Spoločnosť pre otvorené informačné technológie

OTVORENÝ SOFTVÉR VO VZDELÁVANÍ, VÝSKUME A V IT RIEŠENIACH

Zborník príspevkov medzinárodnej konferencie OSSConf 2010

1.–4. júla 2010
Žilina, Slovensko

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
1.–4. júla 2010, Žilina, Slovensko

Organizátori: Miloš Šrámek, Spoločnosť pre otvorené informačné technológie
Tatiana Šrámková, Katedra fyziky, FEI STU Bratislava
Michal Kaukič, Aleš Kozubík, Tomáš Majer, Žilinská univerzita
Lýdia Gábrišová, Ľubica Michálková, Žilinská univerzita
Juraj Bednár, Digmia, Slovensko
Miloslav Ofúkaný, GeoCommunity, Slovensko
Peter Mráz, Kremnica
Slavko Fedorik, SOŠ elektrotechnická, Poprad
Peter Štrba, Spojená škola/Gymnázium M. Galandu, Turčianske Teplice
Ladislav Ševčovič, FEI, Technická univerzita v Košiciach

Editori: Michal Kaukič
Miloš Šrámek
Slavko Fedorik
Ladislav Ševčovič

Recenzenti: Mgr. Juraj Bednár
Mgr. Rudolf Blaško, PhD.
RNDr. Ján Buša, CSc.
Ing. Slavko Fedorik
Ing. Karol Grondžák, PhD.
Mgr. Michal Kaukič, CSc.
Ing. Tomáš Kliment
RNDr. Aleš Kozubík, PhD.
Mgr. Juraj Michálek
doc. RNDr. Štefan Peško, CSc.
Ing. Pavel Stříž, PhD.
RNDr. Ladislav Ševčovič
Ing. Michal Žarnay, PhD.

Vydavateľ: Spoločnosť pre otvorené informačné technológie – SOIT, Bratislava

ISBN 978-80-970457-0-8

Sadzba programom pdfTEX Ladislav Ševčovič

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 1.–4. júla 2010

Editori: Michal Kaukič
Miloš Šrámek
Slavko Fedorik
Ladislav Ševčovič

SPONZORI KONFERENCIE

PODPORILI NÁS

Obsah

ÚVOD

Vážení čitatelia,

tento zborník je jedným z výstupov už druhej prvej samostatnej konferencie na Slovensku, ktorá je venovaná slobodnému a otvorenému softvéru a jeho využitiu vo vzdelávaní, vede, ale aj v ostatných oblastiach, kde sa využívajú informačné technológie. Konferencia je jednou z aktivít občianskeho združenia „Spoločnosť pre otvorené informačné technológie“ (SOIT), `http://www.soit.sk`.

V celosvetovom meradle, obzvlášť v štátoch EU, pozorujeme veľký nárast používania otvoreného softvéru (OSS). Dokonca aj viaceré komerčné firmy ponúkajú riešenia, kde je základom tento softvér. Veľké firmy v oblasti IT, ako Red Hat, Novell (OpenSUSE), Google, HP, IBM nielen deklarujú podporu otvoreného softvéru, ale ho aj vyvíjajú a dávajú ďalej k dispozícii vývojárskej komunite. Tým sa výrazne rozširujú možnosti využitia OSS doma, vo vzdelávaní, výskumnej činnosti a tiež aj v softvérových firmách pre nekomerčné i komerčné riešenia.

Napriek týmto faktom, OSS sa dostáva do povedomia verejnosti ťažšie, ako komerčné operačné systémy a softvér. Šíri sa hlavne v menších komunitách, „ústnym podaním“ a osobnými príkladmi blízkych osôb. Existuje mnoho učiteľov, študentov, výskumníkov alebo prosto nadšencov, ktorí OSS vo svojej práci využívajú, nahrádzajú či dopĺňajú ním drahé komerčné riešenia a pritom získavajú cenné skúsenosti, z ktorých by mohli mať úžitok aj ostatní. Problémom je, ako tieto skúsenosti odovzdať. Jednou z príležitostí môže byť aj naša konferencia.

Sme presvedčení, že otvorený softvér sa môže veľmi dobre uplatniť v oblasti výučby a výskumu na všetkých stupňoch škôl. Zatiaľ čo niektoré softvérové produkty s nedostupným zdrojovým kódom sú široko propagované zjavnou i skrytou reklamou, otvorený softvér a jeho možnosti sú oveľa menej známe. Je to škoda, pretože otvorený softvér v tejto oblasti poskytuje plnohodnotnú náhradu viacerých finančne náročných softvérových produktov (kancelárske balíky, matematický softvér, sieťové služby a administrácia, programy na výučbu rôznych predmetov na základných a stredných školách). Viaceré z týchto otvorených alternatív sú prezentované v materiáloch konferencie (konferenčné DVD a tento zborník).

Výhody otvoreného softvéru sa neprejavujú len v ekonomickej oblasti, ale majú aj významný dopad na myslenie žiakov, študentov i učiteľov. Podporujú tvorivý prístup k softvéru, ktorý sa hĺbavému používateľovi už nejaví ako čierna skrinka ale umožňuje mu prispôsobiť si jeho funkcionalitu vlastným potrebám. Zároveň majú používatelia veľký vplyv na jeho vývoj, pretože ich ohlasy, želania, kritika a chybové hlásenia pomáhajú vývojárom pri ďalšom zdokonaľovaní a šírení príslušného softvérového systému.

Konferencie *Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach* chcú pokračovať v dobrej tradícii vytvorenej predchádzajúcimi podujatiami (dva ročníky špecializovanej sekcie v rámci medzinárodnej konferencie Aplimat v r. 2007 a 2008, *Víkend s Linuxom* v Kremnici v júni 2008). Nie je to len konferencia v klasickom, akademickom ponímaní, ale hlavne neformálne stretnutie ľudí, ktorých spája záujem o otvorený softvér (jeho tvorbu, využitie, komplexnejšie riešenia na základe existujúceho softvéru). Popri recenzovaných

príspevkoch, ktoré sú publikované v tomto zborníku, na podujatí budú predvedené ukážky riešení, kde je základom otvorený softvér. Stručné abstrakty týchto ukážok a prezentácií sú uvedené v závere tohto zborníka.

Všetky príspevky, uvedené v tomto zborníku, ako aj prezentácie, ktoré na konferencii odznejú, budú voľne prístupné na stránke podujatia, `http://ossconf2010.soit.sk`. Okrem toho, na tejto stránke nájdete aj odkaz na konferenčné DVD (DVD médium dostane každý účastník konferencie), ktoré okrem materiálov konferencie obsahuje distribúciu Linux Ubuntu 10.04 s ďalším pridaným softvérom, ktorého sa budú týkať príspevky konferencie.

Táto konferencia je jednou z aktivít SOIT na propagáciu otvoreného softvéru. Sme toho názoru, že čím skôr sa používateľ počítača s otvoreným a slobodným softvérom zoznámi, tým lepšie bude vedieť oceniť jeho prednosti, z ktorých bude neskôr profitovať, či už profesionálne alebo v súkromí. Zároveň sa rozširuje jeho všeobecný rozhľad, keď sa odváži nakuknúť za oponu, mimo javiska, ktoré mu prezentuje svet reklamy a obchodných záujmov softvérových gigantov. Na podporu a propagáciu alternatívneho vyučovania informatiky a ostatných predmetov na základných, stredných a vysokých školách prostredníctvom bezplatného otvoreného softvéru slúži tiež projekt „Slobodný a otvorený softvér pre školy“ s domovskou stránkou `http://sospreskoly.org`. Cieľom projektu je vytvoriť komunitu záujemcov a používateľov otvoreného softvéru na školách všetkých úrovní. Je pokusom o spojenie tých, ktorí vedia ako na to a rovnako aj tých, ktorí si uvedomujú možnosti, ktoré otvorený softvér pre školstvo a spoločnosť poskytuje, ale nevedia ako a kde začať.

Nič sa neurobí samo od seba, takže tento projekt je príležitosťou aj pre vás. Môžeme postupne budovať vzorové učebne s voľne prístupným edukačným softvérom, pracovať na jeho prípadných úpravách a lokalizácii, vytvárať návody a príručky pre správcov učebne a používateľov, aby sme čoraz viac cítili, že v našich snahách nie sme osamotení. Súčasťou snáh projektu *SOS pre školy* je aj presadzovanie (na úrovni vládnej i regionálnych samospráv) otvoreného softvéru v použití pre školské administratívne a ekonomické účely. Na konferenci budú zverejnené výsledky prvého ročníka súťaže o *Cenu SOIT a Liberixu za študentské práce*, ktorú spoločne vyhlásili Spoločnosť pre otvorené informačné technológie a česká všeobecne prospešná spoločnosť Liberix. Hlavným cieľom Ceny je propagácia používania otvoreného softvéru na univerzitách a vyjadrenie uznania učiteľom, ktorí vedením študentských prác podporujú vývoj a využitie otvoreného softvéru.

Tento zborník, materiály konferencie, a aj samotný fakt, že sa toto podujatie mohlo uskutočniť, je kolektívnym dielom organizátorov, editorov, autorov príspevkov a prezentácií, recenzentov i všetkých účastníkov. Myšlienka samostatnej OSS konferencie sa mohla realizovať len vďaka príkladnej spolupráci a dobre odvedenej práci všetkých zainteresovaných. Ďakujeme im a dúfame, že sa aj v budúcnosti budeme stretávať na podobných podujatiach a že nás bude čoraz viac, aby naše hlasy nezanikli v lomoze každodenného zhonu a všadeprítomnej reklamy softvérových firiem. Ďakujeme tiež sponzorom HP Slovensko, Slovenská informatická spoločnosť, Red Hat, Gista, WinStrom, kníhkupectvo U Marečka a Liberix, ktorých sponzorské príspevky umožnili vydanie tohto zborníka, pomohli pri hradení pobytových nákladov viacerým účastníkom z radov učiteľov a prispeli na odmeny laureátov Ceny SOIT a Liberixu.

Organizačný výbor OSSConf 2010

POZVANÍ PREDNÁŠATELIA

Vlastimil Ott je absolventom Sliezskej univerzity v Opave. Od roku 2002 píše články o Linuxe a slobodnom softvéri do mnohých internetových a tlačených médií. Od roku 2005 je šéfredaktorom magazínu LinuxEXPRES a od r. 2008 šéfredaktorom portálu `OpenOffice.cz`. Je korektorom a spoluautorom kníh o Mandriva Linuxe, Ubuntu a ďalších. Od 1. júna 2009 je riaditeľom všeobecne prospešnej spoločnosti Liberix.

Jiří Rybička se zabývá zpracováním textů na počítači již přes 20 let. Těžištěm jeho zájmu je především sazba publikací systémem LaTeX. Je členem výboru Československého sdružení uživatelů TeXu, kde se věnuje zejména začínajícím uživatelům (kniha LaTeX pro začátečníky, vyšla s podporou tohoto sdružení). Na Mendelově univerzitě se věnuje výuce specializovaného předmětu Zpracování textů počítačem a zabývá se problémy zvyšování formální úrovně závěrečných prací.

Petr Kovář se zabývá technickým překladem a psaním technické dokumentace především svobodného a otevřeného softwaru. Studoval aplikované technologie a historické vědy na Univerzitě Hradec Králové a Masarykově univerzitě v Brně. Je členem několika komunitních překladatelských týmů, od roku 2008 koordinuje český překladatelský tým GNOME, je také členem GNOME Foundation a koordinačního týmu GNOME Translation Project. Pomáhá se správou Překladatelského centra `l10n.cz`, které soustřeďuje české komunitně zaměřené lokalizační aktivity a mimo jiné vytváří společnou terminologii a další volně dostupné lokalizační zdroje.

OPEN-SOURCE SOFTWARE VE ŠKOLSTVÍ – ZKUŠENOSTI A POTENCIÁL

OTT, Vlastimil (CZ), Liberix, o.p.s.

Abstrakt. Na portálech `LinuxEXPRES.cz` a `OpenOffice.cz` vycházejí případy použití open source ve školách. Zástupci škol uvádějí důvody, proč open source používat, jakým problémům museli při přechodu čelit a jak je vyřešili. Příspěvek uvede několik škol a jejich zkušeností. Jedná se o školy několika stupňů a zaměření – základní, střední, speciální, univerzity. Některé školy používají open source také ve výuce. Existuje totiž velké množství výukového softwaru, který je k dispozici s neomezující licencí. Jde o programy do běžných předmětů (matematika, fyzika, geometrie, biologie, chemie, zeměpis, výuka jazyků), méně častých předmětů (elektronika, elektrotechnika, astronomie) nebo pro podporu výuky (myšlenkové mapy, schémata, diagramy, testové nástroje a další).

1 Případové studie na portálech Liberixu

Na portálech `LinuxEXPRES`[1] a `OpenOffice.cz`[2] vycházejí články o nasazení svobodného softwaru ve firmách, organizacích a ve školách. Ačkoliv zní tato věta jednoduše a pochopitelně, praxe už tak jednoduchá není. Redaktoři portálů vyvíjejí výraznou aktivitu, aby informace tohoto typu vůbec získali. Firmy a často ani jiné subjekty se nasazením „softwaru zadarmo“ obvykle chlubit nechtějí nebo vůbec nechápou, proč by se o tom mělo psát. Redaktoři se tedy snaží vysvětlovat, proč je takový článek důležitý:

1. motivuje další subjekty, aby open-source software používaly
2. motivuje další subjekty, aby o použití napsaly
3. hraje výraznou marketingovou roli
4. představuje argumentační potenciál

[1] `http://www.linuxexpres.cz/pripadove-studie`
[2] `http://www.openoffice.cz/pouzivaji-openoffice-org`

2 Školy a případy použití open-source softwaru

Rubriky na obou portálech nejsou příliš zavedené, jsou poměrně nové. Dříve články vycházely sporadicky a byly zařazeny do jiných kategorií. Na portálu `OpenOffice.cz` je několik prvních článků z roku 2003, pak se dlouho nic nedělo. Redakce se na tuto tématiku zaměřila od začátku roku 2009 – připravila univerzální dotazník a díky němu do června 2010 vydala na čtyřicet článků.

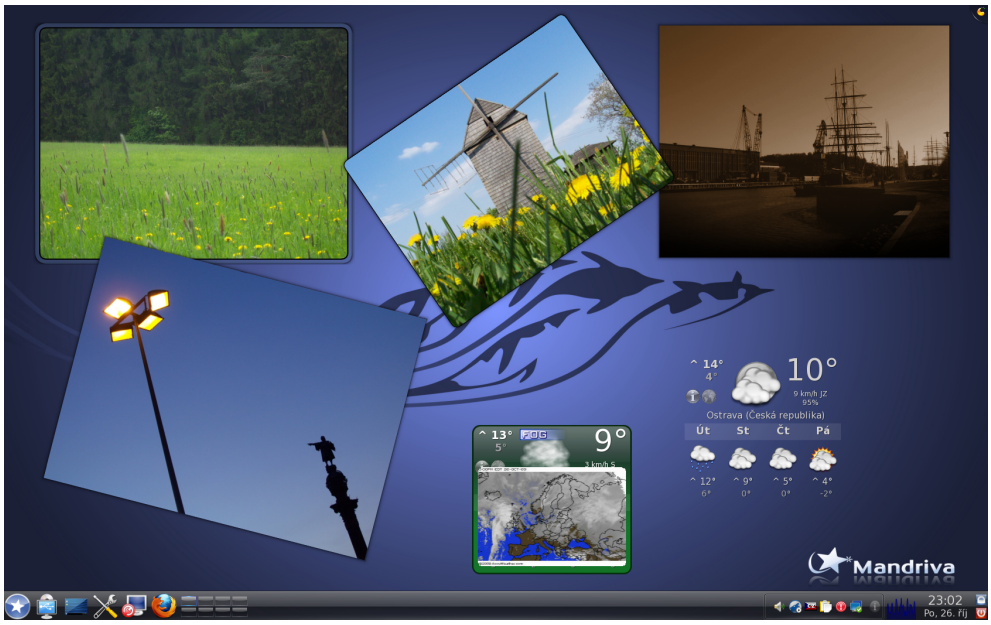
Mezi vyplněnými dotazníky je také určité množství škol a školských zařízení. Mezi prvními vyplnil formulář správce ICT z Jihočeské vědecké knihovny v Českých Budějovicích[3], kde se kancelářský balík OpenOffice.org používá na 120 počítačích pro veřejnost. Následovaly dotazníky vyplněné firmami a poté se přidaly také školy: Střední průmyslová škola sdělovací techniky Panská v Praze[4], Konzervatoř a VOŠ J. Ježka[5], Základná škola s materskou školou Bystričany[6], Gymnázium Jevíčko[7] a další. Začal se projevovat efekt sněhové koule, který jsme podpořili také při osobních setkáních s učiteli (na konferenci Počítač ve škole 2010[8] a spoluprací s Jednotou školských informatiků[9]).

Na portálu `LinuxEXPRES.cz` je situace trochu jiná, zde článek vzniká formou řízeného rozhovoru (obvykle e-mailem). Vzhledem ke komplexnosti a složitosti problematiky není možné sestavit univerzální dotazník. O to cennější získané poznatky jsou, je ovšem pravda, že článků je méně a jejich získání je pracnější. Open-source software používá VOŠZ a SZŠ Hradec Králové[10], a to zejména na serverech, jednotliví učitelé také používají některé open source aplikace:

> Ono je myslím dobré studentům ukazovat programy zdarma, nebo ještě lépe rovnou ty s otevřeným zdrojovým kódem. Nejen, že se jim to může hodit v životě, ale zároveň nejsou odkázáni ve svých přípravách do školy a na hodinu na programy nainstalované pouze ve škole, ale mohou si je také nainstalovat doma na svůj počítač. A hlavně legálně.
>
> – Mgr. Martin Šín, VOŠZ a SZŠ Hradec Králové

Skvělým dokladem toho, že open source je dostatečně kvalitní pro použití na školách, je článek o církevní střední škole v Bojkovicích[11], která používá Ubuntu, OpenOffice.org a do jisté míry výukový software. Na škole probíhají také veřejné vzdělávací kurzy, samozřejmě také s využitím Ubuntu a v něm obsaženého softwaru. Škola open source jednoznačně doporučuje:

[3] `http://www.openoffice.cz/pouzivaji-openoffice-org/jihoceska-vedecka-knihovna-pouziva-openoffice-org`
[4] `http://www.openoffice.cz/pouzivaji-openoffice-org/openoffice-org-na-stredni-skole-panska`
[5] `http://www.openoffice.cz/pouzivaji-openoffice-org/konzervator-a-vos-jaroslava-jezka-pouziva-kancelarsky-balik`
[6] `http://www.openoffice.cz/pouzivaji-openoffice-org/zakladni-skola-s-materskou-skolou-v-bystricanech-pouziva`
[7] `http://www.openoffice.cz/pouzivaji-openoffice-org/gymnazium-jevicko-pouziva-openoffice-org`
[8] `http://www.liberix.cz/liberix-se-ucastnil-konference-pocitac-ve-skole-2010/`
[9] `http://www.liberix.cz/liberix-o-p-s-spolupracuje-s-jednotou-skolskych-informatiku/`
[10] `http://www.linuxexpres.cz/business/nasazeni-svobodneho-softwaru-na-stredni-skole`
[11] `http://www.linuxexpres.cz/business/csos-bojkovice-spokojene-pouziva-ubuntu`



Obr. 1: Mandriva Linux 2010 a pracovní prostředí KDE4

> Doporučila byste používání Linuxu i jiným školám?
>
> ANO! Nebo pro začátek alespoň používání open-source aplikací.
>
> Vysoce hodnotím marketingovou strategii firmy Microsoft. Ale je mi líto, že jí české školství (přinejmenším základní a střední) tolik podlehlo. Přijde mi to nefér vůči studentům, kteří školy opouštějí se znalostí produktů Microsoft a tušením, že něco jiného sice existuje, ale. . .
>
> [. . .]
>
> Rozhodnutí pro Linux mělo důvody finanční i ideologické. Spočítali jsme si, kolik stojí licence na Windows, MS Office a jejich upgrade. [. . .] Můj podíl na rozhodování byl podpořen ještě snahou ušetřit práci - z předchozího zaměstnání jsem měla zkušenost s nákupy a převody licencí a upgradem na postupně přibývající počítače. Po několika letech dala snaha o systematický a průkazný přehled o nakoupeném softwaru docela „zabrat“.
>
> – Ludmila Chládková, učitelka IT

Rozsáhlou školní infrastrukturu postavilo na bázi Linuxu Mendelovo gymnázium v Opavě[12]. Školu navštěvuje cca 800 studentů a pedagogický sbor čítá na sto zaměstnanců. Open-source

[12] http://www.abclinuxu.cz/clanky/rozhovory/mendelovo-gymnazium-v-opave-jak-vyuziva-linux

software použili správci pro vytvoření mnoha serverových služeb, vylepšení bezpečnosti sítě a snadnější správu.

> Pro nasazení Linuxu hrály dva faktory, první je komfort a druhý finance. Na serverech máme zprovozněnou celou škálu služeb jako web, jabber, ftp, mail server, file server, zálohovací server a spoustu dalších. Všechny tyto služby samozřejmě lze realizovat komerčními produkty, ovšem cena by byla neúměrná k šíři využitelnosti, protože určité aplikace jsou např. využívány pouze zlomkem uživatelů, popř. o některých aplikacích uživatelé ani nevědí. Díky Linuxu a open source nás finanční faktor vůbec netíží a linuxové distribuce nabízejí vynikající nástroje pro všechny potřebné činnosti. Mohu si tedy zprovoznit takřka cokoliv a ještě si mohu mezi nástroji vybrat.
>
> – Marek Kočvara, správce sítě

Shrnuto: není známo příliš relevantních argumentů, které by hovořily proti použití open source ve školách. V uvedených případech je používán třeba také komerční operační systém. Je organizačně nebo technicky problematické nahradit informační systémy školy nebo účetnictví, viz dále. Nicméně open-source software má na mnoha školách svou pevnou pozici, na jiných zase výrazný potenciál.

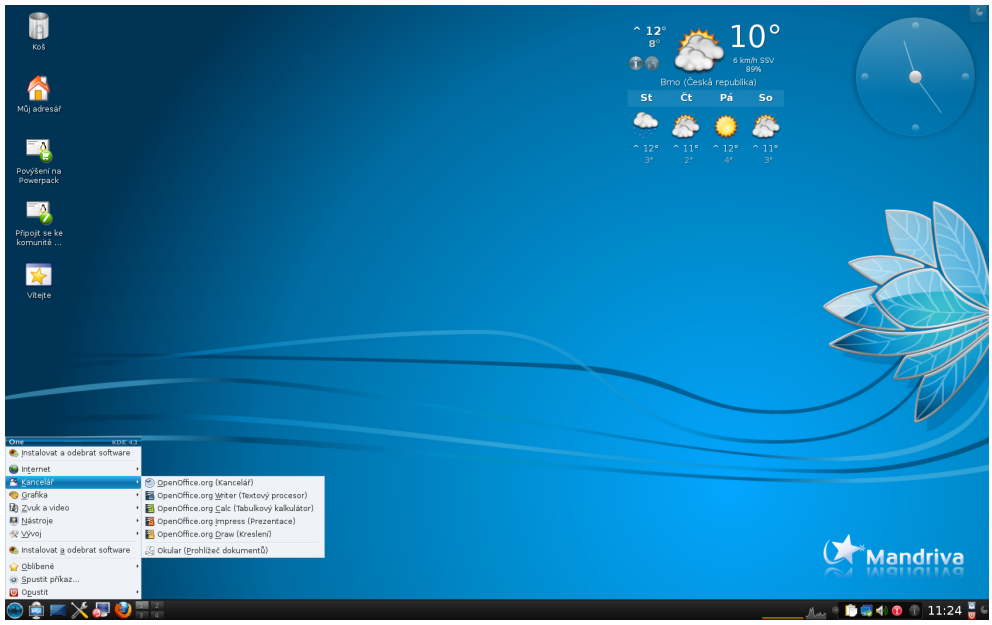
3 Argumenty škol proti open source, které lze vyvrátit

Ačkoliv lze na několika příkladech uvedených výše demonstrovat, že existují školy, které jsou z open source nadšené, přesto nelze bohužel tvrdit, že by měl tento postoj výraznější zastoupení. Objektivně lze fakty doložit (a výše uvedené příklady jsou vybrané ukázky), že školám se použití open-source nástrojů vyplácí. Přesto ze škol slýcháme některé z následujících argumentů proti použití open source. Pokusíme se je vyvrátit.

3.1 Linux? Prý hračka bez využití

To je ta příkazová řádka, ne? Na to my nemáme čas, to je hračka pro pubescenty.

Tragický omyl hraničící s jistou arogancí. Učitel by se neměl a priori vymezovat vůči novým věcem, neboť právě on je (?) nositelem nových podnětů, které předává svým žákům. Nemusí Linux a open source milovat, měl by o něm ale vědět a informace nabídnout žákům. Na školním webu, vypálit jim CD nebo jim poradit, kde mají hledat informace. Linux lze zdarma a legálně získat stažením z internetu nebo jakkoliv jinak. Tvrzení, že „Linux je příkazová řádka“, je předsudek, který se nezakládá na pravdě. Kvalitou svého uživatelského rozhraní dosáhl běžného standardu a zkušenosti mnoha uživatelů dokladují, že žádné problémy s jeho ovládáním nejsou překážkou v jeho používání:



Obr. 2: Mandriva Linux 2010

> Jsem důchodce, 66 let, žiji v Ústí nad Labem. [...] Zkusil jsem instalovat Linux. Mandriva 2007.0 si poradila, nepotřebovala drivery, Fast Tracky a podobné nesmysly. Od té doby používám Linux a jsem spokojen. [...] Používám běžné programy jako OpenOffice.org, Firefox, přehrávače filmů a hudby. V poslední době používám hodně Audacity pro vylepšení MP3, které jsem získal přehráváním LP desek.
>
> –`http://www.linuxexpres.cz/proc-pouziva-linux/vladislav-nevole`

3.1.1 „Zlí“ rodiče

Tvrzení: Rodiče vyžadují, abychom učili Windows a MS Office, protože to je nástroj, se kterým se studenti setkají v zaměstnání – open source nikdo nepoužívá.

Nesmysl. Open-source software a třeba operační systém Ubuntu nebo Mandriva Linux se ovládá stejně jako MS Windows a pro uživatele v tomto ohledu neexistují žádné bariéry nebo rozdíly. Nehrozí, že by se studenti učili něco, co by nevyužili v praxi, protože škola je má učit *dovednosti*, nikoliv klikání na ikony v postupu, který napsal pan učitel jako jediný správný. Také s Ubuntu se studenti naučí ovládat počítač.

Uvedený argument má navíc hliněné základy, protože v době, kdy žáci či studenti opustí školu, budou informační technologie (hardware i software) na jiné úrovni než dnes. Pokud

dnes vládne majorita v oblasti operačních systémů, nikdo nemůže totéž očekávat za pět nebo více let. O to více je potřeba, aby žáci znali principy, nikoliv popisné návody omezené na jeden software konkrétní verze.

Poznámka na okraj. Proč takovým rodičům nevadí, že základní školy často používají devět let starý operační systém na hardwaru, který je ve 44 % ZŠ starší pěti let? A že je učí učitelé, kteří pouze ve 29 % přiznávají, že dokáží pracovat s výukovým softwarem, což zároveň neznamená, že ho používají? Proč zrovna open source má být argumentem pro zhoršení kvality výuky, když jsou zde varovnější faktory? (Zmíněné informace čerpány ze zprávy České školní inspekce Úroveň ICT v základních školách v ČR[13]*)*

3.1.2 Dodavatel

Nám se o počítače stará firma a oni říkali, že Linux nedodávají a že by to stejně bylo dražší.

Zbavte se jich, tahají z vás peníze. Pokud není dodavatel ochoten nebo schopen plnit požadavky zákazníka, pak si má zákazník najít někoho lepšího. Existují firmy, které zajistí provoz učeben i serverů na Linuxu. Technologicky to je řešitelný problém a plně odpovídá požadavkům na optimalizaci a snížení nákladů.

3.1.3 Strach, obavy, předsudky (nevyřčený argument)

Nic o tom nevíme, nerozumíme tomu a máme strach, že se to nepovede. Nechceme se tím zabývat, však současné řešení funguje, tak co. Nemáme důvod měnit.

Informujte se, oslovte Liberix, požádejte o konzultace odborné subjekty, nejen místní firmy. Nechte si předložit nabídky, srovnejte ceny, vyžadujte parametry – chovejte se jako náročný zákazník, kvalitní firma se bude o vaši přízeň ucházet. Školy málo využívají konkurenčního prostředí a často platí nekřesťanské peníze např. za připojení do internetu. Úspory by měly hledat také v případě softwarového vybavení – ušetřit mohou tím, že budou používat kancelářský balík OpenOffice.org nebo Dokumenty Google, linuxový operační systém na školním serveru případně také v učebnách apod.

3.1.4 Peníze

Bude to stát peníze a ty my nemáme. Navíc není jasné, že se něco ušetří.

Peníze můžete ušetřit – nebudete platit licenční poplatky za software, budete ovšem nadále platit za služby a hardware. Míru úspor lze propočítat v konkrétním případě, obecné stanovení zřejmě není možné. Doporučujeme článek *Kolik stojí Linux*[14], kde je nastíněna metodika výpočtu a proveden simulovaný výpočet celkových nákladů (TCO, total cost of ownership[15]).

[13] http://www.csicr.cz/cz/85156-uroven-ict-v-zakladnich-skolach-v-cr

[14] http://www.linuxexpres.cz/business/kolik-stoji-linux

[15] http://en.wikipedia.org/wiki/Total_cost_of_ownership

3.1.5 Neslučitelnost s používaným softwarem

Používáme programy XYZ a ty na tom Linuxu nejedou.

Není třeba násilím migrovat na Linux – výhody open-source softwaru lze využívat také na MS Windows, kde tento software také funguje.[16] Častým argumentem bývá, že na Linuxu

1. *neexistuje účetnictví* – není pravda, existují multiplatformní ekonomické systémy, problém bude s ochotou obsluhy změnit návyky;

2. *nepoběží informační systém školy* – bohužel spíše pravda, školy by měly na výrobce těchto systémů vyvíjet tlak, aby své systémy nabízeli i pro Linux;

3. *nefungují výukové a vzdělávací programy* – částečně pravda; mnohé programy fungují pomocí emulátorů nebo jiných softwarových nástrojů (Wine), jiné mají své linuxové alternativy, pro Linux existuje množství vzdělávacích programů, jsou navíc zdarma;[17]

4. *nebude možné pracovat s dokumenty .doc* a jinými, které na školy zasílá ministerstvo, případně jiné úřady – není pravda, OpenOffice.org to zvládá, úřady nesmí preferovat jeden formát, což se v praxi bohužel porušuje a školy se nebrání.

Je vhodné konstatovat, že existují výrobci, kteří linuxovou platformu ignorují a jejich software pro Linux neexistuje a současně neexistuje vhodná open-source alternativa. Je to případ firmy Adobe, která dává zcela jasně najevo, že své produkty pro Linux nepřipraví, protože se jí to nevyplatí. Podobná situace panuje v případě technických škol a drahého specializovaného softwaru (např. CAD), jejichž linuxové verze k dispozici nejsou. Tento argument proti použití open-source softwaru lze chápat jako oprávněný, není to ale chyba samotné myšlenky open source. Také zde panují konkurenční vztahy, které mohou situaci změnit.

3.1.6 Neuměli bychom to používat

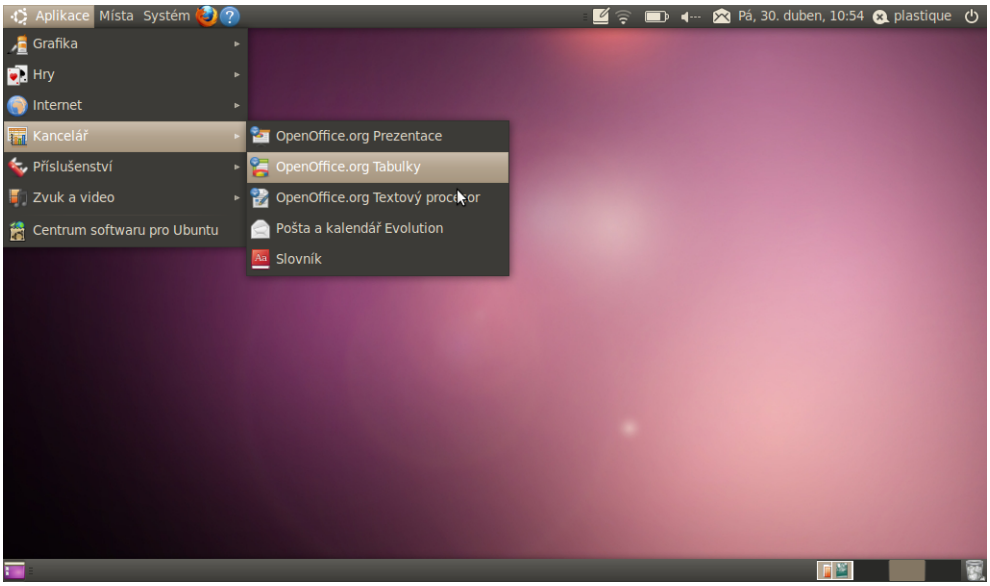
Skryté riziko – drtivá většina učitelů základních škol má pouze základní počítačové dovednosti a počítač umí ovládat na uživatelské úrovni.[18] V případě učitelů středních škol bude stav lepší, ale stále nedostačující. Mnozí učitelé, ačkoliv proškolováni, počítače příliš ovládat neumějí a používat je příliš nechtějí. To chápeme jako zásadní problém, který je nutné řešit zejména koncepčně na jiné úrovni.

Ve spolupráci s *Jednotou školských informatiků* máme připravený vzdělávací program *Volně šiřitelný software pro školství*, který je akreditován na Ministerstvu školství, mládeže a tělovýchovy. Naše nezisková společnost se snaží podobné překážky odstraňovat, obraťte se na nás, poradíme vám a pomůžeme.

[16] `http://www.opensourcewindows.org`

[17] `http://www.linuxexpres.cz/software-pro-skoly`

[18] Podle zprávy ČŠI – `http://www.csicr.cz/cz/85156-uroven-ict-v-zakladnich-skolach-v-cr`



Obr. 3: Ubuntu 10.04 na netbooku

4 Proč by měly školy open source využívat (a chlubit se tím)

Již vydané články o použití open-source softwaru se stávají motivací pro další učitele, aby si našli čas a napsali vlastní článek. Samozřejmě pokud se na škole open source používá. Mnozí učitelé, je jich ovšem stále příliš málo, si uvědomují význam slova *marketing* a *reklama* v kontextu školského prostředí. Vědí, že jejich práce bude oceněna, pokud se informace dostanou k (odborné) veřejnosti. Tomu mohou články výrazně napomoci. Každá reklama škole – byť v relativně malém okruhu uživatelů open source – se může projevit kladně. Pokud škola takto veřejně deklaruje pestrost své výuky a pestrost názorů, může získat punc progresivního vzdělávacího zařízení, které není ve vleku jedné firmy a jednoho dodavatele. Tímto „coming outem“ přitom nemůže nic ztratit.

4.1 Argumenty ve prospěch open-source softwaru

4.1.1 Legálnost

Open source software má také mnohem jednodušší správu licencí,[19] protože drtivá většina softwaru má neomezující licence, které není nutno evidovat či jinak spravovat. Odpadají požadavky na softwarový audit a s tím spojené náklady. Škola i učitelé jdou příkladem v prevenci proti softwarové kriminalitě a věta učitele informatiky „Nahraju vám ten kancelářský

[19] http://www.linuxexpres.cz/business/softwarove-licence-vhodne-pro-pouziti-ve-skolnim-prostredi

balík na flash disk“ získává zcela jiný kontext – šíření open-source softwaru je *žádoucí a naprosto legální*. Učitelé získávají v očích žáků kladné body, protože najednou znají něco jiného, originálního a nového – Linux a open source.

> Dále je důležité studentům ukázat, že argument „potřebuji tento program, ale nemám na něj, tak si ho někde zkopíruji“, neobstojí ve světle faktů. K většině komerčních aplikací je v současnosti možné nalézt alternativy zadarmo, které jsou plně postačující a mnohdy i lepší než jejich komerční protějšky. V tomto smyslu je používání OpenOffice.org důležitým krokem k potírání softwarové kriminality, která je značná především v sektoru studentů a domácích uživatelů.
>
> – Mgr. Martin Raus, Gymnázium Jevíčko[20]

4.1.2 Odolnost proti virům, bezpečnost

Pro linuxové systémy dnes neexistují viry. Existují antivirové programy pro Linux, které obvykle běží na linuxovém serveru, aby chránily počítače s Windows v lokální síti. Není vyloučeno, že viry pro Linux v budoucnu vzniknou, ovšem díky struktuře uživatelských oprávnění bude jejich činnost velmi omezená. Reálná virová hrozba v Linuxu neexistuje, existuje ovšem slabina jako všude jinde – nepoučený uživatel.

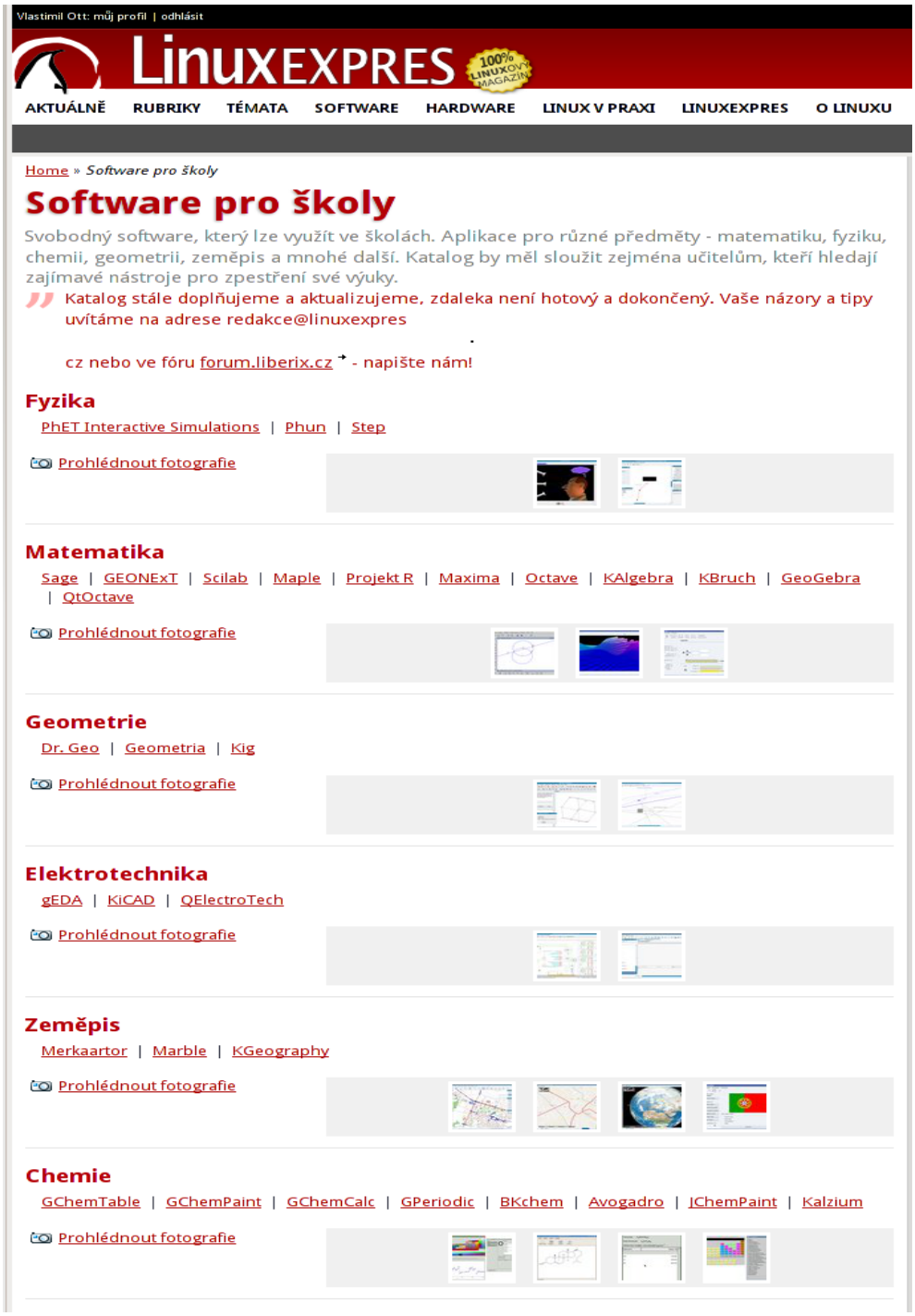
Linuxové systémy jsou velmi bezpečné, nejedná se ovšem o zázrak, i tyto systémy lze napadnout a kompromitovat. Jejich správa vyžaduje pozornost jako každý jiný software. Mají ovšem velkou výhodu – na opravu chyb se nečeká do druhého úterka[21] nebo jiného dne v měsíci, jsou k dispozici v řádu hodin nebo dnů. Linuxová komunita je obrovská a reakce přicházejí velmi rychle. Jakmile používáte Linux a open source, můžete vy nebo správce vašich technologií ihned čerpat z obrovské celosvětové studnice vědomostí a zkušeností, které vám pomohou řešit problémy. (I když se o techniku stará firma, také někde hledá řešení neznámých problémů. A kde jinde než na internetu?)

4.1.3 Inovace

Slovo používané spíše ve firemním a korporátním prostředí má svůj význam také ve školách. Učitelé a studenti získávají obrovské možnosti a know how. Mohou si zdarma nainstalovat, pokud o to mají zájem, množství softwaru, za který by jinde museli platit. Technicky nadaní studenti mají v rukou nástroje pro programování včetně dokumentace, mohou využívat software, který se běžně nasazuje v komerční praxi. Pro matematiku, fyziku a další přírodní vědy jsou k dispozici sofistikované nástroje, díky kterým je možné provádět složité výpočty, generovat grafy nebo simulovat pokusy. Studenti získávají nové možnosti pro středoškolskou odbornou činnost a získané informace a dovednosti mohou konzultovat se zahraničními kolegy. To vede k upevňování komunikačních kompetencí – open-source software je totiž nadnárodní fenomén a znalost světového jazyka je jeho přirozenou součástí.

[20] http://www.openoffice.cz/pouzivaji-openoffice-org/gymnazium-jevicko-pouziva-openoffice-org

[21] http://en.wikipedia.org/wiki/Patch_Tuesday



Obr. 4: Katalog vzdělávacího softwaru na LinuxEXPRES.cz – http://www.linuxexpres.cz/software-pro-skoly

4.1.4 Množství softwaru zdarma

Srovnávat nabídku softwaru pro Windows a Linux je ošemetné a je vhodné přiznat, že mnoho výrobců softwaru linuxovou platformu ignoruje, jak jsme zmínili výše. Podle našich zkušeností většina uživatelů vyžaduje běžně dostupný software, který je součástí linuxových operačních systémů a problém s nedostatkem vhodných nástrojů zde nevzniká. Linuxový operační systém je *plně vybavený* a na rozdíl od konkurence už obsahuje množství programů. Typické nástroje jsou:

webové nástroje – Firefox, Chrome (webový prohlížeč)

komunikace – Thunderbird (pošta), Skype (telefonování přes internet), Pidgin (ICQ, jabber, instant messaging);

kancelář – OpenOffice.org (univerzální kancelářský balík), Scribus (DTP a sazba);

grafika – F-Spot, gThumb a další (prohlížeče obrázků), Digikam (správa fotek), GIMP (bitmapový editor), Inkscape, OpenOffice.org Draw (vektorový editor);

multimédia – Rhythmbox (přehrávač hudby), Totem, SMplayer, VLC (přehrávač videa), Audacity (střih zvuku), Openshot (střih videa)

správa souborů – Nautilus, Dolphin, GNOME Commander, Krusader (správa souborů), Ark (komprimace, správa archivů)

4.2 Uvažujte, porovnávejte, hledejte výhody

Open source není zázrak ani řešení celosvětových problémů. Je to model vývoje softwaru, myšlenkový proud, obchodní princip. Bylo by nerozumné jej ignorovat. Už nyní je jasné, že má své výhody i nevýhody. Zkuste najít výhody a využijte jich. Nemůžete ztratit. Můžete získat.

> S Linuxem pak souzní slova jako dobrodružství, nadšení, spolupráce, komunita, pomoc, občas cesta proti proudu. A mezi studenty jsou jistě „hledači“, kteří v podobných kategoriích nacházejí lidské štěstí. Bylo by tedy dobré, aby Linux jako možnou součást této své životní cesty mohli poznat právě ve škole.
>
> – Ludmila Chládková, učitelka IT

PUBLIKACE V OPEN SOURCE KVALITNĚ A EFEKTIVNĚ

RYBIČKA, Jiří, (CZ)

Úvod

Zpracování textů se dlouhodobě jeví jako nejrozšířenější úloha zejména na osobních počítačích. Prakticky ze dne na den byly opuštěny tzv. „klasické“ technologie horké sazby z kovových písmen a nahrazeny počítačovými programy, které se snaží s různým úspěchem předstírat, že umí vysázet dokument tak, jak to uměli profesionální sazeči v tiskárně. Šmahem byly vyměněny psací stroje používané více než sto let pro přípravu rukopisů a dokumentů menšího rozsahu a nákladu za počítače pracující s knižním písmem.

Všichni ale víme, že samotný program nestačí – je k tomu potřeba i kvalifikovaný uživatel. A zde nastal zásadní rozpor: nástroj pro sazbu se dostal do rukou prakticky každému, kvalifikace však jen málokomu. Masivně se tedy vytvářejí dokumenty, které nesplňují základní kritéria sazby, protože v tom lepším případě se pro jejich úpravu používají pravidla pro psací stroje, jejichž obecná znalost a dostupnost je vyšší než znalost a dostupnost pravidel sazby.

Navíc zde hraje roli i efektivnost přípravy: kdo by se „páral“ s nějakými speciálními mezerami, pomlčkami, zarovnáváním a rozpalem? Přece „stačí“ naházet jakkoliv text na stránky – a je to!

Je jasné, že ti uživatelé, kterým není všechno jedno, se zamýšlejí, jak tuto masově používanou aplikaci zvládnout co nejjednodušším a nejlevnějším způsobem. Těm je určeno několik myšlenek tohoto článku.

Kvalita

Kvalitu můžeme chápat jako míru shody s všeobecně uznávanými pravidly a doporučeními. Jedná se o shodu s typografickými pravidly, pravidly pravopisu, normami a případnými

dalšími doporučeními. Klíčovým problémem v této souvislosti je, zda tvůrce dokumentu umí své požadavky formulovat v souladu s uvedenými pravidly a doporučeními. Vlastnosti použitého programového systému mohou v tomto směru pomoci, ale i klást významné překážky.

Chceme-li vytvořit dokument kvalitně a zároveň také efektivně, můžeme u každého uvažovaného programového systému sledovat uvedená kritéria a snažit se dosáhnout u každého z nich co nejpříznivějšího stavu. Lze říci, že každý systém vyžaduje poněkud jiný přístup a hodně záleží na uživateli, co preferuje, co je schopen a ochoten do řešení vložit a jak je zainteresován na výsledku.

Z počátečního pohledu možná vyplývá, že hlavní část kvalitativních parametrů dokumentů je dána především typografickými pravidly. Ne že by například pravidla pravopisu byla druhořadá, ale v dodržování pravopisu nejsou mezi programovými systémy prakticky žádné rozdíly, stejně tak se jen nepatrně podílí zvolený program na dalších kvalitativních nárocích, jako například na způsobu zápisu bibliografických citací apod.

Typografie je tedy z hlediska tvorby dokumentů poměrně zásadní a tento fakt platí v dnešní době nesrovnatelně více než například před 25 lety, protože narozdíl od minulosti dnes upravují dokumenty lidé, kteří neprošli žádnou typografickou průpravou, přestože disponují dokonalými prostředky pro tvorbu dokumentů, o jakých se dřívějším úpravcům a typografům ani nesnilo.

Řešení kvality je z velké většiny závislé nejspíše na tom, kolik toho uživatel o kvalitě (zejména typografické) ví a do jaké míry je ochoten tato pravidla respektovat a ve svých dokumentech také realizovat.

Efektivnost

Efektivnost je spojena s několika aspekty. Co pravděpodobně všichni chceme?

- ušetřit peníze,
- ušetřit čas,
- zjednodušit práci.

Co je potřeba udělat, abychom jednotlivé cíle splnili?

Ušetřit peníze

Používání volně dostupného softwaru primárně šetří peníze a je velmi důležité v sektorech, kde nejsou obvykle k dispozici finanční prostředky pro pořizování softwaru (například ve školství).

Nejpoužívanější volně dostupné programy pro zpracování textů bezesporu jsou OpenOffice Writer, TEX a jeho nadstavby, Scribus. Tyto programy se velmi liší svým zaměřením, pojetím a všeobecnou znalostí mezi uživateli.

