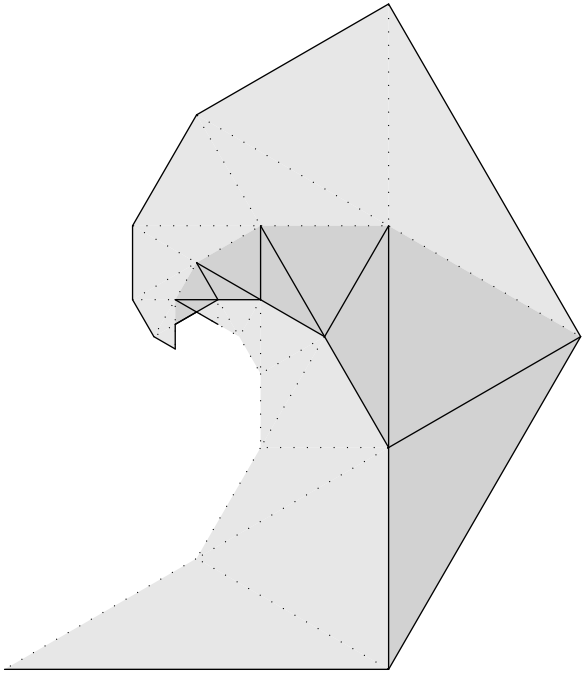
Co však v onom článku chybí, je nějaký náznak, jaké jsou ty správné kombinace přehybů, pomocí kterých docílíme uvedené efekty. Po delším vyhledávání na internetu jsem našel následující Erdélyovy poznámky [5].

> V dané soustavě spidronů jsem každou druhou hranu směřující do středu vzniklého šestiúhelníku přehnul směrem dolů (vznikly přehyby tvaru kopce) a zbývající hrany směrem nahoru (vznikly přehyby tvaru údolí). Vzniklý reliéf pod vlivem vnějších deformací nevykazuje jednoduché lineární vzory, jako například harmonika, ale hrany reliéfu proudí od vrcholů šestiúhelníků do jejich středů jako vír.

Po dlouhém přemýšlení a experimentování jsem došel k závěru, že jedna z těch správných kombinací přehybů je ta, která je na obrázku 7, na kterém je polovina šestiúhelníku a tři semispidrony. Tečkované čáry označují přehyby nahoru (údolí) a plné čáry označují přehyby dolů (kopce).

Pokud si chcete vytvořit obrovskou kresbu a zkoušet přehyby, tady je zdrojový kód obrázku 4, se kterým si můžete hrát. Samozřejmě si zdrojový kód můžete upravit podle libosti.

```
%% soubor glstr9.mp
% obrázky spidronů
% předchozí obrázky po řádek 112
beginfig(5); % pokrytí roviny spidrony
  u := 0.175in;
showhex := false;
showverts := false;
showlines := false;
showcells := false;
rh := false;
showedges := false;  showedges := true;
color cola, colb;
cola := light; colb := dark;
depth := 7;
```



Obrázek 7: Přehyby

```
129 rad := 2u;
130 z0 = (0,0);
131 % základní šestiúhelník
132 for kn := 1 upto 6:
133 z[kn] = (x0-2u,y0) rotatedaround(z0,60*(kn-1));
134 if odd kn:
135   cola := light;
136 else:
137   cola := dark;
138 fi
139  colb := cola;
140 semispid(0, [kn], depth, cola, colb, rh);
141 endfor
142 % kopie původního šestiúhelníku tvořící pokrytí
143 shd := (sqrt 3)/2*rad; % posun nahoru/dolů
144 shr := 3rad;           % posun doleva/doprava
```

```
picture pic[];
pic100 := currentpicture;
pic0 := pic100 shifted (0,2shd);
for kn := 1 upto 6:
  pic[kn] := pic0 rotatedaround(z0,60kn);
  draw pic[kn];
endfor
pic10 = pic100 shifted (0,4shd);
for kn := 1 upto 6:
  pic[10+kn] := pic10 rotatedaround(z0,60kn);
  draw pic[10+kn];
endfor
pic20 = pic100 shifted (3rad, 0);
for kn := 1 upto 6:
  pic[20+kn] := pic20 rotatedaround(z0,60kn);
  draw pic[20+kn];
endfor
endfig;
% další obrázky
end
```

Sám jsem to zkoušel a zjistil jsem, že je velmi složité správně přehnout i jen jeden šestiúhelník vytištěný na celou stránku A4. Proto jsem se rozhodl, že si vždy vezmu jeden spidron, ten přehnu a pak jednotlivé spidrony slepím k sobě lepicí páskou. Nakonec jsem své snažení završil tak, že jsem si raději na internetu prohlížel obrázky spidronových reliéfů, které vytvořili jiní (i když mám podezření, že většina těchto obrázků byla vytvořena na počítači a nebyla vytvořena ručně přehýbáním spidronů a vyfotografováním).

Odkazy

1. WILSON, Peter. Glisterings. *TUGboat*. 2009, roč. 30, č. 1, s. 69–73.
2. WILSON, Peter. Glisterings. *TUGboat*. 2008, roč. 29, č. 2, s. 324–327.
3. PETERSON, Ivars. Swirling seas, crystal balls. *Science News*. 2006, roč. 170, č. 17, s. 266–268.
4. GRÜNBAUM, Branko; SHEPHARD, Geoffrey Colin. *Tilings and Patterns*. W. H. Freeman, 1987.
5. ERDÉLY, Dániel. *Spidron system: A flexible space-filling structure*. 1998. Myšlenka z roku 1979, poprvé prezentováno na 12th International Conference on Crystal Growth (ICCG 1998).

Summary: It Might Work VIII – Drawings

Some aspects of inserting METAPOST figures into (pdf)LaTeX documents are discussed in this paper. The paper also focuses on shapes called spidrons and on manipulations with them.

Keywords: LaTeX, METAPOST, spidron

Peter Wilson, herries.press@earthlink.net
18912 8th Ave. SW
Normandy Park, WA 98166 USA

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:	Duane Bibby
Počet výtisků:	280
Uzávěrka:	9. 12. 2019
Odpovědný redaktor:	Jan Šustek
Redakční rada:	Pavel Haluza, Lukáš Novotný, Vít Novotný, Michal Růžička a Jan Šustek (šéfredaktor)
Vědecká rada:	Ján Buša (předseda), Jiří Demel, Jaromír Kuben (zástupce předsedy), Jiří Rybička a Petr Sojka
Technická redakce:	Vít Novotný
Evidenční číslo MK:	E 7629
Adresa:	CSTUG, Nejedlého 373/1, 638 00 Brno
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`

CONTENTS