Při používání volně dostupných programů se setkáváme většinou s intuitivně chápanými kritérii, která můžeme shrnout do jedné věty: Uživatel zvolí takový program, v němž je schopen a ochoten pracovat (například preferuje interaktivní ovládání). Obvykle první program, se kterým se seznámí, je ten, ve kterém pak pracuje stále a používá ho na všechny aplikace.

Ušetřit čas

Čas je potřebné počítat ve dvou rovinách:

- čas potřebný k naučení a ovládnutí daného programového systému,
- čas potřebný k realizaci určitého dokumentu v daném systému.

Uvažujeme-li obě části, je zřejmé, že časový nárok na ovládnutí systému je investován jednorázově, zatímco čas věnovaný realizaci dokumentů je nutné vynakládat opakovaně. Optimalizace tedy může spočívat v tom, že ovládnutím většího počtu funkcí nebo změnou přístupu k práci na dokumentu lze ušetřit opakované časové ztráty na jednotlivých dokumentech.

Uživatelé však obvykle nemají dostatek trpělivosti k tomu, aby se důkladně seznámili s nástrojem, který používají. Dalším negativním aspektem jsou neustálé změny verzí a uživatelských rozhraní různých produktů, které vedou k nutnosti zvýšit a opakovaně vynakládat časovou investici věnovanou k ovládnutí systému.

Zde asi ani nerozhoduje, zda pracujeme se systémem volně dostupným nebo komerčním, přístup uživatelů je velmi podobný. Komerční programy se však častěji uchylují ke změnám uživatelského rozhraní, protože je to jeden z nástrojů konkurenčního boje a jeden z prostředků opakovaného získávání licenčních poplatků.

Zjednodušit práci

Efektivnost lze intuitivně pochopit i tak, že dosáhneme požadovaného cíle pomocí menšího počtu méně náročných kroků. To souvisí jednak se schopností přenést těžiště práce na počítač, jednak s optimálně zvolenou strategií postupu celého zpracování dokumentu.

V tomto bodě je zcela zřejmé, že problém spočívá ve vybavenosti uživatele vhodnými informacemi, znalostmi a dovednostmi. Těžko přesvědčíte jakéhokoliv autora, že má ke zpracování dokumentu používat vlastní formátovací styly, když neví, co to je a jak se to používá. Bohužel většina příruček k programovému vybavení neřeší kvalitu a možnosti použití jednotlivých nástrojů, protože jen radí, *jak* něco udělat, ale neradí, *proč* právě tuto službu zvolit.

Stejně nevýhodné je používat určitý programový systém pro tvorbu dokumentu, který se svým charakterem vymyká celému zaměření programu a klade na uživatele nepřiměřené nároky při přípravě. Bez schopnosti pracovat ve více různých systémech však tento uživatel nemá jinou volbu a často ani netuší, kde dělá zásadní chybu.

Dokumenty v systému LaTeX kvalitně a efektivně

Uvedli jsme obecně několik aspektů kvality a efektivnosti, soustřeďme se nyní na konkrétní řešení. Takový rozbor bychom mohli (a také měli) udělat pro jakýkoliv systém, který pro zpracování textů používáme nebo máme v úmyslu použít. Z toho pak vyplyne jedna velmi důležitá informace – pro jaký druh dokumentů je daný systém optimální, jakých jeho vlastností si máme všímat a kde lze spatřit nejefektivnější nástroje pro zpracování dokumentu.

Celý postup budeme ilustrovat na systému TeX, přesněji řečeno na jeho nadstavbě LaTeX. V tomto místě nemáme dost prostoru na podrobné seznámení (potřebné informace lze nalézt v četné literatuře), budeme se proto zabývat spíše obecnějšími vlastnostmi.

Kvalita dokumentů a její dosažení

O typografických pravidlech bychom mohli pojednávat na stovkách stránek a o nejrůznějších prvcích včetně jejich možné realizace v počítači byla publikována řada knih a nepřeberné množství elektronických dokumentů. Zaměříme se tedy pouze na některé prvky, jejichž řešení je typické pro systém LaTeX.

Písma a speciální znaky, odstavcová sazba

Kvalitní dokument vysázený knižním písmem splňuje nejnáročnější kritéria *čitelnosti*. Ta je podmíněna precizní sazbou, tedy rovnoměrným rozmístěním světla mezi znaky ve slovech, mezi slovy v řádcích, mezi řádky a odstavci stránky. I ve zcela hladkém textu musí být pro dosažení správného výsledku používána řada nejrůznějších rekvizit, počínaje precizní kresbou písma, přes dostatečné informace o znakovém vyrovnání a slitcích až po promyšlenou distribuci mezislovních mezer v odstavci. V tomto směru lze zcela jednoznačně říct, že systém LaTeX disponuje pravděpodobně nejpropracovanějším algoritmem odstavcové sazby, který může být řízen řadou parametrů, a může tak vyhovět těm nejvyšším nárokům na preciznost a kvalitu výstupu. Použitá písma v základní distribuci pokrývají běžné potřeby a na rozdíl od jiných systémů nevykazují lokalizační a jiné nedostatky či snad hrubé chyby. Jedno z nejlépe propracovaných písem – Knuthovo Computer Modern – je v moderních distribucích nahrazeno rodinou Latin Modern, obsahující především precizně kreslené akcentované znaky a opravující drobné chyby původních Knuthových vzorů. Rodina obsahuje knižní písmo serifové a bezserifové v běžných řezech a také písmo strojopisné. Specialitou je škálování písma – v žádném jiném systému není v různých velikostech písma používán rozdílný obraz vycházející z optimalizace čitelnosti. Kresba velkých stupňů je tedy odlišná od kresby malých stupňů, písmo této základní rodiny je tedy precizní ve všech aplikacích.

Ve speciálních znacích má systém LaTeX rovněž značný náskok před jakýmkoliv jiným programovým systémem. Vysoce kvalitní sazba předpokládá celou škálu rekvizit, kterými systémy založené na bázi TeXu běžně disponují. Jde o široký repertoár mezerových výplňků (kromě předdefinovaných je možné jakýkoliv myslitelný výplněk vytvořit), dále pomlčky

a správně se chovající spojovníky (zdvojování při řádkovém zlomu), uvozovky v několika variantách a širokou paletu znaků a symbolů používaných v odborných textech.

Stránky a sestava dokumentu

Stránkový design je ovlivňován především samotným sazebním materiálem. Jedná se o přístup, kdy prvotní je velikost materiálu, druhotně se pak odvozuje stránkové rozložení. Na rozdíl od tohoto konceptu je v jiných typografických systémech naopak prvotní stránkové rozložení, do něhož se pak jako do připravených „nádob“ takzvaně „nalévá“ sazební materiál.

Otázkou je, který z obou přístupů je lepší. Lze jednoznačně prohlásit, že tato otázka je závislá pouze na charakteru dokumentu. Pro dokumenty typu kniha, odborná zpráva, vědecký článek v časopisu apod. je zcela bezkonkurenčně lepší přístup LaTeXový, naopak pro reklamní leták, jehož tvorba spíše připomíná kreslení v obrazovém editoru, je lepší zmíněný přístup alternativní. V tomto ohledu je vhodné podle charakteru dokumentu zvolit optimální systém pro zpracování.

Řízení stránkového zlomu v systému LaTeX je doplněno o jedinečný koncept plovoucích objektů, tj. prvků, jejichž finální umístění není dáno uživatelem, ale rozmisťovacím algoritmem, který při vhodném nastavení výrazně usnadňuje sesazení celého dokumentu. Nejčastěji se takto obsluhují obrázky a tabulky, patří sem i marginálie (poznámky na okraji). Zcela automaticky pak fungují i poznámky pod čarou, to však není vzhledem k ostatním systémům nic výjimečného.

Koncept sestavy dokumentu uvedeným způsobem velmi kvalitně realizuje vysazení stránek. Lze přesně řídit meziodstavcové mezery, nadpisy a jejich umístění v rámci textu, mezery mezi textem a dalšími objekty, a to vše zcela jednotně, případně s možností určitého akceptovatelného rozmezí. Lze tak dodržet základní typografická pravidla jednotnosti designu a rovnoměrného vyplňování stránek.

Netextové objekty

Pokud jde o matematické výrazy a jejich obdobu ve fyzikálních a jiných odborných textech, lze s jistotou konstatovat, že neexistuje jiný programový systém, který by tuto problematiku řešil kvalitněji a efektivněji. Propracovaný systém matematické sazby s mnoha rekvizitami a přídavnými balíky umožňuje splnit ta nejnáročnější kritéria.

Tvorba tabulek je naproti tomu pravděpodobně nejméně propracovaným prvkem v systému TeX i LaTeX. Přestože existuje poměrně velká skupina rekvizit a přídavných balíků, je v mnoha případech vytvoření tabulky těžkopádné a některé prvky nejsou dostupné vůbec. Na druhou stranu je však potřebné zdůraznit, že pro běžné potřeby tyto rekvizity zcela dostačují a v určitých prvcích lze opět získat kvality, které nejsou v jiných systémech vůbec dostupné, nebo jsou realizovatelné s velkými obtížemi a nároky. Je tedy možné i v oblasti tabulek dosáhnout vysoké typografické kvality.

Obrázky lze do dokumentu v systému LaTeX včlenit několika způsoby. Většinou se předpokládá, že hotový obrázek v rastrovém formátu JPG nebo PNG, či vektorovém formátu

PDF bude vložen do připraveného vynechaného místa dokumentu, to však obvykle není zařízeno přímou sazbou, ale až výstupním ovladačem. Tento způsob práce nepředstavuje velké omezení, neboť samotná příprava obrazu může být realizována jakýmkoliv vhodným obrazovým editorem, který má pro tuto činnost zcela optimální vybavení. Kromě vložení externě připraveného obrazu existují určité omezené možnosti kreslení vektorového obrazu pomocí jednoduchých prvků, což je často využíváno zejména pro jednoduchá schémata nebo nelineární rozmístění libovolných prvků na plochu stránky.

Efektivnost tvorby dokumentů

Uvažujeme-li o efektivní výrobě dokumentů, určitě pro každý programový systém platí, že její podmínkou je především

1. dokonalé využití existujících rekvizit a předdefinovaného způsobu práce systému.

Efektivnost má v systému LaTeX oproti jiným systémům ještě ovšem navíc další rovinu –

2. téměř neomezené možnosti rozšíření a modifikace příkazů a přizpůsobení požadované aplikaci.

Tím se diametrálně odlišuje od jiných systémů, které se uživateli jeví jako uzavřené, tedy množina jejich funkcí je předdefinovaná a konečná.

Systém LaTeX poskytuje uživateli možnost optimálně řešit každý typ dokumentu podle dané situace, přizpůsobit zpracování možnostem vstupu dat a reprezentovat dokument čistě strukturními značkami. Ilustrujme si tyto možnosti na následujících příkladech.

Příklad 1

Z databázového systému získáme soubor `data.csv` a v něm dostáváme tabulku s údaji o zboží, kterou potřebujeme vhodně vysázet. Každý řádek vstupu obsahuje pět údajů (název, počet kusů, nákupní cena bez DPH, prodejní cena bez DPH, sazba DPH), oddělených znakem svislá čára, která je před prvním i za posledním údajem řádku, např.:

```
|Zubní pasta Microsilver     |   2|  130|   167|  20|
```

Takový vstup stačí upravit pro jednodušší následné zpracování automatickým doplněním jednoho příkazu, například:

```
\zbozi|Zubní pasta Microsilver     |   2|  130|   167|  20|
```

K takovému vstupu připravíme soubor:

```
\def\zbozi|#1|#2|#3|#4|#5|{%
      #1 & #2 & #3 & #4 & #5 \\ \hline}
\begin{longtable}{|l|*4{r|}} \hline
Název & Kusů & Nákup & Prodej & DPH \%
      \\ \hline
\endhead
\multicolumn{5}{|r|}{(pokračování na další straně)}
      \\ \hline
\endfoot
\endlastfoot
\input{data.csv}
\end{longtable}
```

Příklad 2

Potřebujeme udělat sbírku příkladů do fyziky. Protože nebudeme vynalézat kolo, posbíráme osvědčené příklady z různých sbírek. Tuto jednotvárnou práci chceme zadat „levné pracovní síle“. Potřebujeme tedy, aby zápis byl co možná nejjednodušší, například takto:

```
\priklad{Vypočtěte, kolik metrů je vysoká věž, z~jejíž střechy
spadla taška až na zem za 3 vteřiny.}{45 metrů}
```

Ve vstupním textu je uvedeno pouze to nejnutnější, aby bylo možné pochopit, o co se jedná. Nejsou zde uvedeny žádné informace o výsledném tvaru. Uvedené značky (příkazy) se nazývají strukturní a strukturní značkování je prvním základním krokem na cestě k efektivnímu dokumentu.

V této fázi stačí tu zmíněnou pracovní sílu pouze instruovat, že každý příklad začne zápisem `\priklad`, že zadání příkladu je v jedné svorce a jeho výsledek je ve druhé svorce. Vůbec nic nemusí vědět o tom, jak bude takový příklad vypadat ve vysázené sbírce, ani nemusí nic vědět o tom, že úplně stejný příklad bude třeba použit pro promítání přes dataprojektor ve výuce nebo bude figurovat jako doplněk k výkladu v učebnici.

Ale i kdybychom sbírku dělali zcela sami a neměli v úmyslu žádné další aplikace, přesto se vyplatí vstup zapisovat stejným způsobem. V každém případě totiž můžeme od sebe oddělit dva procesy – pořizování a úpravy textu, kdy autor uvažuje víceméně pouze o obsahu, a řešení vizuální podoby, kterou může dělat kdokoliv jiný a také neomezeně mnoha způsoby.

Technicky to provedeme doplněním definice makra `\priklad` například takto:

```
\newcounter{ciprikladu}[section]
\def\thecisprikladu{%
        \thesection.\arabic{ciprikladu}}
\def\priklad#1#2{\stepcounter{ciprikladu}
```

```
\par\medskip
\textbf{Příklad \thecisprikladu:} #1 \par
\smallskip \hspace*{\fill}\textit{(#2)}}
```

V tomto jednoduchém případě jsme vytvořili čítač pro číslování příkladů, nařídili jsme jeho nulování v každé sekci a jeho číslo společně s číslem sekce vypisujeme na začátku každého příkladu. Příklady začínají tučným slovem „Příklad“ a odpovídajícím číslem, pak následuje text zadání, za příkladem pak vpravo kurzívou v závorce výsledek.

Lze si snadno představit, že v jiné modifikaci úpravy nemusíme výsledky příkladu vůbec chtít vypisovat, můžeme je dokonce nechat vypisovat v úplně jiné části publikace (například souhrnně v poslední kapitole), zadání příkladů můžeme číslovat jiným způsobem (například při tvorbě písemky můžeme mít „číslování“ A, B atd. a každý příklad může být na samostatné stránce bez výsledků) a podobně.

Co si z rozboru odneseme?

Ze stručného seznámení se systémem LaTeX by měly vyplynout tyto informace:

- Zaměření a základní princip – systém preferuje design sazebního materiálu před designem stránek (prvotní je tedy text, nikoliv předdefinovaná úprava stránky), takže text postupně vyplňuje a tvoří stránky, nenalévá se do připravených stránkových šablon.

- Kvalitativní kritéria – jsou k dispozici prostředky pro precizní sazbu, nepřeberné množství speciálních znaků, neomezené možnosti mezerování, pravděpodobně nejlepší algoritmus odstavcového zlomu, ideální podpora matematických výrazů.

- Efektivnost – systém je volně dostupný (finanční nároky na pořízení jsou nulové včetně pořízení potřebných podpůrných programů), podporuje v plné míře strukturní značkování, umožňuje automatizovat sazbu, je velmi vhodný pro rozsáhlé dokumenty, vyžaduje investici do zvládnutí tvorby vlastních příkazů.

- Omezení – nemá příliš rozsáhlé možnosti stránkového designu, nepracuje obvykle v interaktivním uživatelském rozhraní (nepracuje s náhledem dokumentu – to však vadí jen úplným začátečníkům), má některá omezení v tvorbě tabulek.

Od této chvíle je tedy jasné, kdy a na jaké dokumenty lze systém s úspěchem použít. Bude tedy například velmi efektivní pro sazbu sborníku (opakované úkony s každým článkem, nepříliš složitý stránkový design), ale při tvorbě jednostránkového reklamního letáku se neúměrně nadřeme (chybí propracované funkce pro definici stránkového designu, málo textu, málo možností automatizace).

Pohled do budoucna

Na základě uvedeného příkladu jednoho z mnoha různých systémů jsme předvedli, na co všechno by se měl uživatel soustředit, aby se správně rozhodl pro takový programový systém, který optimálně vyhovuje danému typu dokumentu.

Jenže – který uživatel má čas na to, aby takové rozbory prováděl? Je vůbec efektivní se tím zabývat? Systémů je mnoho a neustále se mění – je možné mít neustále přehled?

Přestože znalosti a informace jsou pro efektivní zpracování dokumentů zcela nezbytné, je na všechny tyto otázky nejčastější odpověď „ne“. Proto by bylo ideální, kdyby tyto rozbory někdo kvalifikovaný udělal dopředu a uživatel mohl získaných výsledků rychle využít.

Tuto myšlenku podporuje navíc možnost rozhodování přenechat počítači, který na základě relevantních informací uživateli nabídne, co má pro svůj konkrétní dokument použít a také *jaké funkce* bude potřebovat a kde se s nimi seznámí.

Na Ústavu informatiky PEF Mendelovy univerzity v Brně vzniká rozhodovací systém pro volbu optimálního programového systému. Principem tohoto systému je, že uživatel zvolí druh dokumentu, který má v úmyslu zpracovat, podle potřeby upřesní požadavky na dokument a svoje vlastní preference (například práce v systému Linux) a rozhodovací stroj odpoví, kterým programem je optimálně daný dokument zpracovatelný a vypíše všechny funkce, které budou potřeba. Ke každé použité funkci je k dispozici nápověda, jak tuto funkci v daném programu použít a kde ji najít. Uživatel tedy nemusí dlouze studovat daný systém, dostane ke svému dokumentu odpovídajícího průvodce.

V současné době je tento rozhodovací systém navržen, přistupuje se k jeho realizaci a shromažďují se potřebná data.

Závěr

Volně dostupné programy nabízejí v oblasti zpracování textů široké možnosti. Chceme-li v této aplikační oblasti postupovat tak, abychom dosáhli co nejkvalitnějšího výsledku s co nejmenším úsilím, musíme jednotlivé programové systémy dostatečně důkladně poznat a ohodnotit.

Příkladem pro zpracování dokumentů určitých typů je systém TEX a jeho nadstavby, kde nejrozšířenější je bezesporu LATEX. V tomto systému se nacházejí unikátní nástroje a prostředky pro dosažení maximálně kvalitního výstupu, jako každý jiný systém má však i svá omezení.

Efektivní zpracování dokumentů tedy nespočívá v absolutní preferenci jednoho programového řešení, ale spíše ve vhodné kooperaci různě zaměřených a různě vybavených programů, z nichž je potřebné využívat ty nejefektivnější funkce. K tomu je nezbytné se alespoň rámcově seznámit s několika reprezentanty, které dostatečně ilustrují možnosti a omezení celé skupiny podobných programů.

Pro optimální rozhodnutí o zpracovávajícím programovém vybavení vzniká automatizovaný systém, jehož principem je poskytnutí podstatných informací pro řešení konkrétního

dokumentu konkrétního uživatele s danými požadavky a preferencemi.

Literatura

- RYBIČKA, J. *LaTeX pro začátečníky*. Brno: Konvoj, 2003.
- RYBIČKA, J. *Typografie a zpracování textů v programu Word* [online]. 2008. [cit. 10. 6. 2010] Dostupné na `http://akela.mendelu.cz/~rybicka/prez/zpract/jaknaword.rtf`
- RYBIČKA, J., TALANDOVÁ, P., PŘICHYSTAL, J. Teoretický model systému pro optimalizovaný proces výběru programového vybavení. *Acta Universitatis Mendelianae Brunensis*, 2010, č. 2 (v tisku)

Kontaktní údaje

Doc. Ing. Jiří Rybička, Dr.,
Ústav informatiky, Provozně ekonomická fakulta,
Mendelova univerzita v Brně, Zemědělská 1, 613 00 Brno, Česká republika,
e-mail: `rybicka@mendelu.cz`

SPONZORSKÁ PREDNÁŠKA

ÚČETNÍ A EKONOMICKÝ SYSTÉM WinStrom FlexiBee

FERSCHMANN, Petr, (CZ)

Jednou z věcí, která dlouho bránila firmám přejít na Linux, byla absence účetnictví. To se ovšem změnilo a multiplatformní účetní a ekonomický systém FlexiBee vstupuje na Slovenský trh. Při této přednášce krátce představíme tento systém, proč by Vás měl zajímat i když nepoužíváte Linux. Popíšeme, jak produkt vznikal, proč podporuje Linux a co nás k tomu vedlo. Ukážeme si také práci systému přes Internet a popíšeme programátorské rozhraní založené na REST API.

Petr Ferschmann je vedoucím vývojového oddělení společnosti WinStrom s. r. o., kde má na starosti vývoj multiplatformního ekonomického systému WinStrom FlexiBee. Je také dlouholetým uživatelem Linuxu (od roku 1997).

OCENENÉ ŠTUDENTSKÉ PRÁCE

CENA ZA ŠTUDENTSKÉ PRÁCE

SOIT (SK) a Liberix (CZ)

Abstrakt. Spoločnosť pre otvorené informačné technológie a spoločnosť Liberix vyhlásili súťaž o najlepšiu bakalársku a diplomovú prácu súvisiacu s otvoreným softvérom a ďalšími otvorenými technológiami. Cieľom súťaže bolo stimulovať záujem o používanie OIT na školách, pretože sú to práve školy, na ktorých je propagácia progresívnych technológií najúčinnejšia. Vypísaním Ceny chcú organizátori vysloviť uznanie pedagógom, ktorí sa touto formou venujú propagácii OIT.

1 Úvod

Cieľovou skupinou Ceny boli slovenské a české univerzity so zameraním najmä na informačné technológie. Okrem diplomu získali laureáti, ktorí sa umiestnili na prvých troch miestach, aj vecnú cenou, a to rovnako študent ako aj vedúci práce. Udeľovanie Ceny SOIT a Liberixu je súčasťou konferencie Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach. Posudzovateľmi prác boli:

Juraj Bednár
Ján Buša
Slavko Fedorik
Andrej Ferko
Roman Fordinál
Pavol Lupták
Marek Mahút
Milan Moravčík
Miloslav Ofúkaný
Jakub Ondrušek
Michal Páleník
Viliam Solčány
Miloš Šrámek
Pavel Stříž

2 Laureáti Ceny

Odborná komisia rozhodla o prvých troch miestach a laureátoch Ceny takto:

1. cena

Native support for deb packages in Spacewalk (Diplomová práca)

- Autor: Lukáš Ďurfina
- Vedúci: Radek Kočí
- Škola: Vysoké učení technické v Brně, Fakulta informačních technologií

Abstrakt. The system Spacewalk is a management tool for the Linux operating systems based on RPM package manager. The aim of thesis is adding support to Spacewalk for DEB package management system, which is connected with Debian, a distribution of Linux operating system. The result is native support of managing Debian system by the Spacewalk, what includes a registration of system, distribution of configuration files, remote scripts running and management of DEB packages.

2. cena

Miniaplikace v desktopových prostředích KDE a GNOME (Bakalárska práca)

- Autor: Petr Šigut
- Vedúci: Marek Grác
- Škola: Masarykova univerzita, Fakulta informatiky, Brno

Abstrakt. Cílem bakalářské práce bylo prozkoumat stav miniaplikací v Linuxu, konkrétně v prostředích KDE a GNOME. Dále se seznámit s prostředky pro získávání informací z webových stránek a webových služeb. Po prozkoumání dostupných poznatků k dané problematice jsme navrhli a implementovali klient-server systém pro zjednodušení vývoje nových miniaplikací.

Práca bola vypracovaná v spolupráci s Red Hat Czech, s. r. o.

3. cena

Problém N telies – metódy a open source softvér (Bakalárska práca)

- Autor: Miroslav Kvaššay
- Školiteľ: Michal Kaukič
- Škola: Žilinská univerzita, Fakulta riedenia a informatiky, Žilina

Abstrakt. Obsahom bakalárskej práce je teoretické spracovanie najznámejších metód, ktoré boli vyvinuté na numerické riešenie problému *N* telies, a praktické otestovanie implementácií niektorých z uvedených metód. Tematicky je práca rozdelená na tri časti, pričom prvá časť predstavuje fyzikálny pohľad na problém *N* telies, druhá časť sa zaoberá vybranými metódami riešenia (pohľad numerickej analýzy) a napokon tretia časť sa zaoberá testovaním implementácií zvolených metód (softvérový pohľad na riešenie problému *N* telies).

3 Prihlásené práce

Okrem ocenených prác, boli do súťaže prihlásené tieto práce (bez poradia):

Anton Mäčko: *Riešenie informačnej infraštruktúry v prostredí Strednej odbornej školy elektrotechnickej v Žiline* (Bakalárska práca)

- Tútor: Peter Palúch
- Vedúci: Ján Király
- Škola: Fakulta riadenia a informatiky, Žilinská univerzita, Žilina

Martin Kompán: *Diskrétna simulácia s použitím modulu SimPy v Pythone* (Bakalárska práca)

- Školiteľ: Mgr. Michal Kaukič. Csc.
- Škola: Fakulta riadenia a informatiky, Žilinská univerzita, Žilina

Jozef Kňažko: *Strategická počítačová hra* (Bakalárska práca)

- Školiteľ: Mgr. Michal Kaukič. Csc.
- Škola: Fakulta riadenia a informatiky, Žilinská univerzita, Žilina

Jozef Hartinger: *Integration of JBoss Seam and RESTEasy* (Bakalárska práca)

- Školiteľ: Ing. Mgr. Zdeněk Říha, Ph.D.
- Škola: Masarykova univerzita, Brno

Ivan Kolesár: *3D vizualizácia drevených kostolíkov* (Bakalárska práca)

- Školiteľ: doc. RNDr. Andrej Ferko, PhD.
- Škola: Fakulta matematiky, fyziky a informatiky, Univerzita Komenského , Bratislava

Pavel Raiskup: *Rozšíření implementace protokolu ftp v knihovně libcurl* (Bakalárska práca)

- Školiteľ: Ing. Kamil Dudka
- Škola: Fakulta informačních technologií, Vysoké učení technické v Brně,

Ondřej Vadinský: *SELinux* (Bakalárska práca)

- Školiteľ: Radomír Palovský
- Škola: Vysoká škola ekonomická, Praha

Radim Hatlapatka: *JBIG2 komprese* (Bakalárska práca)

- Školiteľ: doc. RNDr. Petr Sojka, Ph.D.
- Škola: Masarykova Univerzita, Brno

Martin Kolman: *Flexibilní GPS navigace pro mobilní zařízení s OS Linux* (Bakalárska práca)

- Školiteľ: RNDr. Aleš Horák, Ph.D.
- Škola: Fakulta informatiky Masarykovy univerzity, Brno

Andrej Vanko: *Pohyblivý 3D model ľudského tela* (Bakalárska práca)

- Školiteľ: RNDr. Robert Bohdal, PhD.
- Škola: Fakulta matematiky, fyziky a informatiky, Univerzita Komenského , Bratislava

Peter Schiffer: *Social desktop* (Bakalárska práca)

- Školiteľ: Ing. Jozef Mlích
- Škola: Fakulta Informačních technologií , Vysoké učení technické v Brně

Ivan Šišák: *Nástroj na vizualizaci databázového modelu existující databáze* (Bakalárska práca)

- Školiteľ: Ing. Maroš Barabas
- Škola: Vysoké Učení Technické v Brně, Fakulta informačních technologií

Peter Pajta: *Rozoznávanie gest v aplikácii UML.FRI* (Diplomová práca)

- Školiteľ: : Ing. Tomáš Bača
- Škola: Fakulta riadenia a informatiky, Žilinská univerzita, Žilina

Peter Považanec: *Optimalizácia softvérového riešenia prognózovania časových radov ekonomických dát* (Diplomová práca)

- Školiteľ: Ing. Lucia Pančíková, PhD.
- Škola: Fakulta riadenia a informatiky, Žilinská univerzita, Žilina

Peter Vilhan: *Prostriedky pre správu vzdialených zariadení cez web rozhranie* (Diplomová práca)

- Školiteľ: Ing. Branislav Steinmüller
- Škola: Fakulta informatiky a informačných technológií, Slovenská technická univerzita v Bratislave

LaTeX NIE JE FARBA NA MAĽOVANIE

BLAŠKO, Rudolf, (SK)

Abstrakt. Príspevkom chceme popularizovať používanie systému LaTeX širokou verejnosťou, nielen vedeckou a odbornou obcou. Na začiatku porovnáme jeho výhody a nevýhody, a následne ponúkneme návod, ako môže aj úplný laik vytvárať v tomto systéme svoje jednoduché dokumenty.

1 Úvod

Poznáme mnoho programov, pomocou ktorých môžeme prezentovať v písanej alebo lepšie povedané v grafickej forme. Niektoré sú komerčné, niektoré nie. Niektoré sú viac pohodlné, niektoré menej. Niektoré sa dajú používať priamo bez akýchkoľvek znalostí, k niektorým sú potrebné aspoň elementárne poznatky a k niektorým je potrebné zložiť „maturitu“. Niektoré majú minimálne nároky na softvér a hardvér a niektoré naopak. A z niektorých sa normálny užívateľ s priemernými alebo podpriemernými počítačovými zručnosťami môže zblázniť. Pre jednoduchosť budeme všetky tieto produkty nazývať programy.

Medzi najrozšírenejšie programy medzi verejnosťou patrí známy nemenovaný komerčný produkt a jeho voľne šíriteľný ekvivalent OpenOffice.org. Ich veľkou výhodou je jednoduchosť ovládania – samozrejme pokiaľ máte verziu v jazyku, ktorému rozumiete. Bez akýchkoľvek znalostí môžete písať. Problém je iba v tom, že musíte používať veľmi často myš. Pre bežného užívateľa je to vcelku priateľné riešenie. A to aj napriek tomu, že sú tieto programy „svojhlavé“ a mnohokrát vnucujú užívateľovi svoju vôľu a prispôsobenie tohto počínania na svoj obraz stojí veľkú námahu. Problém je aj v tom, že prvý program funguje iba na platforme operačného systému Microsoft Windows, ktorá je spoplatnená. Kompatibilita medzi uvedenými programami je vcelku dobrá, aj keď nie bezproblémová. Ale ako už bolo spomenuté, pre bežného užívateľa pre normálne kancelárske používanie, pokiaľ sa nekladú vysoké nároky na výstupnú úpravu, sú tieto programy postačujúce a vcelku vhodné.

Alternatívou k uvedeným programom je používať úplne bezplatný systém TEX, resp. jeho najpoužívanejší formát LATEX. Nevýhodou LATEXu je, že úplný laik nedokáže bez predchádzajúcich (síce minimálnych, ale predsa nejakých) znalostí tento program používať. Ďalšia vec, ktorá väčšinou odradí potenciálnych užívateľov, je to, že pisateľ hneď nevidí, čo píše. LATEX funguje podobne ako klasické programovacie jazyky. Napíše sa zdrojový súbor `*.tex` (je to obyčajný textový súbor), ktorý sa preloží prekladačom do `*.pdf` alebo `*.dvi` súboru a ten sa následne vhodným prehliadačom zobrazí. V praxi to nie je také zložité – postačia na to dve systémové okná – v jednom okne je zdrojový textový súbor a v druhom výsledný grafický súbor. Jedným príkazom alebo jedným kliknutím myši sa automaticky `*.dvi` súbor preloží a hneď sa aktualizuje výsledný súbor a vidíme výsledok svojej práce. Existujú aj programy (napr. TEXShell, WinShell, TEXMaker), ktoré tieto problémy riešia za užívateľa.

Existujú aj programy, ktoré fungujú tak, že človek hneď vidí svoje písanie, ale sú komerčné. Koniec koncov je to iba otázka zvyku a priemerne inteligentný človek s tým nemá žiadny problém. Výsledok stojí za to. Nezáleží na tom, v akom operačnom prostredí pôsobíme, aké kódovanie používame, koľko a akých jazykov naraz používame, aký prehliadač používame, akou tlačiarňou tlačíme výsledok a „výsledok je vždy rovnaký“ (samozrejme až na kvalitu a možnosti zobrazovačov, resp. tlačiarní). To znamená, že už napísaný zdrojový súbor môžeme bez problémov modifikovať a beztrestne prekladať v ľubovoľnom inom operačnom systéme a „výsledok je vždy rovnaký“.

Dôležitá je tiež skutočnosť bezpečnosti a jednoduchej možnosti archivácie našej práce. V predtým spomínaných programoch nám jeden zmenený bit znehodnotí celý súbor. Pri TEXu, resp. LATEXu stačí archivovať zdrojové súbory, ktoré sú (vrátane všetkých podporných a konfiguračných) textovými súbormi, to znamená ľahko čitateľnými súbormi a menej náchylnými na poškodenie. Nie nepodstatná je tiež veľkosť archivovaných zdrojových dát. Zdrojové súbory mojej poslednej knihy, ktorá vyšla ako vysokoškolská učebnica v tlačenej forme a mala asi 200 strán popísaných matematickým textom, sa zmestili na klasickú disketu (dnes už nepoužívanú), a spolu mali 1.159.741 bytov. Výsledný súbor `*.dvi` mal 2.884 kB, súbor `*.ps` mal 3.874 kB a súbor `*.pdf` mal 1.721 kB. Súbor `*.doc` (ak zamlčíme skutočnosť, že uvedená kniha sa nedá napísať v tomto formáte v uspokojivom tvare) by obsadil podľa môjho odhadu asi 12.000 kB, t. j. desaťtisíckrát viac!

V Linuxe a jeho klonoch je TEX a LATEX väčšinou súčasťou inštalácie, do ostatných operačných systémov ho treba doinštalovať. V súčasnej dobe to už nie je problém a inštalácia je pohodlná. Keďže prakticky každý národ (okrem Angličanov) má svoju špecifickú abecedu, sú aj rôzne národnostné modifikácie LATEXu. Pre češtinu a slovenčinu je to $\mathcal{C}_{\mathcal{S}}$LATEX.

Nadšenci a užívatelia TEXu sú združení v medzinárodnej organizácii TUG „TEX Users Group“ (Združenie užívateľov TEXu, `http://www.tug.org/`). V Čechách a na Slovensku je jeho súčasťou $\mathcal{C}_{\mathcal{S}}$TUG „Československé sdružení uživatelů TEXu“ (Czechoslovak TEX Users Group, `http://www.cstug.cz/`). Bezplatné inštalácie (TEX Live 2009, resp. TEX Live 2010 pretest) môže každý záujemca nájsť na `http://www.tug.org/texlive/`, resp. `www.tug.org/texlive/pretest.html`.

2 Začíname s LaTeXom

TeX (vyslovujeme „tech“) je počítačový program vytvorený profesorom Donaldom Ervinom Knuthom. Je určený pre sadzbu textu a matematických rovníc pri zachovaní vysokej typografickej úrovne výsledného dokumentu. LaTeX (vyslovujeme „la-tech“, resp. „lej-tech“) je otvorený systém rôznych preddefinovaných maker, ktoré zjednodušujú a zlepšujú prácu pri vytváraní výsledného dokumentu a na sadzbu výsledného textu využíva TeX.

Každý užívateľ (a nielen začiatočník) LaTeXu by mal mať vo svojej knižnici minimálne publikáciu *Ne příliš stručný úvod do systému LaTeX 2ε* [4], ktorá je voľne prístupná (vrátane zdrojových súborov) na internete a Rybičkovu knihu *LaTeX pro začátečníky* [5].

2.1 Zdrojový text

Hlavnou a najdôležitejšou činnosťou pri vytváraní publikácie v programe LaTeX je zápis zdrojových textov, t. j. súborov `*.tex`. Tieto súbory môžeme písať v ľubovoľnom editore, uložiť ich ale musíme ako obyčajné textové ASCII súbory bez formátovacích znakov. To znamená, že sa do nich nesmú dostať riadiace príkazy daného editora. V tomto súbore sú okrem vlastného textu aj „príkazy“, pomocou ktorých LaTeX vysádza výsledný dokument.

Neviditeľné znaky (medzera ␣, tabulátor, koniec riadku) sú spracované LaTeXom ako medzera. Viac po sebe nasledujúcich neviditeľných znakov má význam ako jedna medzera.

Prázdny riadok medzi textom označuje koniec odstavca (za prázdnym riadkom začína nový odstavec). Viac prázdnych riadkov za sebou je spracovaných ako jeden prázdny riadok.

Špeciálne znaky # $ & % ^ _ { } ~ \ majú pre LaTeX zvláštny význam. Pre ich aplikáciu v texte použijeme príkazy `\# \$ \& \% \^{} \_ \{ \} \~ \textbackslash`.

Príkazy v LaTeXe začínajú symbolom \. Za symbolom \ nasleduje špeciálny, resp. nealfanumerický znak (napr. `\␣ \, \& \| \[`) alebo nasleduje meno príkazu zložené výlučne z písmen, pričom sa veľké a malé písmena rozlišujú (napr. `\bf \TeX \end{document}`).

Povinné parametre príkazov sa ohraničujú zátvorkami {}. Ak má príkaz nepovinný parameter, ohraničuje sa zátvorkami []. V texte majú zátvorky [] svoj normálny význam.

Zátvorky `{}` majú špeciálny význam. Vo všeobecnosti ohraničujú parameter príkazu alebo ohraničujú skupinu. **Skupina** je úsek textu (vrátane príkazov), pre ktorý predpokladáme nejaké spoločné vlastnosti, resp. pre ktorý chceme nastaviť pôsobenie nejakého príkazu. Druhý spôsob pre ohraničenie skupiny je pomocou dvojice príkazov `\begin{meno}` a `\end{meno}`, kde `meno` je špeciálne slovo (napr. `center`, `document`, `displaymath`). Takto ohraničený úsek textu sa tiež nazýva **prostredie**.

Prostredia a skupiny sa môžu do seba ľubovoľne vnárať, ale nemôžu sa vzájomne krížiť.

Symbol percenta `%` má špeciálny význam pre poznámkovanie textu. Všetky znaky, ktoré stoja za symbolom %, sú až do konca riadku TeXom pri preklade ignorované.

2.2 Štruktúra dokumentu

Vstupný súbor je zdrojový súbor, na ktorý sa LaTeX nastaví a ktorý prekladá. Vo vstupnom súbore môžu byť odkazy na ďalšie zdrojové súbory, ktoré sa následne bez prerušenia preložia.

Keď LaTeX spracováva vstupný súbor, očakáva jeho pevnú štruktúru v tvare:

```
\documentclass[voľby]{trieda}
    ... preambula ...
\begin{document}
    ... vlastný text ...
\end{document}
```

`\documentclass[voľby]{trieda}` je príkaz, ktorý LaTeX očakáva na začiatku vstupného súboru. **Trieda** je povinný parameter a môže nadobúdať niektorú z hodnôt:

- `article` – pre články, krátke správy, listy, prezentácie, . . . ,
- `report` – pre dlhšie správy, menšie knihy, seminárne a diplomové práce, . . . ,
- `book` – pre skutočné knihy, veľké dokumenty, . . .

Voľby predstavujú nepovinný parameter, ktorý bližšie špecifikuje vlastnosti triedy dokumentu. Najpoužívanejšie z nich sú (ak ich je viac, sú oddelené čiarkou):

- `letterpaper`, `a4paper`, `a5paper`, `b5paper`, `legalpaper` – nastaví veľkosť stránky, ak nie je parameter spomenutý, t. j. implicitne je nastavené `letterpaper`,
- `10pt`, `11pt`, `12pt` – veľkosť písma dokumentu (implicitne je nastavené `10pt`),
- `oneside`, `twoside` – jednostranný výstup (implicitne pre `article` a `report`), resp. dvojstranný výstup (implicitne pre `book`),
- `twocolumn` – text sa bude sádzať do dvoch stĺpcov na stránku,
- `landscape` – výstup bude formátovaný na šírku, t. j. rozmery šírky a výšky sa vymenia.

Preambula je časť vstupného súboru, v ktorej sa nachádzajú tzv. globálne príkazy vzťahujúce sa k celému dokumentu (napr. šírka textu, hlavička, číslovanie stránok, . . .). Môžu sa tu nachádzať naše vlastné príkazy, tzv. **makrá**. Tu sa k dokumentu pripájajú balíčky príkazov tzv. **packages** (súbory `*.sty`), v ktorých sa nachádzajú mnohé rozšírenia LaTeXu (jazykové doplnky, farby, grafické a postskriptové doplnky, matematické a rôzne symboly, rozhodovacie makrá, indexovanie, obtekanie obrázkov, . . .). Vloženie týchto balíčkov sa vykonáva opakovaným použitím príkazu `\usepackage[voľby]{balíček}`, resp. `\usepackage{balíček1,balíček2,balíček3...}`. Parameter **voľby** je nepovinný a bližšie špecifikuje daný balíček.

Pre ilustráciu uvedieme niektoré z týchto balíčkov:

- `czech`, `slovak` – umožní spracovať český, resp. slovenský text vrátane správneho delenia a názvov textových objektov v danom jazyku (Kapitola, Obrázok, . . .),

- `amsmath`, `amssymb`, `amsfonts`, `amscd` – ďalšie príkazy pre matematickú sadzbu,
- `color` – podpora pre prácu s farbami,
- `ifthen` – podporuje rozhodovacie príkazy,
- `makeidx` – podpora pre tvorbu registra,
- `curves`, `bezier` – podpora pre kreslenie kriviek v prostredí `picture`,
- `array`, `hhline` – podpora pre rozšírenie tabuliek,
- `inputenc` – špecifikácia kódovania pre zdrojové súbory,
- `fontenc` – špecifikácia kódovania fontov pre výstupné cieľové dokumenty.

Časť vstupného dokumentu medzi príkazmi `\begin{document}` a `\end{document}` predstavuje hlavnú časť zdrojového textu. Tu sa píše všetok text, ktorý sa má zobraziť vo výstupnom súbore. Tento text môže byť uložený aj v iných zdrojových súboroch a vtedy sa do vstupného dokumentu vkladá príkazmi `\include{meno}` (vloží text na novú stranu), resp. `\input{meno}` (vloží text na miesto, kde sa nachádza príkaz `input`).

Parameter **meno** môže obsahovať cestu k danému súboru, napr. `\input{makra/subor}`. Ak je prípona daného súboru `.tex`, potom príponu nemusíme písať. LaTeX prehľadáva najprv aktuálny adresár, kde sa nachádza vstupný zdrojový súbor a potom adresáre, ktoré má nastavené sytém TeX v parametri `path`.

2.3 Vlastný dokument

Teraz už môžeme písať vlastný dokument. Ak píšeme po anglicky a nemáme žiadne špeciálne požiadavky, potom nám postačí nasledujúca štruktúra vstupného súboru:

```
\documentclass{article}
\begin{document}
 English text ...
\end{document}
```

Ak sa vstupný súbor nazýva `subor.tex`, potom preklad do súborov `subor.dvi`, resp. `subor.pdf` dosiahneme príkazmi `latex subor.tex`, resp. `pdflatex subor.tex` (koncovka `.tex` je nepovinná, takže ju nemusíme písať).

Pre konverziu do postskriptového súboru `subor.ps` musíme použiť nejaký externý program (napr. `dvips`).

Ak vložíme daný text „English text ...“ do súboru `subor1.tex`, potom identický výsledok dostaneme pre:

```
\documentclass{article}
\begin{document}
 \input{subor1}
\end{document}
```

resp.

```
\documentclass{article}
\begin{document}
 \include{subor1}
\end{document}
```

Súbor `subor1.tex` môžeme do dokumentu vložiť aj niekoľkokrát (nemusíme ho prepisovať). Nasledujúce dva príklady nie sú ekvivalentné:

```
\documentclass{article}
\begin{document}
 \input{subor1}
 \input{subor1}
\end{document}
```

resp.

```
\documentclass{article}
\begin{document}
 \input{subor1}

 \input{subor1}
\end{document}
```

V prvom prípade sa text z druhého vloženia pripojí priamo za prvý vstup, t. j. bude tam dvakrát. V druhom prípade je medzi vstupmi medzera, t. j. druhý vstup začne ako nový odstavec. Zato nasledujúce dva príklady sú ekvivalentné. V oboch prípadoch sa uvedený text zobrazí dvakrát a vždy na novej strane.

```
\documentclass{article}
\begin{document}
 \include{subor1}
 \include{subor1}
\end{document}
```

resp.

```
\documentclass{article}
\begin{document}
 \include{subor1}

 \include{subor1}
\end{document}
```

Ak píšeme po slovensky, resp. po česky musíme v preambule túto skutočnosť LaTeXu oznámiť (v opačnom prípade by preklad ignoroval znaky s diakritikou). Máme dve možnosti.

```
\documentclass[a4paper,12pt]{article}
 \usepackage{slovak}
 \usepackage[latin2]{inputenc}
 \usepackage{amsmath,amssymb,amsfonts,amscd}
\begin{document}
 Slovenský text v kódovaní ISO Latin-2 ...
\end{document}
```

V tomto prípade sú zdrojové súbory napísané v kódovaní ISO Latin-2 (kódová stránka ISO-8859-2). Ak vynecháme príkaz `\usepackage[latin2]{inputenc}`, potom kódovanie zdrojových súborov musí byť rovnaké ako implicitné kódovanie v systéme TeX.

Preklad do výstupných súborov `subor.dvi`, resp. `subor.pdf` dosiahneme príkazmi `cslatex subor.tex`, resp. `pdfcslatex subor.tex`.

V súčasnosti sa do popredia dostáva tzv. „babelizovaný LaTeX“, ktorý umožňuje v jednom zdrojovom súbore používať viacero jazykov.

```
\documentclass[a4paper,12pt]{article}
 \usepackage[slovak]{babel}
 \usepackage[IL2]{fontenc}
 \usepackage[latin2]{inputenc}
 \usepackage{amsmath,amssymb,amsfonts,amscd}
\begin{document}
 Slovenský text v kódovaní ISO Latin-2 ...
\end{document}
```

Pre kódovania PC-latin2 (cp852), resp. win1250 (cp1250) má 4. riadok tvar:

```
\usepackage[cp852]{inputenc}   resp.   \usepackage[cp1250]{inputenc}
```

Pre kódovanie unicode UTF8 majú 3. a 4. riadok tvar:

```
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}   resp.   \usepackage[utf8x]{inputenc}
```

Pre preklad súborov v tomto prípade nemusíme použiť CSLaTeX, postačí LaTeX. To znamená, že môžeme použiť príkazy `latex subor.tex`, resp. `pdflatex subor.tex`.

Na záver tejto časti uvedieme príklady súčasného použitia viacerých jazykov. Na prvý pohľad prepínanie medzi angličtinou a slovenčinou nemá význam, však pri zapnutej slovenčine dokážeme písať aj anglický text. Problém je v rozdeľovaní slov na konci riadku (každý jazyk má svoje špecifiká v rozdeľovaní slov) a v slovách, ktoré dopĺňajú text (napr. „Kapitola“, „Chapter“ alebo „Obrázok“, „Obrázek“, „Figure“, . . .). Medzi jednotlivými jazykmi sa prepína príkazom `\selectlanguage{jazyk}`.

```
\documentclass[11pt]{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[russian,slovak,english]{babel}

\begin{document}
\selectlanguage{english}
  English text ...

\selectlanguage{russian}
  Ruskij tekst ...

\selectlanguage{slovak}
  Slovenský text ...

\selectlanguage{english}
  English text ...

\selectlanguage{russian}
  Ruskij tekst ...

  ...

\end{document}
```

Príkaz `\selectlanguage{english}` za `\begin{document}` nemusíme písať, pretože angličtina sa nastaví ako implicitný jazyk (do TeXu sa načíta `\language=0`).

Trochu iný je nasledujúci príklad. Ak by sme ponechali štruktúru dokumentu v nasledujúcom tvare (bez načítania ruštiny), TeX by vyhlásil chybu, pretože ruštinu zatiaľ nepozná

(opäť sa načíta `\language=0`). Musíme mu to explicitne oznámiť príkazom `\selectlanguage{russian}`.

```
\documentclass[11pt]{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[russian,slovak]{babel}

\begin{document}
  Ruskij tekst ...

\selectlanguage{slovak}
  Slovenský text ...

\end{document}
```

2.4 Vzhľad stránky

LaTeX má veľkú výhodu, že o vzhľad stránky a dokumentu sa užívateľ nemusí starať. Štýl stránkovania definuje vzhľad hlavičky a päty stránky a zadáva sa väčšinou v preambule príkazom `\pagestyle{štýl}` (ovplyvní celý dokument od miesta výskytu po koniec dokumentu, prípadne po jeho nový výskyt). Jeho analógia, ktorá ovplyvní iba aktuálnu stránku má tvar `\thispagestyle{štýl}`. Povinný parameter **štýl** môže mať tvar:

- `plain` – hlavička je prázdna, v päte je vycentrované číslo strany (implicitné nastavenie pre triedy `article` a `plain`),
- `empty` – hlavička a päta sú úplne prázdne, nie sú ani čísla strán,
- `headings` – hlavička obsahuje čísla strán a titulkové informácie (nadpisy kapitol a sekcií), päta je prázdna (implicitné nastavenie pre triedu `book`),
- `myheadings` – podobný tvar ako `headings`, ale titulkové informácie musíme nastaviť pomocou príkazov `\markright{pravé_záhlavie}` pre jednostrannú tlač a `\markboth{ľavé_záhlavie}{pravé_záhlavie}` pre obojstrannú tlač `twoside`.

Viac možností ponúka balíček `fancyhdr`.

Príkaz `\pagenumbering{štýl_číslovania}` špecifikuje číslovanie strán, povinný parameter **štýl_číslovania** môže mať tvar:

- `arabic` – čísluje sa arabskými číslami (implicitné nastavenie),
- `roman`, `Roman` – čísluje sa malými, resp. veľkými rímskymi číslami,
- `alph`, `Alph` – čísluje sa malými, resp. veľkými latinskými písmenami.

Štandartne sa čísluje od prvej hodnoty, pre zmenu číslovania môžeme použiť príkaz `\setcounter{page}{nové_číslo_strany}`, resp. `\addtocounter{page}{zväčšenie_čísla_strany_o_túto_hodnotu}`.

2.5 Vzhľad odstavca

Odstavec je ohraničený na začiatku a na konci prázdnym riadkom a o jeho tvar sa stará LaTeX. To znamená, že nezáleží na formáte ako napíšeme zdrojový kód odstavca[1] (nezáleží na počte medzislovných medzier, nezáleží na koľkých riadkoch je text napísaný, . . .).

Na začiatku je odstavec odsadený o hodnotu `\parindent`, ktorú môžeme zmeniť príkazmi: `\setlength{\parindent}{nová_veľkosť}`, resp. `\addtolength{\parindent}{zväčšenie_veľkosti_o_túto_hodnotu}`.

Príkaz `\setlength{\parindent}{10cm}` zmení hodnotu `\parindent` na 10 cm. Toto je nevhodné použitie, pretože keď zmeníme veľkosť písma dokumentu, odsadenie ostane 10 cm. Výhodnejšie je použiť relatívnu veľkosť vzťahujúcu sa k veľkosti písma dokumentu definovaného v príkaze `\documentclass`. Napr. `\setlength{\parindent}{5em}` zmení `\parindent` na 5-násobok šírky písmena „M“ a `\setlength{\parindent}{5.5ex}` zmení `\parindent` na 5,5-násobok výšky písmena „x“. V parametroch príkazov sa desatinné čísla používajú v anglickom tvare s desatinnou bodkou.

Riadkovanie je určené parametrom `\baselinestretch`, ktorý môžeme zmeniť príkazom `\renewcommand{\baselinestretch}{faktor}`, pričom **faktor** je ľubovoľné desatinné číslo. Napríklad hodnota 1.5 zväčší medziriadkové medzery o 50 %.

Pomocou príkazov `\hangafter=číslo`, `\hangindent=veľkosť_zmeny` môžeme meniť tvar odstavca. Tieto príkazy patria medzi základné príkazy systému TeX, z ktorých sú zložené všetky ostatné a nazývajú sa **primitívy**. Vo formáte LaTeX (ako rozšírení TeXu) tieto primitívy bez problémov fungujú a ich zoznam nájdeme napríklad na adrese `http://petr.olsak.net/ftp/olsak/bulletin/primitiv.ps`.

Ak je parameter **číslo** kladný (celé číslo), potom určuje počet riadkov plnej šírky, t. j. takýto počet prvých riadkov bude mať nezmenenú šírku. Ak je tento parameter záporný, potom určuje počet riadkov zmenenej šírky.

Ak je parameter **veľkosť_zmeny** kladný (napr. 1.5cm, 2em), potom budú modifikované riadky krátené zľava o túto hodnotu. Ak je tento parameter záporný, potom krátenie nastane sprava. Ak je nulový, potom krátenie nenastane. Implicitné nastavenie je[2] `\hangafter=1 \hangindent=0pt`.

Po použití príkazov `\hangafter=5 \hangindent=5em` sa v odstavci zobrazí prvých 5 riadkov v normálnej šírke a ostatné sa skrátia zľava o hodnotu 5 em.

Po použití `\hangafter=-3 \hangindent=\parindent` sa v odstavci zobrazia prvé 3 riadky skrátené o hodnotu `\parindent` a ostatné sa zobrazia v normálnej šírke.

Ak použijeme na začiatku odstavca (musí byť prvým príkazom v odstavci) príkaz `\noindent`, potom sa odstavec neodsadí a začne sa sádzať od začiatku riadku. Naopak po použití príkazu `\indent` sa odstavec odsadí o hodnotu `\parindent`. Tento príkaz môžeme použiť v odstavcoch, ktoré sa automaticky neodsadia.

LaTeX implicitne neodsadí prvý odstavec na začiatku sekcie (nepomôže ani príkaz

[1] Pokiaľ ho neprerušíme prázdnym riadkom alebo nepoužijeme formátovacie príkazy na jeho zmenu.

[2] 1 in [palec] = 2,54 cm = 25,4 mm = 72,27 pt [tlačiarenský bod]. Všetky tieto jednotky môžeme použiť.

`\indent`). Na odstránenie tohto nedostatku môžeme definovať vlastný príkaz, napríklad `\def\Prvyindent{\hangafter=-1 \hangindent=\parindent}`.

Ak použijeme predchádzajúce príkazy v preambule dokumentu, majú globálny význam.

2.6 Členenie textu

Pre lepšiu orientáciu v texte je vhodné ho rozčleniť na jednotlivé časti – na kapitoly, sekcie, paragrafy, ... LaTeX obsahuje pre tento účel špeciálne príkazy, ktoré automaticky určia veľkosť a tvar fontu pre daný nadpis, automaticky ho očíslujú, zistia jeho polohu a následne ho zaradia do obsahu aj s jeho číslom. To textu sa zadávajú podľa nasledujúceho vzoru `\chapter[názov_do_obsahu]{názov}`. Ak chýba nepovinný parameter **názov_do_obsahu**, potom sa do obsahu dosadí celý **názov**.

Výsledkom použitia príkazu `\chapter` je na novej strane nadpis[3] „Kapitola 1“ a pod tým **názov**. Jediný problém tohto príkazu je, že sa nedá použiť v triede `article`.

Príkaz `\chapter*{názov}` vypíše iba **názov** a to bez čísla a bez slova „Kapitola“.

Kapitoly môžeme potom deliť na ďalšie podčasti a to v nasledujúcom poradí:

```
\section[názov_do_obsahu]{názov}          resp.  \section*{názov}
\subsection[názov_do_obsahu]{názov}              \subsection*{názov}
\subsubsection[názov_do_obsahu]{názov}           \subsubsection*{názov}
\paragraph[názov_do_obsahu]{názov}
\subparagraph[názov_do_obsahu]{názov}
```

Príkazy `\section`, `\subsection`, `\subsubsection` sú číslované, kým `\paragraph` a `\subparagraph` nie sú. Po ich zadaní sa vypíše iba zodpovedajúce číslo časti a **názov**.

V texte si môžeme nejaké miesto označiť neviditeľnou značkou `\label{návestie}` a potom sa naň odvolávať. Príkazy `\ref{návestie}`, resp. `\pageref{návestie}` vypíšu číslo časti, resp. číslo stránky, kde sa `\label{návestie}` nachádza.

2.7 Vlastný text

V LaTeXe treba rozlišovať medzi textom v obyčajnom a tzv. matematickom móde. Matematické časti textu sa vo vnútri odstavca uzatvárajú medzi znaky `$` a `$`, resp. `\(` a `\)`.

Rozsiahlejšie matematické vzorce je výhodnejšie sádzať na špeciálne riadky. Pre tento prípad existujú matematické prostredia uzavreté medzi znaky `$$` a `$$`, resp. `\[` a `\]`, resp. `\begin{displaymath}` a `\end{displaymath}`, resp. `\begin{equation}` a `\end{equation}`, resp. `\begin{eqnarray}` a `\end{eqnarray}`.

Teraz sa budeme zaoberať normálnym textovým módom. Aj keď mnohé poznatky, ktoré spomenieme sú aplikovateľné aj pre matematický mód.

Prácu nám môžu uľahčiť vlastné makrá, ktoré môžeme na ľubovoľnom mieste dokumentu definovať príkazmi `\newcommand{\meno}[číslo]{definícia_makra}`, resp. `\def{\meno}parametre{definícia_makra}`.

[3] V anglickej mutácii sa napíše „Chapter 1“, t. j. číslo kapitoly.

Takto definované makrá môžu obsahovať niekoľko znakov, ale aj celé strany vrátane skupín a prostredí. Môžu obsahovať až 9 parametrov, ktoré do nich vstupujú.

Príkazom `\newcommand` musíme definovať makro, ktorého **meno** je nové[4] (doteraz v dokumente alebo v systéme TeX nepoužívané). Na premenovanie[5] existujúceho makra použijeme `\renewcommand{\meno}[číslo]{definícia_makra}`. **Číslo** vyjadruje počet parametrov, ktoré môžu vstupovať do makra (maximálne 9). V časti **definícia_makra** sa tieto parametre vyjadrujú ako `#1` (parameter1), ... `#N` (parameterN). Názov **meno** môže obsahovať iba písmena, pričom sa rozlišuje ich veľkosť. Takto vytvorené makro použijeme príkazom `\meno`, resp. `\meno{parameter1}...{parameterN}`.

Uvedieme niekoľko príkladov na použitie príkazu `\newcommand`:

```
\newcommand\MPMakro{Vidím ťa.}          \MPMakro
\newcommand\List[2]{#1 píše list #2.}   \List{Jano}{Ferovi}
\renewcommand\MPMakro[1]{Vidím #1.}     \MPMakro{Fera}
```

Príkaz `\def` je TeXový primitív a funguje podobne ako `\newcommand`, ale ak makro s názvom **meno** existuje, potom ho bez opýtania prepíše. Navyše musíme všetky **parametre** vypísať aj s ohraničením do definície, okrem štandartného ohraničenia zátvorkami {}. Predchádzajúce príklady potom budú vyzerať nasledovne:

```
\def\MPMakro{Vidím ťa.}             \MPMakro
\def\List[#1]#2{#1 píše list #2.}   \List[Jano]{Ferovi}
\def\MPMakro#1{Vidím #1.}           \MPMakro{Fera}
```

Na poznámky písané pod čiarou slúži príkaz `\footnote{text_pod_čiarou}`.

Text zvýrazníme pomocou príkazu `\emph{zvýraznený_text}`. Ak je pôvodný text písaný kurzívou, potom sa **zvýraznený_text** prepne do kolmého módu a naopak z kolmého módu sa **zvýraznený_text** prepne do kurzívy.

Ak použijeme prepínač `\it`, *potom do konca dokumentu, resp. do ďalšieho výskytu prepínača písma bude text vysádzaný kurzívou.* Ak príkaz použijeme v tvare `{\it náš_text}`, resp. `\textit{náš_text}`, potom kurzívou bude vypísaný iba **náš_text**.

Analogicky fungujú príkazy `\bf`, `\textbf` (**polotučné písmo**), `\sl`, `\textsl` (*sklonené písmo*), `\sc`, `\textsc` (KAPITÁLKY), `\tt`, `\texttt` (strojopisné písmo), `\em`, `\textem` (*vyznačovacie písmo* – analogické s `\emph`), `\rm`, `\textrm` (antikva – kolmé písmo) a `\sf`, `\textsf` (bezpätkové písmo).

Na zmenšenie alebo zväčšenie písma môžeme použiť nasledujúce príkazy. Pomer zmeny sa nemení a veľkosť všetkých písiem je odvodená od implicitného nastavenia veľkosti normálneho písma. Tiež sú prepínače (medzi rôznymi veľkosťami písma) a ich použitie je úplne rovnaké ako v predchádzajúcom odstavci a môžeme ich navzájom medzi sebou kombinovať: `\tiny` (najmenšie písmo), `\tiny\bf` alebo `\bf\tiny` **(najmenšie polotučné písmo)**, `\scriptsize` (veľmi malé), `\footnotesize` (malé), `\small` (menšie), `\normalsize`

[4] Ak makro s týmto názvom existuje, systém TeX nám vynadá.

[5] Ak makro s týmto názvom neexistuje, systém TeX nám vynadá.

(normálne písmo), `\large` (väčšie), `\Large` (veľké), `\LARGE` (veľké), `\huge` (veľké) a `\Huge` (najväčšie písmo).

TEX a LATEX majú veľmi veľký potenciál a dokážu toho oveľa oveľa viac. Ich možnosti ďaleko prekračujú kapacitu tohto článku. Sú to napríklad farebné možnosti (balíček `color`), obrázky (prostredie `picture`, balíček `graphicx`), tabuľky (prostredie `tabular`, balíček `longtable`), veľké množstvo rôznych matematických a aj nematematických symbolov, register pojmov, rôzne fonty písiem, ľubovoľné zmenšenie či zväčšenie písma, prostredia na tvorenie zoznamov, možnosť vkladania rôznych externých súborov, atď. Na záver spomeniem ešte jeden odkaz `http://www.cheat-sheets.org/saved-copy/latexsheet.pdf`, kde sa nachádza na dvoch stranách stručný prehľad základných príkazov LATEXu.

Ďakujem všetkým nadšencom, ktorí sa venujú rozvíjaniu TEXu. Táto práca vznikla na propagáciu LATEXu.

Literatúra

[1] BALDA, M.: *Výpočty a diagramy v LATEXu*. Praha, Zpravodaj Československého sdružení uživatelů TeXu, č. 2, ročník 14, CSTUG, 2004, str. 54–110, ISSN 1211-6661, `http://bulletin.cstug.cz/pdf/bul_042.pdf`.

[2] KNUTH, D. E.: *The TEXbook*, Volume A of *Computers and Typesetting*, Addison-Wesley Publishing Company (1984), ISBN 0-201-13448-9.

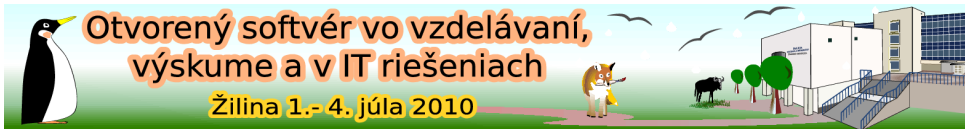
[3] KOPKA, H. – DALY, P. W.: *LATEX – Podrobný průvodce*, Brno, Computer Press, 2004, ISBN 80-722-6973-9.

[4] OETIKER, T. – PARTL, H. – SCHLEGL, E. – HYNA,I. – KOČER, M. – SÝKORA, P.: *Ne příliš stručný úvod do systému LATEX 2ε*, „public domain“ dokument, 1998, `http://www.penguin.cz/~kocer/texty/lshort2e/lshort2e-cz.pdf`, resp. `http://ftp.cstug.cz/pub/tex/CTAN/info/lshort/slovak/Slshorte.pdf` (Buša J. ml. a st., slovenský preklad).

[5] RYBIČKA, J.: *LATEX pro začátečníky*, Brno, KONVOJ 2003, ISBN 80-7302-049-1.

Kontaktná adresa

RNDr. Rudolf BLAŠKO, PhD., Katedra matematických metód, Fakulta riadenia a informatiky, Žilinská univerzita, Univerzitná 8215/1, 010 26 Žilina, Slovenská Republika, `beerb@frcatel.fri.uniza.sk`, `http://frcatel.fri.uniza.sk/~beerb/`.



WEKA AS A TOOL FOR CLASSIFICATION TASKS

BOHÁČIK, Ján, (SK)

Abstract. Classification belongs to the main tasks in the Data Mining stage of the Knowledge Discovery in Databases. The aim of this paper is to compare several classification data mining algorithms using the Weka software tool implemented in Java and issued under the GNU General Public License. Some of results of the author are also described.

1 Introduction

Data repositories are expanding very quickly and contain immense amounts of various data. According to some estimates these amounts doubling every twenty months [4]. Due to it, information and database systems are widely used. The current generation of database systems is based mainly on a small number of primitives of Structured Query Language (SQL) used for manipulating with relational databases which consist of relations (tables) [7]. The tables have attributes in columns and instances in rows. Often, these tables contain more and more covert information that cannot be found out and transformed into knowledge by classical SQL queries. Dependencies are searched so that knowledge can be extracted. There are several ways how to use found dependencies, which classification is one of the most used ones from.

Classification is often solved in the Data Mining stage of the Knowledge Discovery in Databases. Given a set of training instances (known instances) $\boldsymbol{V}$ where each instance $\boldsymbol{e} \in \boldsymbol{V}$ is described by attributes $\boldsymbol{A} = \{A_1, \ldots, A_k, \ldots, A_M\}$, $A_k = \{a_{k,1}, \ldots, a_{k,l}, \ldots, a_{k,N_k}\}$, $a_{k,l}$ is a possible value of the attribute, and classified into a class $c_j \in C$ where $C = \{c_1, \ldots, c_j, \ldots, c_O\}$ is a set of possible classes, the task is to build a model that predicts the class of an unseen instance. This model can be used

in systems where a decision is required on the basis of known data. Systems of this kind are described more in [11, 12]. An example of using in transportation is a prediction of accidents on the basis of risk factors [3]. Among most popular classification models are the Nearest Neighbor Classifier model, the Naive Bayes Classifier model and the Fuzzy Decision Tree model. These models are compared in this paper on the basis of their error rates in classification. A modification of using the Fuzzy Decision Tree model is described. The algorithms for making the first two models are implemented in Weka [13] – an object-oriented machine-learning Java software tool issued under the GNU General Public License. The algorithm for making the Fuzzy Decision Tree model is implemented in my object-oriented Java software tool Fuzzy Rule Miner [2].

The paper is organized as follows. Section 2 explains object-oriented Java tool Weka. In Section 3, principles of building the Nearest Neighbor Classifier model and the Naive Bayes Classifier model are stated. The principle of building the Fuzzy Decision Tree model and two ways how to use it are described in Section 4. The results of experiments are in Section 5. Section 6 concludes this paper.

2 Weka

Weka (http://www.cs.waikato.ac.nz/ml/weka/, Waikato Environment for Knowledge Learning) is a machine-learning tool developed at the Waikato University in the New Zealand [13]. Originally written in C, Weka has been completely rewritten in Java and is compatible with almost every computing platform. It can cope with pre-processing and data analysis, classification models, association models, and evaluation metrics. There are three modes of Weka operation: a) GUI, b) command-line and c) Java API. Java API allows to make computer programs for solving classification tasks. In the following lines, there is a code showing how to make attributes and instances:

```
// Declare an attribute with numerical values:
  Attribute A1 = new Attribute("A1");
// Declare an attribute with nominal values:
  FastVector A2NomVal = new FastVector(2);
  A2NomVal.addElement("a2,1");
  A2NomVal.addElement("a2,2");
  Attribute A2 = new Attribute("A2", A2NomVal);
// Declare the class attribute:
  FastVector CVal = new FastVector(2);
```

```
  CVal.addElement("c1");
  CVal.addElement("c2");
  Attribute C = new Attribute("C", Cval);
// Declare the vector of attributes A and class C:
  FastVector AandC = new FastVector(3);
  AandC.addElement(A1);
  AandC.addElement(A2);
  AandC.addElement(C);
// Create training instances V:
  Instances V = new Instances("V", AandC, 10); // 10 is the initial capacity
  V.setClassIndex(2);
// Create an instance and add it:
  Instance e1 = new Instance(3);
  e1.setValue((Attribute)AandC.elementAt(0), 1.0);
  e1.setValue((Attribute)AandC.elementAt(1), "a2,2");
  e1.setValue((Attribute)AandC.elementAt(3), "c1");
  V.add(e1);
```

In the *weka.classifiers* package, the most important class is *Classifier*. It is a general scheme for any classification model in Weka. *Classifier* contains two significant methods, *buildClassifier()* and *classifyInstance()*. The former is for building the classification model, the latter is for determining the value of class attribute C when the values of all attributes in $\boldsymbol{A}$ are known (i.e. classification).

3 Nearest Neighbor Classifier and Naive Bayes Classifier

Nearest Neighbor Classifier (NNC) assumes that all instances correspond to points in the n-dimensional space $\boldsymbol{R}^n$ [8]. NNC is a type of lazy learning where the function is only approximated locally and all computation is deferred until classification. During learning, all points with known classes are remembered. When a new point is classified, the k-nearest points to the new point are found (k is a positive integer, usually small) and are used with a weight for determining the class of the new point (the instance with unknown class). For the sake of increasing classification accuracy, greater weights are given to closer points during classification. The simple version of the algorithm is easy to implement by computing the distances from the instance with unknown class to all stored known instances $\boldsymbol{V}$, but it is computationally intensive, especially when the cardinality of $\boldsymbol{V}$ grows. Many nearest neighbor classifier models have been proposed; these

generally seek to reduce the number of distance evaluations. Its implementation in Weka is in the *weka.classifiers.lazy* package and it can be used as follows:

```
// Create a new Nearest Neighbor Classifier:
Classifier NNC = (Classifier)new IBk();
NNC.buildClassifier(V);
```

Naive Bayes Classifier (NBC) is a probabilistic classification model based on Bayes' theorem. It represents each instance in $\boldsymbol{V}$ as a #($\boldsymbol{A}$)-dimensional vector of attribute values $[a_{1,l_1}, a_{2,l_2}, \ldots, a_{M,l_M}]$, #($\boldsymbol{A}$) is the cardinality of $\boldsymbol{A}$. Given that there are O classes $c_1, \ldots, c_j, \ldots, c_O$, the classifier predicts that an unknown instance with known values of vector $X = [x_1, \ldots, x_k, \ldots, x_M]$) belongs to the class c_j having the highest posterior probability conditioned on X. In other words, X is assigned to class c_j if and only if $P(c_j/X) > P(c_k/X)$ for $1 \leq k \leq O$ and $j \neq k$. NBC is very simple, it requires only a single scan of the data, thereby providing high accuracy and speed for large databases. However, inaccuracies arise due to a) the simplified assumptions involved and b) a lack of available probability data or knowledge about the underlying probability distribution [9, 13]. Its implementation in Weka is in the *weka.classifiers* package and it can be used as:

```
// Create a new Naive Bayes Classifier:
Classifier NBC = (Classifier)new NaiveBayes();
NBC.buildClassifier(V);
```

4 Fuzzy decision tree classification with more leaf nodes

In this section, two ways how to use a fuzzy decision tree (FDT) for determining the class of an instance $\boldsymbol{e}$ are described. The FDT is made with a top-down greedy ID3-like heuristic based on [6]. Before it is built, attributes are transformed into linguistic terms $\boldsymbol{A} = \{A_1, \ldots, A_k, \ldots, A_M\}$, $A_k = \{a_{k,1}, \ldots, a_{k,l}, \ldots, a_{k,N_k}\}$, $a_{k,l}$ is a linguistic term of the linguistic variable A_k, and classes are transformed into class linguistic terms $c_j \in C$, $C = \{c_1, \ldots, c_j, \ldots, c_O\}$ is the class linguistic variable. For each $a_{k,l} \in A_k \in \boldsymbol{A}$, $c_j \in C$ and $\boldsymbol{e} \in V$, membership degree $a_{k,l}(\boldsymbol{e})$ ($c_j(\boldsymbol{e})$) to which $a_{k,l}$ (c_j) is the value of A_k (C) for instance $\boldsymbol{e}$ is determined. The value of a membership degree is in the continuous interval 0 to 1. Number 1 means the highest possibility that $a_{k,l}$ (c_j) is the value of A_k (C), number 0 means the lowest possibility that $a_{k,l}$ (c_j) is the value of A_k (C). The transformations and computing membership degrees can be done for example with algorithm [5]. During building, main activities are the association of a linguistic variable with a node and the decision if a node or a leaf should be associated with a branch coming from a

node. The former is done with computing cumulative information as a criterion. The linguistic variable with the maximal cumulative information is chosen. The latter uses a stopping criterion based on the frequency of the branch. The FDT is made as follows. The root node is made. Linguistic variable $maxA_k$ is associated with it after cumulative information is computed for all linguistic variables in $\boldsymbol{A}$. A branch coming from the root node is made for each $a_{k,l} \in maxA_k$ and it is associated with the $a_{k,l}$. These branches are considered unprocessed. For each unprocessed branch, the decision if an internal node or a leaf node is connected with it is made. Then the branch is considered processed. If an internal node was connected with it, a linguistic variable is associated with it. The process of building finishes when all branches are considered processed. Let the FDT have R leaf nodes $\boldsymbol{L} = \{l_1, \ldots, l_r, \ldots, l_R\}$. During building the FDT, also value F_j^r for each leaf node l_r and each linguistic term c_j is made. This value F_j^r means the certainty degree of the linguistic term c_j attached to the leaf node l_r.

Classification of an instance $\boldsymbol{e}$ using the FDT means determining the values of membership degrees $c_j(\boldsymbol{e})$ for all $c_j \in C$. It is also supposed that membership degrees $a_{k,l}(\boldsymbol{e})$ for all $a_{k,l} \in A_k \in \boldsymbol{A}$ are known. The first way how to conduct classification is based on [10]. It was initially meant for classification in the crisp case. In the crisp space, a linguistic term either is the value of a linguistic variable or it is not. In other words, either $a_{k,l}(\boldsymbol{e})=1$ or $a_{k,l}(\boldsymbol{e})=0$ for all $a_{k,l} \in A_k \in \boldsymbol{A}$. Also, just one $a_{k,l}(\boldsymbol{e})=0$ for all $a_{k,l} \in A_k$, all the others equal 0. In the fuzzy case, there can be more than one $a_{k,l}(\boldsymbol{e})>0$ for all $a_{k,l} \in A_k$. For the purpose of using this first way, $a_{k,l}(\boldsymbol{e})$ with the maximal value among all $a_{k,l} \in A_k$ is set to 1 and all the others are set to 0. The instance $\boldsymbol{e}$ with these membership degrees is called rounded instance $\boldsymbol{e}$. In the FDT, a path from the root node to the leaf node l_r where terms a_{k_1,l_1}, a_{k_2,l_2}, …, a_{k_q,l_q} associated with the branches in the path respectively match with $a_{k,l}$ with $a_{k,l}(\boldsymbol{e})=1$ of the rounded instance $\boldsymbol{e}$. After that, $c_j(\boldsymbol{e})=F_{j_q}^r$.

The other way for classification of an instance $\boldsymbol{e}$ uses one or more paths from the root node to the leaf nodes. The reason why it is like this is because there may be several $a_{k,l}(\boldsymbol{e})>0$ for a node associated with A_k in the FDT. That is why there may be several paths, from the root node to the leaf nodes, whose all branches coming from the nodes are associated with $a_{k,l}$ with $a_{k,l}(\boldsymbol{e})>0$. Because all $a_{k,l}(\boldsymbol{e})$ do not equal 1 in general in the path, it is clear that each path of this kind should be included in the final value of $c_j(\boldsymbol{e})$ with a certain weight. It is defined for instance $\boldsymbol{e}$ and the path from the root node to the leaf node l_r as follows:

$$W_r(\boldsymbol{e}) = \prod_{a_{k,l} \in \boldsymbol{PATH}_r} a_{k,l}(\boldsymbol{e}), \tag{1}$$

where $\boldsymbol{PATH}_r$ is a set of all linguistic terms $a_{k,l}$ associated with the branches in the path from the root node to the leaf node l_r. Membership degrees $c_j(\boldsymbol{e})$ for all $c_j \in C$ are computed as follows:

$$c_j(\boldsymbol{e}) = \sum_{r=1}^{R} F_j^r \cdot W_r(\boldsymbol{e}). \tag{2}$$

If, in the first or the second way, classification only into one class linguistic term $c_j \in C$ is required, instance $\boldsymbol{e}$ is classified into $c_j \in C$ whose $c_j(\boldsymbol{e})$ is maximal. Software tool Fuzzy Rule Miner of the author contains similar classes to Weka.

5 Experiments

The main purpose of the experiments is to compare the two ways how to use a fuzzy decision tree based on [6] (FDT) for determining the class of an instance $\boldsymbol{e}$ with each other and with Nearest Neighbor Classifier (NNC) and Naive Bayes Classifier (NBC). The algorithm for building the FDT and the two ways of determining the class of an instance $\boldsymbol{e}$ are implemented in object-oriented Java tool Fuzzy Rule Miner of the author [2]. The NNC and NBC are implemented in Weka [13] – an object-oriented machine-learning Java software tool issued under the GNU General Public License.

The experiments are carried out on selected machine-learning databases [1]. First, if it was required the databases were fuzzified according to [5]. Then each database was randomly separated into two parts. One part contained 70% of the database and it was used for building classification models. The other part contained 30% of the database and it was used for verification of the classification models. This process of separation and verification was repeated 100 times. Partial error rates for respective databases were obtained. The error rate was finally calculated as the ratio of the number of mis-classification combinations to the total number of combinations.

The results of the experiments are in Table 1 where FDT-M denotes fuzzy decision tree based on [6] and the second way of determining the class of an instance $\boldsymbol{e}$ stated in Section 4, FDT-O denotes fuzzy decision tree based on [6] and the first way of determining the class of an instance $\boldsymbol{e}$ stated in Section 4, NNC denotes Nearest Neighbor Classifier and NBC denotes Naive Bayes Classifier. The last row contains average error rates for all databases and respective classification models. It can be seen that FDT-M which uses several paths from the root node to

leaf nodes for classification is almost 10% better that FDT-O and it is the best of all classification models compared here.

Table 1: Error rates for respective classification models

Database	**Error rate for a database and a classification model**			
	FDT-M	**FDT-O**	**NNC**	**NBC**
BUPA	0.4174	0.4205	0.3910	0.4416
Ecoli	0.2022	0.2654	0.2046	0.1547
Glass	0.3988	0.4647	0.3335	0.5394
Haberman	0.2624	0.2609	0.3471	0.2453
Iris	0.04067	0.02956	0.05044	0.04556
Pima	0.2436	0.2563	0.3091	0.2483
Wine	0.04566	0.06509	0.05094	0.02697
Average	**0.2301**	**0.2518**	**0.2409**	**0.2431**

6 Conclusions

In this paper, object-oriented machine-learning Java software tool Weka was described with focus on classification models and classification tasks. Mechanisms of using a fuzzy decision tree for determining class linguistic terms of instances were investigated. Two ways were described. One is an analogy of the crisp case when only one path from the root node to a leaf node is used. The other is a generalization of the former where several paths are taken into consideration with certain weights. The generalized way has almost 10% better results on selected databases and also it is the best of all compared classification models.

7 References

[1] ASUNCION, A., NEWMAN, D., J.: UCI Machine Learning Repository [`http://www.ics.uci.edu/mlearn/MLRepository.html`], Irvine, University of California, Department of Information and Computer Science, CA, 2007.

[2] BOHACIK, J., MATIASKO, K., LEVASHENKO, V.: Software for making and using fuzzy rules, Journal of ELECTRICAL ENGINEERING, Vol. 57, pp. 85-88, 2006, ISSN 1335-3632
[3] CHANG, L.-Y., CHEN W.-C.: Data mining of tree-based models to analyze freewayaccident frequency, Journal of Safety Research, Vol. 36, No. 4, pp. 365-375, 2005, ISSN 0022-4375.
[4] JOHN, G. H.: Enhancements to the Data Mining Process, Standford: PhD Thesis, 1997, Standford University
[5] LEE, H.-M., CHEN, C.-M., CHEN, J.-M., JOU, Y.-L.: An efficient fuzzy classifier with feature selection based on fuzzy entropy. IEEE Transaction on Systems, Man, and Cybernetics – Part B: Cybernetics 3, Vol. 31, pp. 426-432, 2001
[6] LEVASHENKO, V., ZAITSEVA, E.: Usage of new information estimations for induction of fuzzy decision trees. Proc of the 3rd IEEE International Conference on Intelligent Data Engineering and Automated Learning, Kluwer Publisher, UK, pp. 493-499, 2002
[7] MATIASKO, K.: Databazove systemy (Database systems), Zilina: University of Zilina, 2009, ISBN 80-7100-968-7
[8] MITCHELL, T. M.: Machine Learning. USA: McGraw-Hill Companies, 1997, ISBN 0-07-042807-7
[9] MITRA, S., ACHARYA, T.: Data mining – multimedia, soft computing, and bioinformatics. NJ: Wiley, 2003, ISBN 0-471-46054-0
[10] QUINLAN, J. R.: Induction of decision trees. Machine Learning, No. 1, pp. 81-106, 1986
[11] TRNKA, A.: Proposal of application datawarehouses into control process. In: Proc. of the International Doctoral Seminar, AlumniPress, Smolenice, pp. 343-348, 2009, ISBN 978-80-8096-088-9
[12] TRNKA, A., TANUSKA, P.: Datawarehousing and data mining methodologies as a one of the possible benefits in control process. In: Aktualnye problemy i innovacii v ekonomike, upravlenii, obrazovanii, informacionnych technologijach : materialy mezdunarodnoj naucnoj konferencii (12-15 maja 2009 goda, Stavropol-Kislovodsk) - Vol. 5, No. 3, pp. 84-87., 2009, ISSN 2074-1685.
[13] WITTEN, I. H., FRANK, E.: Data mining: Practical machine learning tools and techniques (2nd edition). San Francisco: Morgan Kaufman, 2005, ISBN 0-12-088407-0

Contact address

Ján BOHÁČIK (Ing., PhD.), Department of Informatics, FRI ZU in Žilina, Univerzitná 8215/1, 010 26 Žilina, Jan.Bohacik@fri.uniza.sk

This work is made with the support of

Center of excellence for systems and services of intelligent transport

ITMS code of the project 26220120028

University of Zilina in Zilina

ERDF - European Regional Development Fund

Project is financially supported by the sources of EC

TERMINÁLOVÁ SIEŤ PO ROKU

FEDORIK, Slavko, (SK)

Abstrakt. V príspevku v krátkosti zhrniem ročné skúsenosti s používania tejto technológie, jej klady i zápory, i to čo sa podarilo a čo nie. Hlavné ťažisko prezentácie je však zverejnenie výsledku prieskumu, ktorý som urobil medzi svojimi študentmi. Ťažiskom prieskumu boli ich postoje a názory na používanie operačného systému Linux a FOSS pri vyučovaní. Ich pohľad na používanie tenkých klientov ako takých, ako aj ich názory a skúsenosti s používaním slobodného a otvoreného softvéru mimo vyučovacieho procesu.

1 Úvod

Je to rok, čo som v našej škole vytvoril laboratórium výpočtovej techniky, založené na technológii tenkých klientov, prihlasujúcich sa k terminálovému serveru. Základom technológie je projekt Linux Terminal Server (Project) – skrátene LTSP [1]. Pre zavedenie tejto technológie som použil distribúciu Ubuntu 9.04.

Použitie terminálov nie je v oblasti počítačov novinkou, je známe už z dôb sálových počítačov, keď sa viacerí používatelia pripájali k centrálnemu bodu pomocou textových terminálov. V súčasnosti už terminál nemusí znamenať blikajúcu zelenú obrazovku s prácou v textovom režime. Súčasné terminály sú plne grafické, teda poskytujú grafické používateľské rozhranie.

2 Popis technológie

Predmetom nie je detailný popis technológie, ale pre úplnosť uvediem aspoň krátke zhrnutie základných princípov.

Projekt Linux Terminal Server je riešenie, ktoré pridáva linuxovému serveru podporu tenkých klientov, teda klientov, ktoré nemajú nainštalovaný vlastný systém (často bez vlastného pevného disku). Pomocou tejto technológie je možné vytvoriť prostredie (označované ako prostredie chroot), ktoré umožňuje:

- štartovanie (bootovanie) systému po sieti, prostredníctvom protokolu BOOTP alebo DHCP;
- vzdialené prihlasovanie používateľov k serveru s využitím autentifikácie pomocou protokolu SSH;
- vzdialenú prácu na serveri, a to šifrovane (využitím tunela ssh) alebo nešifrovane (priama relácia X);
- spúšťanie lokálnych aplikácií – nainštalovaných priamo na klientoch;

Aktuálny stav LTSP umožňuje použitie ako tenkých klientov, tak i takzvaných tučných klientov, teda klientov, ktorý majú časť alebo všetky aplikácie nainštalované lokálne. Pomocou LTSP je možné transparentné využívanie lokálnych prenosných médií (USB), prenos zvuku a prípadne i využite iného lokálneho hardvéru, ako webové kamery, či programátory mikrokontrolérov...

Pomocou tejto technológie je možné vybudovať cenovo efektívne siete, ktoré dokážu na hardvérovo slabých strojoch využívať plnú výpočtovú silu servera.

3 Použitý hardvér

Pre vybudovanie terminálovej siete som použil:

- terminálový server Hewlett-Packard, vybavený procesorom Xeon 4 x 2000 MHz, 4 GB RAM, dvomi integrovaným sieťovými rozhraniami a dvomi 500 GB SATA diskami;
- tenké klienty RB-851 MINI, vybavené procesorom Intel 800 MHz, 256 MB RAM, grafickou kartou VIA Unichrome, zvukovou kartou AC97, sieťovou kartou a bez pevného disku
- pre 15 klientov je použitá samostatná sieť 100base-T, prepojená štandardným prepínačom (switch)

Použité hardvérové vybavenie plne postačuje pre bežnú prácu, ale skúsenosti ukázali:

- 4 GB RAM pre server a pätnásť klientov, pracujúcich v prostredí Gnome, neposkytuje dostatočnú rezevu pre rozširovanie a použité množstvo pamäte sa často blížilo k hranici 3,5 GB (cca 90%)
- 64 MB grafická karta (zdieľaná pamäť) je na hranici použiteľnosti graficky náročných aplikácií (spôsobuje trhané zobrazovanie pri posúvaní okien alebo prepínaní plôch)
- prepínač 100base-T nepostačuje na prenos videa, keď jeden klient dokázal využiť až 70 Mb/s (teraz je flashplayer zakázaný);

Obrázok 1: RB–851 MINI, s rozmermi $170 \times 124 \times 85\,\text{mm}$

- všetky ostatné komponenty a parametre poskytujú pohodlnú rezervu na dlhodobé využívanie terminálovej siete.

Vzhľadom k tomu, že rozšírenie RAM servera a nahradenie prepínača pre GB sieť nie je problém (s nákladom cca 300 €), jediným úzkym hrdlom je veľkosť pamäte (a výkon) grafickej karty klientov.

4 Používaný softvér

Aj vzhľadom na množstvo dostupnej RAM som počas školského roka experimentoval a študentom som menil východzie grafické prostredie. Najmä z dôvodu minimalizácie použitia pamäte, ale toto experimentovanie umožnilo žiakom zoznámiť sa s viacerými spôsobmi realizácie grafického prostredia, počnúc Fluxbox a OpenBox, pokračujúc FVWM-Crystal a LXDE až nakoniec ku Gnome.

V učebni vybavenej terminálovou sieťou študenti pri výuke najčastejšie používajú nasledujúci softvér:

- OpenOffice 3.1 – pre vyučovanie práce s kancelárskymi aplikáciami;
- FireFox 3.5 a Opera 10.10 – pre prístup na web;
- Ktechlab – pre simulácie elektronických obvodov;
- Kicad – pre kreslenie elektrotechnických schém a návrh plošných spojov.

5 Pohľad študentov

Všetko, čo som doteraz uviedol je môj pohľad na výkon terminálovej siete a považujem ho za dosť kritický. Samotné zriadenie Linuxovej učebne sa stretlo s nevôľou niektorých

kolegov, ktorí to označili až za zbytočne vyhodené peniaze. Oba tieto dôvody, ale nie len tie, ma viedli k príprave jednoduchého dotazníka, obsahujúceho pätnásť otázok, ktorý som na konci školského roka dal vyplniť študentom, ktorí sa učia v mojej triede.

Cieľom môjho malého výskumu bolo overiť niekoľko téz, ktoré zaznievajú od mojich kolegov, či v internetových diskusiách:

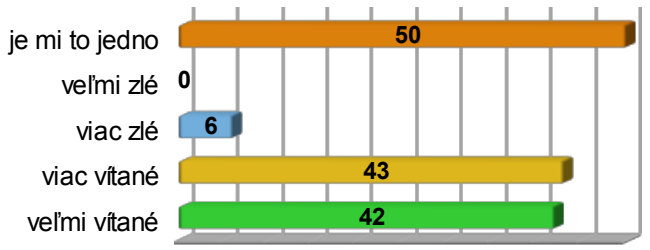
- vyučovanie menšinového softvéru (linux, otvorený softvér) je pre študentov nezaujímavé;
- používanie vzdialenej práce po sieti je pomalé;
- bezplatná legálnosť otvoreného softvéru je jeho výhodou;

A tiež ma zaujímalo, aké skúsenosti s Linuxom, či otvoreným softvérom si študenti prinášajú z domu, čo si myslia o používaní otvoreného softvéru firmami (budúcimi zamestnávateľmi) a či uvažujú o používaní otvoreného softvéru do budúcnosti.

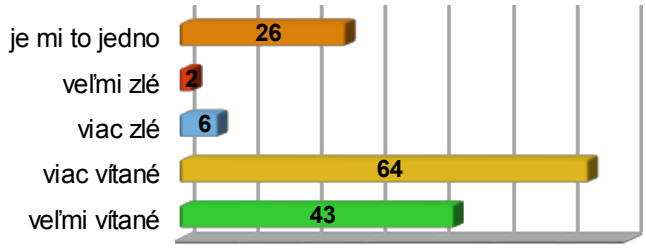
6 Výsledok prieskumu

V prieskume som rozdal 120 dotazníkov a zozbieral odpovede 141 žiakov, celkom z 13 tried. Týchto trinásť tried je približne rovnomerne rozložených medzi všetky ročníky, ktoré študujú v našej škole, pričom sa nejedná o všetkých študentov, pretože v niektorých triedach učím iba jednu skupinu. Výsledky prieskumu predkladám v grafickej podobe po jednotlivých otázkach, v poradí tak, ako boli v dotazníku, pričom čísla v jednotlivých stĺpcov udávajú počet odpovedí pre danú možnosť.

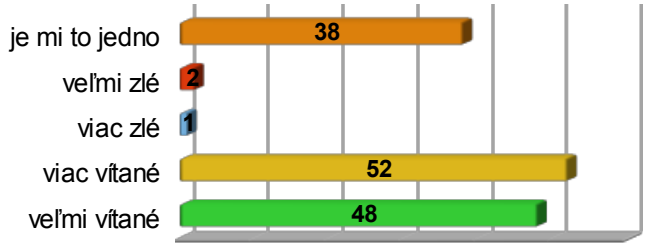
1. Vyučovanie v inom operačnom systéme ako Windows je pre Vás:



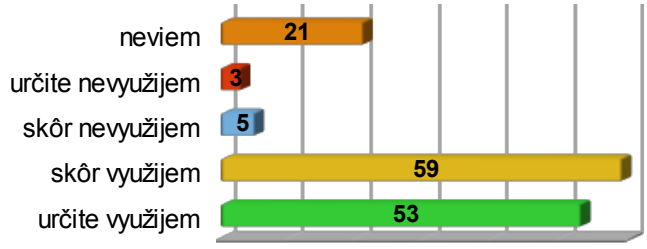
2. Vyučovanie používania iných programov, než na aké ste zvyknutý je pre Vás:



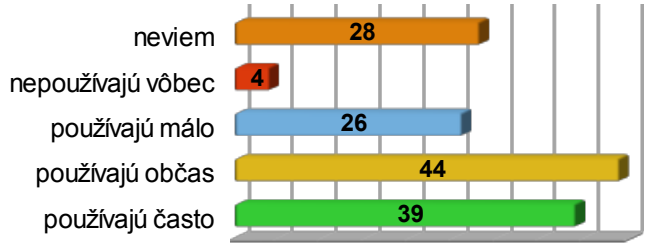
3. Používanie otvoreného softvéru pri vyučovaní je pre Vás:



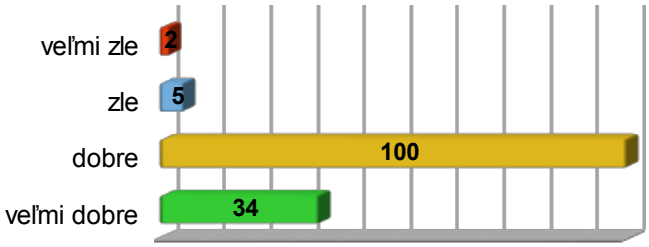
4. Vedomosti a zručnosti, získané pri vyučovaní s otvoreným softvérom:



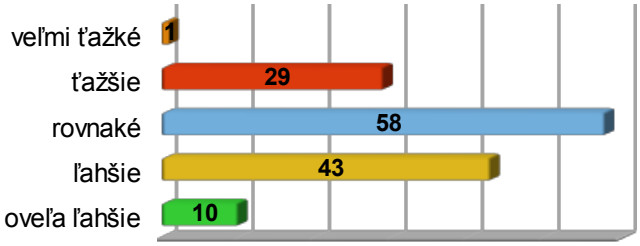
5. Myslím si, že otvorený softvér firmy:



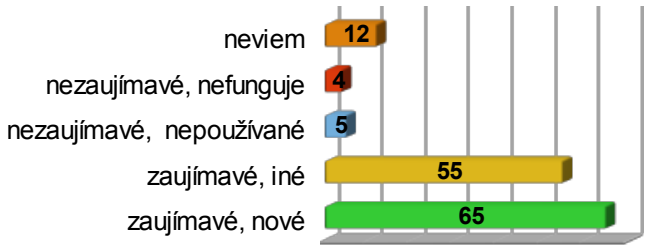
6. V operačnom systéme Linux sa Vám pracuje:



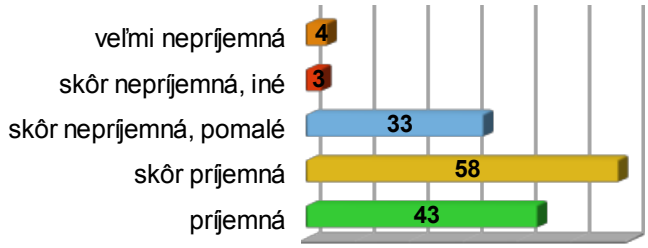
7. Používanie operačného systému Linux je v porovnaní s Windows pre Vás:



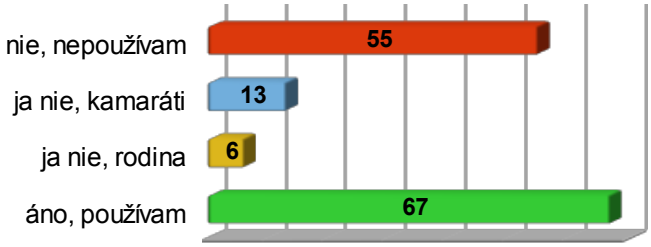
8. Používanie tenkých klientov (tých maličkých počítačov) je pre Vás:



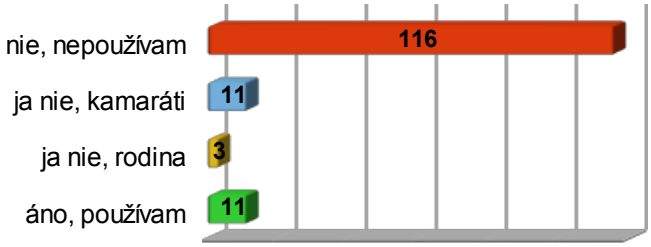
9. Práca s tenkými klientmi (tými maličkými počítačmi) je pre Vás:



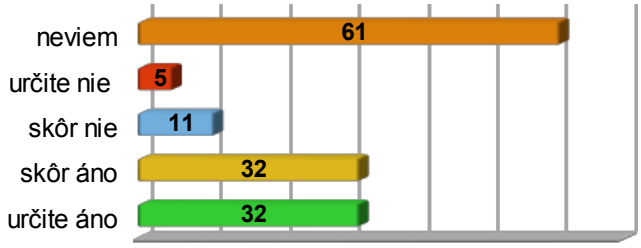
10. Používate doma otvorený softvér?



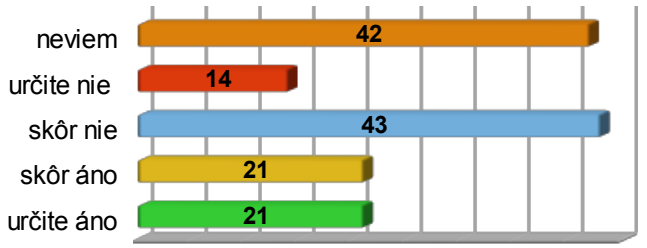
11. Používate doma operačný systém Linux?



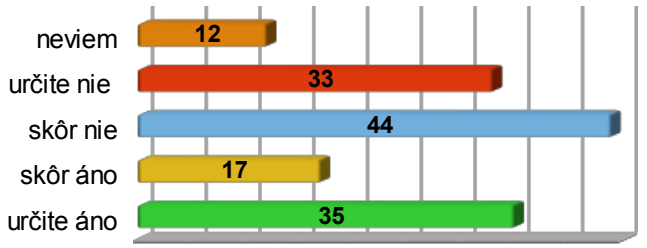
12. Uvažujete o používaní otvoreného softvéru v budúcnosti?



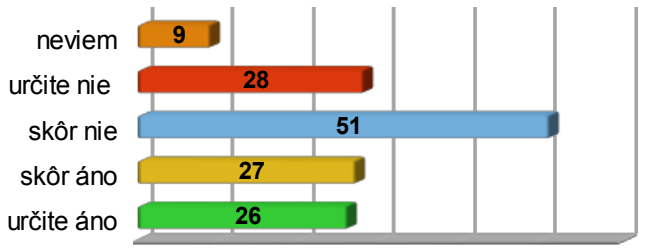
13. Uvažujete o používaní operačného systému Linux?



14. Používate doma legálny softvér?



15. Je pre vás legálnosť softvéru dôležitá?



7 Vyhodnotenie prieskumu

Výsledky prieskumu ma veľmi milo prekvapili, takú pozitívnu odozvu od žiakov som neočakával, najmä v prvej časti (otázky 1 -. 4). V rozhovore po vyplnení dotazníka som ešte zisťoval, čo chceli povedať odpoveďou "Je mi to jedno", ale i túto odpoveď možno brať pozitívne, pretože v rozhovore študenti potvrdili postoj k softvéru ako k nástroju, teda že im je naozaj jedno, aký nástroj používajú. Týmito svojimi odpoveďami vyvrátili negatívne názory na používanie otvoreného softvéru vo vyučovaní.

V ďalšej časti vyvrátili časté tvrdenia o ťažkom používaní Linuxu v porovnaní s Windows (otázky 6 a 7), samozrejme študenti vystupujú čisto ako používatelia, ktorí majú minimálnu možnosť konfigurácie systému.

Osobne najväčšie obavy som mal z hodnotenia práce tenkých klientov (otázky 8 a 9), ale i tu prevážili pozitívne postoje a dokonca hoci boli v dotazníku všetky otázky uzatvorené, jeden študent mi pripísal, že nepociťuje žiadny rozdiel oproti plne vybavenému stroju. Prial by som vám vidieť ten údiv v očiach prvákov, keď zistia, že počítač je tá malá skrinka na stole a mnohí z nich proste neveria, že by to fungovať mohlo.

Zaujímavé výsledky ukazuje zisťovanie používania otvoreného softvéru študentmi doma (otázky 10 – 13). Drvivá väčšina študentov doma Linux nepoužíva a stretla sa s nim až na mojich hodinách. Na základe tohto zistenia možno hodnotiť ich pozitívny postoj k používaniu Linuxu jednak ako prekvapenie, že Windows nie je jediný funkčný operačný systém a jednak ako osobnú skúsenosť, ktorá vyvracia časté názory internetových fór o nepoužiteľnosti Linuxu a otvoreného softvéru všeobecne. Z diskusie po vyplnení dotazníkov však vyplynulo i to, že mnohí doma používajú otvorený softvér (najmä FireFox), ale nevedia, že to otvorený softvér je.

Z posledných dvoch otázok ďalej vyplynulo, že mladú generáciu legálnosť softvéru veľmi netrápi. Nie je to prekvapujúce zistenie, je však alarmujúce, pretože v priebehu školského roka sa priznávalo k nelegálnemu používaniu oveľa vyššie percento študentov. Ukazuje to však i to, že na zaujatie mladej generácie pre myšlienky otvoreného softvéru nie je táto téma veľmi účinná.

8 Záver

Prezentované skúsenosti i výsledky prieskumu medzi žiakmi ukazujú, že voľba pre použitie terminálového riešenia učebne bola v mojom prípade správna, a to napriek zmieneným nedostatkom. Použitie tohto riešenia výrazne znížilo náročnosť na správu softvéru v učebni a výrazným spôsobom zaujalo študentov.

Napriek uvedeným skutočnostiam a napriek jednoduchosti inštalácie a zavedenia terminálovej siete v Linuxe, neodporúčam uvedené riešenie pre začiatočníkov v oblasti správy Linuxových serverov a desktopov. Miestami dochádza k pádu programov, či k ich zaseknutiu a vyťaženiu jedného alebo viacerých jadier procesora na 100 %. Stretávam sa i s problémami, keď študenti vypnú klienta bez toho aby sa odhlásili, čo je potom potrebné urobiť manuálne za nich.

Literatúra

[1] Linux Terminal Server Project – domovská stránka `http://ltsp.org/`

[2] Ubuntu – domovská stránka `http://www.ubuntu.com/`

Kontaktná adresa

Ing. Slavko FEDORIK,
Stredná odborná škola elektrotechnická,
Hlavná 1, 059 51 Poprad-Matejovce, `slavko.fedorik@soumatej.sk`

ZMENY VO VÝUČBE UNIXU NA FEI STU OD ŠK. ROKU 2010/2011

FODREK, Peter, (SK); FOLTIN, Martin (SK); BLAHO, Michal (SK)

Abstrakt. Unix sa, pre bývalý študijný program automatizácia a dnešný štúdijný program Priemyselná informatika (PI), učí už približne 20 rokov. V analýze [1] prednesenej na akcii 11th Real-Time Linux Workshop (RTLWS11) konanej, pod vedením Real Time Linux Foundation Working Group neziskovej organizácie Open Source Automation Development Lab (OSADL), dňa 28.–30. septembra 2009 v Drážďanoch sme analyzovali príčiny neobľúbenosti predmetu. Existujú tri typy príčin. Jedným typom je prístup väčšiny ZŠ a SŠ vo výučbe informatiky k Linuxu a programovaniu. Druhá oblasť je nedostatok počítačov s predinštalovaným Linuxom. Tretia oblasť problémov je neprispôsobenie pedagogických metód zmeneným podmienkam. A práve túto skupinu problémov chceme riešiť pomocou prístupov uvedených v tomto príspevku.

1 Úvod

Výučba systémového paralelného programovania v Unixe, na úrovni utilít, prešla za približne 20 rokov drastickými zmenami. Učilo sa postupne na operačných systémoch a distribúciách GNU/Linux

a) QNX

b) Solaris 2.0

c) Mandrake Linux 9.2 +Debian

d) Fedora Core 4 +Debian

Dvaja starší autori príspevku, v roku 1997, zažili, ako študenti, Solaris 2.0 s Open Windows na dvoch X-termináloch a telnetový prístup z MS-DOSu na solarisový server z ďalších 8 počítačov v dvoch učebniach. Študenti sa striedali tak, že jeden týždeň boli v jednom a druhý v inom laboratóriu. V každom laboratóriu bol prítomný iný z dvoch cvičiacich,

ktorý si miesto nemenili pri výmene študentov. To dávalo možnosť porovnávať ako aj prijať rôzne prístupy k problému. Navyše bol predmet v 3. nominálnom ročníku štvorročného bakalárskeho štúdia a predchádzal mu predmet Operačné systémy.

Na Operačných systémoch (OS) sa učilo skriptovanie v tcsh na súborových systémoch so zákernosťami: ako súbor s nula až troma medzerami alebo netlačiteľnými znakmi na začiatku mena súboru, či symbolický link na vyššiu úroveň v strome s tým, že sa hodnotilo stratou 10 z 15 bodov to, že sa skript nezacyklil. Posledné zadanie z OS bolo na medziprocesovú komunikáciu (IPC) a procesy v jazyku C. OS sa učil v letnom semestri 2. nominálneho ročníka ako povinne voliteľný predmet. Keďže však bol jediným povinne voliteľným predmetom a študenti si museli zvoliť najmenej jeden povinne voliteľný predmet z ponuky, bol de facto povinným. Potom, ako sme OS absolvovali, sa predmet OS začal hodnotiť tak, že namiesto dvoch týždňov sa muselo vypracovať zadanie za dve 50-minútové hodiny. To viedlo k tomu, že v priebehu dvoch rokov získali zápočet len 4 študenti nášho študijného odboru (vtedy ešte). Pritom ročníky mali 120 a 100 študentov. Keďže predmet učila vtedajšia Katedra informatiky a výpočtovej techniky (dnešná FIIT) vytvorila sa „alternatíva“ vo forme neUnixového predmetu. Toto považovali KIVT za podraz a urážku, čím sa začal proces delenia fakulty na FEI a FIIT.

Je to smutné, lebo z teoretických znalostí z OS sa dalo ťažiť dodnes. Len vďaka nim bolo možné pochopiť nové algoritmy plánovania procesov pre OS reálneho času, ktoré boli prednesené na RTLWS11 v Drážďanoch.

Faktické zrušenie predmetu OS navyše zásadne zmenilo podmienky výučby predmetu. Hoci na OS mal každý študent len jednu konkrétnu formu IPC a na vtedajšom predmete Programovacie prostriedky reálneho času sa preberali všetky formy a prostredky na IPC, knižnica nCurses a sockety, čo je však len špeciálna forma IPC, bol predmet OS veľkou pomocou. Našou výhodou bolo aj to, že ak sme chceli používať Internet a nemali sme iný prístup k nemu, museli sme od prvého ročníka používať prístup cez telnet z MS-DOSu na server DECEF (z DEC na ElektroFakulte) prípadne k DECEFu pristupovať pomocou textových terminálov vdt52s. Prehliadače boli lynx a www. Na DECEFe bežal Ultrix t. j. Unix od Digitalu (DEC =Digital Equipment Corporation), neskôr FreeBSD, openBSD a netBSD. Predmet OS sa učil na vdt52s na serveri ALF (podľa procesoru DEC Alpha AXP). „Na ALFovi“ bol okrem prehliadačov www a lynx aj NCSA Mosaic, ktorý sa dal spustiť na X-termináloch. DECEF bežal na architekúre VAX, dnes je aj on PC-serverom s BSD.

Najmladší autor Ing. Blaho, absolvoval predmet na Mandrake (po zlúčení s Connectivou sa dnes volá Mandriva) so serverom KAR10 na Debiane. Spoluautor Fodrek učí dnes predmet na Fedore. Ak si pripomnieme, že Výpočtové stredisko FEI, a s ním aj Centrálna počítačová učebna bežia na Windows 7 a učebňa na predmet má dualboot s Windows XP, vychádza jasne, že treba zmeniť prístup k výučbe. Navyše LDAP, NFS a ssh server KAR10 bol vymenený za KAR11. Stále sa však toleruje prístup na KAR11 (KAR je skratka bývalej Katedry automatizácie a regulácie, ktorá je po zlúčení súčasťou Ústavu riadenia a priemyselnej informatiky) cez ssh a priamo sa študentom ponúka Windows based ssh klient Putty.

2 Aktuálny stav výučby

Napriek uvedeným zmenám v pripravenosti študentov na štúdium predmetu nedošlo k zmene filozofie a prevedenia predmetu. Skôr naopak. Pri zmene bakalárskeho štúdia sa skrátili cvičenia z troch na dve hodiny týždenne. Nedôveryhodne pôsobia dnešným študentom aj prednášky na priesvitkách ukazovaných cez spätný projektor. Vrcholom ale je, že vedúci cvičení, ktorý nie je autorom, prichádza občas na cvičenia s notebookom s Windows. Pritom študentom hovorí, aký je Unix a Linux dobrý.

Problém vypukol aj, keď sa predmet učil jeden rok aj pre študentov štúdijného programu Aplikovaná informatika (AI). Úspešnosť študentov v získavaní zápočtov AI bola 44% a PI bola 88%. Aj za tých 88% sme dostali ústne napomenutie od vtedajšieho riaditeľa ústavu spolu s príkazom zvýšiť priechodnosť predmetu. Študenti AI už na ďalší rok nemali v programe tento predmet. Práve 44% je podľa Saeda Dehnadiho [2] a iných štúdií percento študentov IT schopných naučiť sa programovať.

Keďže študenti AI, ktorí nezískali zápočet sa nedostavili ani na riadny ani na opravný zápočet, ktorý spočíval v odovzdaní 3. zadania, nemohli sme konať inak. Zarážajúci, ale bol rozdiel medzi študentami programu, kde bol predmet nový, a programu, kde už poznali zadania daného typu. V reakcii na to sme zmenili zadania na také, že sa neopakujú a ani nemôžu. Ide o simuláciu sieťových regulačných obvodov, kde na cca. 62 študentov je 13 regulačných obvodov. 12 z regulačných obvodov má po dva riadené systémy a pre každý typ riadených systémov je bežne 6 rôznych prenosových funkcií najmenej však 3 prenosové funkcie. To dáva od 118 do 438 rôznych zadaní. Stačí však vygenerovať každých 4-5 rokov nové prenosové funckie a zadania sa nebudú opakovať. Predtým to bolo 15 rôznych technologických procesov, ktoré však nemali daný matematický opis a tak boli stále rovnaké.

Výsledkom tejto zmeny bol inzerát v internej sieti študentského domova, ktorý ponúkal po 2 000 Sk za vypracovanie každého z dvoch kontrolných zadaní. Tieto zadania sa odovzdávajú v 5. a 9. týždni semestra a majú hodnotu po 10b. Naviac tam bola ponuka 5 000 Sk za záverečné zadanie za 30 bodov. Na inzerát nikto nereagoval, lebo sa študentom, podľa výrazu dvoch najkvalitnejších študentov v ročníku, neoplatilo za také peniaze vypracovať zadanie kolegom. Zaujímavé je, že študent, ktorý sa priznal k autorstvu inzerátu, vypracoval nakoniec zadania sám na 46 z 50 možných bodov. Ale na záver podotkol, že mu to zabralo príliš veľa času.

O rok sme teda porovnávali časy potrebné na vypracovanie zadaní s časmi uvedenými v štúdijnom programe. Platí totiž pravidlo, že počet kreditov za predmet sa rovná súčtu týždenného počtu hodín prednášok, cvičení a samostatnej domácej práce resp. prípravy na predmet. Keďže predmet má 5 kreditov, 2 hodiny prednášok a 2 hodiny cvičení, má byť teda 1 hodina domácej prípravy týždenne. Ako to ale vyzerá v skutočnosti? Študenti uvádzajú, že pri riešení prvého zadania doma strávili 15–18 hodín. Pri druhom zadadní to bolo 14–15 hodín a pri treťom 40–45 hodín. To dáva dohromady 70–80 hodín domácej prípravy za semester. To je však obrovský nepomer k plánovanej jednej hodine týždenne, teda 12–13 hodinám za semester. Naša veková kategória mala iba jedno 50 bodové zadanie,

ale bolo odovzdávané od 8. týždňa, pričom najlepší jedinci nemali dva týždne pred minimálnym týždňom odovzdávania čo robiť. A to si ešte sami museli najskôr do 3. týždňa vymyslieť zadanie, aby tam boli všetky preberané oblasti, a ktoré nakoniec odobril cvičiaci. Na riešenie sme teda mali 3. týždne a celkovo riešenie zabralo 25–30 hodín (pol dňa týždenne v prestávkach vo vyučovaní a pol soboty alebo nedele doma).

Vtedy, keď sme prieskum robili, tak mal Ing. Fodrek názor, že obsah predmetu pokrýva len 50% reálnych potrieb programátora systémov reálneho času. Po RTLWS11 ho upravil tak, že predmet pokrýva len 25% potrebných vedomostí. Pri časovej náročnosti predmetu je to však neriešiteľný problém.

3 Realizované zmeny

Ostáva aplikovať zmeny vo výučbe. Dve už nastali, a hoci problém len zvýraznia, majú pozitívny vplyv na vzdelanosť. Prvou je rozdelenie dvojíc na jednotlivcov. Tým sa časová náročnosť de facto zdvojnásobí. Aj keď sa vie, že aj tak dominantnú časť, alebo celé zadanie, vyriešil jeden študent a druhý sa mu odvďačil vypracovaným zadaním z iného predmetu. Druhá je zmena názvu. Po Programovacích prostriedkov reálneho času máme dnes Sotvér riadiacich systémov I. Od šk. roku 2010/2011 sa bude predmet volať Unix/Linux – systémy reálneho času. Zároveň sa zmenila dotácia na 3 hodiny cvičení a zostali dve hodiny prednášok ako aj hodina predpokladanej domácej práce. Z toho vyplýva pridanie kreditu. Predmet sa presunul z letného do zimného semestra II. nominálneho ročníka. V súvislosti s odchodom prednášajúcej do dôchodku ku 1. 11. 2010 sa mení aj prednášajúci. Podľa študijného programu by ním mal byť terajší bežný cvičiaci Fodrek.

4 Zamietnuté zmeny

Existoval návrh na spojenie predmetov Základy systémov reálneho času (ZSRT), ktorý sa premenuje na Java – grafické rozhranie RS. Keďže pôvodne bolo ZSRT náhradou za Operačné systémy, bolo by to logické. Navyše by to umožnilo presunúť výučbu Javy z Windows na GNU/Linux. Tak isto by bolo možné mať jediný vývojový prostriedok na oba predmety napr. Eclipse. Eclipse ovláda ako C používané v Unix predmete, tak aj Javu. Navyše Java predmet mal dotáciu 2 hodiny prednášok a 3 hodiny cvičení (po novom má 2 hodiny cvičení). Po spojení by vznikol predmet so 4 hodinami prednášok a 5 hodinami cvičení týždenne. Zmena bola navrhnutá spolu s pridaním hodín samostatnej práce na tri. To sa zamietlo, a dokonca by sa znížil počet hodín samostatnej práce na jedinú hodinu, z dvoch hodín na oba predmety v súčte. Navyše po novom je Java – grafické rozhranie RS povinne voliteľným predmetom, kde sa volí jeden z dvoch.

Z dôvodu nedostatku financií a odmietnutia grantu KEGA nemohlo byť laboratórium určené na cvičenia renovované po stránke hardvéru a ani softvéru hoci boli začaté predbežné rokovanie a akademickej licencii Red Hat Enterprise Linux MRG Realtime.

5 Navrhované zmeny v prístupe k prednáškam

V prístupe k prednáškam sa navrhuje použiť notebook s Unixom. Je to dokonca podmienka vedenia ústavu pre pokračovanie predmetu. Problémom je aký notebook vybrať? Na našom trhu by malo byť niekoľko notebookov s Linux-om ako OEM operačným systémom. Ich rozvrstvenie v podstate vystihuje pozícu Linuxu všeobecne. Bezplatné Linuxy v najnižšej triede a platené Linuxy v najťažších kalibroch majú silnú pozíciu, ale v bežnom segmente prakticky neexistujú. Väčšina dodávateľov počítačov v SR neponúka počítače s Unixom, majú len Windows, maximálne ak ponúkaju PC bez Operačného systému.

Táto zmena spôsobí zmenu teoretických prednášok do praktickejšej formy s ukážkami konkrétnych programov, ktoré sa doteraz robili na cvičeniach. Umožní sa tým využiť lepšie čas cvičení. Nejaká miera teórie je ale nutná. Je potrebné nájsť vhodný pomer. Momentálne je to 3:1 v prospech teórie. Plán je zmeniť to na 1:1 až 1:2 v jej neprospech. Navrhuje sa aj sprístupniť prednášky mimo laboratória. To však predpokladá ochranu voči plagiátorstvu zo strany STU. Terajší stav je taký, že materiály sú dostupné len v laboratóriu cvičení. Je to preto, lebo nastal prípad, že sa na inej univerzite objavila doslovná kópia prednášok terajšej prednášajúcej, len so zmeneným menom prednášajúceho. Vedenie však odmietlo ísť do právneho sporu a tým chrániť autorské práva k prednáškam.

6 Navrhované zmeny na cvičeniach

Na cvičeniach sa navrhuje preberať zadania každé cvičenie resp. na 10 cvičeniach po 5 bodov, alebo každý týždeň po 2 body a 30 bodov za záverečné zadanie. Tak isto sa navrhuje, aby sa obmedzili, ak nie zrušili, prezentácie demonštračných príkladov na cvičeniach. Tak budú k dispozícii len zadania jednotlivých cvičení a odpadne podobný problém s autorským právom ako bol na prednáškach.

7 Zmeny v obsahu predmetu

Vzhľadom na to, že dnes sa „čaká“ na ukončenie témy procesy na prednáškach a až potom sa pristúpi k jej preberaniu na cvičeniach, druhé a tretie cvičenie sa preberala knižnica nCurses (new curses=nové zaklínadlá, nové kliatby, nové prekliatia). Vzľadom na snahu, aby nás, na konci semestra, študenti opäť nepreklínali, tak túto časť plánujeme zrušiť, prípadne presunúť na koniec semestra v menšom rozsahu.

Uvoľnené miesto plánujeme využiť na cvičenie obsluhy systému, ktoré sa teraz cvičí len v prvom týždni semestra. Prípadne môže byť súčasťou tejto kapitoly aj InstallFEST. Namiesto asi 1/3 prvej prednášky a jedného cvičenia by sa obsluha učila 1-2 prednášky a 1-2 cvičenia. Téma procesy sa bude prednášať dva namiesto terajších skoro troch týždňov. Cvičenia ostanú v dvojtýždňovom rozsahu. Túto tému totiž zvládajú všetci študenti. Téma

signály ostane na jednotýždňovej dotácii na prednáškach a cvičeniach, hoci sú v nej drobné nedostatky, z pohľadu koncových znalostí študentov.

Téma časovače a čas spôsobuje študentom najväčšie problémy z celého predmetu, aj preto, že sa učila v 5. týždni, keď sa preberalo zadanie prvého bloku. Preto ju navrhujeme rozšíriť na dva týždne. Ide totiž o kľučovú tému predmetu. Téma však využíva procesy aj signály. Časovače oznamujú svoje vypršanie štandardne pomocou signálov. Sockety a spolupráca s DNS, ktoré ako-tak zvládli všetci študenti sa budú učiť rovnako ako teraz dva týždne. Ide o tretiu najdôležitejšiu tému predmetu. Vlákna robia len drobné problémy, tak ostanú na týždňovej dotácii. Možno sa presunú pred procesy alebo signály, keďže sú procesom podobné a signály im vedia byť doručené.

Nasleduje medziprocesová a medzivláknová komunikácia a synchronizácia (rady správ, rúry/pipe, zdieľaná pamať, semafory ...). Ide o druhú najdôležitejšiu tému. Preto jej venujeme zvyšok semestra. V rámci obsluhy sa budeme po novom venovať ladiacemu nástroju gdb. Viac sa budeme venovať gcc/g++/ln a tvorbe Makefile. Tak isto sa sem presunie dodatková téma o tvorbe binárneho spúšťacieho súboru z viacerých zdrojových kódov. Toto totiž kód sprehľadní, rovnako ako delenie kódu do funkcií, ktorému sa študenti bránia.

8 Podporné aktivity

Pokúsili sme sa otvoriť krúžok obsluhy Unixu pre stredné školy. Individuálne sa prihlásila jedna študentka a hromadne prišla 13-členná skupina tretiakov zo súkromného gymnázia. Priviedol ju ich učiteľ, absolvent FMFI UK. Z tej skupiny prejavilo záujem študovať na technickej škole asi 10 študentov, z nich len traja mali záujem o krúžok. Krúžok teda stroskotal a pravdepodobne sa nebude už konať.

9 Styk s praxou

Súborom neriešiteľných problémov je dať do súladu požiadavky trhu práce a mentálne možnosti študentov. Pán Klaus nedávno hľadal programátora v C pod Unixom na embedded systémy [3]. Podmnožinou embedded systémov sú hard realtime systémy, ktoré by mali vedieť tvoriť absolventi Priemyselnej informatiky. Uveďme si teda otázky príjímacieho testu pána Klausa, ku ktorým som pridal komentár kedy sa tá-ktorá časť učí na našom predmete.

Znalostní test

- **Jaké znáte mechanizmy meziprocesní komunikace v Linuxu a stručně je popište i s případnými klady a zápory.** (obsah otázky sa učí 1/5 I. bloku. dve tretiny II. bloku a celý III.blok)
- **Jaké znáte nástroje pro distribuovanou (různé instance OS) meziprocesní komunikaci a stručně je popište i s případnými klady a zápory?** (v celom druhom bloku a na poslednej prednáške, ktorá nie je obsahom cvičení)

- **Co je asynchronní komunikace (v rámci IPC) a kdy je výhodná?** (v 1/5 prvého bloku, v 1/6 druhého bloku a v 1/3 tretieho bloku)

- **Potřebujete, aby vašich 5 aplikací mezi sebou v rámci jednoho OS a jednoho uživatele komunikovaly - předávaly si krátké zprávy. Mezi jakými topologiemi komunikace můžete vybírat a jaké jsou jejich klady a zápory.** (v 2/5 I. bloku, v celom II. bloku a v celom III. bloku)

- **Napište krátký program, který pustí najednou 5 podprocesů. Všechny podprocesy „usnou" náhodně na 5-10 sekund a bez další akce se ukončí. Rodič čeká na všechny procesy a jakmile se jeden potomek ukončí, vypíše na stdout číslo procesu, který skončil. Pokud skončily všechny procesy, automaticky se ukončí také. K jakému problému může dojít a jak ho budete řešit?** (v 2/5 prvého bloku a v 1/3 druhého bloku)

- **Pokud spustíte program z předchozího bodu a po vzniku podprocesů zmáčkněte ctrl-c, co se stane s těmito podprocesy? A proč? Jak byste změnili program, abyste změnili toto chování?** (v 1/5 prvého bloku)

- **Píšete aplikaci, ve které potřebujete zpracovávat data každou 1ms. Napište tedy smyčku, která bude číst stdin a tato data vypisovat na stdout průměrnou rychlostí jeden znak za 1ms bez ohledu na rychlost dat na vstupu. Popište všechny možné problémy, které musíte řešit, jejich původ a jejich řešení.** (v 3/5 prvého a 1/3 druhého bloku)

- **Píšete grafickou aplikaci v Linuxu, jejíž součástí má být oblast okna, které bude co nejvěrněji emulovat příkazovou řádku - terminál v rozlišení** 80×25**. Jakou rychlou techniku byste použili a na co je potřeba dát pozor?** (Táto oblasť sa neučí pre jej náročnosť. Ale jedno z možných riešení matne súvisí s nCurses, ktoré sa učia na druhom a treťom cvičení prvého bloku. nCurses ale chceme „obetovať" v prospech obsluhy systému.)

- **Co je window manager a jaká je jeho úloha? Pomůže vám v tzv. kiosk módu aplikace např. web browser? Proč?** (Táto oblasť sa neučí, nepriamo sa spomenie jednou vetou ako voľba gdm. Korektný KIOSK mód (t. j. bez Window managera) je oriešok aj pre kóderov jadra, ktorí sú našimi absolventami.)

- **Potřebujete monitorovat chování vaší aplikace tj. vsunete do ní posílání zpráv v místech, které potřebujete sledovat. Navrhněte, jak tuto aplikaci sledovat na stovkách stanic v lokální síti? Pokud navrhnete více možností, vysvětlete jejich klady a zápory.** (Táto oblasť sa korektne nepreberá. Využiť pri, otázkou, navrhovanom nekorektnom riešení sa dá druhý blok, 1/3 tretieho bloku a obsah prednášky, ktorá sa nepreberá na cvičení.)

- **Vytvořili jste web browser postavený na existující knihovně. Při spuštění se ale vizuálně nenačítá žádná stránka. Co byste zkusili zjistit jako první a jak?** (Toto sa neučí, iba ak okrajovo v druhom bloku a na inom predmete.)

- **Potřebujete zjistit, zda v poslední hodině vaše aplikace na některé ze stovek stanic vygenerovala chybovou hlášku. Jak to zjistíte a jaké nástroje k tomu použijete?** (Táto téma sa, de facto, neučí, lebo je o expertnom ovládaní povelového riadku a aj po pridaní času bude v predmete len rýchlokurz povelového riadku.)

- **Potřebujete, aby se vaše aplikace resp. sada aplikací spouštěla automaticky po nastartování Linuxu (kiosk mód - tj. uživatel pracuje jen s vaší aplikací). Jak to zajistíte? Je to spojeno s nějakým omezením či rizikem?** (Toto niečo medzi témou Window manager a predchádzajúcou otázkou. Preto sa to neučí. Niečo sa preberie po zmene.)

- **Zjistili jste, že vaše aplikace v produkčním nasazení je v ojedinělých případech (jednou za dva dny) nestabilní a padá. Odhadujete, že vyřešení bude vyžadovat minimálně týden práce. Co uděláte, jako rychlou a dočasnou úpravu?** (Učiť toto neprichádza do úvahy. Veď študenti nevedia dobre ani len analýzu kódu. Syntéza je neskonalý problém. Ešte ako-tak zvládnu implemetáciu s využitím demo príkladov. A o zvyšku životného cyklu programu, okrem dvoch zo 65 študentov, ani len nepočuli)

- **Potřebujete v knihovně webkit povolit 3D transformace, které fungují jen v Safari verzi a v ostatních jsou zakázané. Zkuste lehce odhadnout, jak budete postupovat.** (Táto téma sa čiastočne učí na inom predmete.)

Ako vidno z testu je jeden predmet príliš málo na obsiahnutie problematiky. Úvodom do ťažších oblastí je bakalárska práca Michala Praženku: „Analýza plánovačov v Linuxe“ z mája 2010. Tento dištančný študent sa, ako prvý u nás, a asi aj na svete, aspoň okrajovo, zaoberal vplyvom I/O plánovačov na hard realtime. Doterajšie práce sa zoaberali len plánovačmi procesov. A pritom vstupno-výstupné operácie sú pre deadline rizikovejšie. Plánovanie procesov je dôležité pre akademikov pre obmedzenie latencie. Praktikov zaujíma deadline teda latencia+ čas vykonania úlohy. A v tej druhej časti je plánovanie I/O kritické. Aj napriek problémom sa teda občas objaví dobrý študent, ktorý vytvorí kvalitnú záverečnú prácu. Pred pánom Praženkom bol podobný Vladimír Chren, ktorý urobil, na prelome rokov 2007–8, za 45 dní kernelový port CANOpen protokolu. O to isté sa pokúšali francúzi od roku 2002 do roku 2005 a neuspeli. Globálne tu uspel koncern VW, ktorý vyvíjal CAN protokoly a ovládače od roku 2002 a až v marci 2008 mal funkčný kód. CANOpen od VW fungovalo plnohodnotne vo vanila jadre až v septembri 2009. Pritom VW v januári 2010 predstavilo prvé plne linuxové auto na svete: Audi A8 model roku 2011. Ostatné značky koncernu budú plne linuxové do konca roku 2014. BMW majú zatiaľ Linux len na audiosystéme.

Problémom je, že takých študentov je minimum a prax a predmet sú stavané na ich schopnosti. Toto potvrdzuje aj jedna z popredných vývojárov Google, absolventka Žilinskej univerzity, Petra Popluhárová. V rozhovore [4] sa vyjadrila jasne o prijímaní do Google:

„Bola to poriadna zaberačka. Nikoho nezaujímalo, aký mám titul alebo vzdelanie, ale dostávala som záludné otázky z programovania. Musela som riešiť algoritmy a rôzne hlavolamy, podľa ktorých softvéroví inžinieri zisťujú, ako to uchádzačom o prácu myslí.“

A to treba povedať, že Google sídli v oblasti Silicon Valley (medzi južným San Franciscom a San José). Priamo v Silicon Valley, alebo do 20 km okolo neho sídli centrála, alebo IT pobočka 2.,4. a 5. najlepšej univerzity sveta v technických vedách podľa Academic Ranking of World Universities z čínskeho Šanghaja. V programovaní a IT sú to dokonca 1., 3. a 4. najlepšia univerzita. Ide o miestnu Stanfordovu univerzitu (Leeland Stanford junior University) a University of California at Berkeley ako aj Pittsburgskú Carnegie Mellon University s IT pobočkou Carnegie Mellon Silicon Valley. V programovaní sa medzi uvedené univerzity vklienila len celkovo najlepšia technika: MIT (Massachusetts Institute of Technology) zo 105-tisícovej „Bostonskej Petržalky“ Cambridge, MA. Napriek tomu nemá vzdelanie a ani ukončená škola žiadny vplyv na prijímanie do práce v oblasti nami vyučovaného predmetu.

Cambridge, Massachusets by mal byť vzorom pre vzdelávacie systémy. Na 105 tisíc ľudí má 14 univerzít. 11 z nich sú college, teda naše odborné vysoké školy. Tie môžu vydávať len bakalárske diplomy. Jedna zo 14 je obyčajnou univerzitou a dve sú výskumné univerzity. Tie majú na jedného študenta bakalárskeho štúdia troch študentov inžinierskeho/magisterského štúdia. Na jedného študenta druhého stupňa pripadá na výskumných univerzitách päť doktorandov. Najväčším problémom predmetu je, že je stavaný na uvedený model výskumnej univerzity, čo je u nás neexistujúca kategória univerzít. Navyše predmet nerešpektuje mentálne danosti študentov a ich nedostatočnú prípravu na nižších stupňoch školského systému. Vieme rešpektovať len požiadavky zamestnávateľov. To je ako regulácia s neznámym obmedzením akčného zásahu. Pokúsime sa to zmeniť.

10 Záver

Navrhli sme zmeny s ohľadom na skvalitnenie výuky predmetu pre nepripravených študentov. Napriek problémom dúfame v úspešnú implementáciu. V tomto momente sa už tvoria prezentácie na predmet, ktorý sa začne vyučovať v septembri.

Literatúra

[1] Fodrek, P.: *Problems regarding teaching Linux programming for Industrial realtime programmers* In: Eleventh Realtime Linux Workshop+In:Linux Weekly News[online]. Dráždany+Boulder, Colorado : TU Dresden,OSADL+Eklektix, Inc. 2009, do-

stupné na http://lwn.net/images/conf/rtlws11/papers/proc/p12.pdf, citovane 10. 5. 2010

[2] Dehnadi, S. – Bornat, R.: *The camel has two humps (working title)*[online]. Londýn : Middlesex University,. 2006, dostupné na http://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf citovane 10. 5. 2010

[3] Klaus, J.: *Znalostní test – programátor C*[online]. Mountain View,California: Posterous(Y Combinator), 2010, dostupné na http://jaroslavklaus.posterous.com/znalostni-test-programator-c citovane 14. 5. 2010

[4] Vilhanová, D.: *Slovenská googláčka* In Emma [online]. Bratislava: Emma online (Plus 7, s. r. o.), 2008, dostupné na http://emma.pluska.sk/emma/clanky/kariera-a-peniaze/uspesne-zeny/slovenska-googlacka.html citovane 14. 5. 2010

Kontaktná adresa

Peter FODREK (Ing. , PhD.),
výskumný pracovník, Oddelenie Informačných a komunikačných technológií, Ústav riadenia a priemyselnej informatiky. FEI STU v Bratislave, Ilkovičova 3,
812 19 Bratislava, peter.fodrek@stuba.sk
Martin FOLTIN (Ing. , PhD.),
vedúci, Oddelenie Informačných a komunikačných technológií, Ústav riadenia a priemyselnej informatiky. FEI STU v Bratislave, Ilkovičova 3,
812 19 Bratislava, martin.foltin@stuba.sk
Michal Blaho (Ing.),
pedagogický pracovník, Oddelenie Informačných a komunikačných technológií, Ústav riadenia a priemyselnej informatiky. FEI STU v Bratislave, Ilkovičova 3,
812 19 Bratislava, michal.blaho@stuba.sk

OPEN SOURCE NÁSTROJE NA TVORBU PARALELNÝCH A DISTRIBUOVANÝCH APLIKÁCIÍ

GRONDŽÁK, Karol (SK)

Abstrakt. Riešenie mnohých komplexných úloh si vyžaduje veľké množstvo výpočtov. Pre ich urýchlenie boli navrhnuté mnohé riešenia, založené na paralelnom použití viacerých procesorov. V tomto článku popíšeme takéto architektúry a aj nástroje, ktoré je možné použiť na vytvorenie paralelnej, alebo distribuovanej aplikácie. Dôraz je kladený na nástroje, ktoré sú voľne dostupné.

1 Úvod

Vznik elektronických počítačov v päťdesiatych rokoch minulého storočia dal ľudstvu k dispozícii nástroj na automatické a rýchle vykonávanie výpočtov. Prvé počítačové systémy boli nesmierne nákladné a využívali sa pre riešenie vedecko–technických problémov či už vo vojenskom alebo civilnom sektore.

Významnýmn faktorom, ovplyvňujúcim výkonnosť počítača je počet inštrukcií, ktoré je jeho procesor schopný vykonať za jednotku času. Jednou z možností, ako zvýšiť počet inštrukcií vykonaných za jednotku času daného procesora je zvýšenie jeho taktovacej frekvencie. Takéto zvyšovanie je od istej úrovne technologicky veľmi náročné a existujú aj isté fyzikálne hranice, ktoré nie je možné prekročiť.

Iným spôsobom na zvýšenie výkonnosti procesora je zmena spôsobu organizácie spracovania inštrukcií (napr. pomocou pipeline a pod.). Na tomto princípe boli vybudované prvé *superpočítače* v šesťdesiatych rokoch minulého storočia [1, 2]. Ich nevýhodou bola vysoká cena.

S príchodom lacných, sériovo vyrábaných mikroprocesorov, sa ukázalo, že ich prepojením je možné zostaviť lacnejšie *superpočítače* s porovnateľným výkonom. Tak vznikli skupiny počítačov prepojených v rámci jednej organizácie (klastre), ktoré bolo možné využiť na vysokovýkonné počítanie.

S príchodom internetu vznikla obrovská sieť navzájom prepojených, geograficky oddelených, počítačov. To umožnilo rozšíriť myšlienku klastra na počítače, ktoré sú od seba geograficky vzdialené. Dobrovoľníci môžu poskytnúť nepoužívaný výpočtový výkon svojho počítača projektom, ktoré tento výkon môžu využiť na riešenie výpočtovo náročných úloh.

2 Klasifikácia paralelných architektúr

Základnou myšlienkou moderných paralelných a distribuovaných výpočtových systémov je paralelné spracovanie údajov. Existuje viacero spôsobov, ako to dosiahnuť. Jednotlivé spôsoby predstavujú rôzne paralelné a distribuované architektúry.

Jednou z hlavných charakteristík architektúry je mohutnosť toku inštrukcií. Ten môže byť buď jednoduchý, alebo násobný. Podobne aj tok spracúvaných údajov môže byť jednoduchý, alebo násobný. Kombináciou týchto dvoch charakteristík paralelných a distribuovaných systémov dostaneme štyri možné architektúry. Takúto klasifikáciu zaviedol Flynn [3].

Ak počítačový systém spracúva jeden tok inštrukcií a jeden tok údajov, označíme ho ako Single Instruction Single Data (SISD) architektúru. Táto architektura v podstate ani nie je paralelná, vyskytuje sa len kvôli konzistencii klasifikácie. Do tejto kategórie patria všetky jednoprocesorové počítačové systémy.

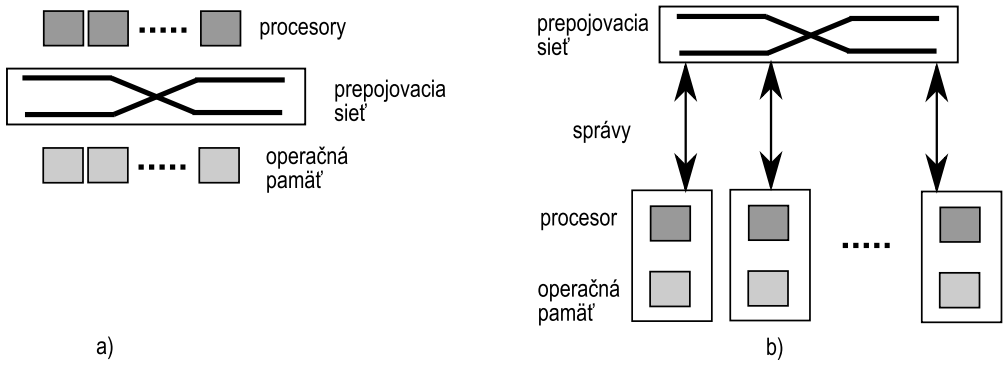
Logicky ďalšou architektúrou je Single Instruction Multiple Data architektúra. Ako už názov hovorí, ide o systémy, kedy sa v jednom čase vykonáva jedna inštrukcia, ale nad viacerými údajmi súčasne. Takýto spôsob spracovania sa označuje ako dátový paralelizmus. Je typický pre úlohy, kde sa spracúvajú veľké množstvá údajov tým istým algoritmom. Svoje uplatnenie našiel pri spracovaní obrazu a zvuku.

Moderné zobrazovacie adaptéry (GPU, graphics processing unit) a špecializované procesory na spracovanie zvuku (DSP, digital signal processor) patria do tejto kategórie. V súčasnosti výrobcovia zobrazovacích adaptérov poskytujú programové rozhranie pre sprístupnenie výpočtového výkonu GPU používateľom [4]. Do tejto kategórie možno zaradiť aj prvé superpočítače.

Kategória Multiple Instruction Single Data (MISD) nie je príliš rozšírená. Je to zrejme preto, že neexistuje relácie medzi touto architektúrou a bežne používanými programovými konštrukciami.

Poslednou architektúrou je Multiple Instruction Multiple Data (MIMD), ktorá je používaná v moderných superpočítačoch. Predstavuje skupinu procesorov, z ktorých každý vykonáva svoju činnosť na pridelených údajoch.

V mnohých prípadoch je potrebné, aby jednotlivé procesory počas výpočtu medzi sebou komunikovali. Spôsob komunikácie závisí od hardvérovej architektúry počítačového systému. V princípe najjednoduchšou je komunikácie pomocou zdieľanej pamäte. Ak procesory zdieľajú spoločnú operačnú pamäť, zapisovaním a čítaním z vopred dohodnutých adries je možné vymieňať údaje. Takéto architektúry označujeme ako systémy so zdieľanou pamäťou (obr. 1a).



Obrázok 1: Počítačový systém a) so zdieľanou pamäťou, b) s distribuovanou pamäťou

V prípade geograficky oddelených autonómnych procesorov neexistuje možnosť komunikácie pomocou zdieľanej pamäte. Každý procesor má svoju vlastnú operačnú pamäť. Na výmenu údajov musia procesory použiť komunikačnú sieť. Takéto architektúry predstavujú systémy s distribuovanou pamäťou (obr. 1b).

3 Voľne dostupné nástroje pre tvorbu paralelných a distribuovaných aplikácií

Ako bolo vyššie povedané, existujú rôzne architektúry paralelných a distribuovaných výpočtových systémov. Ich špecifiká treba brať do úvahy pri tvorbe aplikácií pre takéto systémy. Jednotlivé architektúry majú svoje špecifiká, ktoré ich predurčujú pre použitie pri riešení rôznych typov úloh.

3.1 Systémy so zdieľanou pamäťou

Ak viacero procesorov zdieľa spoločnú operačnú pamäť, hovoríme o systéme so zdieľanou pamäťou. Typickými predstaviteľmi tejto architektúry sú viacprocesorové systémy, ktoré zdieľajú operačnú pamäť a sú riadené jedinou inštanciou operačného systému (označované ako Symmetric MultiProcessing, SMP systémy). V súčasnosti môžeme ako systémy so zdieľanou pamäťou označiť aj viacjadrové procesory.

Paralelná aplikácia môže byť realizovaná viacnásobným súčasným vykonaním toho istého programu. Komunikácia medzi procesmi je však vo väčšine operačných systémov pomerne drahá operácia (z hľadiska počtu inštrukcií potrebných na jej realizáciu). Preto moderné operačné systémy poskytujú mechanizmus vlákien, ktorý umožňuje definovať a vykonávať rôzne úlohy v rámci vykonávania jedného procesu. Podpora vlákien môže byť implementovaná na úrovni samotného operačného systému, alebo na používateľskej úrovni pomocou knižníc.

Štandard POSIX.1c, Threads extensions (IEEE Std 1003.1c-1995) definuje aplikačné programové rozhranie pre tvorbu vlákien, manipuláciu s nimi a ich synchronizáciu. Implementácie tohoto štandardu možno nájsť vo forme knižnice pthreads vo väčšine operačných systémov založených na UNIXe.

Nevýhodou knižnice pthreads je nutnosť explicitnej paralelizácie kódu. Tvorca aplikácie musí určiť časti kódu, ktoré možno vykonať paralelne a zabezpečiť paralelné vykonanie týchto častí. Pri návrhu sa tvorca môže dopustiť chýb, ktoré môžu viesť až k úplnému znefunkčneniu aplikácie (napr. nesprávnym použitím synchronizačných príkazov, a pod.). V súčasnej dobe však neexistuje plne automatický kompilátor, ktorý by dokázal bez ľudského zásahu vytvoriť paralelnú aplikáciu na základe analýzy zdrojového kódu.

Dodávatelia systémov so zdieľanou pamäťou však v snahe podporiť svoje produkty vytvorili konzorcium, ktoré špecifikovalo aplikačné programové rozhranie OpenMP [5, 6]. Toto rozhranie definuje sadu príkazov pre prekladač a knižničné funkcie, pomocou ktorých je možné vytvárať prenositeľné paralelné aplikácie. Tvorca aplikácie označí pomocou špeciálnych príkazov pre prekladač časti kódu, ktoré majú byť vykonané paralelne. Prekladač vygeneruje automaticky kód, ktorý zabezpečí paralelné vykonanie označenej časti.

Voľne dostupným prekladačom, podporujúcim štandard OpenMP je vďaka projektu GOMP [7] aj prekladač GNU GCC od verzie 4.4. Vlastné implementácie štandardu OpenMP ponúkajú vo svojich produktoch firmy IBM (XL C/C++/Fortran), Oracle, Intel, Hewlett-Packard, Microsoft (Visual Studio 2008 C++) a Cray (Cray C/C++/Fortran) [6].

3.2 Systémy s distribuovanou pamäťou

Moderné superpočítače sú tvorené množstvom procesorov, z ktorých má každý svoju lokálnu operačnú pamäť. Pre prístup do operačnej pamäte iného procesora je potrebné vytvoriť špeciálny mechanizmus. Jednou z možností je komunikácia pomocou zasielania správ. Tento mechanizmus zvolilo konzorcium dodávateľov superpočítačov, univerzít a výskumných laboratórií, ktorí vytvorili špecifikáciu Message Passing Interface (MPI). Je to špecifikácia komunikácie medzi počítačovými systémami vytvorená s ohľadom na tvorbu distribuovaných aplikácií. Umožňuje zasielanie správ medzi dvomi, resp. viacerými procesormi, pričom poskytuje aj nástroje na synchronizáciu komunikácie. Medzi najrozšírenejšie voľne dostupné implementáce patria MPICH2 a OpenMPI.

MPICH2 je projekt vyvíjaný v Argonne National Laboratory v spolupráci s mnohými univerzitami a výskumnými centrami [8]. Vďaka tejto spolupráci je možné MPICH2 prevádzkovať na bežne dostupnom hardvéri, na špecializovaných zariadeniach pre komunikáciu (InfiniBand, Myrinet, 10Gb Ethernet, a pod.), ako aj na proprietárnych počítačových systémoch (Blue Gene, Cray, a pod.). Je šírený voľne aj so zdrojovými kódmi. Podporuje platformy Linux (IA32 a x86-64), Mac OS/X (PowerPC a Intel), Solaris (32 a 64 bitov) a Windows.

Ďalším projektom, dostupným pod BSD licenciou je OpenMPI [9]. Je vyvíjaný konzorciom, ktoré zahŕňa viaceré univerzity (TU Drážďany, TU Chemnitz, University of Houston,

University of British Columbia, University of Stuttgart, University of Tennessee, ...) ako aj komerčné firmy (CISCO, IBM, a ďalšie).

Inou cestou sa vybrali tvorcovia systému BOINC (Berkeley Open Infrastructure for Network Computing) [10]. Uvedomili si, že vďaka internetu existujú stovky tisíc navzájom prepojených počítačov, ktorých procesory nie sú vždy plne využité. Vytvorili preto platformu, ktorá umožňuje riadenie poskytovania výpočtovej kapacity procesorov. BOINC predstavuje platformu, na ktorej môžu záujemcovia o masívne paralelné výpočty vytvoriť svoju aplikáciu. Ďalej je potrebné získať dobrovoľníkov, ktorí sú ochotní poskytnúť výpočtový výkon svojich procesorov na takýto projekt.

Túto infraštruktúru využívajú mnohé známe a populárne projekty, akými sú:

- **SETI@home** – projekt hľadania mimozemského inteligentného života,
- **FreeHAL** – projekt štúdia umelej inteligencie,
- **NFS@home** – projekt riešenia rozkladu veľkých prvočísiel,
- **NQueens@home** – hľadanie riešenia problému N kráľovien na šachovnici, pre $N > 19$,

a mnoho iných [10].

4 Záver

Jednou z ciest, ako urýchliť výpočtovo náročné úlohy je ich paralelizácia. V závislosti od charakteru riešeného problému je potrebné zvoliť vhodnú architektúru a nástroj na vytvorenie paralelnej aplikácie.

V tomto príspevku boli stručne predstavené jednotlivé paralelné a distribuované architektúry, ako i softvérové prostriedky pre tvorbu aplikácií pre tieto systémy. Dôraz bol kladený na nástroje, ktoré sú voľne dostupné.

Pre počítačové systémy so zdieľanou pamäťou existuje špecifikácia OpenMP, ktorá definuje inštrukcie pre prekladač na generovanie paralelného kódu. Príkladom voľne dostupného kompilátora kompatibilného so špecifikáciou OpenMP je projekt GNU GCC.

Pre systémy s distribuovanou pamäťou je de facto štandardom špecifikácia Open Message Interface. Aj tu existuje niekoľko implementácií s otvoreným zdrojovým kódom, napr. MPICH2 vyvíjaný v Argonne National Laboratory a OpenMPI, vyvíjaný konzorciom univerzít, výskumných ústavov a firiem.

Literatúra

[1] WILKINSON, B – Allen, M.: *Parallel Programming*. Upper Saddle River : Pearson Prentice Hall, 2005, ISBN 0-13-140563-2

[2] BUYYAM R. (editor):*High Performance Cluster Computing: Programming and Applications, Vol. 2*. Upper Saddle River : Prentice Hall PTR, 1999, ISBN 0-13-013785-5

[3] FLYNN, M. J.: *Very High Speed Computing Systems*, Proc. IEEE, Vol. 12, 1966, s. 1901–1909

[4] KIRK, D. B. – HWU W. W.: *Programming Massively Parallel Processors*. Burlington : Elsevier, 2010, ISBN 978-0-12-381472-2

[5] QUINN, M. J.: *Parallel Programming in C with MPI and OpenMP*. Boston : Mc Graw Hill, 2003. ISBN 007-123265-6.

[6] OpenMP [online]. Dostupné na internete: `http://openmp.org/wp/`

[7] Projekt GOMP [online]. Dostupné na internete: `http://gcc.gnu.org/projects/gomp/`

[8] Projekt MPICH2 [online]. Dostupné na internete: `http://www.mcs.anl.gov/research/projects/mpich2/`

[9] Projekt OpenMPI [online]. Dostupné na internete: `http://www.open-mpi.org/`

[10] Projekt BOINC [online]. Dostupné na internete: `http://boinc.berkeley.edu/`

Kontaktná adresa

Karol GRONDŽÁK (Ing., PhD.),
Katedra informatiky, FRI ŽU v Žiline, Univerzitná 1
010 26 Žilina, `Karol.Grondzak@fri.uniza.sk`

ARCHITECTURE DEVELOPMENT OF WEB BASED APPLICATION FOR A CONSTRUCTION COST PREDICTION

KANIK, Tomasz, (PL)

Abstract. This paper describes the idea and first steps of development of intended commercial application for construction cost prediction. Study of different architecture models and development techniques is described at the very beginning. Then pros and cons of OSS and commercial tools are analysed. Based on that the future shape of project is envisioned and described up to current state. Server-based environment solutions are described too.

1 Overview of main requirements for an application

It should provide construction cost pre-calculation for non-professional builder. During building process real cost calculation is performed by systematically filling real price cost according to created model. It is based on free web technologies. Interface is user friendly. A hierarchical structure of aggregate items of construction allow different, independent price range calculation. Availability of tender creation for multiple group of construction items and their distribution by different channels – for example by email. Summarized report based on available data.

2 Establishing of the overall structure of an application

Architectural design represents an early stage of the system design process. It represents the link between specification and design processes, often carried out in parallel with some specification activities, which involve identifying of major system components and their communications. Right architecture design prevents to disestablish the system, or to reduce it efficiency by dynamic specification changes [1].

Design process has few, practical rules. First is to localize operations to minimise sub-system communication and increase system performance. Second is to use a layered

architecture with critical assets in inner layers. Third, critical component isolation. The last, fourth, is to include redundant and self-contained components in the architecture for better maintainability [2].

The architectural model of a system may conform to a generic architectural model or style. An awareness of these styles can simplify the problem of defining system architectures. However, most of large systems are heterogeneous and do not follow a single architectural style.

Analysis of different model characteristic show that three-tier model is the best fit for our requirements. Three-tier is a client–server architecture in which the user interface, functional process logic ("business rules"), computer data storage and data access are developed and maintained as independent modules, most often on separate platforms. Apart from the usual advantages of modular software with well-defined interfaces, the three-tier architecture is intended to allow any of the three tiers to be upgraded or replaced independently as requirements or technology change. For example, a change of operating system in the presentation tier would only affect the user interface code. Idea of the three-tier model is shown in Fig.1.

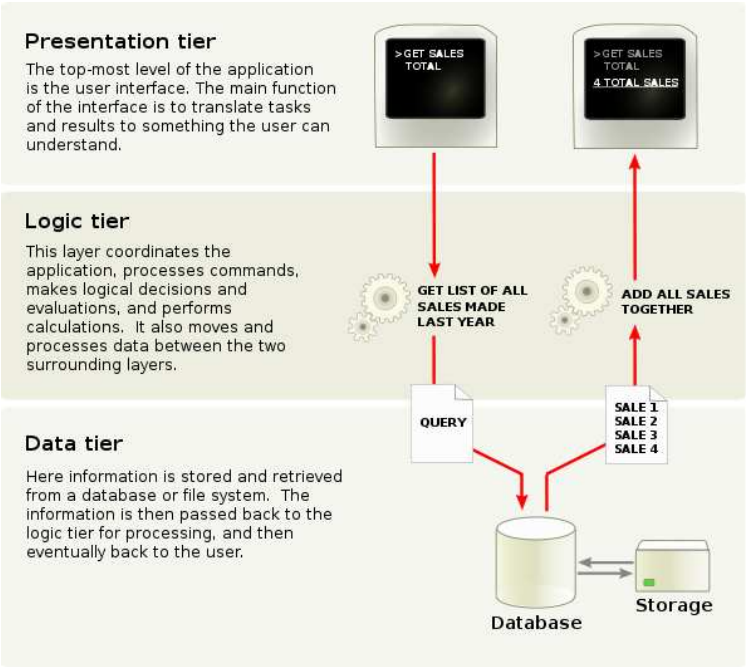


Figure 1: Visual overview of a Three-tiered application [2]

3 Specific application development methodology

The methodology choice we made is fundamentally for future model development by succeeded team. That is why we performed robust sensitivity analysis, which show that the right choice is Incremental Model. It is most appropriate, because requirements for future development are not well understood and expectations rapidly changing.

The Incremental model is the combination of linear and iterative methodologies with the primary objective of each being to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process. Our idea is that the initial software concept, requirements analysis, design of architecture and system core are defined using the waterfall approach, followed by iterative prototyping, which culminates in implementation of the final prototype (i.e., working system) [3].

Strengths of this solution: Potential exists for exploiting knowledge gained in an early increment as later increments are developed. Moderate control is maintained over the life of the project through the use of written documentation and the formal review and approval/sign-off by the user and information technology management at designated major milestones. Stakeholders can be given concrete evidence of project status throughout the life cycle. Helps to mitigate integration and architectural risks earlier in the project. Allows delivery of a series of implementations that are gradually more complete and can go into production more quickly as incremental releases. Gradual implementation provides the ability to monitor the effect of incremental changes, isolate issues and make adjustments before the organization is negatively impacted [4].

Weaknesses found : When utilizing a series of mini-Waterfalls for a small part of the system before moving on to the next increment, there is usually a lack of overall consideration of the business problem and technical requirements for the overall system. Since some modules will be completed much earlier than others, well-defined interfaces are required. Difficult problems tend to be pushed to the future to demonstrate early success to management [4].

4 Software modelling tool for creating the application architecture

It sounds funny or silly for someone who never created a large project, that it is not so easy to find good tool with support of UML2, MDA or BPMN models. Open source solution mostly don't cover any of above standards at all. Simple applications created by different teams, groups or people are good for learning and can cover requirements of semester project, but it is all. That is why, after long search and many tries our project architecture was created using commercial solution of IBM company - Rational Software Modeler. Experience we gather is expressed bellow, in description of each software we use.

Dia is a free and open source general-purpose diagramming software. It uses a controlled

single document interface similar to GIMP. Dia has a modular design with several shape packages available for different needs: flowchart, UML, network diagrams, circuit diagrams, and more. It does not restrict symbols and connectors from various categories from being placed together [5].

It is really smart and useful tool when you need to draw overhead diagram, but drawing large amount of UML diagrams with it is road across hell. There is no project which glues different diagrams together and no creator or hot key for frequently used operations is available.

Umbrello UML Modeller is a free software UML diagram application available natively for Unix. It handles all the standard UML diagram types. It can reverse engineer code written in C++, IDL, Pascal/Delphi, Ada, Python, and Java, as well as import XMI files generated by external tools and export to various programming languages. It allows simple collaborative development by exporting to DocBook and XHTML formats [6].

An idea of creating code from diagrams or back to them sounds good but reality shows it is still under development, moreover in non-native environment (non-KDE) the application freezes few times and after process was killed, the project had broken structure. Some problems with grouping of objects and changes of interaction between diagram occurred.

StarUML is an open source UML tool. The stated goal of the project was to replace larger, commercial applications such as Rational Rose and Borland's Together. Now it is no longer under development. StarUML was written in Delphi, which is one of the reasons why it is no longer maintained [7].

This tool nearly meets our requirements, except it is currently missing object, package, timing and interaction overview diagrams. It is huge mistake that project was canceled and no successor seems to appear. We were positively surprised that manipulation with diagrams was really good and number of properties allow to specify exactly what we want. Maybe some changes in IDE can be done, which increase ergonomic model space use.

Enterprise Architect, (EA), commercial software produced by Sparx Systems, is a collaborative modelling, design and management platform based on UML 2.1 and related standards (i.e. BPMN, XMI). Agile, intuitive and extensible with fully integrated, powerful domain specific high-end features at a fraction of the cost of many competitors. An enterprise wide solution for visualizing, analysing, modelling, testing and maintaining a wide range of systems, software, processes and architectures [8].

Definitely it is not intuitive software, despite of the claims on EA homepage. It is a mess of creators and views. It takes a while to prepare environment to do exactly what we want. Maybe it is good that so many futures are provided but it makes this software less practical, no one looks from the viewpoint of designer who has no time to try gadgets but have a lot of work to do. Object manipulation and diagrams interaction are not so good as it should be.

IBM Rational Software Modeller (RSM) made by IBM's Rational Software division, is a UML 2.0-based visual modelling and design tool. Rational Software Modeller is built on the Eclipse open-source software framework and is used for visual modelling and model-

driven development (MDD) with UML for creating applications and web services. RSM is engineered as a plugin that sits on top of the open-source Eclipse development platform [9].

There are not many application I can say are nearly ideal, but RSM and their advanced version RSA (IBM Rational Software Architect) looks like dream of all software architects. It is multi-platform software, has integrated most of commonly used modelling standards, has ergonomic modelling space, has good integration between different diagrams, allows describe model in amount of properties exactly as it is needed. Using this software is a pleasure, our project is well documented and every time we need to check or specify some details we can easily find it.

5 Choosing programming language

Language choice is a topic filled with opinions. Most programmers have a favourite language, and most programmers have languages they hate. Programming in a given language engenders similar responses to doing plumbing repairs with a given box of tools. In general the goal of computing should be to produce the most powerful and reliable tools possible with the least amount of effort and resources consumed. That is why our choice of language is made based on overall goals and requires various language to be reviewed. Bellow, popular languages we consider to use are reviewed in few words.

Java is a multi-platform language that is especially useful in networking. Of course, the most famous usage of Java is on the web, with Java applets, but Java is also used to build stand-alone cross-platform programs. Since it resembles C++ in syntax and structure, learning Java is usually quite easy for most C++ programmers. Java offers the advantages provided by object-oriented programming, such as reusability; on the other hand, it can be difficult to write highly efficient code in Java, and Swing, its primary user interface, is notoriously slow. Nevertheless, Java has increased in speed in recent years, and version 1.5 offers some new features for making programming easier [10].

PHP is a common language for website design that is sometimes used as a scripting language in *nix. PHP is designed for rapid website development, and as a result it contains features that make it easy to link to databases, generate HTTP headers, and so forth. As a scripting language, it contains a relatively simple set of basic components that allow the programmer to quickly get up to speed, though it does not have more sophisticated object-oriented features [10].

C♯ is a multi-paradigm programming language encompassing imperative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It is one of the programming languages designed for the Common Language Infrastructure. By design, C♯ is the programming language that most directly reflects the underlying Common Language Infrastructure (CLI). Most of its intrinsic types correspond to value-types implemented by the CLI framework. However, the language specification does not state the code generation requirements of the compiler: that is, it does not state that a C♯ compiler must target a Common Language Runtime, or generate Common Intermediate Language (CIL),

or generate any other specific format. Theoretically, a C♯ compiler could generate machine code like traditional compilers of C++ or FORTRAN [11].

Ruby is a dynamic, reflective, general purpose object-oriented programming language that combines syntax inspired by Perl with Smalltalk-like features. It was influenced primarily by Perl, Smalltalk, Eiffel, and Lisp. Ruby supports multiple programming paradigms, including functional, object oriented, imperative and reflective. It also has a dynamic type system and automatic memory management; it is therefore similar in varying respects to Python, Perl, Lisp, Dylan, Pike, and CLU [12].

Python is a general-purpose high-level programming language whose design philosophy emphasizes code readability. Python aims to combine "remarkable power with very clear syntax", and its standard library is large and comprehensive. Its use of indentation for block delimiters is unusual among popular programming languages. Python supports multiple programming paradigms, primarily but not limited to object oriented, imperative and, to a lesser extent, functional programming styles. It features a fully dynamic type system and automatic memory management, similar to that of Scheme, Ruby, Perl, and Tcl. Like other dynamic languages, Python is often used as a scripting language, but is also used in a wide range of non-scripting contexts [13].

Due to the use of multi-tier model with not only web pages but back-end system too, we decide to use Java language as the one, which meet our requirements. It is important for us to choose language, which is not on margin or end-route of their life cycle – future perspective is of the same importance as today ability the language has. For now, Apache Wicket is used as web framework and support of Hibernate Technologies for base tier is planning. Implementation of other/new framework or technologies will depend on future requests or needs.

6 Choosing database engine

Project specification says that application have to be based on free technologies, it means that no commercial database engine can be used. That is, why there is no commercial engine description. Language chosen for project development (Java) allows to use any kind of currently known database, so we are not obliged to make choice based on language restrictions. Some of database engines, we consider to use, are described bellow.

HSQLDB (Hyper Structured Query Language Database) is a relational database management system written in Java. It has a JDBC driver and supports a large subset of SQL-92, SQL-99, and SQL:2003 standards. It offers a fast, small (around 600 kilobytes in the standard version) database engine which offers both in-memory and disk-based tables. Embedded and server modes are available [14].

MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. It is currently the most popular open source database server in existence. On top of that, it is very commonly used in conjunction with PHP scripts to create powerful and dynamic server-side applications. MySQL has

been criticized in the past for not supporting all the features of other popular and more expensive DataBase Management Systems. However, MySQL continues to improve with each release, and it has become widely popular with individuals and businesses of many different sizes [15].

Firebird is a relational database offering many ANSI SQL standard features that runs on Linux, Windows, and a variety of Unix platforms. Firebird offers excellent concurrency, high performance, and powerful language support for stored procedures and triggers. The Firebird Project is a commercially independent project of C and C++ programmers, technical advisers and supporters developing and enhancing a multi-platform relational database management system based on the source code released by Inprise Corp (now known as Borland Software Corp) [16].

PostgreSQL is a powerful, open source object-relational database system. It has architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It runs on all major operating systems, including Linux, UNIX, and Windows. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored procedures (in multiple languages). It includes most of SQL:2008 data types. It also supports storage of binary large objects, including pictures, sounds, or video. It has native programming interfaces for C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, among others, and exceptional documentation [17].

A database engine we are looking for will have long term influence on future of the project, even if database independent technology, like Hibernate, is used. HSQL is the database for development, tests, single client solution and generally small projects. Last few years Firebird is used in large governments project, mainly for it is performance, but it have weaker support from community. MySQL, well known database for PHP and other applications installed on LAMP servers, has good support and years of development. Even considering that all, we feel that it is still not what we looking for. There is lot of non standard functions, not so good performance and scalability and not enough security too. At the end we realize it, our choice should be PostgreSQL. It is database engine for large and critical projects in government and commercial solutions, many times compared to Oracle or DB2. PostgreSQL has very good performance with large amount of data, has mechanisms to maintain the database and is prepared for long up-time work.

7 J2EE application server

GlassFish is an open source application server project led by Sun Microsystems (Oracle now) for the Java EE platform. The proprietary version is called Sun GlassFish Enterprise Server. GlassFish is free software, dual-licensed under two free software licences: the Common Development and Distribution License (CDDL) and the GNU General Public License (GPL) with the classpath exception. GlassFish is based on source code released by Sun and Oracle Corporation's TopLink persistence system. It uses a derivative of Apache Tomcat as the

servlet container for serving Web content, with an added component called Grizzly which uses Java NIO for scalability and speed [18].

JBoss Application Server (or JBoss AS) is free software/open-source Java EE-based application server. Because it is Java-based, the JBoss application server operates cross-platform: it is usable on any operating system that supports Java. JBoss AS was developed by JBoss, now a division of Red Hat [19].

Apache Tomcat is an open source servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run. Apache Tomcat powers numerous large-scale, mission-critical web applications across a diverse range of industries and organizations [20].

No special requirements are formed for application server for now, because it can be changed without influence on most of the project. We would like to create one EAR file with our project and make it available on all application servers, which support this file format. For development environment Apache TomCat is chosen for it's simple configuration, management and good support from their community.

8 Development IDE

There is no specification or recommendation for development environment. Only two most popular IDE are presented, because the use of other environments is imperceptible. Someone can say that Visual Studio is recently used to develop Java programs, and it is true, but not for web-based development. That small difference cause that only two, well known IDE's, are used in real development. It is thanks to provided support and functionality, which help programmers to make their job and not to think about technical dependencies.

NetBeans refers to both a platform framework for Java desktop applications, and an integrated development environment (IDE) for developing with Java, JavaScript, PHP, Python, Ruby, Groovy, C, C++, Scala, Clojure, and others (for a complete overview, visit the netbeans website). The NetBeans IDE is written in Java and runs everywhere where a JVM is installed, including Windows, Mac OS, Linux, and Solaris. A JDK is required for Java development functionality, but it is not required for development in other programming languages. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans platform (including the NetBeans IDE) can be extended by third party developers [21].

Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It is written primarily in Java and can be used to develop applications in Java and, by means of various plug-ins, other languages including C, C++, COBOL, Python, Perl and PHP. The IDE is often called Eclipse ADT for Ada, Eclipse CDT for C/C++, Eclipse JDT for Java and Eclipse PDT for PHP. The initial codebase originated from VisualAge. In its default form it is meant for Java developers, consisting of the Java Development Tools (JDT). Users can extend

its capabilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Released under the terms of the Eclipse Public License, Eclipse is free and open source software [22].

Our team chosen Eclipse as development environment used for this project. This decision is among based on their support for advanced SVN plugins, Ant building script, remote library import and good integration with development application server selected before. Moreover Rational Software Modeler is based on Eclipse, thanks to this our team has good, past experience with that environment.

9 Current state of project and development

After all preparation we created well documented architecture model. Based on that model, development work is started and first prototype is being prepared. Simulation of production environment is created based on Debian system and TomCat application server. There is remote access enabled to this machine for any team member – it is for test, how developed programs behave in real simulated environment. Source code is stored in SVN repository, team members IDE's are prepared to remotely communicates with it. This works as back-up and helps to keep source updated during different modules development process. Simple database model is kept on the server, to help the members to provide tests on their local machines too, thanks to VPN tunnelling.

10 Conclusions

All steps described before show how to make reliable and well-considered decision on large scale project. Few of this step seems to be not important at the beginning but in the future can cause large changes, even cause the project rebuild. Technologies in use today may change tomorrow, so we have to provide mechanisms to implement any of them, even if we do not have their specification now. This article is not the receipt for a good project, it just puts lights on tools and technologies available today but which may not be well recognised.

Literatúra

[1] SOMMERVILLE I.: *lectures of Software Engineering*, University of Southern California, 2008

[2] Wikipedia, `http://en.wikipedia.org/wiki/Multitier_architecture`, 2010-06-06

[3] Wikipedia, `http://en.wikipedia.org/wiki/Software_development_methodologies`, 2010-06-06

[4] Centers for medicare and medicaid services: *Selecting a development approach*, Washington, Rev. 2008-03-27

[5] Wikipedia, http://en.wikipedia.org/wiki/Dia_(software), 2010-06-06

[6] Wikipedia, http://en.wikipedia.org/wiki/Umbrello_UML_Modeller, 2010-06-06]

[7] Wikipedia, http://en.wikipedia.org/wiki/StarUML, 2010-06-06]

[8] Sparx Systems Pty Ltd, http://www.sparxsystems.com/products/ea/features.html, 2010-06-06

[9] Wikipedia, http://en.wikipedia.org/wiki/IBM_Rational_Software_Modeler, 2010-06-06

[10] SUH E., *The Tower of Babel – An Exploration of Programming Languages*, http://www.cprogramming.com/langs.html, 2010-06-07

[11] Wikipedia, http://en.wikipedia.org/wiki/C_Sharp_(programming_language), 2010-06-07

[12] Wikipedia, http://en.wikipedia.org/wiki/Ruby_(programming_language), 2010-06-07

[13] Wikipedia, http://en.wikipedia.org/wiki/Python_(programming_language), 2010-06-07

[14] Wikipedia, http://en.wikipedia.org/wiki/HSQLDB, 2010-06-08

[15] Erack Network, http://www.tizag.com/mysqlTutorial/, 2010-06-08

[16] Firebird Foundation Incorporated, http://www.firebirdsql.org, 2010-06-08

[17] PostgreSQL Global Development Group, http://www.postgresql.org, 2010-06-08

[18] Oracle Corporation, http://www.glassfish.org, 2010-06-08

[19] Red Hat, Inc., http://www.jboss.com, 2010-06-08

[20] The Apache Software Foundation, http://tomcat.apache.org, 2010-06-08

[21] Oracle Corporation, http://netbeans.org, 2010-06-08

[22] The Eclipse Foundation, http://www.eclipse.org/resources/, 2010-06-08

Contact

Tomasz KANIK (Bc),
Fakulta riadenia a informatiky,
Univerzitná 8215/1, 010 26 Žilina,
tkanik@o2.pl

3D V TECHNOLÓGII FLASH POMOCOU OPEN SOURCE KNIŽNÍC

KOLESÁR, Ivan, (SK)

Abstrakt. Príspevok sa venuje jednej z možností ako na internete vizualizovať 3D obsah. A tou je vizualizácia pomocou programovacieho jazyka Actionscript 3.0, ktorý je súčasťou Flashu, a pomocou ktorého sa dajú voľne vytvárať aplikácie na tejto technológii. Prvé sú v príspevku načrtnuté možnosti, ktoré ponúka jazyk Actionscript 3.0 pre vytváranie aplikácií, ktorými sú napríklad spracovanie a vykresľovanie obrázkov, prehrávanie videa a zvuku, načítavanie xml súboru a mnoho ďalšieho. Neskôr sú rozobrané možnosti zobrazovania 3D pomocou open source knižníc Away3D, Papervision3D a Sandy3D. Tie sú následne porovnané z hľadiska využívania techník počítačovej grafiky, z hľadiska rýchlosti vykresľovania a je aj porovnaný proces spracovania scény a jej vykreslenia do 2D obrázka. Všetky tieto porovnania sú vyhodnotené a sú z nich vyvodené závery pre jednotlivé knižnice. Nakoniec je v príspevku rozobraná budúcnosť vizualizácie 3D obsahu na technológii Flash pomocou jazyka haXe, ktorý dokáže výsledný kód skompilovať do Flash aplikácie alebo do zdrojových kódov iných platforiem ako Java, C++ a Javascript.

1 Úvod

Adobe Flash je multimediálna platforma, využívaná pre prehrávanie videa, animácií a pridávanie interakcie na webovské stránky. Je to taktiež jedna z technológií, ktorá na internete sprostredkúva webaplikácie. Samotná technológia existuje už od roku 1996 a odvtedy sa neustále naďalej vyvíja a vychádzajú jej nové verzie. Najaktuálnejšia je verzia 10.1, ktorej vývoj sa uberá aj smerom na operačný systém Android 2.2 a hardvérovú akceleráciu pre niektoré operácie ako dekódovanie H.264 videa a hardvérovú akceleráciu na grafických kartách pre vykresľovanie. Táto technológia sa skladá z animácií, prezentovaných ako jednotlivé vrstvy obrázkov, zvládaní prúdového preposielania videa a zvuku a skriptovacieho jazyka Actionscript 3.0. Hlavnými výhodami tejto technológie sú jej multiplatformovosť, keďže existujú Flash player prehrávače pre viacero najznámejších operačných systémov, a veľká rozšíriteľnosť medzi používateľmi internetu, kde podľa štúdií od spoločnosti Adobe až 99% používateľov má nainštalovaný a povolený prehrávač. Jej rozšírenie je pozorovateľné aj pri obľúbených webových službách ako YouTube alebo Facebook.

2 Actionscript 3.0

Actionscript 3.0 je voľne dostupný skriptovací jazyk, ktorý vlastní spoločnosť Adobe. Pôvodne bol vyvinutý pre tvorbu 2D vektorovej animácie vo Flashi. Jeho prvá verzia sa objavila v roku 2000 a bola prístupná v prehliadači Flash player 4, kde sa postupne upravovala syntax pre objektovo orientované programovanie, pridávala sa správa prúdoveho spracovania a nakoniec v najnovšej tretej verzii i limitovaný prístup k hardvérovej akcelerácii.

Syntax jazyka je odvodená od syntaxe ECMAScript, podľa štandardu ECMA262 piatej edície. Je teda syntakticky podobný aj s ďalšími populárnymi skriptovacími jazykmi ako Javascript a JScript. Existuje preň mnoho vývojových prostredí, medzi najznámejšie patrí napríklad Adobe Flash od spoločnosti Adobe alebo open source projekty swftools, FlashDevelop a mnoho iných. Práve pomocou tohto jazyka je možné v technológii Flash voľne vytvárať bohaté webaplikácie obsahujúce animácie, video, audio a bohatú interaktivitu.

3 Flash 3D

Pod pojmom 3D vo Flashi máme konkrétne na mysli schopnosť vykresľovať trojrozmerné scény pomocou rôznych vykresľovacích techník v technológii Flash. Konkrétne Flash 10 už má v sebe podporu pre vykresľovanie scény, avšak zatiaľ je to stále bez pokročilejších vykresľovacích techník akými sú napríklad shadere, textúry. Tie sa dajú nájsť hlavne v externých pomocných knižniciach slúžiacich pre vykresľovanie 3D scény. Takýchto knižníc sa dá na internete nájsť mnoho, v tomto článku sú bližšie porovnávané tie, ktoré sú open source, majú bohatú paletu techník pre zobrazovanie, veľkú komunitu používateľov, mnoho návodov a príkladov. Sú to knižnice Away3D, Papervision3D a Sandy3D. Každá z nich je zameraná pre odlišný typ aplikácie ktorý má vykresliť a tak majú rôzne vlastnosti napríklad pri kvalite vykresľovania pri mnohých objektoch, rýchlosti vykresľovania, počtu dostupných techník a podobne. Preto boli tieto knižnice podrobne preskúmané, porovnané a bol pre každú vypracovaný samostatný záver. Preskúmavanie knižníc sa dialo kvalitatívne, kde pri každej knižnici sa bralo do úvahy, ktoré techniky podporuje a ktoré nie, taktiež sa pri každej knižnici načrtol ich renderovací kanál. Ďalej sa porovnávalo kvantitatívne, kde sa na daných knižniciach pomocou externých aplikácií a testovania zistilo ich správanie pri niektorých úlohách počítačovej grafiky a ich celková rýchlosť vykresľovania scény.

3.1 Spoločné znaky knižníc

Ako prvé, krátko spomenieme, čo majú dané knižnice spoločné. Každá z týchto knižníc v sebe obsahuje základné vykresľovanie priestorovej scény, reprezentovanej množinou trojuholníkov alebo sústavou geometrických primitívov, dokážu vykresliť flat shader, phong shader, enviroment shader a cell shader. Dokážu prehrávať a spracovávať animácie objektov v scéne, vlastné tiene objektov a podporujú aj vykresľovanie textúr na objektoch.

3.2 Away3D

Je najmladšia z porovnávaných knižníc. Primárne je zameraná pre sprístupnenie čo najväčšieho počtu pokročilých grafických techník. Oproti ostatným dvom knižniciam ponúka naviac nahrávanie scény z mnohých formátov, pričom tými rozšírenejšími sú formáty Wavefront, Collada a MD2. Má v sebe priamo nástroje aj pre exportovanie zobrazovanej scény do formátov Wavefront a vlastných dátových formátov AWD alebo AS3. Nastavovanie zobrazovania scény prezentuje pomocou kamerového systému, kde hlavným plusom tejto knižnice je množstvo typov kamier, ktoré si môže vývojár vybrať, pričom každá sa dá vnútorne nastavovať podľa parametrov a to dokonca aj na úrovni výberu šošoviek, napríklad ortogonálnej, perspektívnej alebo sférickej.

Animácie objektov knižnica spracováva na dvoch úrovniach, na globalnej transformácii celého objektu alebo na animáciach na lokálnych úrovniach v rámci objektu. Tie dokáže ovládať buď pomocou pomocnej kostrovej štruktúry modelu alebo prehrávať ako postupnosti množín trojuholníkov. Knižnica obsahuje sadu tried reprezentujúcich jednotlivé typy svetiel, bodové, smerové, ambientné, a pre simuláciu jednoduchého projekčného tieňa. Ďalej ponúka triedy nástrojov procedurálneho spracovania scény, vyťahovanie, zrkadlo, kopírovanie a pod. Pre realistickejšie zobrazovanie scény ponúka aj nástroje pre zobrazovanie odrazu šošoviek a HDR textúr (High Dynamic Range, jedná sa o obrázky s vysokým jasovým rozsahom). V poslednom rozšírení knižnice bola pridaná možnosť dynamického prevodu scény do BSP (Binary Space Partition, rozdelenie scény do stromu polpriestorov) a vďaka tomu aj rýchlejšiemu vykresleniu veľkých scén.

3.3 Papervision3D

Knižnica, ktorá má najväčšiu komunitu spomedzi porovnávaných. Má dostatočnú paletu nástrojov a techník a poväčšine ak je žiadaná nejaká technika, ktorá nie je v oficiálnom vydaní, je možné ju nájsť na internete vytvorenú od nadšenca z komunity. Má podporu pre väčšinu známych modelovacích aplikácií, napríklad Blender, a ponúka rozšírené nástroje pre nahrávanie, spracovanie a dokonca aj exportovanie scény vo formáte Collada. Jej nastavenie zobrazovacieho kanála spočíva vo výbere typu kamery a v nastavení parametrov kamery.

Na rozdiel od Away3D neponúka výber šošovky, knižnica poskytuje iba perspektívny pohľad a má menej typov kamier. Pre vývojárov však knižnica ponúka špeciálny typ kamery, ktorý priamo v okne stále zobrazuje jej základné parametre. Dokáže taktiež prehrávať animácie uložené vo formáte Collada, neponúka však ich procedurálne ovládanie. Z pokročilejších unikátnych techník počítačovej grafiky je v nej používanie displacement mapovania. Avšak podobne ako Away3D ponúka triedy pre zobrazovanie 3D textu s vlastným fontom. Taktiež pomocou komunity bolo do nej pridané zobrazovanie odrazu šošoviek a projekcie jednoduchého tieňovania.

3.4 Sandy3D

Najstaršia z porovnávaných knižníc, v súčasnosti sa jej vývoj presúva na skriptovací jazyk haXe, avšak ešte stále sa vyvíja aj pre jazyk AS3. Na rozdiel od Away3D a Papervision3D neposkytuje knižnica načítavanie scény z formátu Collada, ani širší výber typu kamery a jej nastavovania. Animácie podporuje iba z formátu MD3 a to formou prehrávania postupnosti množín trojuholníkov. Jej hlavným prínosom je jej silné prepojenie s fyzikálnym modelom knižníc simulujúcich fyzikálne javy WOW a jiglibflash.

Tabuľka prehľadu technológií jednotlivých knižníc

	Away3D	Papervision3D	Sandy3D
Kamerový systém	✓	✓	✓
Geometrické primitíva	✓	✓	✓
3D text	✓	✓	X
Materiály a shadere	✓	✓	✓
Svetlá a tiene	✓	✓	✓
Animácie	✓	✓	✓
Spracovanie udalostí	✓	✓	✓
Podpora 3d aplikácií	3DsMax, Blender	3DsMax, Blender	3DsMax, Blender
Podpora 3d formátov	DAE, KMZ, OBJ, MD2, ASE, MAX	KMZ, MAX, ASE, DAE, MD2	MAX, ASE, DAE, MD2

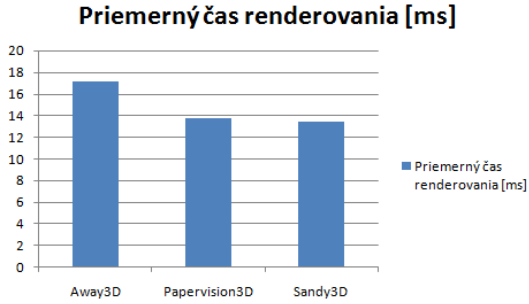
Obrázok 1: Tabuľka prehľadu knižníc, ich podpory jednotlivých súčastí 3D počítačovej grafiky

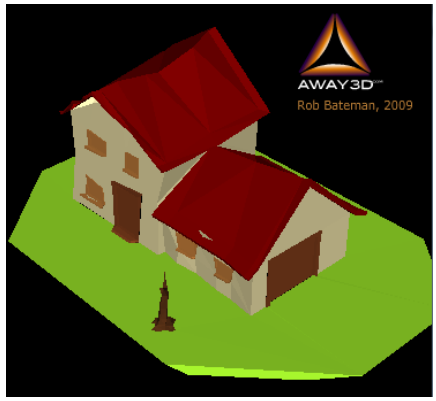
4 Porovnávacie aplikácie

Pre každú knižnicu bola navrhnutá sada testovacích aplikácií. Tie mali porovnať ako dobre si jednotlivé knižnice poradia s niektorými úlohami počítačovej grafiky a ako rýchlo dokážu vykresľovať 3D scénu. Pre všetky tieto aplikácie je použitý rovnaký model scény pri porovnávaní. Bolo nazbieraných 50 meraní z webovej stránky `http://www.divan.tym.cz/flash` a z nich boli vypracované závery.

4.1 Základne zobrazenie scény

Prvý typ aplikácie zobrazuje základnú geometriu scény s jednoduchými farebnými materiálmi.



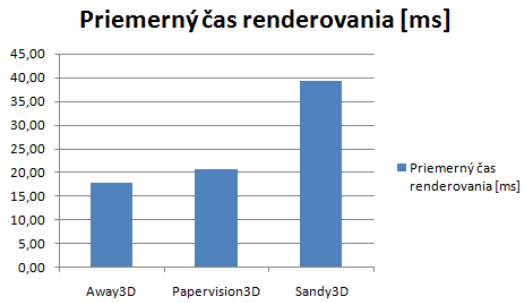


Obrázok 2: Prvý typ aplikácie pre testovanie

Podľa grafu na obrázku 2 je vidieť, že tento typ úlohy dokážu mierne rýchlejšie vykresliť knižnice Papervision3D a Sandy3D. Sandy3D má však problémy korektne preusporiadávať trojuholníky scény a vznikajú v nej artefakty.

4.2 Zobrazenie scény s textúrami

Geometricky sa zobrazuje rovnaká scéna, avšak tá už nie je rozdelená do viacerých objektov, ale spojená do jedného a na tomto celom objekte je použitá jedna textúra.

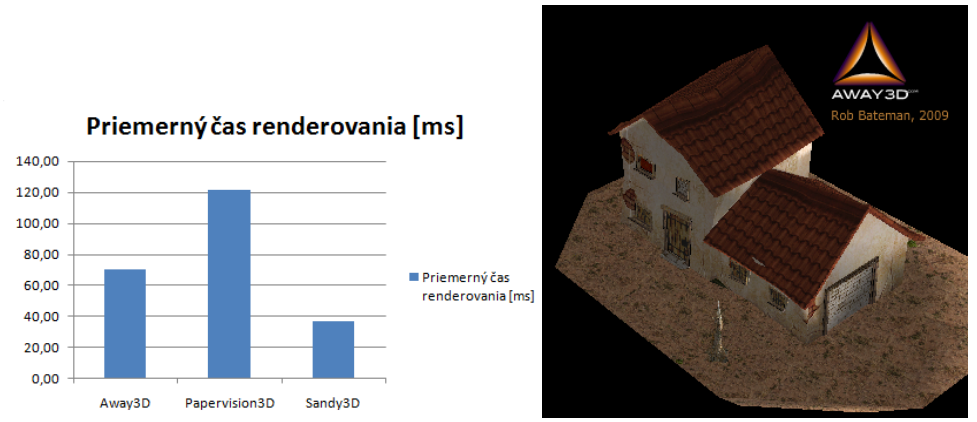


Obrázok 3: Druhý typ aplikácie pre testovanie

Pri tomto meraní je badať, že Away3D si lepšie poradí so scénou ak je uložená do jedného objektu ako do viacerých a taktiež, že knižnica Sandy3D má menšie problémy zobraziť otextúrovaný objekt.

4.3 Zobrazenie scény s phong shaderom

Opäť geometricky rovnaká scéna, obsiahnutá v jednom objekte s rovnakou textúrou ako predchádzajúca aplikácia, avšak je pridaný phongov osvetľovací model.



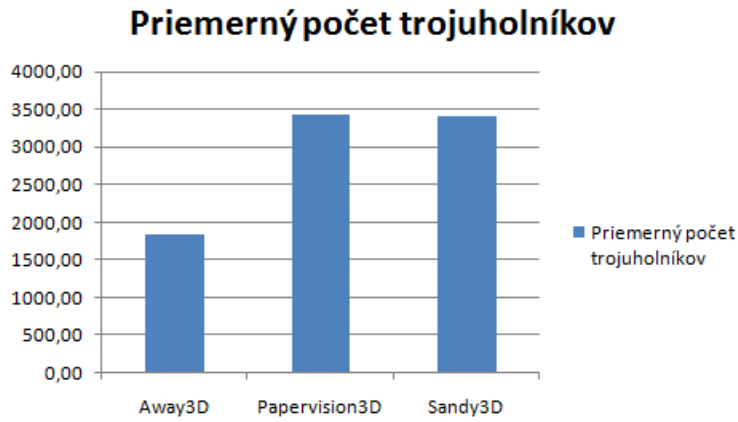
Obrázok 4: Tretí typ aplikácie pre testovanie

Tu je práve vidieť výborne zvládnutý matematický model knižnice Sandy3D, ktorá ako jediná v priemernom čase dokázala korektne vykresliť scénu v reálnom čase. Away3D síce vykresľovala pomalšie, avšak spomedzi meraných knižníc práve jej výstup bol takmer úplne bez artefaktov. Pri tomto meraní najhoršie dopadla knižnica Papervision3D, ktorá má problém časovo a aj korektne vykresliť phong shading.

4.4 Rýchlosť renderovania jednotlivých knižníc

Posledný typ aplikácie meria rýchlosť vykresľovania 3D scény jednotlivých knižníc. Tá je udávana v počte trojuholníkov, ktoré dokáže knižnica vykresliť v reálnom čase. Aplikácia samotná zobrazuje dynamickú scénu, v ktorej sa pohybuje stúpajúci počet planét až kým aplikácia nezistí, že rýchlosť vykreslenia stúpla nad 40 ms.

Z obrázka 5 je zrejmé, že cenou za bohatú paletu techník platí knižnica Away3D rýchlosťou vykresľovania. Naproti nej ostatné knižnice sa držia pomerne vysoko na hodnote 3 500 vykreslených trojuholníkov v reálnom čase.



Obrázok 5: Graf rýchlosti vykreslenia 3D scény pri jednotlivých knižniciach

4.5 Vyhodnotenie a závery

Každá z porovnávaných knižníc jedinečným spôsobom vykresľuje 3D scénu a tak sa každá zíde pre rôzne typy aplikácií.

Away3D je knižnicou pre aplikácie, ktorých koncoví používatelia majú silnejšie počítače, avšak sú zameraní na zobrazovanie scén pomocou pokročilejších grafických techník, poprípade čo najreálnejšieho zobrazovania. Dobre sa takto hodí pre zobrazenie scén s nie komplikovanou geometriou, ale s dôrazom pre verné vykreslenie.

Papervision3D na rozdiel od Away3D neponúka takú paletu techník pre zobrazovanie, avšak snaží sa to kompenzovať rýchlosťou vykresľovania. Taktiež tým, že táto knižnica má tak bohatú komunitu na webe existuje mnoho rozšírení a ukážok pre ňu. Práve to, že je ľahko prístupná, rýchla a má dostatočné techniky pre vykresľovanie 3D scény ju robí výbornou knižnicou pre tvorbu webaplikácií.

Sandy3D je najstaršou knižnicou pre vykresľovanie 3D scén, oproti ostatným má výborne vytvorený matematický model, má najviac návodov a príkladov zo všetkých knižníc a svojím prepojením s externými časticovými a fyzikálnymi knižnicami sa veľmi dobre hodí pre zobrazovanie fyzikálnych alebo matematických simulácií.

Budúcnosť 3D na webe

Aj keď momentálne je Flash platforma jednou z vhodnejších pre vykresľovanie 3D scén na webe, samotný vývoj Flash prehliadačov sa postupne začína zameriavať skôr na mobilné zariadenia ako na počítačové zostavy. Okrem toho sa pre web dlhodobejšie chystá nový štandard HTML 5, ktorý pomocou rozšírenia WebGL a tagu canvas bude schopný zobrazovať 3D scénu priamo s hardvérovou akceleráciou. V nedávnej dobe vznikla reakcia pre

existenciu veľkého množstva skriptovacích jazykov pre vývoj webaplikácií. Je ním open source jazyk haXe, ktorý má za cieľ zjednotiť tvorbu aplikácií inak tvorených v Javascripte, Jave, Actionscripte, PHP alebo v C++. Syntax tohto jazyka je podobný ako Javascript alebo Actionscript, keďže syntax prebral od ECMAScript štandardu. Obsahuje aj kompilátor, ktorý dokáže aplikáciu z jazyka haXe prekompilovať do týchto jazykov alebo aj priamo do Flash aplikácie, ktorá je optimálnejšia oproti originálnemu Flex kompilátoru a dokáže poskytnúť v niektorých prípadoch až 30 % zrýchlenie. Hlavnou výhodou haXe je, že dokáže vyvíjať rovnaké aplikácie ako pre Flash tak aj pre Javascript, ktorým bude v budúcnosti sprístupnený aj webGL.

Literatúra

[1] Žára, J.: *Moderní počítačová grafika*. Brno : Computer Press, 2004. ISBN 80-251-0454-0.

[2] *Adobe, Flash Player Statistics*. [online] 2010 [cit. 9.3.2010] Dostupné z `http://www.adobe.com/products/player_census/flashplayer`

[3] *ActionScript 3.0 Language and Components Reference*. [online] 2010 [cit. 9.3.2010] Dostupné z `http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3`

[4] *ECMAScript*. [online] 2010 [cit. 9.3.2010] Dostupné z `http://www.ecmascript.org`

[5] *Away3D Flas Engine*. [online] 2007 [cit. 9.3.2010] Dostupné z `http://www.away3d.com`

[6] *Papervision3D*. [online] 2006 [cit. 9.3.2010] Dostupné z `http://blog.papervision3d.org`

[7] *Sandy 3D engine (AS3 & AS2) for Adobe Flash*. [online] 2005 [cit. 9.3.2010] Dostupné z `http://flashsandy.org`

[8] Kolesár, I.: *Testovanie flash 3D knižníc*. [online] 2010 [cit. 9.3.2010] Dostupné z `http://www.divan.tym.cz/flash`

[9] *haXe, Welcome to haXe!*. [online][cit. 9.3.2010] Dostupné z `http://haxe.org`

Kontaktná adresa

Ivan Kolesár,
Katedra aplikovanej informatiky, FMFI UK v Bratislave,
Mlynská Dolina, 842 48 Bratislava,
`kolesar.ivan@gmail.com`

IMPLEMENTÁCIA HEURISTIKY PRE TVORBU ROZVRHU SLUŽIEB VODIČOV AUTOBUSOV POMOCOU PYFUZZY

LADOVSKÝ, Tomáš, (SK)

Abstrakt. Článok rieši problém rozpisovania služieb vodičom autobusov pomocou fuzzifikovanej heuristiky day-by-day. Heuristika priraďuje turnusy vodičom postupne pre každý deň plánovaného časového obdobia a neuvažuje s následkami rozhodnutí, ktoré vznikli v minulosti. Tento nedostatok heuristiky vniesol do plánovaných rozhodnutí (priradení) neistotu, čo je dôvodom zavedenia fuzzifikácie. Neistotu článok modeluje pomocou fuzzy inferenčného systému a implementuje ju pomocou balíčka pyFuzzy.

1 Úvod

Problém rozpisovania služieb vodičom autobusov (The Bus Driver Rostering Problem – DRP) pozostáva zo zobrazenia množiny vodičov do množiny turnusov pre každý deň danej časovej periódy. Tento problém je v článku definovaný s ohľadom na viacero právnych predpisov, ktorými stanovené podmienky musia byť dodržané. Všetkým vodičom musíme priradiť turnus alebo deň voľna a to pre každý deň plánovaného obdobia. Preto pre každého vodiča musí tvorca rozpisu služieb vyhotoviť postupnosť turnusov a dní voľna pre uvažovanú časovú periódu. V článku je DRP obmedzovaný viacerými právnymi predpismi [1] a pravidlami spoločností pomocou nasledujúcich *ťažkých podmienok*:

(h1) každý turnus musí byť pridelený maximálne jednému vodičovi;
(h2) každý vodič môže dostať pridelený maximálne jeden turnus z vopred zvolenej podmnožiny množiny všetkých turnusov (pracovné dni, víkend, sviatky, leto, zima) alebo deň voľna;
(h3) každý vodič musí odpočívať minimálne 11 hodín medzi dvoma nasledujúcimi turnusmi.

Nazývajú sa ťažkými podmienkami, pretože sa pri vytváraní rozpisu služieb nesmú porušiť. Sú predpísané právom, zmluvami a pravidlami autobusovej spoločnosti. S ohľadom na

ťažké podmienky dobré rozpisy služieb obvykle obsahujú malé diferencie medzi celkovými rozvrhnutými pracovnými hodinami každého vodiča. Ďalej je dobrý rozpis služieb charakterizovaný tým, že vodičom sa opakujú rovnaké alebo podobné turnusy vzhľadom na trasu a/alebo obtiažnosť. V článku sa zaoberáme dvoma charakteristikami dobrého rozpisu služieb (*ľahké podmienky*) a to:

(s1) zrovnomerňovaním celkových kumulovaných denných pracovných časov vodičov za celé časové obdobie rozpisovania služieb;
(s2) maximalizáciou frekvencie opakovania rovnakých alebo podobných turnusov.

V ľahkých podmienkach sú zahrnuté ciele rozpisovania služieb a tie optimalizujeme. Na riešenie problému je zvolená heuristika day-by-day [2], ktorá najskôr nájde priradenia medzi vodičmi a turnusmi pre prvý deň plánovaného časového obdobia, potom pre druhý, atď. Hlavným jej nedostatkom je, že neuvažuje s následkami rozhodnutí, ktoré vznikli v minulosti.

2 Matematická formulácia

Rozpisovanie služieb vodičom autobusov znamená, že pre každý deň z množiny dní $\mathscr{D} = \{1,2,\ldots,n\}$ sa priradí vodičovi V_i z množiny vodičov $\mathscr{V} = \{V_1,V_2,\ldots,V_m\}$ taký turnus T_k z množiny turnusov $\mathscr{T} = \{T_1,T_2,\ldots,T_t\}$ alebo deň voľna, aby nedošlo k porušeniu žiadnej ťažkej podmienky počas optimalizovania ľahkých podmienok. Parameter n označuje počet dní, parameter m udáva počet vodičov a parameter t udáva počet turnusov. **Turnus** $T_k = (z_k, e_k, c_k)$ je definovaný usporiadanou trojicou nasledovných parametrov: časom začatia z_k; časom ukončenia e_k a denným pracovným časom vodiča c_k turnusu $T_k \in \mathscr{T}$. **Rozpis služieb** $R = (x_{ijk})$ modelujeme pomocou binárnej premennej x_{ijk} nasledovne: $x_{ijk} = 1$ ak i-ty vodič má pridelený v j-ty deň k-ty turnus; inak je $x_{ijk} = 0$ pre $(V_i, j, T_k) \in \mathscr{V} \times \mathscr{D} \times \mathscr{T}$. Táto premenná vyjadruje rozhodnutia tvorcu rozpisov služieb pri prideľovaní turnusov vodičom za celé časové obdobie rozpisovania služieb. Nech s je aktuálny deň heuristiky day-by-day, potom **rozpis služieb v aktuálnom dni** modelujeme premennou y_{ik} následovne: $y_{ik} = 1$ ak i-ty vodič má pridelený v aktuálny deň k-ty turnus; inak je $y_{ik} = 0$ pre $(V_i, T_k) \in \mathscr{V} \times \mathscr{T}$.

Algoritmus day-by-day

K1:	Inicializuj aktuálny deň na prvý deň plánovaného obdobia, $s := 1$.
K2:	Pomocou vopred zvolenej *metódy rozpisovania služieb v aktuálny deň* získaj optimálne riešenie rozpisu služieb v aktuálnom dni Y^*.
K3:	Pre daný aktuálny deň s aktualizuj rozpis služieb R, teda $x_{isk} := y^*_{ik}, (V_i, T_k) \in \mathscr{V} \times \mathscr{T}$.
K4:	Ak je aktuálny deň rovný poslednému dňu obdobia rozpisovania služieb ($s = n$) potom koniec, R je rozpisom služieb na celé plánované obdobie, inak prejdi na nasledujúci deň ($s := s+1$) a goto K2.

Metóda rozpisovania služieb v aktuálny deň je postup riešenia problému rozpisovania služieb v aktuálny deň. Riešením metódy rozpisovania služieb v aktuálny deň je optimálny rozpis služieb v aktuálnom dni $Y^* = (y^*_{ik}), (V_i, T_k) \in \mathscr{V} \times \mathscr{T}$.

3 Implementovanie heuristiky day-by-day

Fuzzifikovaná heuristika je implementovaná v programovacom jazyku *Python* [4]. Dôvodom jeho výberu je množstvo balíčkov, rýchle prispôsobenie programátora k Pythonu, open source, množstvo dokumentácie a jednoduché paralelne programovanie. V dnešnej dobe existujú pre Python dva balíčky, ktoré implementujú fuzzy množiny a vykonávajú na nich operácie fuzzy logiky. Sú to *pyFuzzyLib* a *pyFuzzy*. Balíček *pyFuzzy* je zvolený z nasledujúcich dôvodov: jednoduché programovanie fuzzy množín a fuzzy pravidiel (v zdrojovom kóde alebo súbor flc); možnosť zobrazovania vstupov a výstupov graficky; možnosť grafického zobrazenia fuzzy pravidiel; množstvo príkladov. Fuzzy inferenčný systém implementovaný v pyFuzzy je typu MIMO (Multiple Inputs Multiple Outputs). Programátorovi ponúka pyFuzzy veľký výber zo vstupných a výstupných druhov fuzzy čísiel (množín). Ďalej má programátor možnosť výberu z rôznych noriem a konoriem, ktoré môže použiť pri tvorbe pravidiel, agregácií a/alebo konfigurovania defuzzifikácie. V závislosti od ich výberu je programátor schopný vytvoriť rôzne inferenčné metódy (Mamdami, Larsen, [3]). Náš fuzzy inferenčný systém pozostáva z dvoch vstupov, piatich pravidiel a jedného výstupu (MISO – Multiple Inputs Single Output). Nasledujúce kroky vedú k vytvoreniu fuzzy inferenčného systému.

3.1 Vytvorenie fuzzy inferenčného systému

Kód inicializácie systému zapíšeme v Pythone takto:

```
import fuzzy.System
system = fuzzy.System.System()
```

Trieda *fuzzy.System* koordinuje celý fuzzy systém (bázu dát, bázu pravidiel, fuzzifikačné rozhranie, defuzzifikačné rozhranie).

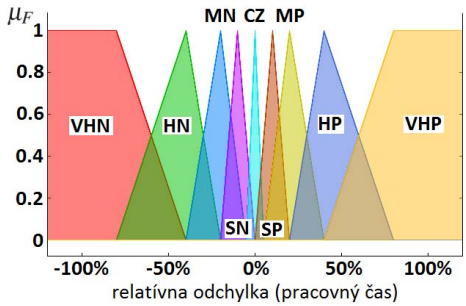
3.2 Definovanie bázy dát fuzzy množín vstupov

Nech s-ty deň mesiaca ($1 \leqq s \leqq 28$) je aktuálnym dňom fuzzifikovanej heuristiky day-by-day. Predpokladáme, že i-ty vodič by mal dostať pridelený k-ty turnus, ktorý začína v s-tý deň. Na základe tohoto predpokladu vieme spočítať nasledujúce relatívne odchýlky:

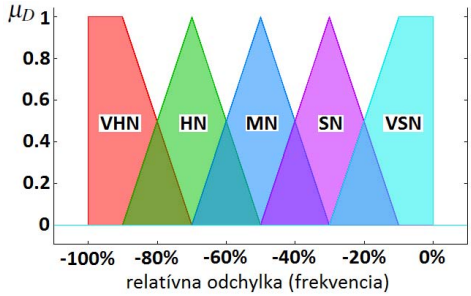
- *Relatívna odchýlka reálneho od ideálneho denného pracovného času* $d_{ik}(s,R)$. Táto relatívna odchýlka môže byť pozitívna alebo negatívna.
- *Relatívna odchýlka reálnej od ideálnej frekvencie* $f_{ik}(s,R)$ podobných turnusov. Táto relatívna odchýlka môže byť iba negatívna.

Pre tvorcu rozpisu služieb je hodnota $d_{ik}(s,R)$ ($f_{ik}(s,R)$) veľmi dôležitá. Čím je hodnota $d_{ik}(s,R)$ bližšie k nule ($f_{ik}(s,R)$ väčšia), tým je väčšie uprednostnenie tvorcu rozpisu služieb, aby priradil i-tému vodičovi k-tý turnus na začiatku s-tého dňa.

Nech D je fuzzy lingvistická premenná vyjadrujúca relatívnu odchýlku reálneho denného pracovného času od ideálneho, ktorá nadobúda nasledujúce hodnoty: **VHN** – veľmi veľká



(a) Funkcie príslušnosti fuzzy množín VHN, HN, MN, SN, CZ, SP, MP, HP, VHP; popisujúce relatívnu odchýlku reálneho denného pracovného času od ideálneho



(b) Funkcie príslušnosti fuzzy množín VHN, HN, MN, SN, VSN; popisujúce relatívnu odchýlku reálnej od ideálnej frekvencie podobných turnusov

Obrázok 1: Funkcie príslušnosti fuzzy množín vstupov; lingvistických premenných D a F

negatívna relatívna odchýlka; **HN** - veľká negatívna relatívna odchýlka; **MN** - stredná negatívna relatívna odchýlka; **SN** – malá negatívna relatívna odchýlka; **CZ** – relatívna odchýlka blízko nule; **SP** – malá pozitívna relatívna odchýlka; **MP** – stredná pozitívna relatívna odchýlka; **HP** – veľká pozitívna relatívna odchýlka; **VHP** – veľmi veľká pozitívna relatívna odchýlka. Na obrázku 1(a) sú zobrazené jednotlivé funkcie príslušnosti k predošlým fuzzy množinám.

Nech F je fuzzy lingvistická premenná vyjadrujúca relatívnu odchýlku reálnej frekvencie od ideálnej, ktorá nadobúda nasledujúce hodnoty: **VHN** – veľmi veľká negatívna relatívna odchýlka, **HN** - veľká negatívna relatívna odchýlka, **MN** – priemerná negatívna relatívna odchýlka, **SN** – malá negatívna relatívna odchýlka, **VSN** – veľmi malá negatívna relatívna odchýlka. Na obrázku 1(b) sú zobrazené jednotlivé funkcie príslušnosti k predošlým fuzzy množinám. Kód predošlých fuzzy množín vstupov zapíšeme v Pythone nasledujúco:

```
from fuzzy.InputVariable import InputVariable
from fuzzy.Adjective import Adjective
from fuzzy.set.Polygon import Polygon
from fuzzy.set.Triangle import Triangle
from fuzzy.fuzzify.Plain import Plain
irregularity = InputVariable(fuzzify = Plain(), min = -100, max = 100)
system.variables["irregularity"] = irregularity
irregularity.adjectives['VHN']=Adjective(Polygon([(-100,1),(-80,1),(-40,0)]))
irregularity.adjectives['HN']=Adjective(Polygon([(-80,0),(-40,1),(-20,0)]))
irregularity.adjectives['MN']=Adjective(Polygon([(-40,0),(-20,1),(-5,0)]))
irregularity.adjectives['SN']=Adjective(Polygon([(-20,0),(-10,1),(0,0)]))
irregularity.adjectives['CZ']=Adjective(Triangle(0,5,5))
irregularity.adjectives['SP']=Adjective(Polygon([(0,0),(10,1),(20,0)]))
irregularity.adjectives['MP']=Adjective(Polygon([(5,0),(20,1),(40,0)]))
irregularity.adjectives['HP']=Adjective(Polygon([(20,0),(40,1),(80,0)]))
irregularity.adjectives['VHP']=Adjective(Polygon([(40,0),(80,1),(100,1)]))
```

Podobne píšeme kód aj pre fuzzy premennú „frequency“. Trieda *fuzzy.InputVariable* implementujúca vstupnú fuzzy premennú, pozostáva z niekoľkých fuzzy hodnôt, ktoré implementuje trieda *fuzzy.Adjective*. V našom prípade to môže byť napríklad hodnota „VHN“ fuzzy premennej „relatívna odchýlka reálneho denného pracovného času od ideálneho“.

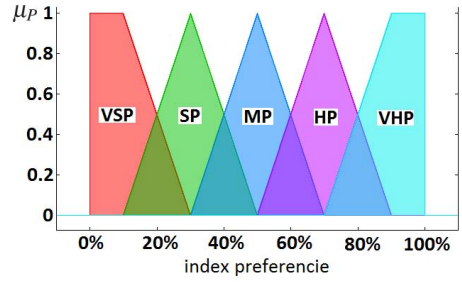
Balík *fuzzy.set* obsahuje triedy implementujúce fuzzy množiny, ktoré majú napríklad tvar trojuholníka `Triangle()`, lichobežníka `Trapez()`, ostrej hodnoty `Singleton()`, polygónu `Polygon()` a veľa iných. Balík *fuzzy.fuzzify* pozostáva z tried, ktoré slúžia na určenie stupňa, ktorým jednotlivé vstupy prislúchajú fuzzy množinám definovaných funkciami príslušnosti. Tieto triedy sa starajú o zobrazenie ostrej hodnoty do prislúchajúcej fuzzy množiny.

3.3 Definovanie bázy dát fuzzy množín výstupov

Nech P je fuzzy lingvistická premenná vyjadrujúca silu uprednostňovania (preferencie) tvorcu rozpisu služieb, ktorá nadobúda nasledujúce hodnoty: **VSP** – veľmi slabé uprednostňovanie, **SP** – slabé uprednostňovanie, **MP** – stredne silné uprednostňovanie, **HP** – silné uprednostňovanie, **VHP** – veľmi silné uprednostňovanie. Kód predošlých fuzzy množín výstupu zapíšeme v Pythone takto:

```
from fuzzy.norm.Min import Min
from fuzzy.norm.Max import Max
from fuzzy.defuzzify.COG import COG
from fuzzy.OutputVariable import OutputVariable
INF, ACC = Min(), Max()
COG = COG(INF=INF, ACC=ACC, failsafe = 0., segment_size=0.5)
preferen = OutputVariable(defuzzify = COG, min = 0, max = 100)
system.variables['preference'] = preferen
preferen.adjectives['VSP']=Adjective(Polygon([(0,0),(0,1),(10,1),(30,0)]))
preferen.adjectives['SP']=Adjective(Polygon([(10,0),(30,1),(50,0)]))
preferen.adjectives['MP']=Adjective(Polygon([(30,0),(50,1),(70,0)]))
preferen.adjectives['HP']=Adjective(Polygon([(50,0),(70,1),(90,0)]))
preferen.adjectives['VHP']=Adjective(Polygon([(70,0),(90,1),(100,1),(100,0)]))
```

Index preferencie $p_{ik}(s,R)$ vyjadruje uprednostňovanie tvorcu rozvrhu, či priradí i-tému vodičovi k-ty turnus, ktorý začína v s-tý deň. Nech $0 \leqq p_{ik}(s,R) \leqq 1$, $V_i \in V, T_k \in T, s \in D$. Čím je index uprednostnenia väčší, tým je tvorca rozpisu služieb presvedčenejší, že dôjde k priraďovaniu nad uvažovanými vodičmi a turnusmi v aktuálny deň. Na obrázku 2 sú zobrazené jednotlivé funkcie príslušnosti fuzzy množín popisujúcich silu uprednostňovania tvorcu rozpisu služieb. Balík *fuzzy.norm* pozostáva z rôznych t-noriem a t-konoriem, ktoré sú implementované triedami. Napr. operátory maxima, minima, algebraické sčítanie alebo algebraické násobenie. Tieto operátory sa používajú pri vyhodnocovaní pravidiel a pri agregácií, teda zjednotení výstupov každého pravidla. Balík *fuzzy.defuzzify* pozostáva z niekoľkých tried, ktoré vykonávajú proces defuzzifikácie. Ide o opačný proces k fuzzifikácii, teda triedy zobrazujú fuzzy množiny na ostré hodnoty. Najznámejšie sú COG (Center of Gravity) a COGS (Center of Gravity for Singletons). Trieda *fuzzy.OutputVariable*



Obrázok 2: Funkcie príslušnosti fuzzy množín VSP, SP, MP, HP, VHP určujúce index preferencie

implementuje výstupnú fuzzy premennú. Pri jej inicializácii treba určiť metódu defuzzifikácie. Pozostáva z niekoľkých fuzzy hodnôt.

3.4 Definovanie bázy pravidiel

Teraz definujme bázu pravidiel. Vieme, že s je aktuálny deň a R je rozpis služieb. Taktiež vieme, že relatívna odchýlka reálneho denného pracovného času od ideálneho je značená ako $d_{ik}(s,R)$. Ďalej relatívna odchýlka reálnej od ideálnej frekvencie podobných turnusov je značená ako $f_{ik}(s,R)$ a index preferencie je značený ako $p_{ik}(s,R)$. Kvôli čitateľnosti zapisujeme v nasledujúcom texte $d_{ik}(s,R), f_{ik}(s,R)$ a $p_{ik}(s,R)$ iba ako d, f a p.

Báza pravidiel

R1:	IF d is {VHN,HN,MN,SN,SP,MP,HP,VHP} AND f is VHN THEN p is VSP
R2:	IF (d is CZ AND f is VHN) OR (d is {VHN,HN,MN,SN,SP,MP,HP,VHP} AND f is HN) OR (d is {VHN,HN,HP,VHP} AND f is MN) OR (d is {VHN,VHP} AND f is SN) THEN p is SP
R3:	IF (d is CZ AND f is HN) OR (d is {MN,SN,SP,MP} AND f is MN) OR (d is {HN,HP} AND f is SN) OR (d is {VHN,VHP} AND f is VSN) THEN p is MP
R4:	IF (d is CZ AND f is MN) OR (d is {MN,SN,SP,MP} AND f is SN) OR (d is {MN,SN,SP,MP} AND f is VSN) THEN p is HP
R5:	IF (d is CZ AND f is SN) OR (d is {SN,CZ,SP} AND f is VSN) THEN p is VHP

Predošlú tabuľku pravidiel zapíšeme v Pythone takto (iba prvé pravidlo):

```
from fuzzy.Rule import Rule
from fuzzy.operator.Compound import Compound
from fuzzy.operator.Input import Input
OR, AND = Max(), Min()
system.rules['preference index is VSP'] = Rule(
    adjective = preferen.adjectives["VSP"],
    operator=Compound(OR,
        Compound(AND,
            Compound(OR,
                Input(system.variables["irregularity"].adjectives["SN"]),
                Input(system.variables["irregularity"].adjectives["MN"]),
                Input(system.variables["irregularity"].adjectives["HN"]),
                Input(system.variables["irregularity"].adjectives["VHN"])
            ),
            Input(system.variables["frequency"].adjectives["VHN"])
        ),
        Compound(AND,
            Compound(OR,
                Input(system.variables["irregularity"].adjectives["SP"]),
                Input(system.variables["irregularity"].adjectives["MP"]),
                Input(system.variables["irregularity"].adjectives["HP"]),
                Input(system.variables["irregularity"].adjectives["VHP"])
            ),
            Input(system.variables["frequency"].adjectives["VHN"])
        )
    ),CER=CER
)
```

Podobne píšeme kód aj pre zostávajúce pravidlá R2 až R5. Trieda *fuzzy.Rule* implementuje bázu pravidiel. Balík *fuzzy.operator* pozostáva z operátorov, ktoré vytvárajú bázu pravidiel.

Napríklad trieda `Compound()` slúži na skladanie zložitejších pravidiel pomocou noriem a konoriem. Trieda `Input()` slúži na zviazanie fuzzy hodnoty k uvažovanému vstupu (tvorba výroku).

3.5 Vykreslenie grafov

Príkaz `createDoc(system)` vykresľuje grafy fuzzy premenných vstupov (irregularity, frequency) a výstupov (index preferencie) zadefinovaných v báze dát. Príkaz `create3DPlot()` vykresľuje 3D obrázok riešenia výstupu FIS vzhľadom na vstupy.

```
from fuzzy.doc.plot.gnuplot import doc
d = doc.Doc()
d.createDoc(system)
d.create3DPlot(system,"irregularity","frequency","preference")
```

Balíček *fuzzy.doc.plot.gnuplot* je potrebný na vykreslenie vstupných fuzzy množín a výstupných fuzzy premenných a to v 2D a 3D prevedení. Na vykreslenie je použitý program gnuplot [7] cez rozhranie Gnuplot.py.

3.6 Získanie výstupnej hodnoty

Inštancia vstupných hodnôt relatívnej odchýlky pracovných kumulovaných časov od ideálnych je napríklad rovná hodnote $irr = 13$ a relatívna odchýlka frekvencie opakovania podobných turnusov je rovná hodnote $freq = -54$ (priradenie jedného vodiča a jeden turnus). Výslednú hodnotu indexu preferencie spočítame pomocou príkazu `system.calculate(...)`. Ta naplní slovník "'out"' novou hodnotou výstupu.

```
irr, freq, out = 13, -54, {"preference" : 0}
system.calculate(input={'irregularity':irr,'frequency':freq},output=out)
print output
```

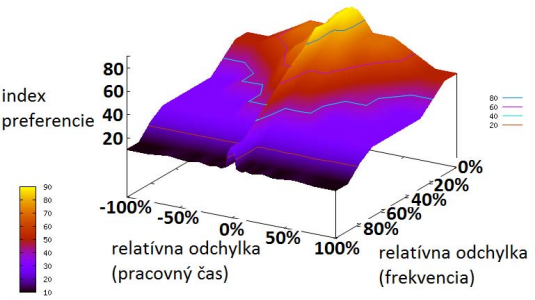
Výstupnou hodnotou FIS systému je ostrá hodnota indexu preferencie y (output). Defuzzifikáciu robíme pomocou najčastejšie používanej metódy 'ťažiska' (Center of gravity – COG). Teraz vieme spočítať index preferencie pre i-teho vodiča a k-ty turnus. Nasledujúci algoritmus počíta maticu indexu preferencie $P = (p_{ik})_{m\times t}$ pre všetkých vodičov a turnusy z daných množín $\mathscr{V}$ a $\mathscr{T}$. Nech s je aktuálny deň a R je rozpis služieb.

Algoritmus výpočtu matice indexu preferencie (ceny) $P = (p_{ik})_{m\times t}$

K1:	Inicializuj index vodiča $i := 1$ a index turnusu $k := 1$.
K2:	Ak je $(i = m)$ a $(k = t)$ potom **STOP** inak goto **K3**.
K3:	Pre hodnoty vstupu $x_1 := d_{ik}(s,R)$ a $x_2 := f_{ik}(s,R)$ spočítaj pomocou navrhnutého fuzzy inferenčného systému hodnotu výstupu y a zapíš ju do matice $p_{ik}(s,R) := y$. Goto **K4**.
K4:	Ak $(k = t)$ potom $i := i+1$ a $k := 0$ inak $k := k+1$. Goto **K2**.

Obrázok 3 zobrazuje výstupnú hodnotu fuzzy inferenčného systému pre každého vodiča a turnus. V podstate ide o 3D-graf matice $P = (p_{ik})_{m\times t}$.

Nech u_{ik} je binárny parameter rozpisovania služieb, ktorý je definovaný nasledovne: $u_{ik} = 1$ ak i-ty vodič môže dostať pridelený k-ty turnus; inak $u_{ik} = 0$ pre $(V_i, T_k) \in \mathscr{V} \times \mathscr{T}$. Tento parameter definuje tvorcovi rozpisu služieb, či môže danému vodičovi priradiť určitý turnus. Taktiež môžeme týmto parametrom definovať náročnosť turnusov, senioritu, atď. Nech v_i je binárny parameter rozpisovania služieb, ktorý je definovaný nasledovne: $v_i = 1$ ak i-ty vodič môže pracovať v aktuálny deň; inak $v_i = 0$ pre $V_i \in \mathscr{V}$. Tento parameter slúži na modelovanie požadovaných dní voľna zo strany vodičov, plánované školenia zo strany zamestnávateľa, zdravotnú starostlivosť a iné. Nech w_k je binárny parameter rozpisovania služieb, ktorý modeluje rozhodnutia: $w_k = 1$ ak musíme odjazdiť k-ty turnus v aktuálny deň; inak $w_k = 0$ pre $T_k \in \mathscr{T}$. Nie každý turnus sa musí odjazdiť v daný deň. Napr. turnusy pre voľné dni sa nesmú odjazdiť cez týždeň.



Obrázok 3: Správanie sa fuzzy inferenčného systému vzhľadom na jeho vstupy

Nech s je aktuálnym dňom heuristiky a r_i je čas ukončenia priradeného turnusu i-teho vodiča v $(s-1)$-vý deň a ďalej nech platí, že ak $\left(\frac{z_k+1440-r_i}{660} < 1\right)$ alebo $(u_{ik} = 0)$ potom $b_{ik} = -\infty$ inak $b_{ik} = p_{ik}(s,R)$. Pomocou tejto implikácie zaručujeme splnenie ťažkej podmienky (h3). Teraz keď poznáme ceny priraďovacej úlohy b_{ik}, môžeme spočítať zovšeobecnenú priraďovaciu úlohu priradenia turnusov vodičom pre uvažovaný aktuálny deň. Výsledkom je optimálne riešenie rozpisu služieb v aktuálnom dni Y^*, viď. druhý krok heuristiky.

$$\max\{\sum_{V_i \in \mathscr{V}} \sum_{T_k \in \mathscr{T}} b_{ik} : \sum_{V_i \in \mathscr{V}} y_{ik} = w_k, T_k \in \mathscr{T}; \sum_{T_k \in \mathscr{T}} y_{ik} \leqq v_i, V_i \in \mathscr{V}; y_{ik} \geqq 0, (V_i, T_k) \in \mathscr{V} \times \mathscr{T}\}$$

4 Reálna inštancia problému

Pre každý deň z množiny dní $\mathscr{D} = \{1,2,\ldots,28\}$ priraď vodičovi V_i z množiny vodičov $\mathscr{V} = \{V_1, V_2, \ldots, V_{107}\}$ deň voľna alebo turnus T_k z množiny turnusov $\mathscr{T} = \{T_1, T_2, \ldots, T_{179}\}$, pričom nedôjde k porušeniu žiadnej ťažkej podmienky a k maximálnemu zlepšeniu ľahkých podmienok. Časť turnusov T_k je zadaná v tabuľke 1. Turnusy s identifikačným číslom 1 až 107 sú turnusy, ktoré sa majú odjazdiť počas pracovných dní. Turnusy s číslami 108 až 179 sa majú odjazdiť počas víkendov. Vzhľadom na túto skutočnosť sú definované parametre u_{ik}, v_i, w_k pre celé obdobie rozpisovania služieb. Algoritmus day-by-day je napísaný v programovacom jazyku Python (Python Programming Language [4]). Na výpočet priraďovacej úlohy je použité GLPK (GNU Linear Programming Kit [5]). Získavame prijateľný 28 dňový rozpis služieb vodičov autobusov (tabuľka 2), ktorého relatívna odchýlka kumulovaného pracovného času od ideálneho (obrázok 4(a)) sa sústreďuje okolo nuly. Na obrázku 4(b)

Tabuľka 1: Dvadsaťosemdňová inštancia problému rozpisovania služieb

	Počet vodičov	Počet dní	Počet turnusov
	107	28	179
Turnus	Začiatok (min)	Koniec (min)	Pracovný čas (min)
T_1	298	460	239.4
T_2	793	980	227.8
⋮	⋮	⋮	⋮
T_{178}	522	711	219.0
T_{179}	260	700	480.3

Tabuľka 2: Dvadsaťosemdňový rozpis služieb vodičov autobusov

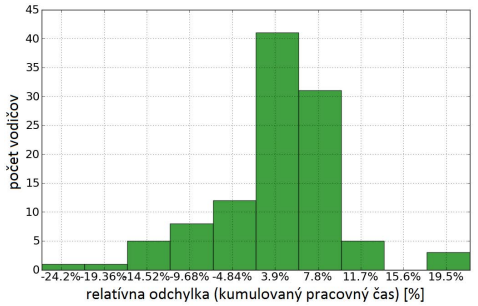
$\mathscr{V}/\mathscr{D}$	1Pon	1Uto	1Str	1Štv	1Pia	1Sob	...	4Sob	4Ned	s_i (min)
V1	T1	T29	T18	T35	T12	T150	...	T147	T154	11400.0
V2	T3	T5	T4	T8	T8	T109	...	T109	T111	10186.0
V3	T2	T11	T8	T26	T5	T110	...	T111	T109	10696.6
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮
V105	T104	T45	T9	T45	T92	T154	...	T174	Off	9928.5
V106	T107	T94	T95	T67	T52	T152	...	T158	T168	11832.6
V107	T106	T79	T24	T88	T16	Off	...	T138	T159	10414.2

vidíme zobrazenú frekvenciu opakovania sa podobných turnusov pridelených vodičom za celé obdobie rozpisovania služieb. Povšimnime si, že turnusy T_1 až T_{72} majú značné zníženú frekvenciu opakovania priradených turnusov a turnusy T_{73} až T_{179} majú zasa zvýšenú frekvenciu priradených turnusov (diagonála od zhora dole).

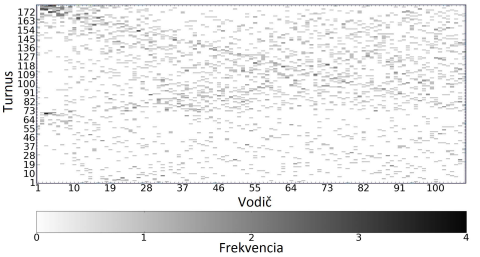
5 Záver

Článok sa venoval problematike rozpisovania služieb vodičom autobusov vo verejnej autobusovej doprave na linkách do 50 km. Na vyriešenie problému bola zvolená heuristická metóda day-by-day, ktorá vytvára rozpis služieb po rade pre každý deň plánovaného obdobia. Slabinou (neistotou) algoritmu je, či vytvorený rozpis služieb v uvažovaný deň vedie k optimálnemu výsledku. Z tohoto dôvodu je časť algoritmu fuzzifikovaná a riešená ako fuzzy inferenčný systém. Vstupom systému sú kandidáti na riešenie (odchýlky od ideálneho pracovného času a frekvencie podobných turnusov) a výstupom je index preferencie. Tento index slúži v zovšeobecnenej priraďovacej úlohe ako cena priradenia medzi turnusmi a vodičmi. Iným dôvodom zavedenia fuzzy inferenčného systému je uľahčenie tvorby rozpisu služieb vzhľadom na skúsenosti tvorcov rozpisov (fuzzy premenné a ich hodnoty, fuzzy pravidlá).

Prečo sa v článku snažíme minimalizovať nerovnomernosti v pracovných výkonoch vodičov autobusov? Dôvodov je viacero. Jedným z nich je vplyv na morálku vodičov. Mo-



(a) Relatívna odchýlka kumulovaného pracovného času od ideálneho za celé obdobie rozpisovania služieb



(b) Frekvencie opakovania podobných turnusov priradených vodičom za celé obdobie rozpisovania služieb

Obrázok 4: Relatívna odchýlka a frekvencia navrhnutého 28 dňového rozpisu služieb vodičov autobusov

rálka vodičov a tak isto aj práceschopnosť (menej absencií) sa zvyšuje s rovnomernejšími pracovnými výkonmi. Prečo článok maximalizuje u vodičov frekvenciu rovnakých alebo podobných turnusov? Tak isto aj v tomto prípade je dôvodov viacero. Napríklad bezpečnosť a pohodlie vodičov autobusov (návyk na trasu turnusu, tieto turnusy sú odjazdené bezpečnejšie). No netreba zabúdať aj na to, že sa turnusy nesmú opakovať príliš často. Je možné, že samotná trasa turnusu a jej rôznorodosť pôsobí na vodičov tak, že zostávajú pozorní a že neupadajú do mdlôb z dlhého času sedenia za volantom. Potom by sa ponúkalo viac priestoru pre tvorcu rozpisov služieb, aby sa početnosť podobných turnusov mohla zvýšiť.

Balíček pyFuzzy je veľmi užitočný nástroj. Uľahčuje tvorbu fuzzy množín a prácu s fuzzy logikou pri tvorbe aplikácií. Jeho výhodami je napríklad ľahké modelovanie neistoty, jednoduchý návrh bázy pravidiel a jednoduchý návrh bázy dát (oproti pravdepodobnostným rozdeleniam). Jeho nevýhodami sú napríklad výpočtová rýchlosť inferenčného systému, chýba užívateľská dokumentácia a čiastočná kompatibilita s Windows.

Poďakovanie Táto práca vznikla s podporou grantovej agentúry VEGA vrámci riešenia projektu 1/0135/08 „Optimalizačné problémy v logistických a dopravných systémoch“.

Literatúra

[1] POLIAK, M., GNAP, J., *Práca vodičov nákladných automobilov a autobusov a používanie tachografov*, Žilinská univerzita v Žiline/EDIS-vydavateľstvo ŽU, ISBN 978-80-8070-989-1, (2009).

[2] THEODOROVIC, D., LUČIĆ, P., *A fuzzy set theory approach to the aircrew rostering problem, Fuzzy Sets and Systems, Vol 95, Issue 3*, 261–271,(1998).

[3] LEE, K.H., *First Course on Fuzzy Theory and Applications*, str. 236–243, ISBN 3540229884, (2005)

[4] Python Programming Language, `http://www.python.org/`

[5] GNU Linear Programming Kit (GLPK), Package for solving large-scale linear programming (LP) and mixed integer programming (MIP), `www.gnu.org/software/glpk`

[6] Python fuzzy package, http://pyfuzzy.sourceforge.net/
[7] Gnuplot, A portable command-line driven graphing utility, http://www.gnuplot.info/

Kontaktná adresa

Tomáš LADOVSKÝ (Ing.),
Fakulta riadenia a informatiky, Žilinská univerzita,
Univerzitná 1, 010 26 ŽILINA,
tomas.ladovsky@fri.uniza.sk

VYUŽITIE FOSS PRI PROGRAMOVANÍ MIKROKONTROLÉROV ATMEL

LAJČIAK, Pavol, (SK)

Abstrakt. Článok opisuje použitie otvoreného softvéru pri tvorbe aplikácií pre 8-bitové RISC mikrokontroléry Atmel AVR. Opisuje postup tvorby aplikácií. A to inštaláciu a konfiguráciu potrebných softvérových nástrojov, vytvorenie programu v jazyku C, kompiláciu a napálenie do čipu. Tiež popisuje programátory vhodné na napálenie hotovej aplikácie do mikrokontroléra. Kľúčové slová: mikrokontrolér, Atmel, programátor, jazyk C

Úvod

Algoritmy sú všade okolo nás. Algoritmy používajú rastliny, zvieratá, ľudia, ale aj neživá príroda. Sú do nich vložené samotným Tvorcom. Neživá príroda, rastliny i ľudia sa vyvíjajú podľa určitých algoritmov bez toho, aby si to uvedomovali.

Okrem týchto algoritmov v prírode, existujú algoritmy, ktorými môžeme popísať ľudské konanie a správanie. Existuje napríklad algoritmus na uvarenie čaju, uvarenie chutného obeda atď.

Tiež sa stretávame s algoritmami, ktoré sú zakomponované do strojov a prístrojov, s ktorými sa stretávame v každodennom živote. Sú napríklad súčasťou práčiek, chladničiek, nápojových automatov, riadiaceho systému kúrenia a klimatizácie, a mnohých ďalších elektronických pomocníkov.

Človek sa od nepamäti snažil zjednodušiť a zefektívniť isté činnosti a tak si uľahčiť isté činnosti. A tak vynaliezal rôzne stroje a prístroje, ktoré najskôr sám ovládal. Neskôr tieto stroje automatizoval, aby pracovali sami bez ľudskej obsluhy.

Zo systémového pohľadu sa na stroj môžeme pozrieť ako na spojenie riadiacej jednotky, výkonových členov, senzorickej a akčnej časti. Takúto koncepciu môžeme v podstate nájsť vo všetkých strojoch a prístrojoch.

V minulosti riadiaca časť mohla byť realizovaná mechanickým spôsobom. Príkladom môže byť napríklad hracia skrinka, ktorá ako riadiacu jednotku používala otočný valček, na

ktorom bol systém výčnelkov a pomocou týchto výčnelkov bola naprogramovaná melódia. Zmena melódie bola drahá a zdĺhavá.

Dnes ekvivalent takejto hracej skrinky môžeme nájsť v rôznych kníhkupectvách vo forme hracích pohľadníc.

S nástupom éry počítačov a mikrokontrolérov, sa konštrukcia automatizovaných a riadiacich systémov značne zjednodušila.

Aby takýto systém pracoval, musí mu tvorca, človek vdýchnuť algoritmus, podľa ktorého sa tento systém správa. Pomocou mikrokontroléra, v ktorom je uložený program, v spolupráci s výkonovými členmi, aktormi a senzormi môžeme vytvoriť akýkoľvek stroj alebo prístroj.

1 Vývojové nástroje

Existuje veľa vývojových nástrojov pre radu RISC mikrokontrolérov Atmel AVR pre operačný systém Windows. Samotná firma Atmel dala k dispozícii integrované vývojové prostredie AVR Studio.

AVR Studio spolupracuje s FOSS balíkom WinAVR, ktorý obsahuje nástroje potrebné na vývoj aplikácií platforme Atmel AVR. Obsahuje kompilátor avr-gcc, debuger avr-gdb a programátor avrdude a ďalšie.

Čo sa týka OS GNU/Linux, existuje veľa aplikácií pre vývoj aplikácii na platforme Atmel AVR. V repozitároch distribúcie Ubuntu 10.04 sú obsiahnuté tieto balíčky: ava, avarice, avr-libc, avra, avrdude, avrdude-doc, avrp, avrprog, binutils-avr, gcc-avr, gdb-avr, simulavr, uisp, sdcc, sdcc-doc, sdcc-libraries, sdcc-ucsim, usbprog, usbprog-gui. Môžeme ich nainštalovať pomocou príkazu apt-get install ¡názov_balíčka¿ alebo pomocou správcu balíkov synaptic.

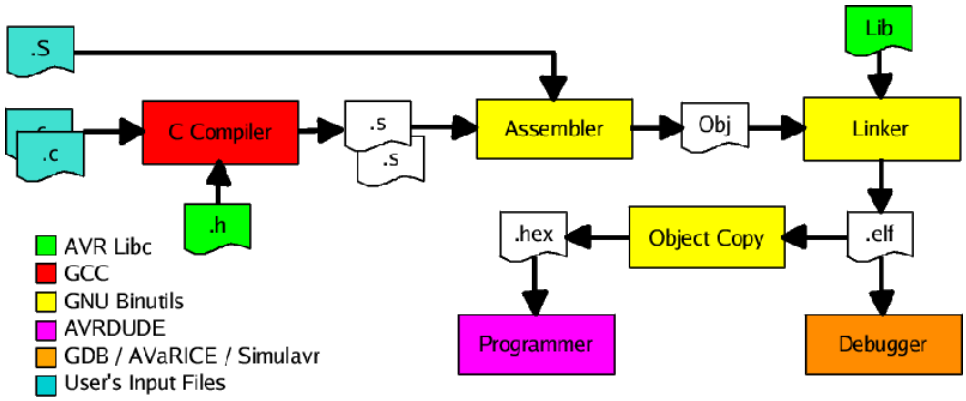
1.1 Asembléry

ava je pokročilý asembler a linker pre 8-bitovú rodinu mikrokontolérov Atmel AVR. Tak ako preprocesor jazyka C poskytuje prácu so segmentmi a virtuálnymi symbolmi. Syntax tohoto asembléra nie je kompatibilná so syntaxou asembléra od firmy Atmel.

avra je asembler pre 8-bitovú rodinu mikrokontolérov Atmel AVR. Z väčšej časti je kompatibilný so syntaxou asembléra od firmy Atmel, ale pridáva nové vlastnosti ako lepšiu podporu makier a prídavné direktívy preprocesora.

1.2 Reťazec vývojových nástrojov

Na obrázku 1 je ukázaný reťazec vývojových nástrojov, ktorý je potrebný pre tvobru aplikácií pre platformu 8-bitových RISC mikrokontolérov pre vývoj jazyku C.



Obrázok 1: Reťazec vývojových nástrojov [1]

1.3 Kompilátor

GCC je skratka pre GNU Compiler Collection. Ide o veľmi flexibilný kompilačný systém. Má rôzne front-endy pre rôzne jazyky. Obsahuje aj veľa back-endov, ktoré generujú kód asembléra pre veľa rôznych architektúr procesoror a hostiteľských operačných systémov. Všetky zdieľajú spoločný middle-end, obsahujúci generickú časť kompilátora, vrátane mnohých optimalizácií.

Z pohľadu GCC je hostiteľským systémom ten, na ktorom kompilátor beží. Cieľovým systém je ten, pre ktorý sá kód prekladá. A zostavovací systém je systém, na ktorom bol kompilátor zostavený zo zdrojového kódu. Ak kompilátor má spoločný hostiteľský a cieľový systém, hovoríme o natívnom kompilátore. Ak hostiteľský a cieľový systém sú rôzne, hovoríme o krížovom kompilátore (cross-compiler).

GCC je odlišný od ostatných kompilátorov. GCC sa zameriava na preklad vysokoúrovňových jazykov do asembléru cieľovej platformy. Balíček gcc-avr je kompilátor GNU C pre cieľovú platformu AVR. AVR GCC obsahuje tri kompilátory - kompilátor pre jazyk C, C++ a Ada. [1]

1.4 GNU AVR Binutils

Programy v balíčku binutils-avr sú používané na manipuláciu s binárnymi a objektovými súbormi, ktoré boli vytvorené pre architektúru Atmel AVR. Skratka GNU Binutils znamené „Binary Utilities“. Obsahuje assembler (gas), linker (ld) a veľa ďalších utilít, ktoré pracujú s binárnymi súbormi a sú vytvorené ako časť reťazca vývojárskeho softvéru.

Nástroje, ktoré boli vytvorené pre prácu s AVR začínajú s prefixom „avr-“. Napríklad, meno pre asembler, ktorý sa natívne volá „as“,(dokonca v dokumentácii GNU asembler je nazývaný ako „gas“). Ale, keď je skompilovaný pre prácu s cieľovou platformou AVR stane sa s neho „avr-as“.

Nasleduje zoznam programov, ktoré sú obsiahnuté v balíčku AVR Binutils:

- avr-as – asembler,
- avr-ld – linker,
- avr-ar – vytvára, modifikuje alebo extrahuje z knižníc (archívov),
- avr-ranlib – generuje index obsahu knižnice (archívu),
- avr-objcopy – kopíruje a prekladá objektové súbory do rôznych formátov,
- avr-objdump – zobrazuje informácie z objektových súborov vrátane disasemblovacích informácií,
- avr-size – vypíše veľkosti sekcii a celkovú veľkosť,
- avr-nm – vypíše symboly z objektových súborov,
- avr-strings – vypíše tlačiteľné reťazce zo súborov,
- avr-strip – Discard symboly zo súborov,
- avr-readelf – zobrazí obsah súborov vo formáte ELF,
- avr-addr21ine – konvertuje adresy do súboru a riadku,
- avr-c+-—-filt – Filter na odkódovanie (demangle) kódovaných symbolov C++.

1.5 Knižnica jazyka C

avr-libc je štandardná knižnica jazyka C, používaná na vývoj programov pre mikrokontoléry Atmel AVR, ktorá sa Ubuntu 10.04 LTS nachádza vo verzii 1.6.7. Balíček obsahuje statické knižnice ako aj potrebné hlavičkové súbory. Poskytuje podmnožinu štandardnej knižnice C pre 8-bitové RISC mikrokontoléry Atmel AVR a základný kód potrebný pre štart väčšiny aplikácií.

1.6 Debugovacie nástroje

Balíček gdb-avr bol kompilovaný pre cieľovú architektúru AVR. GDB je debuger na zdrojovej úrovni, schopný prerušiť program na ľubovolnom špecifikovanom riadku, zobraziť hodnoty premenných, a určiť chybu, pokiaľ nastala. V súčasnosti funguje pre jazyky C, C++, Fortran, Modula 2 a programy v jazyku Java. Je povinnou výbavou pre každého seriózneho programátora. Tento balíček je primárne určený pre AVR vývojárov.

Okrem debugera gdb-avr pod OS GNU/Linux môžeme používať balíček avarice, pokiaľ cielový mikrokontolér disponuje JTAG rozhraním a my máme programovací hardvér, pomocou ktorého môžeme tento mikrokontolér pripojiť. Program avarice prekladá príkazy medzi diaľkovým debugovacím protokolom GDB a protokolom AVR JTAG ICE. AVR JTAG sa používa na debugovanie procesora v reálnom aplikácii - doske. Pomocou neho samozrejme tiež môžete naprogramovať AVR mikrokontolér.

1.7 Simulátor

simulavr simuluje rodinu mikrokontrolérov Atmel AVR, emuluje vzdialený cieľ pre gdb a zobrazuje informácie o registroch a pamäti v reálnom čase.

1.8 Programátory

Balíček avrdude je programátor slúžiaci na čítanie/zápis/manipuláciu s obsahom ROM a EEPROM pamäti mikrokontroléra s použitím techník ISP. Balíček avrdude-doc obsahuje dokumentáciu ku konfigurácii a prevádzkovaniu balíčka avrdude.

Balíček avrp je programátor slúžiaci na programovanie FLASH/EEPROM pre 8-bitovú rodinu RISC procerorov Atmel AVR. Tiež umožňuje programovanie rady Atmel AT89. Podporuje minimálne štyri rôzne programovacie zariadenia, vrátane vlastnej vývojovej dosky s ISP programovaním od Atmelu.

Balíček avrprog môže programovať AVR mikrokontroléry a používa paralelný port ako programovacie zariadenie. Zariadenie je programované v ISP móde. Súčasťou balíka je schéma vyžadovaného hardvéru, ktorý je navrhnutý tak, aby bolo efektívny a lacný. `http://avrprog.sourceforge.net`

Programátor uisp je vyžadovaný na programovanie AVR mikrokontolérov objektovým kódom vytvoreným asemblérom/linkerom ava, gas alebo gcc. Podporuje programovanie ISP pomocou vývojovej/prototypovej dosky STK500 a mnohé ďalšie extrémne lacné programáry pripojitelné na paralelný port. Tiež može byť použitý na programovanie mikrokontolérov Atmel AT89S51 a AT89S52.

usbprog je programovací nástroj používaný na nahradenie firmvéru v programovacom zariadení USBprog. Dokáže automaticky získať zoznam dostupných firmvérov z internetu. Vybraný frrmvér dokáže stiahnuť a priamo nahrať do programovacieho zariadenia USBprog.

Môžete jednoducho nainštalovať rôzny firmvér priamo cez USB. Programovacie zariadenie dokáže programovať a debugovať AVR a ARM mikrokontoléry ako USB-RS232 prevodník, ako JTAG rozhranie alebo ako jednoduchý vstupno-výstupné rozhranie s 5 linkami. Je to program ovládaný z príkazového riadku. Balíček usbprog-gui poskytuje grafické rozhranie pre balíček usbprog.

2 Ďalšie nástroje

sdcc je kompilátor jazyka C pre rodiny mikrokontolérov Intel MCS51, AVR, HC08, PIC a Z80. Obsahuje kompilátory, asembléry and linkery. Balíček sdcc-doc poskytuje dokumentáciu a príklady a balíček sdcc-libraries obsahuje hlavné knižnice pre balíček sdcc.

sdcc-ucsim je simulátor mikrokontolérov. Obsahuje rozšíriteľnú podporu pre rôzne rodiny mikrokontolérov. V súčasnosti podporuje rodiny mikrokontolérov Intel MCS51, AVR, HC08, PIC a Z80.

Okrem týchto balíčkov existujú balíčky integrovaných vývojových prostredí KontrollerLab a cdkjavr. Podľa aktivity projektu sa zdá, že balíček cdkavr sa už dlhšiu dobu nevyvíja. O trocha lepšie je na tom balíček KontrollerLab, ktorý síce nie je v stabilnej verzii, no jeho posledná verzia je z roku 2008.

3 Programovanie mikrokontrolérov

Programovanie mikrokontolérov Atmel AVR môžeme prevádzať rôznymi spôsobmi. Prvým spôsobom je programovanie technikou ISP (In Circuit Serial Programing), ktorú podporujú všetky procesory z tejto rodiny. Táto technika umožňuje programovanie priamo v obvode, bez toho, aby sme museli mikrokontrolér vytiahnuť zo zariadenia. Programovanie sa vykonáva pomocou ISP konektora, ktorý môže byť 6 pinový alebo 10 pinový. Oba konektory obsahujú signály MOSI, MISO, RST, CLK, VCC, GND. Pri návrhu zariadenia, v ktorom mikrokontrolér chceme programovať pomocou ISP, je potrebné dbať na to, aby na pinoch, ktoré sa používajú na ISP programovanie, nebola žiadna nízkoimpedančná záťaž, alebo aby sa dala počas programovania odojiť.

Existuje aj druhý spôsob programovania pomocou rozhrania JTAG. Toto rozhranie poskytuje väčšie možnosti ako ISP programovanie. No jeho nevýhodou je, že rozhranie JTAG obsahujú viacpinové a drahšie mikrokontroléry. Preto sa týmto programovaním sa nebudeme naďalej zaoberať, nakoľko je to nad rámec tohoto článku.

4 Programátory – softvér

Na to, aby sme aplikáciu, ktorú pre cieľový mikrokontrolér vytvoríme, mohli napáliť do pamäti FLASH/EEPROM mikrokontroléra, potrebujeme programovací softvér a programovací hardvér. V literatúre sa obyčajne pre obe veci používa rovnaký pojem programátor.

Máme dve krajné možnosti vybrať si vhodný programovací softvér a zostrojiť k nemu programovací hardvér. Alebo vybrať si programovací hardvér a nájsť k nemu vhodný programovací softvér.

Moja situácia bola taká, že som chcel používať viacero programovacích hardvérov a tak som hľadal vhodný programovací softvér, ktorý by mi to umožňoval a je priebežne udr-

žiavaný. Mojim požiadavkám najviac vyhovoval balíček avrdude. Môžeme ho využívať z príkazového riadku.

Pokiaľ vám nevyhovuje práca v príkazovom riadku existujú k balíčku avrdude rôzne GUI (Graphics User Interface), ktoré prácu s ním zjednodušujú. Asi najlepším GUI pre OS GNU/Linux je AVR8 Burn-O-Mat, ktorý je napísaný v jazyku Java a tým pádom ide o multiplatformný nástroj. Na domácej stránke tohoto projektu môžete nájsť aj online nástroj na výpočet poistiek mikrokontoléra. Okrem neho existujú ďalšie GUI pre OS GNU/Linux ale nie sú také vyspelé, alebo nie sú aktuálne. Ide o balíčky avrdude-gui a gnome-avrdude. Pre operačný systém Windows existuje Khazama AVR Programmer, ktorého kvality sú porovnateľné s aplikáciou AVR8 Burn-O-Mat.

Okrem toho existujú ďalšie programovacie softvéry, ktoré môžu dobre poslúžiť pre programovanie mikrokontolérov v spojení so rôznym programovacím hardvérom. Ide o programy avrp, avrprog, uisp, usbprog, usbprog-gui.

5 Programátory – hardvér

Programátorov existuje veľké množstvo. Keď si pozrieme dokumetáciu k programovaciemu softvéru (avrdude, avrp, avrprog, uisp), zistíme, že softvér podporuje veľké množstvo proprietárnych i open source programátorov.

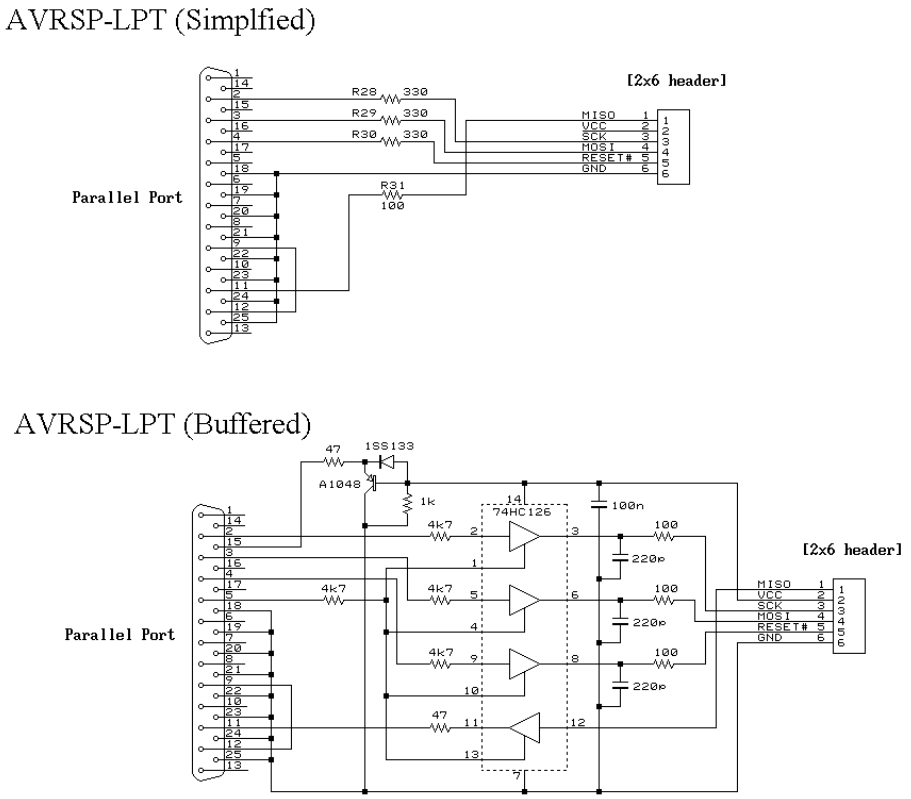
Existujú programátory pripojiteľné na sériový, paralelný a USB port. Jednoduché programátory obsahujú len niekoľko málo diskrétnych súčiastok. Najednodnoduchší na paralelný port obsahuje len 4 rezistory, 25 pinový LPT konektor a 6 alebo 10 pinový ISP konektor. Prípadne ISP konektor môžeme vynechať a vodiče/signály pripojiť priamo na príslušné piny mikrokontroléra.

Na nasledujúcom obrázku je ISP programátor na paralelný port, ktorého autorom je Elm-Chan. [3] Jeho výhodou je extrémne malá zložitosť. Programátor obsahuje len konektor na paralelný port a štyri rezistory a ISP konektor. ISP konektor môžeme vynechať, pokiaľ vodiče pripojíme priamo na vývody mikrokontoléra.

Jeho hlavnou nevýhodou je veľká citlivosť paralelného portu na preťaženie a skrat. Vyvarujte sa prípadnému skratu a pretaženiu, lebo paralelný port sa veľmi ľahko zničí. Napájanie je potrebné riešiť z iného zdroja. Ja osobne používam napájanie z USB portu.

V domovskom priečinku je potrebné vytvoriť súbor .avrduderc, ktorý je používateľským konfiguračným programom pre avrdude. Treba do neho zapísať nasledujúce riadky. Týmto sa pridá definícia programátora s názvom avrsplpt a nadefinuje priradenie pinov pre port LPT. Následne takto nadefinovaný programátor možeme používať pomocou avrdude.

```
programmer
  id    = "avrsplpt";
  desc  = "Elm-chan parallel programmer";
  type  = par;
  reset = 4;
```



Obrázok 2: Elm-Chan ISP LPT programátor

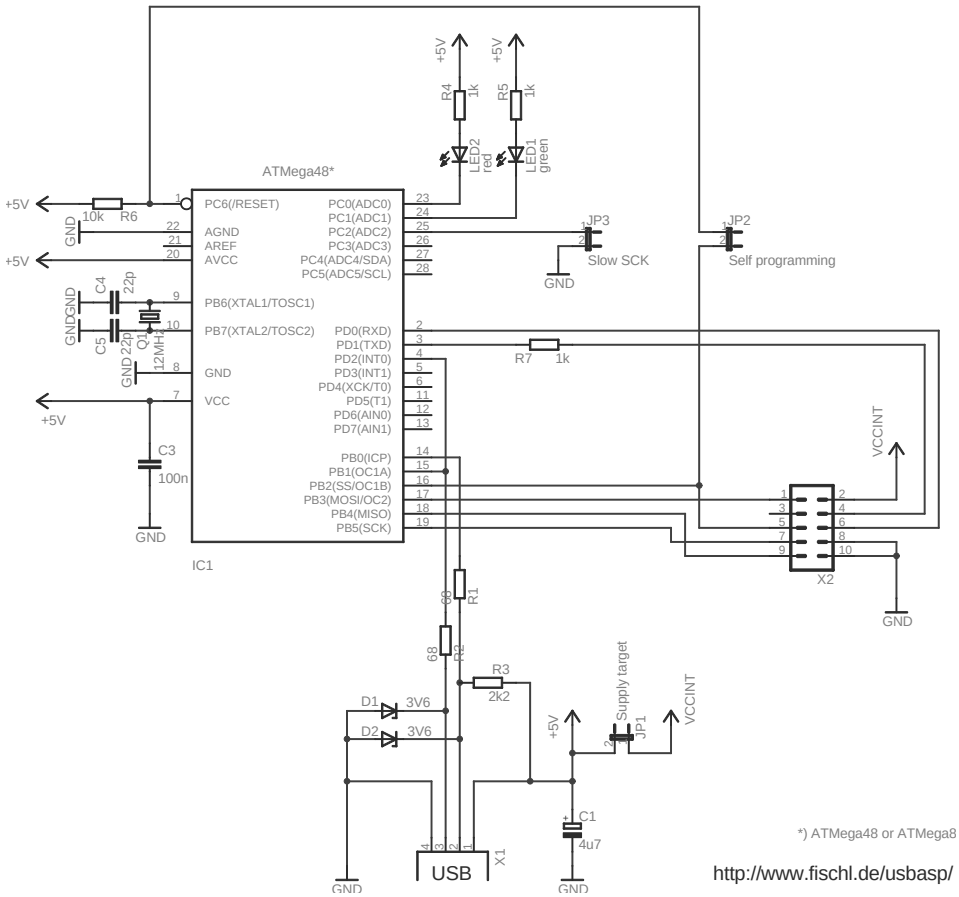
```
  sck   = 2;
  mosi  = 3;
  miso  = 11;
;
```

Programovanie mikrokontroléra pomocou takéhoto programátora na paralelnom porte vykonáme príkazom.

```
avrdude -p m16 -c avrsplpt -U flash:w:name.hex
```

Ďalšou možnosťou je použitie programátora USBasp. Ide o programátor pripájaný na USB port. Jeho srdcom je ATmega8. Ak si tento programátor chceme zostrojiť doma, potrebujeme naprogramovať ATmega8A. Na jeho programovanie môžeme použiť vyššie zmienený ISP programátor na paralelný port.

Ak vás zaujímajú podrobnosti, ako niektorý takýto programátor zostaviť, informácie nájdete na mojej internetovej stránke `http://pavolmaria.org` v sekcii elektronika. [4]



Obrázok 3: ISP programátor USBasp

6 Zapojenie ISP konektora a umiestnenie ISP pinov na mikrokontoléroch

V tab. 1 vidíme umiestnenie ISP pinov na jednotlivých vybraných mikrokontoléroch, v tab. 2 zapojenie 6 pinového a v tab. 3 zapojenie 10 pinového ISP konektora.

7 Vybrané mikrokontoléry

Čo sa týka výberu mikrokontroléra, zvažoval som rôzne faktory. Vybral som si tieto mikrokontoléry: pre malé projekty mikrokontolér ATtiny2313A, pre stredne veľké projekty mikrokontolér ATmega8A a pre veľké projekty mikrokontolér ATmegal6A. Samozrejme, že

Tabuľka 1: Umiestnenie ISP pinov na mikrokontoléroch

	ATtiny2313A	ATmega8A	ATmegal6A
RESET	1	1	9
SCK	19	19	8
MISO	18	18	7
MOSI	17	17	6
VCC	20	7	10
GND	10	8	11

Tabuľka 2: Zapojenie 6 pinového ISP konektora ATMEL

1	MISO	2	VCC
3	SCK	4	MOSI
5	RST	6	GND

Tabuľka 3: Zapojenie 10 pinového ISP konektora ATMEL

1	MOSI	2	VCC
3	LED	4	GND
5	RST	6	GND
7	SCK	8	GND
9	MISO	10	GND

je možné použiť aj ďalšie mikrokontoléry. No je potrebné, aby boli podporované jednotlivými aplikáciami, ktoré na vývoj používate.

Ak chcete váš vývoj uskutočňovať v jazyku C, je potrebné, aby podpora daného mikrokontoléra bola zahrnutá v knižnici avr-libe. Zoznam podporovaných mikrokontolérov nájdete v dokumentácii ku knižnici avr-libc.

Ak chcete váš vývoj uskutočňovať v asembléri je potrebné, aby podpora konkrétneho mikrokontroléra bola zahrnutá v balíčku s asemblérom avra, ava.

8 Vzorový príklad

Ak chceme začať vývoj pre 8-bitové mikrokontoléry Atmel AVR potrebujeme mať nainštalované príslušné nástroje. Potrebujeme mať programovací hardvér. Potrebujeme mať mikrokontolér, pre ktorý budeme vyvíjať aplikácie. Pre demonštračný príklad som vybral procesor ATmegal6A. Tento mikrokontolér som vybral preto, lebo má dostatočne veľa vý-

Tabuľka 4: Parametre vybraných mikrokontolérov

Mikrokontrolér	ATtiny2313A	ATmega8A	ATmegal6A
Flash (Kbytes)	2	8	16
EEPROM (Bytes)	128	512	512
SRAM (Bytes)	128	1024	1024
PDIP Pins	20	28	40
F.max (MHz)	20	16	16
Vcc(V)	1.8-5.5	2.7-5.5	2.7-5.5

vodov a periférii a tak sa dá v budúcnosti dá skúšať široká paleta najrôznejších úloh. Cena tohoto mikrokontoléra je približne 6 Eur.

Ak by sme chceli používať tento postup vo vyučovaní, treba zvážiť, či nie je vhodnejšie použitie mikrokontoléra ATmega8A, ktorého cena je približne 2 Eur

```
/* ledblink.c, an LED blinking program */
#include<avr/io.h>
#include<util/delay.h>

void sleep(uint8_t millisec)
    {while(millisec)
        {_delay_ms(1);/* 1 ms delay */
         millisec--;
        }
    }
main()
    {DDRC |=1<<PC2;  /* PC2 will now be the output pin */
        while(1)
            {PORTC &= ~(1<<PC2);/* PC2 LOW */
             sleep(100);/* 100 ms delay */

             PORTC |=(1<<PC2); /* PC2 HIGH */
             sleep(100);/* 100 ms delay */
            }
    }
```

Vzorový príklad je prevzaný z [5]
Preklad súboru zo zdrojového kódu do objektového kódu spravíme pomocou:

```
avr-gcc -mmcu=atmega16 Os ledblink.c o~ledblink.o
```

Vytvorenie .hex súboru urobíme príkazom

```
avr-objcopy -j .text -j .data -O ihex  ledblink.o  ledblink.hex
```

Na naprogramovanie mikrokontroléra Atmega16A pomocou USBasp a avrdude použijeme príkaz

```
avrdude -p m16 -c usbasp -e -u flash:w:ledblink.hex
```

avrdude môžeme použiť na čítanie/zápis/kontrolu poistiek mikrokontroléra, EEPROM a FLASH pamäti.

9 Využitie mikrokontrolérov a hotové projekty

Mikrokontroléry môžeme využiť na rozličné činnosti. Prvá úloha, ktorá býva pomocou mikrokontrolérov realizovaná je blikajúca LED. Potom to môžu byť rôzne svetelné efekty, ovládanie tlačítok a maticovej klávesnice, čítanie hodnot z A/Č prevodníka, riadenie PWM a mnohé ďalšie. Na internete nájdeme aj množstvo projektov realizovaných pomocou AVR mikrokontrolérov a to napríklad projek LCD2USB alebo aj USBasp, ktoré ako svoje jadro používajú Atmega8A mikrokontrolér.

Zaujímavé projekty nájdeme na stránke Objective Developemet `http://www.obdev.at/products/vusb/index.html`. Táto firma je autorom virtuálneho USB portu V-USB (firmvér), ktorý slúži na priame pripojenie USB bez iných obvodov.

Záver

Cieľom tohto článku bolo ukázať možnosti FOSS pri vývoji aplikácií pre cieľovú platformu Atmel AVR. Pri vývoji bolo ukázané použitie kompilátora gcc-avr, príkazu avr-copy z balíčka avr-binutils a programovacieho softvéru avrdude v spojení s programovacím hardvérom USBasp a Elm-Chan programátorom na paralélny port.

Tento článok bol napísaný až teraz, no myšlienky naň dozrievali asi päť rokov. Sú za nim skryté hodiny práce a to čítanie dokumetácie, zostavovanie programovacieho hardvéru a testovanie jednotlivých FOSS nástrojov, vhodných na vývoj pre platformou Atmel AVR tak, aby bol vývojový cyklus uzavretý.

Literatúra

[1] Documentation:AVR GCC/AVR GCC Tool Collection
`http://www.avrfreaks.net/wiki/index.php/Documentation:AVR_GCC/AVR_GCC_Tool_Collection`

[2] AVR Libc user manual
http://savannah.nongnu.org/download/avr-libc/avr-libc-user-manual-1.6.5.pdf.bz2

[3] Elm-Chan ISP LPT programmer
http://elm-chan.org/works/avrx/avrx_lpt.png

[4] USBASP - Sériový programátor procesorov Atmel AVR na USB port
http://www.pavolmaria.org/index.php?id=elektronika/avrspusbasp

[5] AVR Microcontrollers in Linux HOWTO
http://tldp.org/HOWTO/Avr-Microcontrollers-in-Linux-Howto/x207.html

Kontaktná adresa

Pavol LAJČIAK (Ing.),
Katedra informatiky PF KU v Ružomberku, Hrabovská cesta 1,
034 01 Ružomberok, pavol.lajciak@pavolmaria.org

VYUŽITÍ MATEMATICKÉHO PROGRAMU SAGE V ANALÝZE REDISTRIBUČNÍCH SYSTÉMŮ

MUŽÍKOVÁ, Karolína, (SK)

Abstrakt. Matematický program Sage je volně dostupný software, který je určený pro matematické výpočty a grafická znázornění. Pomocí grafického rozhraní v internetovém prohlížeči je nabízeno interaktivní grafické uživatelské prostředí, ve kterém lze s využitím programovacího jazyka Python řešit řadu matematických úloh. Hlavním námětem této práce je rozbor fungování a aplikace programu Sage při analýze redistribučních systémů. Teorie redistribučních systémů spadá pod oblast teorie her a zkoumá rozdělování výplat v systémech s více hráči. Práce je zaměřena na rozbor redistribuce výplat v systému se třemi hráči, což lze s využitím programu Sage přehledně znázornit v trojrozměrném prostoru.

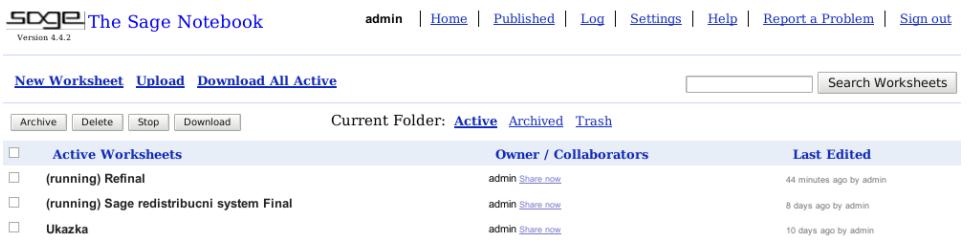
1 Úvod

Tento článek seznamuje s matematickým softwarem Sage, který je volně šiřitelný pod všeobecnou veřejnou licencí GNU (GNU General Public License). Vývoj tohoto open source programu probíhá od roku 2005. Jde o software, který lze výhodně využít při matematických výpočtech a jejich grafickém znázornění. Představuje tak konkurenci drahým komerčním programům jako je například Matlab, Maple a Mathematica [2]. V současné době je k dispozici pro většinu platforem včetně Linux nebo Microsoft Windows. S programem Sage lze pracovat buď v textovém okně terminálu nebo používat interaktivní grafické rozhraní zobrazené v internetovém prohlížeči. Program Sage je tvořený v programovacím jazyce Python. Jde o dynamický, interaktivní, objektově-orientovaný programovací jazyk, který byl prvně publikován v roce 1991. Python je vyvíjen jako open-source projekt, stejně tak jako samotný Sage [4].

2 Základy práce v matematickém programu Sage

Při zpracovávání tohoto článku bylo používáno grafické rozhraní programu Sage zobrazené v internetovém prohlížeči (příkazem `sage -notebook` v okně terminálu). Použita byla

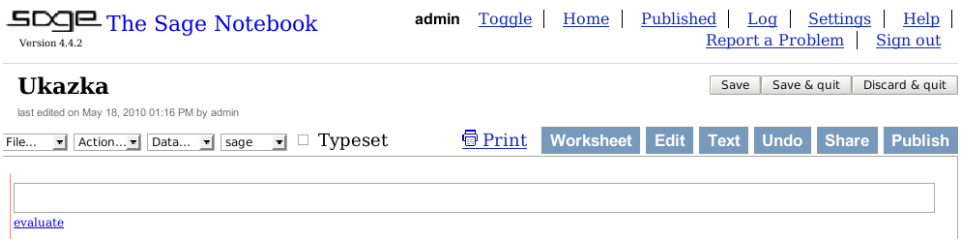
verze 4.4.2. Úvodní uživatelská nabídka po přihlášení do programu je ukázána na Obr. 1. Úvodní strana nabízí možnost výběru existujících pracovních sešitů (Active Worksheets)



Obrázek 1: Úvodní strana programu Sage

nebo založení nového sešitu (New Worksheet). Celková nabídka je uživatelsky velmi přehledná a nabízí standardní volby (ukládání a nahrávání souborů, odstraňování souborů, přejmenování názvů souborů, převod souborů do textové podoby, možnosti sdílení souborů, aj.).

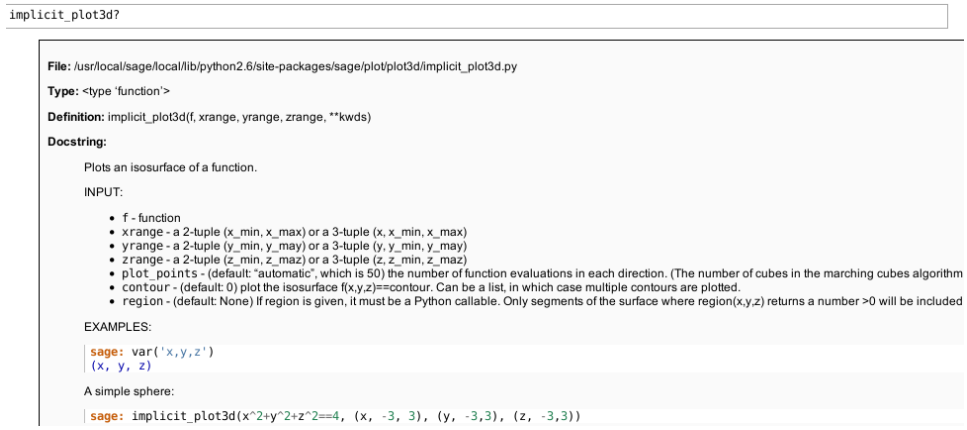
V pracovním sešitě (Obr. 2) se příkazy zadávají do jednotlivých fialových obdélníkových polí. V každé buňce může být i několik příkazů zároveň. Je možné psát každý nový příkaz na nový řádek dané buňky nebo jednotlivé příkazy na jednom řádku oddělovat středníkem. Hotový příkaz se spustí volbou evaluate nebo kombinací kláves `Shift + Enter`. Tím se pod polem s příkazem zobrazí buď výsledek konkrétního výpočtu nebo se vygeneruje obrázek. Pro ukládání se používá příkaz Save v pravé horní liště, případně příkaz `Save & quit`, který



Obrázek 2: Nový pracovní sešit

konkrétní pracovní sešit uloží a uzavře. Při ukončení práce je nutné odhlášení z programu Sage (Sign out) a následné použití kláves `Ctrl + C` v terminálu.

Program Sage nabízí velmi podrobnou nápovědu, a to v různých podobách. Uživatelsky nejvhodnější a nejpřehlednější je Sage Tutorial a Sage´s Reference Manual, které obsahují všeobecné informace o jednotlivých funkcích a velké množství příkladů [6]. V samotném programu lze využít znak otazníku umístěný za nějaký název objektu, který vypíše o daném objektu základní informace, včetně informace o typu funkce, její syntaxi a uvádí příklad použití. Příklad fungování této nápovědy je ukázán na Obr. 3 pro funkci `implicit_plot3d`, která byla v analýze redistribučního systému také použita.



Obrázek 3: Nápověda k funkci `implicit_plot3d`

3 Teorie redistribučních systémů

Teorie redistribučních systémů je aplikací a rozšířením oblasti teorie her. Redistribuční systém je takový systém, ve kterém dochází k nějakému přerozdělení odměn oproti výkonům, které podali členové tohoto systému. Pokud dochází v systému k redistribuci prostředků mezi členy, kteří tento systém vytvářejí, a tato redistribuce neodpovídá výkonům, které jednotliví členové vykonali, snižuje se tím výkon celého systému. Výzkumu této oblasti se velmi intenzivně věnuje vědecký kolektiv na Vysoké škole finanční a správní v Praze [1].

Model elementárního redistribučního systému, jak ho navrhuje Valenčík a Budinský [1], zahrnuje pouze tři hráče A, B, C. Každý z těchto hráčů má stejný vliv na daný systém, libovolní dva hráči mohou vytvořit koalici, všechny koalice jsou možné a rovnoprávné a všichni hráči jsou informováni o tom, jaká je jejich výkonnost. Výkonnost hráčů A, B, C je ve stejném pořadí oceněna hodnotami 6 :4:2 (jednotka není pro účel zkoumání redistribučního systému exaktně daná, může jít např. o tisíce peněžních jednotek).

Tento základní model popisuje redistribuční rovnice, která vyjadřuje snížení efektivnosti systému v důsledku odchylky výplat od výkonnosti. Rovnice popisující všechny možnosti rozdělení výplat je

$$x+y+z = 12-\eta\sqrt{(x-6)^2+(y-4)^2+(z-2)^2} \tag{1}$$

kde $x+y+z$ představuje součet skutečných výplat, hodnota 12 vyjadřuje maximální možnou výplatu $(6+4+2)$, koeficient η je koeficient snížení výkonnosti systému (používá se hodnota $\eta = 0.5$; pokud je $\eta = 0$, tak redistribucí nedochází ke snížení efektivnosti).

Výraz pod druhou odmocninou představuje funkci prostorové vzdálenosti rozdělení skutečných výplat od výplat podle výkonnosti hráčů.

Uvedenou elementární redistribuční rovnici lze znázornit v trojrozměrném prostoru. Vznikne tím redistribuční plocha. Na této ploše leží dva významné body, a to bod V rozdělení

výplat podle výkonu (s hodnotou souřadnic [6, 4, 2]) a tzv. rovnostářský bod R (s hodnotou souřadnic po zaokrouhlení [3,51; 3,51; 3,51]), který popisuje takové rozdělení výher, kdy všichni tři hráči dostanou stejnou odměnu.

Model elementárního redistribučního systému umožňuje popis různých typů vyjednávání a výsledky tohoto vyjednávání zobrazovat jako vyjednávací trajektorie na redistribuční ploše.

Z pohledu teorie her je výše popsaná situace v redistribučním systému hrou s nekonstantním součtem tří inteligentních hráčů, ve kterých lze vytvářet koalice a je možné hrát hru opakovaně.

4 Použití programu Sage v analýze redistribučního systému

Pro zobrazení redistribuční plochy, její analýzu a výpočet významných bodů redistribučního systému byl použit matematický program Sage. Sage znázorňuje obrázky programem Jmol [3], který je v programovacím jazyce Java, a díky tomu lze pomocí myši obrázky různě otáčet a přibližovat, a tím lépe analyzovat celý objekt.

4.1 Znázornění elementární redistribuční plochy

Pro znázornění plochy v trojrozměrném prostoru se obvykle používá funkce `plot3d`. To lze využít pouze tehdy, jestliže je hodnota z funkcí dvou proměnných x a y, tedy $z = f(x,y)$. Tuto podmínku bohužel nesplňuje rovnice elementárního redistribučního systému, protože jde o funkci tří proměnných x,y,z. V důsledku toho se pro znázornění redistribuční plochy musí použít funkce `implicit_plot3d`, která zobrazuje rovnici ve tvaru $f(x,y,z) = 0$ [5]. Povinnými vstupními argumenty příkazu je samotná znázorňovaná funkce a také rozměr os x,y,z. Rozměr os lze zapsat buď trojicí údajů `(x, xmin, xmax)` nebo postačuje uvedení minimální a maximální hodnoty na konkrétní ose např. `(xmin, xmax)`. Dále je možné doplnit další požadavky na podobu zobrazovaného objektu, a to např. barvu, zobrazení os, velikost a průhlednost objektu.

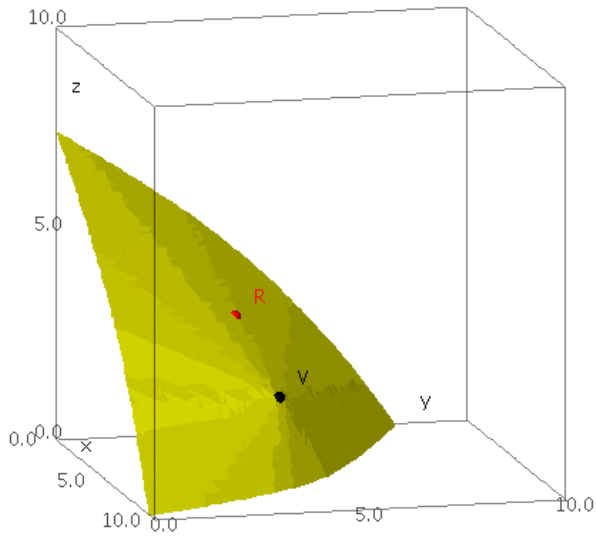
Zobrazení popisků os v 3D prostoru ovšem není úplně nejsnazší. První možností, která je uvedena i v následujím příkladě je, že se popisky tří os znázorní jako textová pole, ovšem každé toto textové pole se musí přesně umístit v souřadnicovém systému. Tento způsob není příliš praktický, protože při jakékoliv změně v obrázku se mohou popisky skrýt a znova se musí hledat souřadnice, ve kterých je popisek osy vidět. Druhou možností je po vygenerování obrázku použít pravé tlačítko myši a zvolit `Style/Axes`. Tím se uprostřed nákresu barevně zobrazí tři pomocné osy s popiskem. Přestože lze měnit barvu tohoto pomocného útvaru, tak velmi snižuje přehlednost celého obrázku.

Pro zobrazení popisků bodů v souřadnicovém systému jsem také používala popisek typu textové pole, pro jehož umístění se musí rovněž určit souřadnice. Na Obr. 5 jsou kromě samotné redistribuční plochy a popisků os znázorněny i dva významné body. Bod označený písmenem V je bod rozdělení výplat podle výkonnosti, kdy každý účastník systému získá přesně tolik, kolik odpovídá jeho uskutečněnému výkonu. V tomto bodě je součet výplat

```
# Zakladni redistribucni plocha, rucni popisky os, bod V (deleni vykonove), bod R (deleni rovnostarske)
x, y, z = var ('x, y, z')
f = x + y + z - 12 + 0.5 * sqrt((x-6)^2 + (y-4)^2 + (z-2)^2)
osax = text3d('x',(1, 0.5, 0)); osay = text3d('y',(0, 9, 0.5)); osaz = text3d('z',(0, 0.5, 8.5))
V = (6, 4, 2); R = (3.51, 3.51, 3.51)
implicit_plot3d (f, (0,10), (0,10), (0,10), color = 'yellow') + osax + osay + osaz + point3d (V, size = 8, color =
'black') + point3d (R, size = 8, color = 'red') + text3d ('V', (6.5, 4.5, 2.5), color = 'black') + text3d ('R', (4,
4, 4), color = 'red')
```

Obrázek 4: Příkaz pro znázornění redistribuční plochy, popisků os, bodů V a R

všech hráčů největší. Čím dále od tohoto bodu, tím více hodnota součtu výplat klesá. Bod označený písmenem R je bod rovnostářského rozdělení výplat, kdy všichni účastníci daného systému získají stejnou výplatu. Těmito dvěma body musí procházet každá redistribuční plocha.



Obrázek 5: Redistribuční plocha

4.2 Výpočet důležitých bodů redistribuční plochy

Elementární rovnice redistribuční plochy obsahuje odmocninu, což představuje významný problém při matematických výpočtech v aplikaci Sage. Dosud jsem neobjevila jinou možnost řešení, než si rovnici pro konkrétní příklad umocnit a až potom zadávat upravený tvar rovnice. Tento způsob není vůbec praktický a může s sebou nést i chyby vzniklé v průběhu ručního výpočtu, včetně případné chyby zaokrouhlovací.

Celkem jsem analyzovala a hledala souřadnice následujících bodů redistribuční plochy:

- bodu R, který popisuje rovnostářské rozdělení výplat v systému,

- tří průsečíků redistribuční plochy s osami souřadnic x, y, z (body Px, Py, Pz),
- tří důležitých bodů ležících v jednotlivých souřadnicových plochách (body X, Y, Z).

Analýza bodu R, který popisuje rovnostářské rozdělení výplat v systému

Tento bod popisuje takové rozdělení, kde všichni tři účastníci dostanou stejnou výplatu $(x = y = z)$. Redistribuční rovnice se proto vyjádří jako funkce jediné proměnné x. Na Obr. 6 je vidět, že si Sage neporadil s vyřešením rovnice obsahující odmocninu. Tvar rovnice pouze částečně upravil, ale ponechal ve výrazu druhou odmocninu a také neznámou proměnnou x. Následně jsem již zadávala umocněný a upravený formát rovnice. Umocněním se bohužel vždy získá druhý kořen rovnice, který z hlediska jeho interpretace ve vztahu ke zkoumanému objektu není správný. Správným kořenem rovnice je přibližná hodnota $x = 3,51$. Souřadnice bodu R jsou tak $[3,51; 3,51; 3,51]$. Druhý kořen rovnice není správný, protože suma souřadnic by převyšovala hodnotu 12, která je maximální možná, a kořen by nevyhovoval původní redistribuční rovnici.

```
# SOURADNICE ROVNOSTARSKEHO BODU (x = y = z)
# Nevyreseni rovnice s odmocninou
x = var('x')
g = -3*x + 12 - 0.5*sqrt((x - 6)**2 + (x - 4)**2 + (x - 2)**2)
solve(g,x)
```

```
    [x == -1/6*sqrt(3*x^2 - 24*x + 56) + 4]
```

```
# Uprava puvodni rovnice
x = var('x')
g = 8.25*x^2 - 66*x + 130
s = solve(g, x, solution_dict = True)
x1, x2 = float(s[0][x]), float(s[1][x])
print x1, x2
```

```
    3.50763403608 4.49236596392
```

```
# Overeni vypoctu dosazenim do puvodni rovnice
x = var('x')
g = -3*x + 12 - 0.5*sqrt((x - 6)**2 + (x - 4)**2 + (x - 2)**2)
g.substitute(x = x1); g.substitute(x = x2)
```

```
    -8.88178419700125e-16
    -2.95419578350399
```

Obrázek 6: Příkaz pro zjištění souřadnic rovnostářského bodu

Analýza průsečíků redistribuční plochy s osami souřadnic

Pro průsečík redistribuční plochy s osou z (bod Pz) platí, že ostatní souřadnice tohoto průsečíku jsou nulové $(x = y = 0)$. Tyto hodnoty se dosadí do rovnice redistribuční plochy a opět tím vznikne rovnice o jedné neznámé. Rovnice se musela nejprve umocnit a až potom vyřešit programem Sage. Umocněním rovnice vzniknou opět dva kořeny řešení, ale jeden z kořenů nevyhovuje původní elementární redistribuční rovnici, což lze v programu Sage ověřit. Stejný postup výpočtu byl použit i pro zbylé dva průsečíky Px a Py.

```
# PRUSECIK PLOCHY S OSOU z (x = y = 0)
# Sage nevyresi rovnici s odmocninou
z = var('z')
g = -z + 12 - 0.5*sqrt((6)**2 + (4)**2 + (z- 2)**2)
solve(g,z)
```

```
    [z == -1/2*sqrt(z^2 - 4*z + 56) + 12]
```

```
# Uprava puvodni rovnice
z = var('z')
g = 0.75*z^2 - 23*z + 130
s = solve(g, z, solution_dict = True)
Pz1, Pz2 = float(s[0][z]), float(s[1][z])
print Pz1, Pz2
```

```
    7.47344925163 23.193217415
```

```
# Overeni
z = var('z')
g = -z + 12 - 0.5*sqrt((6)**2 + (4)**2 + (z- 2)**2)
g.substitute(z = Pz1); g.substitute(z = Pz2)
```

```
    0.000000000000000
    -22.3864348300688
```

```
Pz = (0, 0, Pz1)
```

Obrázek 7: Příkaz pro výpočet průsečíku redistribuční plochy s osou z

Analýza důležitých bodů ležících v jednotlivých souřadnicových plochách

Jde o velmi významnou skupinu bodů, které jsou důležité při analýze vyjednávání v redistribučních systémech. Na Obr. 8 je uveden příkaz pro výpočet souřadnic bodu Z, který leží v rovině xy, a tak je jeho z-ová souřadnice nulová.

Proměnná y se vyjádří pomocí proměnné x (např. z rovnice $y = ax$ a dosazením souřadnic bodu $V = [6,4,2]; 4 = 6a; a = 2/3; y = 2/3x$).

Elementární redistribuční rovnice se vyjádří jako rovnice jediné proměnné x. Vypočítané hodnoty `x1, x2` a `y1, y2` se dosadí do původní elementární redistribuční rovnice, a tím se zjistí správné kořeny dané rovnice, které jsou potom hledanými souřadnicemi bodu Z. Všechny tyto vypočtené důležité body lze znázornit na redistribuční ploše (příkaz pro

```
# Vypocet souradnic bodu Z v plose XY, z = 0, y = 2/3x
x = var('x')
g = (5/3*x - 12)**2 - 0.25*((x-6)**2+(2/3*x-4)**2+4)
s = solve(g, x, solution_dict = True)
x1, x2 = float(s[0][x]), float(s[1][x])
print x1, x2
```

```
    6.5662188881 8.19240180156
```

```
y1 = 2/3 * x1; y2 = 2/3 * x2; print y1, y2
```

```
    4.37747925873 5.46160120104
```

```
# Zpetne dosazeni x1 a y1 do rovnice redistribucni plochy (lepsi vysledek)
r = 12 - x1 - y1 - 0.5*sqrt((x1 - 6)**2 + (y1 - 4)**2 + (0 - 2)**2); print r
```

```
    2.44249065417534e-15
```

```
# Zpetne dosazeni x1 a y1 do rovnice redistribucni plochy (horsi vysledek)
r = 12 - x2 - y2 - 0.5*sqrt((x2 - 6)**2 + (y2 - 4)**2 + (0 - 2)**2); print r
```

```
    -3.30800600518427
```

```
# SOURADNICE bodu Z v plose XY (z = 0)
Z = (x1, y1, 0); print Z
```

```
    (6.5662188880998889, 4.3774792587332589, 0)
```

Obrázek 8: Příkaz pro výpočet souřadnic bodu Z, ležícího v rovině xy

zobrazení Obr. 9, znázornění Obr. 10). Průsečíky s osami souřadnic jsou zobrazeny modrou barvou, důležité body v jednotlivých souřadnicových plochách zelenou barvou.

```
print Px, Py, Pz
```

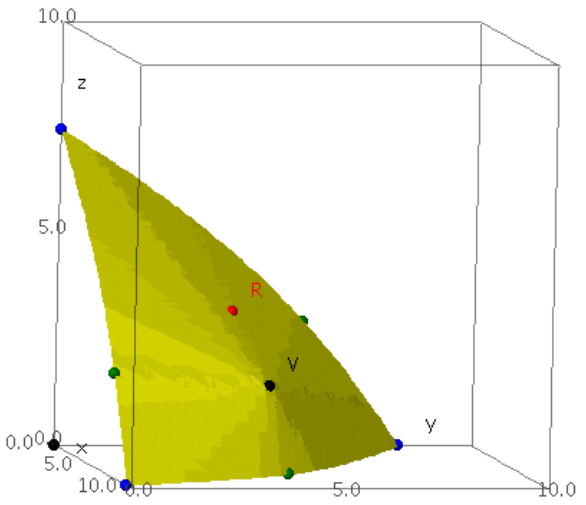
```
    (9.2390477143047676, 0, 0) (0, 8.2030935234448936, 0) (0, 0,
    7.4734492516322693)
```

```
print X, Y, Z
```

```
    (0, 5.8807373824995413, 2.9403686912497706) (7.4010311744816315, 0,
    2.467010391493877) (6.5662188880998889, 4.3774792587332589, 0)
```

```
# Znazorneni dulezitych bodu, pruseciku s osami (Px, Py, Pz), pocatku P, bodu V a R, pruseciku v souradnicovych
plochach (X, Y, Z)
x, y, z = var ('x, y, z')
f = x + y + z - 12 + 0.5 * sqrt((x-6)^2 + (y-4)^2 + (z-2)^2)
osax = text3d('x',(1, 0.5, 0)); osay = text3d('y',(0, 9, 0.5)); osaz = text3d('z',(0, 0.5, 8.5))
P = (0, 0, 0); V = (6, 4, 2); R = (3.51, 3.51, 3.51)

implicit_plot3d (f, (0,10), (0,10), (0,10), color = 'yellow') + osax + osay + osaz + point3d (V, size = 8, color =
'black') + text3d ('V', (6.5, 4.5, 2.5), color = 'black') + point3d (R, size = 8, color = 'red') + text3d ('R', (4,
4, 4), color = 'red') + point3d (P, size = 8, color = "black") + point3d (Px, size = 8, color = "blue")+ point3d
(Py, size = 8, color = "blue") + point3d (Pz, size = 8, color = "blue") + point3d (X, size = 8, color = "green") +
point3d (Y, size = 8, color = "green") + point3d (Z, size = 8, color = "green")
```

Obrázek 9: Příkaz pro znázornění redistribuční plochy a důležitých bodů



Obrázek 10: Redistribuční plocha a důležité body

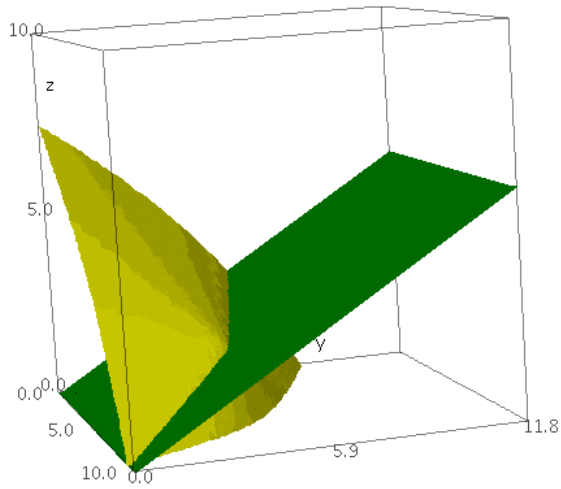
4.3 Znázornění pomocných vyjednávacích rovin

Při analýze redistribučních systémů je velmi důležité sledovat různé typy vyjednávání jednotlivých hráčů. Na redistribuční ploše lze jednotlivé vyjednávací trajektorie znázornit. Základní jsou tři vyjednávací trajektorie při dohodě dvou hráčů o rozdělení svých odměn podle jejich výkonu. V této situaci si dva hráči dělí výplatu v poměru ke svým výkonům

```
# Znazorneni jedne ze tri pomocnych rovin
x, y, z = var ('x, y, z')
f = x + y + z - 12 + 0.5 * sqrt((x-6)^2 + (y-4)^2 + (z-2)^2)
X1 = (0, 2*5.88073738249954, 2*2.94036869124977)
X2 = (10, 2*5.88073738249954, 2*2.94036869124977)
P = (0, 0, 0)
Pxx = (10, 0, 0)
osax = text3d('x',(1, 0.5, 0)); osay = text3d('y',(0, 9, 0.5)); osaz = text3d('z',(0, 0.5, 8.5))
implicit_plot3d (f, (0,10), (0,10), (0,10), color = 'yellow') + polygon3d ([X2, X1, P, Pxx], color = "green") + osax
+ osay + osaz
```

Obrázek 11: Příkaz pro znázornění redistribuční plochy a jedné z pomocných rovin

a třetímu zůstane jen tolik, kolik mu ostatní dva ponechají. Trajektorie tohoto vyjednávání se protínají v jednom bodě, a to v bodě V se souřadnicemi [6, 4, 2]. Trajektorie začínají v bodech X, Y, Z, které byly vypočteny, a končí v průsečících redistribuční plochy s osami souřadnic, tedy v bodech Px, Py, Pz. Pomocná rovina byla znázorněna s využitím funkce



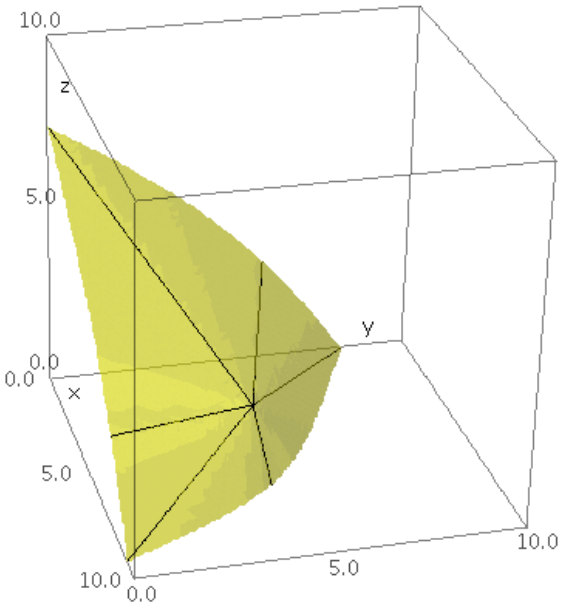
Obrázek 12: Redistribuční plocha a jedna z pomocných rovin

`polygon3d`. Základním argumentem této funkce jsou body, které se mají spojit do jednoho útvaru. Při zpracování této části jsem zjistila, že podoba výsledného útvaru záleží na pořadí bodů v zadání příkazu, což může být významná komplikace při řešení matematických úloh. Vyjednávací trajektorii tvoří společné body této pomocné roviny a redistribuční plochy.

Všechny tři pomocné roviny rozdělí redistribuční plochu na šest dílčích oblastí. Každá pomocná rovina prochází bodem V, určujícím rozdělení podle výkonu, a počátkem souřadnicového systému. Vyjednávací trajektorie popisující situaci dělení podle výkonu jsou celkem tři. Každou z trajektorií lze získat jako průsečnici redistribuční plochy a pomocné roviny tak, jak je ukázáno na Obr. 14.

```
# Vyjednavaci trajektorie
x, y, z = var("x y z")
f = x + y + z - 12 + 0.5 * sqrt((x - 6)**2 + (y - 4)**2 + (z - 2)**2)
osax = text3d('x',(1, 0.5, 0)); osay = text3d('y',(0, 9, 0.5)); osaz = text3d('z',(0, 0.5, 8.5))
p = implicit_plot3d (f, (0,10), (0,10), (0,10), color = 'yellow', opacity = 0.6)
l11 = line3d ([(6.56621888809989, 4.37747925873326, 0), (6,4,2)], color = "black")
l12 = line3d ([(6,4,2), (0,0,7.47)], color = "black")
l21 = line3d ([[7.40103117448163, 0, 2.46701039149388], [6,4,2]], color = "black")
l22 = line3d ([[6,4,2], [0,8.2,0]], color = "black")
l31 = line3d ([[0, 5.88073738249954, 2.940368691249777], [6,4,2]],color = "black")
l32 = line3d ([[6,4,2], [9.24,0,0]],color = "black")
show (p + l11 + l12 + l21 + l22 + l31 + l32 + osax + osay + osaz)
```

Obrázek 13: Příkaz pro zobrazení vyjednávacích trajektorií



Obrázek 14: Vyjednávací trajektorie na redistribuční ploše

Závěr

Matematický program Sage se ukázal být velmi vhodným softwarem pro grafické znázornění redistribučních systémů. Práce v grafickém rozhraní je uživatelsky velmi příjemná. Syntaxe příkazů je logická. Realizace některých matematických operací a popisků grafických objektů nebyla uživatelsky úplně jednoduchá, ale to může být i tím, že jsem doposud možná neobjevila všechny dostupné funkce a možnosti programu. Rovněž se na vývoji programu Sage neustále pracuje, a tak je pravděpodobné, že se systém stane ještě dokonalejším.

V případě redistribučních systémů je tato práce pouhým úvodem k jejich rozboru. S využitím programu Sage je vhodné zkoumat různé typy vyjednávacích trajektorií, vliv tvorby koalic mezi účastníky na možnost ovlivnění redistribuce a hledání rovnovážných situací.

Literatura

[1] BUDINSKÝ, P. – VALENČÍK, R.: Teorie redistribučních systémů. In: Politická ekonomie, roč. 13, 2009, č. 5, s. 644 – 659

[2] HAUPT, M.: Systém počítačové algebry Sage. Bakalářská práce. Brno, 2009. Dostupné na internetu [cit. 2010-06-08]: http://www.math.muni.cz/~{}plch/diplomky/Sage.pdf

[3] Jmol. Dostupné na internetu [cit. 2010-06-10]: http://jmol.sourceforge.net/

[4] PYTHON – programovací jazyk. Dostupné na internetu [cit. 2010-06-08]: http://sk.wikipedia.org/wiki/Python_%28programovac%C3%AD_jazyk%29

[5] SAGE. Oficiální stránka. Dostupné na internetu [cit. 2010-06-08]: http://sagemath.org/

[6] SAGE Reference Manual, 3D Graphics. Dostupné na internetu [cit. 2010-06-08]: http://www.sagemath.org/doc/reference/plot3d.html

Kontaktní adresa

Karolína MUŽÍKOVÁ (Ing.),
Katedra manažérskych teórií FRI ŽU v Žiline,
Univerzitná 8215/1, 010 26 Žilina,
Karolina.Muzikova@fri.uniza.sk

PREDSTAVENIE PROJEKTU FREEMAP.SK A JEHO SLUŽIEB

PÁLENIK, Michal, (SK), JAMEČNÝ, Tibor, (SK)

Abstrakt. `Freemap.sk` je slovenskou časťou projektu `openstreetmap.org`. V prvej časti článku sa budeme venovať medzinárodnému projektu openstreetmap, najmä s dôrazom na licenciu pod ktorou funguje. Na príkladoch konkurenčných služieb a licenčných bariér pri ich používaní si ukážeme dôvody jeho vzniku. V druhej časti článku si predstavíme zaujímavé služby a nástroje portálu `Freemap.sk`. Z pohľadu pohybu po prírode sú zaujímavé vrstvy turistických chodníkov, cyklochodníkov a náučných chodníkov, ktoré v spojitosti s vrstevnicami a reliéfom ukazujú realitu v teréne. Z pohľadu vyučovacieho procesu predstvíme vrstvu wikipédiu (ktorá ukazuje priestorovo rozmiestnené dôležité objekty), vrstvu vysvetlení názvov (ktorá prehľadne ukazuje prečo sa daný objekt volá práve tak). Naznačíme ako pridávať nové objekty do týchto vrstiev. Ďalšou zaujímavou vrstvou sú historické mapy, ktoré zobrazia historické mapy nad súčasnými. Z ostatných služieb si predstavíme najmä ako vložiť mapu na svoju stránku a ako jednoducho zobraziť viacero bodov s popisom. Z ďalších služieb si predstavíme možnosť tvorby geosprievodcu, historického sprievodcu a možnosť hlásenia polohy.

1 Úvod

Projekt OpenStreetMap vznikol ako odpoveď na neexistenciu voľne dostupných konzistentných geografických dát. Licencia týchto dát umožňuje akékoľvek, aj nepredpokladateľné a inovatívne použitie (rešpektujúc určité pravidlá).

Počas niekoľko rokov svojej existencie sa k samotným dátam podarilo pripojiť veľa zaujímavých služieb, ktoré sprístupňujú pokročilé používanie aj bežným, netechnickým užívateľom. Tieto siahajú od rôznych online služieb, cez tlačené mapy až po aplikácie do mobilných telefónov.

2 Licencie dát

Copyright je množina práv priznaných autorovi originálneho diela, vrátane práva kopírovať, rozširovať a meniť dielo. Tieto práva môžu byť licencované [1]. Slovo „licencia“ znamená povolenie (vydané autorom diela) používať dané dielo (podľa dohodnutých podmienok v licencii) [2]. Väčšina licencií obsahuje len základné práva, ako „právo použitia na súkromné účely bez možnosti zmeny licencovaného diela a bez možnosti ďalšej distribúcie bez písomného súhlasu autora“. Príkladom môže byť článok v internetových novinách, ktorý si môžeme zadarmo prečítať, ale už nemôžeme jeho obsah skopírovať na vlastnú stránku a zmeniť ho.

Samozrejme môžeme osloviť autora článku so žiadosťou o udelenie licencie na zmenu a ďalšie šírenie jeho článku, ten našej žiadosti pravdepodobne vyhovie, ale za menšiu alebo väčšiu úplatu. Dosť rozšírená je aj taká forma licencie, kde môžeme pôvodné dielo šíriť ďalej pri dvoch podmienkach:

1. dielo nebude zmenené,
2. dielo bude použité len na nekomerčné účely.

Ani takáto forma licencie nie je vhodná pre šírenie máp, keďže všetky mapy obsahujú nejaké nepresnosti (chýba spojenie niektorých ciest, prípadne bola dokončená nová cesta). Nahlasovanie takýchto nepresností autorovi mapy trvá buď dlho, alebo je v niektorých prípadoch ignorované úplne. Ako príklad reštriktívnej licencie slúži licencia na Google Maps:

- bez písomného súhlasu nie je možné [3]: kopírovať, prekladať, upravovať alebo odvodiť nové dielo z máp; rozširovať mapy tretím osobám; používať mapy pre navigáciu (napr. v aute),
- za akcepovanie licenčných podmienok sa považuje aj použitie služby (čiže pozretie google mapy) [4],
- Google môže bez oznámenia vopred nielen zmeniť mapy a spôsob ich používania, ale dokonca aj úplne zastaviť poskytovanie máp,
- používať Google mapy je možné len v rámci internetového prehliadača [5], takže ak potrebujeme zobrazovať mapy v nami vytvorenom programe, nemôžeme použiť Google Maps bez porušenia licenčných podmienok.

Na druhej strane projekt OpenStreetMap využíva licenciu Creative Commons Attribution-ShareAlike 2.0 [6], skratka CC-by-SA. Detailnejší popis [7]:

- **A**ttribution (by): dielo je dovolené kopírovať, rozširovať, prezentovať a odvodiť nové dielo len ak je uvedený zdroj dát (OpenStreetMap) aj s odkazom na licenciu (CC-by-SA).

- **S**hareAlike (SA): odvodené nové dielo je možné rozširovať len pod identickou alebo jej podobnou licenciou.
- Licencia neobsahuje klauzulu non-commercial, takže dáta je možné použiť aj na komerčné účely, napríklad predávať tlačené mapy.

V blízkej budúcnosti sa plánuje zmena licencie na ODbl [8]. Zmena licencie by mala ujednotiť a spresniť súčasnú licenciu. Licencia nebola stavaná na použitie pre databázy. Napríklad podmienka citovania pri 250 tisícoch autorov sa ťažko spĺňa, v praxi sa aj tak používal OpenStreetMap ako zdroj dát, a nie individuálni autori. Zásadné body používania ostávajú nezmenené [9].

3 Vrstvy

Väčšina máp, dostupných na interete, má veľmi úzke zameranie. Niektoré slúžia len ako autoatlas, tie vyspelejšie dokážu aj naplánovať trasu (Google Maps), prípadne sú zamerané len na jednu oblasť, ako je napr. turistika (`turistickamapa.sk`). Tie vyspelejšie umožňujú prepínať účel mapy podľa požiadaviek používateľa, napr. na turistickej mape zobraziť aj cyklotrasy (`mapy.cz`). Spôsob zobrazenia máp je daný poskytovateľom mapy a používateľ nemá žiadne, prípadne len limitované možnosti jeho zmeny: napríklad portál `turistickamapa.sk` ponúka možnosť „zobraziť hrady“ na mape, ale neponúka už možnosť zobrazenia náučných chodníkov.

Na zobrazovanie dodatočných informácií na mape sa používajú vrstvy. Takisto ako pri maľovaní prekryje novšia vrstva farby tú staršiu, prekryje na mape vrstva s užšie zameranými informáciami (napr. hrady) vrstvu so všeobecnejšími informáciami (napr. cestná cieť, reliéf). Nanášané vrstvy nenesú okrem užitočnej informácie (napr. ikony hradov) žiadne ďalšie informácie, preto je zvyšok vrstvy priesvitný. Toto umožňuje kombinovať viacero vrstiev na seba, napr.

1. podkladovú vrstvu bude tvoriť cestná sieť
2. na ňu bude nanesená vrstva „vrstevnice“
3. potom sa nanesie vrstva „turistické chodníky“
4. dodatočne sa aplikuje vrstva „náučné chodníky“
5. a nakoniec pribudne vrstva „hrady“.

Proces kombinovania vrstiev sa deje automaticky podľa zvolených vrstviev používateľa. Výsledná mapa vyzerá, ako keby tam boli všetky informácie zakreslené „natvrdo“, čiže vyššie zmienený príklad bude obsahovať rovnaké informácie ako tlačená turistická mapa kúpená v obchode.

Portál `freemap.sk` umožňuje zobraziť mapy v troch kategóriách:

- autoatlas – vrstvy: fotografie, počasie, wikipédia, hrady a adresné body
- turistika – vrstvy: turistika, cyklotrasy, náučné chodníky, vrstevnice, geocaching + rovnaké vrstvy ako autoatlas
- vlastné – tu je možné navoliť si vlastnú kombináciu vyššie spomenutých vrstiev + ďalšie dostupné vrstvy: reliéf, názvy miest a obcí, podklad – cestná sieť, objekty (POI).

V čase pribúdania ďalších vrstiev bude treba prehodnotiť zobrazovanie vrstiev a zaviesť špecializované portály. Už existuje špecializovaná turistická mapa (`http://turistika.freemap.sk`).

Vznik dodatočných vrstiev závisí od množstva zmapovaných objektov pre danú vrstvu a od odozvy používateľov. Keďže sa jedná o slobodnú mapu, môže si každý záujemca vytvoriť vlastnú mapu, záleží len od jeho technických schopností a možností (vlastný server, atď). Pre menej náročných záujemcov ponúka freemap možnosť zobrazenia vlastného výberu objektov na vlastnej stránke cez aplikáciu EmbeddedFreemap [10].

4 Nástroje

OpenStreetMap je často chápaný ako polotovar, ktorý treba dorobiť pre potreby koncového užívateľa. Pre najbežnejšie služby pripravil portál `freemap.sk` viacero nástrojov.

4.1 Wikipédia

`Freemap.sk` úzko spolupracuje s inými slobodnými projektami a wikipédia je jedným z nich. Na mape zobrazujeme rôzne objekty, ktoré sú na slovenskej wikipédii zahrnuté a georeferencované. Ďalšia vrstva zobrazuje, prečo sa daný objekt volá práve tak. Ak je ulica pomenovaná po osobnosti, zobrazí jej profil na wikipédii, ak je námestie pomenované po historickej udalosti, ukáže o nej ďalšie informácie.

4.2 EmbeddedFreemap

EmbeddedFreemap je jednoduchá aplikácia, ktorá umožňuje zobraziť mapu na stránkach používateľa. Jej použitie je veľmi jednoduché, stačí pár kliknutí myšou a zobrazí sa vygenerovaný kód, ktorý treba vložiť do obsahu svojich stránok. Sprievodca umožňuje nastaviť:

- veľkosť mapy – šírka a výška
- konkrétny výrez mapy (napr. celé Slovensko, prípadne iba časť mesta)
- zvýrazniť konkrétne miesto na mape vlajkou (napr. poloha školy, firmy, . . .)

V prípade pokročilejších nastavení (ako napríklad: zobrazenie pobočiek firmy na mape Slovenska) je už nutná malá miera technických zdatností.

4.3 Nahrávanie trás

Ak máme záznam prejdenej trasy z GPS zariadenia, môžeme si ho nechať zobraziť na mape. Takto môžeme záznam vizuálne analyzovať (napr. aké zaujímave pamiatky sa nachádzajú v blízkosti), prípadne môže poslúžiť na vysvetlenie cesty inej osobe (lepšie raz vidieť, ako 100-krát počuť :)).

4.4 Vytlačenie mapy

Podobne ako pri EmbeddedFreemap, stačí si len vybrať oblasť mapy, ktorú chceme mať na papieri, vybrať si vrstvy, a zobrazí sa nám stránka, ktorú rovno vytlačíme. Takto vytlačenú mapu je možné potom legálne kopírovať, rozdávať kamarátom i predávať.

4.5 Vyhľadávanie

Vyhľadávanie miest (Žilina), ulíc (Hurbanova, Žilina), trás a iných objektov (napr. Gerlachovský štít, alebo „Fak. riadenia a informatiky ŽU“). Vyhľadávanie zobrazí výsledky na mape ale umožňuje i použitie jednoduchého API založeného na XML alebo JSON. Vyhľadávanie umožňuje i reverzný geocoding, teda vyhľadanie slovného popisu adresy na zadaných súradniciach.

4.6 Plánovač trás

Plánovač trás vyhľadá najkratšiu alebo najrýchlejšiu cestu medzi dvoma bodmi na mape. Počiatočný a koncový bod trasy je možné zadať ručne (ako pri nástroji „Vyhľadávanie“) alebo jednoduchšie je zvoliť ho pomocou kliknutia myšou na mape. Vyhľadávať je možné pre auto, bicykel alebo osobu idúcu pešo. V budúcnosti plánujeme plánovanie trás aj pre iné skupiny (napr. vozíčkari alebo použijúc MHD).

4.7 Sledovanie polohy

Pre sledovanie aktuálnej polohy nejakého objektu (napr. osoby, auta) je postačujúce mať GPS zariadenie a mobilný telefón s aktivovaným dátovým prenosom. O zbieranie dát, ich uchovávanie, štatistiky a zobrazenie na mape sa postará portál `freemap.sk`. Funkčnosť celej aplikácie bola otestovaná počas dvadsaťdňového pochodu z Dukly do Brezovej pod Bradlom, po červenej turistickej trase s názvom Cesta hrdinov SNP (ktorá je súčasťou medzinárodnej diaľkovej trasy E8).

Sledovanie polohy je možné v jednoduchom automatickom móde (iba zaznamenávanie polohy v určené časové okamihy) alebo i v pokročilom manuálom móde. V manuálnom móde môže užívateľ pridávať komentáre alebo fotky. Má tak k dispozícii referencovaný denník.

4.8 Fotogaléria a turistický sprievodca

Nástroj podobný ako sledovanie polohy je fotogaléria. Užívateľ môže nahrať svoje fotky na server a prepojiť ich s polohou na mape. V spolupráci s Technickou univerzitou v Košiciach sme vyvinuli špeciálnu aplikáciu pre telefóny s operačným systémom Android, ktoré dokážu priamo uploadovať fotky na server `freemap.sk` ako i na vlastný server.

Ďalšou možnosťou vyžitia mapy je pre turistického alebo historického sprievodcu. Na serveri umožňujeme nahrať a referencovať poznámky. Tieto môžu obsahovať turistické informácie, informácie o historických udalostiach alebo akékoľvek iné informácie.

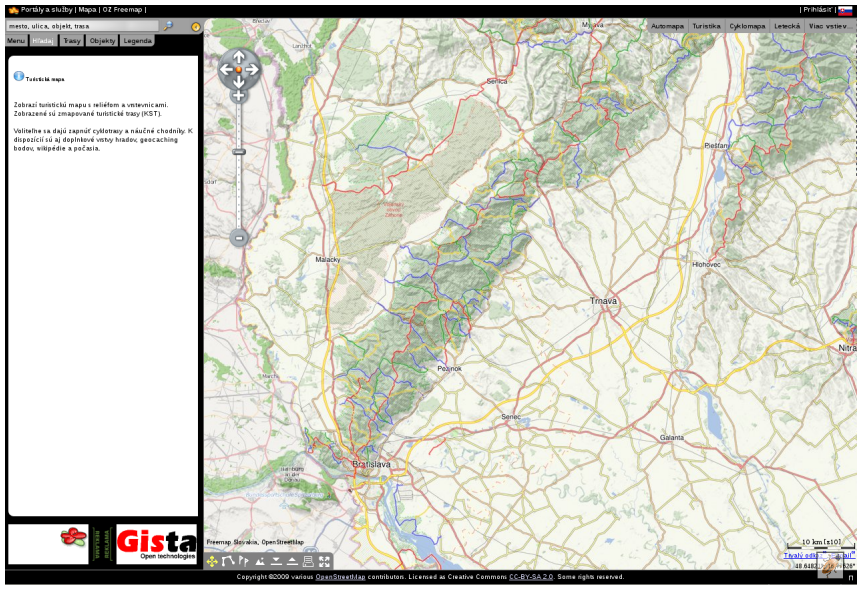
4.9 Mobilné telefóny

V rámci projektu `freemap.sk` ponúkame na stiahnutie aj rôzne aplikácie pre mobilné telefóny [11]. Týchto aplikácií je relatívne veľa, pre rôzne platformy na mobilných telefónoch. Väčšina z nich pracuje v offline móde, teda nepotrebujú prístup na internet. Toto je veľmi dôležité v zahraničí (kde je roamingový internet veľmi drahý) alebo v teréne (kde nie je prístup na mobilnú sieť ani na internet). Niektoré používajú vektorové dáta, ktoré umožňujú vyhľadávanie trás a objektov. Iné sú v rastrovom formáte, ktorý je krajší na bežný pohľad a lepšie zobrazuje vrstevnice.

5 Záver

Projekt `freemap.sk` je dobrovoľnícky projekt založený na open source technológiách a slobodných dátach z projektu OpenStreetMap. Ako iné slobodné projekty, je vytváraný dobrovoľníkmi so zázemím vlastného občianskeho združenia. Podarilo sa nám vytvoriť a sprístupniť nástroje použiteľné aj pre bežných, netechnických užívateľov, ako je vidieť aj na nasledujúcej snímke obrazovky zo stránky portálu.

Keďže sme dobrovoľnícky a otvorený projekt, existuje veľa plánovaných aktivít, nástrojov a služieb [12]. Tieto plánované nástroje sú niekedy recesistické (napríklad mapa pre Austrálčanov s juhom na hornej polovici mapy, ascii art mapa), informačné (polohy vlakov, štatistiky obcí), iné užívateľské rozhrania (napr. WMS), ako i medzinárodne zaujímavé (generovanie renderovacích stylesheetov, lepší import wikipédie). Samostatnou kapitolou je zobrazovanie histórie. V blízkej budúcnosti plánujeme zobrazovať historické mapy ako prekrytie nad súčasnou mapou. Dlhodobejšou aktivitou je zobrazovanie zmien v mape: užívateľ by si mohol vybrať rok, v ktorom chce mapu vidieť. Zobrazilo by to ako mesto v danom čase vyzeralo, ktoré ulice ešte neboli postavené. Keďže sme otvorená komunita, do prác sa môže zapojiť ktokoľvek a doplniť nové nečakané služby a nástroje. Toto hovorí aj motto projektu OpenStreetMap [13]: Projekt začal lebo väčšina máp, o ktorých si myslíte že sú voľne dostupné, majú právnické alebo technické obmedzenia, ktoré zamedzujú ľuďom používať ich tvorivými, produktívnymi alebo neočakávanými spôsobmi.



Literatura

[1] http://en.wikipedia.org/wiki/Copyright
[2] http://en.wikipedia.org/wiki/License
[3] http://www.google.com/intl/en_ALL/help/terms_maps.html
[4] http://www.google.com/accounts/TOS
[5] http://code.google.com/apis/maps/terms.html
[6] http://wiki.openstreetmap.org/wiki/OpenStreetMap_License
[7] http://en.wikipedia.org/wiki/Creative_Commons_licenses
[8] http://wiki.openstreetmap.org/wiki/Open_Database_License
[9] http://www.opendatacommons.org/licenses/odbl/summary/
[10] http://wiki.freemap.sk/EmbeddedFreemap
[11] http://wiki.freemap.sk/MapaDoMobilu
[12] http://wiki.freemap.sk/WishList
[13] http://wiki.openstreetmap.org/wiki/Main_Page

Kontaktná adresa

Michal PÁLENÍK,
Freemap Slovakia, Matičná 8/A5,
900 28 Ivanka pri Dunaji, michal.palenik@freemap.sk

PROTOKOL KERBEROS V – ANALÝZA A KONFIGURÁCIA

PISARČÍK, Peter, (SK)

Abstrakt. Článok ponúka pohľad na centrálny autentifikačný protokol s názvom Kerberos vyvinutom na Massachusetts Institute of Technology, pričom sa hlavne venuje analýze protokolu vo verzii 5. Cieľom článku je aj v krátkosti predstaviť grafické webovo orientované konfiguračné rozhranie pre administráciu distribučného centra kľúčov MIT implementácie Kerbera V.

1 Úvod

Internet je v súčasnosti pre mnohých takmer každodenným „elektronickým chlebom“, ba pre určité skupiny medzinárodnej spoločnosti sa stal neodmysliteľnou súčasťou ich života. Toto konštatovanie zároveň paradoxne koreluje s konštatovaním možného zlyhania celosvetového sieťového média vo forme najrôznejších útokov – útokov ľudského charakteru. Práve z tohto titulu sa v progrese Internetu stále viac venuje pozornosť autentifikačným procesom, na základe ktorých je možné eliminovať negatívny dopad kyberokriminality. Systém Kerberos je silným autentifikačným mechanizmom, ktorý okrem bezpečnej autentifikácie ponúka aj službu jedného prihlásenia (single sign-on). Je teda jedným z vyhovujúcich riešení v procese zabezpečenia siete Internet.

2 Protokol Kerberos V

Pojem Kerberos

Kerberos (grécky $\kappa\varepsilon\rho\beta\varepsilon\rho o\varsigma$, Kérberos, latinsky Cerberus) je meno bytosti z gréckej a rímskej mytológie, ktorá je bežne zobrazovaná v podobe viachlavého psa. Úlohou tejto kreatúry bolo stráženie vstupnej brány do podsvetia, v ktorom vládol boh Hádes so svojou manželkou Perzephoneou.

História

Projekt Kerberos je pokračovaním projektu Athéna, ktorého vývoj a výskum začal v roku 1983. Projekt zastrešovali spoločnosti IBM, Digital Equipment Corporation a MIT. Cieľom projektu bolo vytvorenie počítačového prostredia, ktoré by bolo zložené až z tisíc pracovných staníc s využitím heterogénneho hardvéru; teda cieľom bolo zlepšenie kvality vzdelávania na inštitúte. Výsledkom projektu Athéna bolo vytvorenie mnohých technológií, ktoré sú široko využívané dodnes, ako je napríklad X Window Systém a protokol Kerberos. Keď projekt Athéna skončil v roku 1991 výpočtové prostredie bolo premenované na Athéna systém a je stále využívané mnohými v MIT komunite.

Ako už bolo vyššie uvedené, sieťový protokol Kerberos vznikol na výskumnej pôde projektu Athéna. Aktuálnou verziou protokolu je verzia 5, ktorá bola vytvorená Johnom Kohlom a Cliffordom Neumanom v roku 1993 a je dokumentovaná ako RFC 1510 (novelizovaná verzia RFC 4120).

Základné ciele protokolu Kerberos

1. **Obojstranná autentifikácia.** Nakoľko je protokol Kerberos autentifikačným protokolom, ponúka overenie identity klienta pristupujúceho k určitej službe, avšak okrem centrálnej autentifikácie klientov Kerberos ide ďalej, keď ponúka aj autentifikáciu opačným smerom, teda klientov dokáže uistiť o identite služieb, ku ktorým sa pripájajú.

2. **Ochrana hesiel.** Jedným z častých problémov sieťových protokolov je hrozba odpočúvania autentifikačných údajov. Preto bol Kerberos navrhnutý s ohľadom na ochranu autentifikačných údajov. V protokole Kerberos sa diskrétne heslo nikdy neposiela priamo po sieti, avšak sa špeciálnym spôsobom upravuje a následne využíva ako zdieľaný šifrovací kľúč, ako bude popísané ďalej.

3. **Systém jedného prihlásenia.** Systém jedného prihlásenia je najzaujímavejším bodom návrhu protokolu Kerberos. Vychádza z predpokladu, že užívateľ po prihlásení k svojmu počítaču využíva služby mnohých ďalších počítačov (sieťových systémov). Autentizovať sa pri prístupe ku každej z týchto služieb a ešte prípadne opakovane je prinajmenšom nepohodlné. Samozrejme teraz neberieme do úvahy, že niektoré aplikačné programy, z hľadiska zvýšenia užívateľského komfortu, ponúkajú uloženie hesiel, čo je potenciálne bezpečnostné riziko. Kerberos sa problém snaží vyriešiť prostredníctvom systému jedného prihlásenia – akonáhle sa niekto, alebo niečo, raz autentifikuje voči systému Kerberos, už nie je potrebná žiadna ďalšia autentifikácia pri prístupe k službám, ktoré využívajú systém Kerberos.

Inovácie v protokole Kerberos V

1. **Otvorenosť pre nové kryptografické algoritmy** – dátové štruktúry sú navrhnuté tak,

aby bola možná implementácia akéhokoľvek kryptografického algoritmu, na rozdiel od verzie 4, v ktorej bola dátová štruktúra pevne zviazaná s algoritmom DES. Výhodou oproti verzii 4 je tiež možnosť využitia rozdielnych kryptografických algoritmov v každej z nasledujúcich správ: lístok, odpoveď, relačný kľúč.

2. **Zápis protokolu pomocou technológie ASN.1** (Abstract Syntax Notation One), ASN.1 popisuje reprezentáciu štruktúry dát, kódovanie a dekódovanie dát a ich prenos. Je súborom formálnych pravidiel umožňujúcich popis objektov nezávisle na ich technickej architektúre (viď. piata sekcia RFC 4120 dokumentu).

3. **Spätná kompatibilita s verziou 4**, zabezpečená tzv. 5-to-4 transformátorom lístkov (implementačne známe ako krb524).

4. **Zmena tvaru principálov**[1] na nasledujúcu formu:
   meno_užívateľa/meno_inštancie@názov_kerberovej_domény

5. **eliminovanie dvojitého šifrovania**, ktoré sa vyskytovalo v komunikácii autentifikačného servera (AS) a servera pre prideľovanie lístkov (TGS)

6. **Prepožičiavateľné lístky** (forwardable) – užívateľ môže požiadať o tento typ lístka, ktorý mu umožní jeho prepožičiavanie inému systému, čím sa daný systém stane oprávneným na základe tohto lístka požadovať služby bez nutnosti opätovného zadávania hesla užívateľa; špeciálnym prípadom takéhoto lístka je TGT[2].

7. **Proxy lístky** (proxiable) – sú podobné prepožičiavateľným lístkom v tom, že môžu byť prenesené na iný hostiteľský počítač. Avšak proxy TGT lístok môže byť použitý len na získanie lístka služby, v žiadnom prípade nemôže byť použitý na získanie nového TGT lístka pre vzdialený hostiteľský počítač.

8. **Obnoviteľné lístky** (renewable) – v Kerberos IV. bola životnosť lístkov limitovaná ako ochrana pred odcudzením. Kerberos V prináša dve rôzne schémy životnosti, ktoré kombinujú dlhú životnosť s bezpečnosťou lístkov s krátkou životnosťou. Ak užívateľ požiada o obnovenie lístka, získa lístok so štandardnou životnosťou a obnoviteľnú životnosť. Lístok je tak platný len počas štandardnej životnosti avšak môže byť predložený KDC[3] zo žiadosťou o predĺženie životnosti. KDC môže takúto žiadosť odmietnuť. Ak však je žiadosť schválená KDC vráti iný lístok, na základe ktorého sa užívateľ

[1] Je to reťazec, ktorý slúži k identifikácii užívateľa alebo služby. Každý principál má tvar: `meno/inštancia@REALM`, kde meno je obvykle užívateľské meno, alebo meno služby, REALM udáva meno kerberovej domény a inštancia je nepovinný reťazec, vďaka ktorému môže mať jeden užívateľ viac principálov.

[2] TGT (Ticket Granting Ticket) – je lístkom na lístky, čo znamená, že ide o špeciálny typ lístka, vďaka ktorému je možné získať ďalšie lístky. Až keď klient vlastní TGT lístok, ktorý sa získava od KDC, môže žiadať a získať lístky pre autentifikáciu voči rôznym sieťovým službám.

[3] KDC (Key Distribution Center) je centrum pre distribúciu kľúčov a je tak srdcom Kerbera. Úlohou KDC je manažovanie užívateľských účtov a účtov aplikačných serverov. Zároveň sa stará o prihlasovanie užívateľov do kerberovej domény. Tento pojem (KDC) však v sebe skrýva dve služby a to: autentifikačná služba (AS –

môže ďalej identifikovať. Tento proces môže byť opakovaný až kým obnoviteľnosť lístka úplne neskončí.

9. **Lístky s preddefinovanou životnosťou** – každý lístok môže byť akceptovaný len v čase ktorý je definovaný lístkom. Štandardne sa pri požiadaní o vydanie lístka vydá lístok platný od aktuálneho času s konfiguračne preddefinovanou časovou exspiráciou. V prípade lístkov s preddefinovanou životnosťou ide o určenie začiatku platnosti lístka v budúcnosti. Tento typ lístka sa zvykne využívať napríklad v prípade periodického spúšťania určitých sieťových systémových služieb. Napriek uvedenej výhode, tento typ lístka nie je často v praxi využívaný a niektoré implementácie ako napr. Active Directory od spoločnosti Microsoft túto podporu nezahŕňajú.

10. **Preautentifikácia** – pôvodný protokol Kerberos IV. nebol odolný voči lokálnym útokom hrubou silou a slovníkovým útokom. Tento nedostatok spôsobil, že bolo možné získať kombináciu užívateľského mena a hesla (autentifikačný server KDC vždy odoslal šifrovanú správu, na ktorú bolo možné off-line aplikovať spomínané útoky). K eliminácii týchto typov útokov došlo v protokole Kerberos V zavedením tzv. preautentifikácie. Preautentifikácia vyžaduje, aby žiadateľ dokázal svoju identitu pred tým ako mu KDC vydá TGT lístok.

    V špecifikácii protokolu sa nachádza niekoľko typov preautentifikácií, avšak reálne došlo k implementácii len šifrovaného časového odtlačku (PA-ENC-TIMESTAMP). Preautentifikácia je riadená zo strany KDC špeciálnymi pravidlami. Ak užívateľ požaduje získanie TGT lístka vo forme autentifikácie voči AS, ale KDC vyžaduje preautentifikáciu, zašle KDC žiadateľovi lístka chybovú správu (KRB_ERROR) namiesto štandardnej AS_REP. Táto chybová správa hovorí klientovi, že je nutná preautentifikácia. Klient musí teda vygenerovať požadované autentifikačné dáta a znova odoslať AS_REQ správu spolu s preautentifikačnými dátami. Ak je preautentifikácia akceptovaná, začína štandardná výmena správ protokolu Kerberos.

Činnosť protokolu Kerberos V

Protokol Kerberos pracuje vo viacerých fázach. Súhrne môžeme hovoriť, že ide o tri fázy, pričom každá z fáz sa skladá z otázky a odpovede.

AS_REQ

Ide o počiatočnú požiadavku zo strany klienta adresovanú autentifikačnému serveru, ktorej cieľom je získanie TGT lístka. V tejto fáze klient zasiela svoje autorizačné dáta. Celá táto požiadavka putuje sieťou bez šifrovania a vyzerá nasledovne:

AS_REQ = (PrincipálKlient , PrincipálSlužba , IP, LT, (TS))

Authentication Service) a služba pre výdaj lístkov (TGS – Ticket Granting Service). V niektorých sieťach je viac ako jedno KDC. V tomto prípade hovoríme o KDC hlavnom (master) a KDC podriadených (slave).

PrincipálKlient je principál asociovaný s užívateľom, ktorý sa autentifikuje; PrincipálSlužba je principál asociovaný so službou, o ktorú klient žiada (ide o reťazec kbrbtgt/REAL@REALM); IP_zoznam (IP) je zoznamom IP adries, ktoré určujú hostiteľský počítač, na ktorom je možné využiť získaný lístok; a nakoniec životnosť (LT) určuje maximálny platný čas pre lístok, ktorý sa bude používať. Je potrebné podotknúť, že hoci sa zdá zbytočné pridávať do požiadavky principál služby, keď v konečnom dôsledku je jasné, že pôjde o KRBTGT principál, predsa je potrebné si uvedomiť, že toto miesto môže byť využité na zadanie konkrétnej služby, ktorú jedinú chce užívateľ využiť, a preto žiada autentifikačný server priamo o lístok pre konkrétnu službu. Tým sa preskočí fáza žiadania TGS.

Ďalšia skutočnosť sa týka IP zoznamu, ktorý môže byť aj prázdny. V tomto prípade môže byť získaný lístok využitý na akomkoľvek hostiteľskom počítači. Toto riešenie umožňuje klientom, ktorí sa nachádzajú za NAT bezproblémovo využívať požadované služby; Niekedy klient pridáva do požiadavky aj svoj aktuálny čas – časové razítko (TS), čo je však len z dôvodu, že KDC, ak časové razítko klienta nie je v povolenom rozsahu, môže klienta hneď varovať, že pre využívanie služieb daného systému Kerberos nie je synchronizovaný.

AS_REP

Ide o odpoveď na predchádzajúcu požiadavku, ktorú odosiela autentifikačný server klientovi, potom čo prekontroloval či sa principál klienta a služby nachádzajú v KDC databáze (ak čo i len jeden principál v databáze chýba je klientovi zaslané chybové hlásenie). Autentifikačný server vytvára odpoveď nasledovne:

1. Náhodne vytvorí relačný kľúč, ktorý sa stáva tajným kľúčom zdieľaným klientom a TGS (SKTGS);

2. Vytvorí TGT lístok, ktorý obsahuje klientsky principál a principál služby, ďalej zoznam IP adries (dáta sú preberané z AS_REQ). Pridá tiež dátum a čas ako časové razítko (TS), životnosť (LT) a nakoniec relačný kľúč (session key - SKTGS), čím vznikne nasledujúca konštrukcia:

$$\mathbf{TGT = (PrincipálKlient, krbtgt/REALM@REALM, IP, TS, LT, SKTGS)}$$

3. Autentifikačný server vygeneruje a odošle klientovi odpoveď: TGT lístok, ktorý bol vyššie popísaný, zašifrovaný s použitím tajného kľúča TGS (KTGS); principál_služby (krbtgt/REALM@REALM), časové razítko (TS), životnosť (LT) a relačný kľúč (SKTGS), to všetko šifrované použitím tajného kľúča klienta (využitím funkcie string2key), ktorý žiadal o službu. Teda odpoveď vyzerá:

$$\mathbf{AS\_REP = \{\ PrincipálSlužba, TS, LT, SKTGS\}KKlient\ \{\ TGT\ \}KTGS}$$

Z týchto konštrukcií môže na prvý pohľad vyvstávať otázka duplicity informácií, ale je potrebné si uvedomiť, že tým že sú informácie v TGT šifrované použitím tajného kľúča servera, nie sú čitateľné pre klienta, a preto musia byť zopakované, čo slúži pre verifikáciu spojenia medzi klientom a KDC.

Vo chvíli, keď klientský počítač prijme odpoveď autentifikačného servera vyžiada si od užívateľa heslo. Heslo slúži ako jeden zo vstupných parametrov, na základe ktorých sa vytvorí špeciálny reťazec, ktorý využíva funkcia string2key(), ktorej výstupom je tajný kľúč klienta. Takto sa dešifruje časť správy, ktorú KDC zašifrovalo využitím tajného kľúča klienta. Ak teda užívateľ je skutočne tým, za ktorého sa vydáva a teda zadal správne heslo, dešifrovanie je úspešné a on získava relačný kľúč a TGT, prostredníctvom ktorých môže žiadať o ďalšie služby (lístky).

TGS_REQ

Užívateľ sa v predchádzajúcej fáze úspešne autentifikoval a získal teda TGT lístok. V tejto chvíli vlastní užívateľ lístok (TGT), prostredníctvom ktorého môže žiadať od TGS (KDC) lístky pre rôzne služby, voči ktorým je samozrejme oprávnený. Práve to je druhá fáza, ktorá začína žiadosťou klienta, ktorú odosiela TGS v podobe TGS_REQ, v ktorej žiada o lístok pre konkrétnu službu.

Požiadavka je konštruovaná nasledovne. Najprv sa vytvára tzv. „autentifikátor“ zložený z užívateľského principála a časového razítka (TS) klientského počítača, pričom tieto dva údaje sú šifrované relačným kľúčom (SKTGS), ktorý užívateľ prijal v predchádzajúcej fáze:

$$\textbf{Autentifikátor} = \{\textbf{PrincipálKlient, TS}\}\textbf{SKTGS}$$

Následne sa vytvorí celá požiadavka, ktorá obsahuje: principál služby, o ktorej lístok sa žiada, životnosť (LT) a autentifikátor, ktorý bol popísaný v predchádzajúcom bode, k čomu sa pripája TGT lístok, ktorý je šifrovaný kľúčom TGS (KTGS)

$$\textbf{TGS\_REQ} = (\ \textbf{PrincipálSlužba, LT, Autentifikátor}) \{\ \textbf{TGT}\ \}\textbf{KTGS}$$

TGS_REP

Vo chvíli keď príde požiadavka klienta na TGS, TGS najprv prekontroluje či principál požadovanej služby existuje v KDC databáze. Ak existuje, z databázy sa načíta tajný kľúč, ktorým sa dešifruje TGT, čím sa extrahuje relačný kľúč, ktorý sa využije na dešifrovanie autentifikátora. Pred generovaním odpovede klientovi sa verifikujú nasledujúce podmienky:

- či platnosť TGT neskončila,
- či principál klienta obsiahnutý v autentifikátore koreluje s tým, ktorý sa nachádza v TGT,
- či autentifikátor sa nenachádza vo vyrovnávacej pamäti odpovedí,
- či zoznam IP adries nie je prázdny a v prípade, že nie, či zdrojová IP adresa žiadateľa sa nachádza v tomto zozname.

Vyššieuvedené podmienky potvrdzujú, že TGT skutočne patrí užívateľovi, ktorý vytvoril požiadavku, a teda TGS môže vytvoriť adekvátnu odpoveď. Táto prebieha v nasledujúcich krokoch:

1. Náhodne sa vytvorí relačný kľúč, ktorý bude tajným zdieľaným kľúčom medzi klientom a službou (SKSlužba).

2. Vytvorí sa lístok služby (TSlužba) obsahujúci: principál klienta, principál služby, zoznam IP adries (IP), časové razítko KDC (TS), životnosť (LT) a nakoniec relačný kľúč (SKSlužba)

TSlužba = (PrincipálKlient, PrincipálSlužba, IP, TS, LT, SKSlužba)

3. Vytvorí sa celková správa – odpoveď obsahujúca: lístok služby (ako ukazuje schéma vyššie) šifrovaný využitím tajného kľúča služby (KSlužba) a záznam obsahujúci principál služby, časové razítko (TS), životnosť (LT) a nový relačný kľúč (SKSlužba), všetko zašifrované s použitím relačného kľúča extrahovaného z TGT (SKTGS). Odpoveď teda vyzerá takto:

TGS_REP = { **PrincipálSlužba, TS, LT, SKSlužba**}**SKTGS** { **TSlužba**}**KSlužba**

Klient – žiadateľ príjme takúto odpoveď pričom aplikuje na ňu relačný kľúč, ktorý má uložený vo vyrovnávacej pamäti, a pomocou ktorého dešifruje časť odpovede, ktorá obsahuje nový relačný kľúč, prostredníctvom ktorého bude následne komunikovať s aplikačným serverom služby. Úlohou klienta v tomto kroku je: do svojej vyrovnávacej pamäte, vložiť nový relačný kľúč ako aj lístok služby, ktorý sa použije pri prístupe k danej službe.

AP_REQ

Klient, ktorý získal lístok pre prístup k požadovanej službe (t. j. lístok a príslušný relačný kľúč), sa v tejto fáze kontaktuje s aplikačným serverom pre prístup k požadovanej službe prostredníctvom AP_REQ správy. Táto správa je tvorená ad hoc na rozdiel od predchádzajúcej správy, do ktorej bol zapojený KDC, a teda variuje v závislosti od aplikácie (služby aplikačného servera). Môžeme však uvažovať nad nasledujúcou stratégiou:

1. Klient vytvorí autentifikátor obsahujúci užívateľský principál a časové razítko (TS), to všetko šifrované relačným kľúčom (SKSlužba), ktorý je zdieľaný s aplikačným serverom

Autentifikátor = { **PrincipálKlient , TS** }**SKSlužba**

2. Klient vytvorí správu požiadavky obsahujúcu: lístok služby (TSlužba), ktorý je šifrovaný tajným kľúčom danej služby (KSlužba) a autentifikátor, ktorý bol vytvorený klientom

AP_REQ = Autentifikátor { **TSlužba** }**KSlužba**

Vo chvíli doručenia takejto požiadavky, aplikačný server dešifruje lístok služby s použitím svojho tajného kľúča, na základe čoho získa relačný kľúč pre komunikáciu s klientom, a ktorý súčasne použije na dešifrovanie autentifikátora. Na overenie či žiadateľ je ten, za ktorého sa vydáva a súčasne či má žiadateľ právo pristúpiť k požadovanej službe, aplikačný server verifikuje nasledujúce podmienky:

- či platnosť lístku služby nevypršala,
- či principál klienta obsiahnutý v autentifikátore je totožný s tým, ktorý obsahuje lístok,
- či sa autentifikátor nenachádza vo vyrovnávacej pamäti, prípadne či mu nevypršala platnosť,
- či zoznam IP adries (extrahovaný z lístka) nie je prázdny a v prípade, že nie, či klient – žiadateľ komunikuje z niektorej z IP adries zoznamu.

AP_REP

Poslednou z trojice odpovedí v procese autentifikácie protokolom Kerberos je odpoveď aplikačného servera, v ktorej dosvedčuje klientovi, že je naozaj tým serverom, ktorý klient požadoval. Avšak táto správa nie je vždy požadovaná. Klient žiada o ňu len v prípade, ak je nevyhnutná obojstranná autentifikácia. Tým je zjavná jedna z výhod protokolu Kerberos, kedy sa neoveruje len identita klienta, ale súčasne aj identita aplikačného servera.

Bezpečnosť protokolu Kerberos V

Protokol Kerberos je kryptografickým protokolom, a tým spadá do oblasti exaktných empirických vied. Vyžaduje si teda formálny dôkaz svojej bezpečnosti. Takýto dôkaz je možné vytvoriť za pomoci špeciálnej metodiky, tzv. BAN logiky, alebo progresívnejšej GNY logiky. Obidve metodiky si kladú za cieľ formálne dokázať bezpečnosť objektu, pričom sa zameriavajú na: presnú definíciu cieľov, ktoré má objekt dosiahnuť; definíciu postupov a vzťahov subobjkektov objektu; dokazovanie celkovej bezpečnosti cez čiastkové činnosti; vylúčenie závislosti na nestabilných a neoverených predpokladoch atď. Protokol Kerberos V. je bezpečný nakoľko existuje formálny dôkaz jeho bezpečnosti realizovaný pomocou spomenutých metodík. V ďalšom texte sa však skôr pozrieme na konkrétnu implementáciu ochrany proti možným útokom.

Slovníkový útok a útok hrubou silou

Kerberos V. tým, že je otvorený pre najnovšie kryptografické algoritmy, efektívne predlžuje dobu pre uhádnutie kľúča; súčasne je pridaná podpora preautentifikácie, ktorá znemožňuje off-line útoky na vydané TGT lístky.

„Replay“ útok

Protokol Kerberos má niekoľko zabudovaných ochrán, ktorými predchádza úspešnosti „replay útoku“ a administrátor by nikdy nemal zanedbávať aktivovanie týchto ochrán:

1. **Zoznam adries v lístku** – ak klient požaduje lístok od KDC môže vložiť do žiadosti zoznam sieťových adries, z ktorých bude komunikácia platná. Tento zoznam sieťových adries je prenášaný cez celú komunikáciu protokolu Kerberos.

2. **Ochrana založená na čase** – ak klient požaduje využitie kerberizovanej služby vygeneruje súčasne autentifikátor, ktorý je odosielaný s lístkom k požadovanej službe ako súčasť autentifikácie. Autentifikátor obsahuje časové razítko, ktoré je šifrované relačným kľúčom generovaným KDC. Keď požadovaná služba získa autentifikátor dešifruje ho za pomoci získaného relačného kľúča a časové razítko overí voči svojmu lokálnemu času. Ak rozdiel týchto dvoch časov je viac ako päť minút služba zamietne lístok a odmietne autentifikovať užívateľa.

3. **„Replay" vyrovnávacia pamäť** – každá kerberizovaná služba udržiava vyrovnávaciu pamäť prijatých autentifikátorov. Ak služba prijme autentifikátor, ktorý sa už nachádza vo vyrovnávacej pamäti zamietne požiadavku o prístup. V opačnom prípade služba akceptuje požiadavku a autentifikátor pridá do vyrovnávacej pamäte.

Útok „man-in-the-middle"

Základným cieľom útoku je sfalšovanie identity požadovaného servera, ku ktorému sa klient chce pripojiť. Teda útok prebieha spôsobom, pri ktorom útočník skrýva svoju identitu a vydáva sa za aplikačný server, ku ktorému klient pristúpi. Týmto útočník začína komunikáciu s klientom. Následne sa útočník pokúsi upraviť správy zasielané užívateľom tak, aby získal prístup na reálne požadovaný server. Útočník je teda v pozícii prostredníka, na základe čoho sa nazýva aj tento útok. Dobrou správou je, že Kerberos protokol má už priamo zabudovanú ochranu proti tomuto typu útoku. Ako je známe Kerberos ponúka obojstrannú autentifikáciu, čo teda znamená, že nielen klient je povinný sa autentifikovať, ale aj aplikačný server musí potvrdiť svoju identitu (samozrejme ak je to vyžadované).

3 Konfiguračné rozhranie

Z predchádzajúcich kapitol je možné vidieť, že protokol Kerberos V. je silným nástrojom pre administrátorov počítačovej siete a zároveň užitočným pomocníkom pre tých, ktorý často pracujú s rozličnými sieťovými (ale aj lokálnymi) službami vyžadujúcimi autentifikáciu. A práve z tohto hľadiska bolo navrhnuté riešenie: rozhranie pre konfiguráciu protokolu Kerberos V., alebo presnejšie jeho MIT implementácie.

Nakoľko konfiguračné rozhranie sa neustále vyvíja, na tomto mieste by som rád uviedol len URL, kde sa projekt nachádza a odkiaľ je možné stiahnuť zdrojové kódy a nájsť zaujímavosti týkajúce sa vývoja spomínaného rozhrania

`http://project.xdata.sk/kerberos`.

4 Záver

Protokol Kerberos sa vyvíja už takmer 30 rokov a ponúka funkcionalitu, ktorá má čo ponúknuť aj súčasnému kyberopriestoru. A práve z tohto hľadiska bolo žiadúce vytvorenie

nástroja, ktorý by uľahčil implementáciu a konfiguráciu tohto protokolu do konkrétnych počítačových sietí a tým poukázal na možnosti, ktoré v sebe protokol Kerberos, zvlášť vo verzii V. skrýva. Verím, že vytvorený konfiguračný systém nájde uplatnenie v reálnej praxi, a stane sa každodenným pomocníkom mnohých sieťových administrátorov.

Literatúra

[1] BURDA, Z.: Kerberos: Instalace a použití. In.: `http://kerberos.zdenda.com` (2006)

[2] GARMAN, J.: Kerberos: The Definitive Guide. Ó Reilly : USA, 2003

[3] SMITH, R.: LINUX ve světe WINDOWS. GRADA : Praha, 2006

[4] MIT: In.: `http://web.mit.edu/kerberos/www/`

[5] RICCARDI F.: The Kerberos protocol and its implementations. In.: `http://www.kerberos.org/software/tutorial.html` (2007)

[6] MIGEON, J.: The MIT Kerberos Administrator's How-to Guide. MIT Kerberos Consortium, 2008

[7] DOSTÁLEK, L.: Velký průvodce protokoly TCP/IP: Bezpečnost. Computer Press, Praha, 2001

[8] WIKIPEDIA: In.: `en.wikipedia.org`; `cs.wikipedia.org`; `sk.wikipedia.org`

Kontaktná adresa

Peter PISARČÍK (RNDr., PhDr.),
Ústav informatiky, Prírodovedecká fakulta, UPJŠ v Košiciach,
Jesenná 5, 040 01 Košice,
`pisarcik@gmx.net`

OD NUL A OBČASNÝCH JEDNIČEK KE GENEROVANÝM KNIHÁM FONTÍKŮ A KANDŽÍKŮ

STŘÍŽ, Pavel, (CZ)

Abstrakt. Článek je náhledem do světa generovaných knih náhledů písem. Ať už písem s rozsahem 128 či 256 znaků (nespisovně sedmi- a osmibiťáky) nebo písem a jejich glyfů používaných v sazbě ČJKV (čínština, japonština, korejština, vietnamština; nespisovně šestnáctibiťáky). Smyslem článku není poukázat na všechny nuance téměř ročního projektu, spíš má čtenáři zamotat hlavu tím, cože to v tom TEXu, jeho přátelích a ve svobodném software, lze všechno připravit. Čtyři vygenerované knihy vydal nakladatel Martin Stříž.

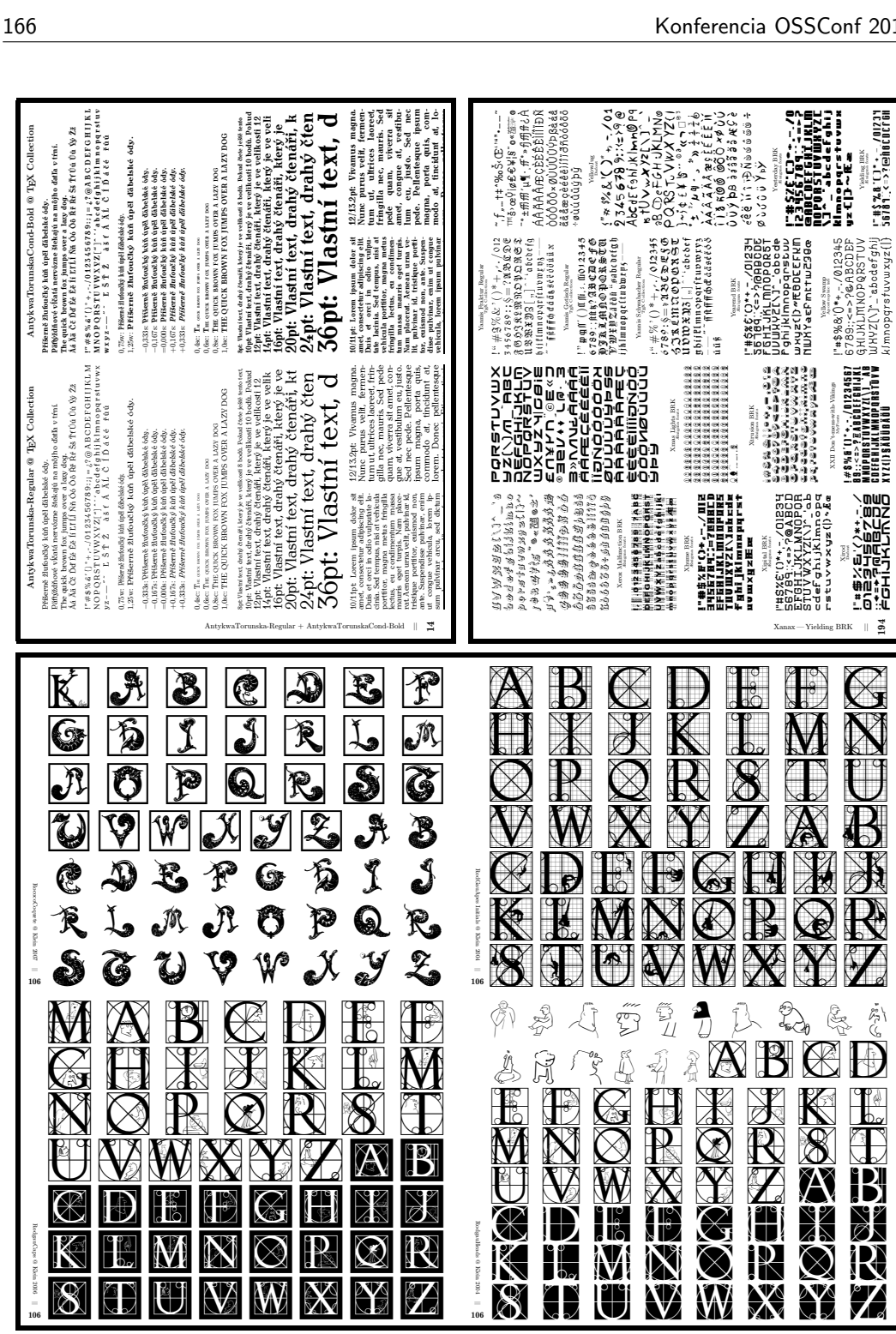
1 Úvod aneb o jednom snu

Když jsem byl žák končící povinnou školní docházku (začátek devadesátých let minulého století), tak jsme jako děcka řádili na osmibiťácích. Záhy přichází osobní počítače s mikroprocesorem, operačním systémem DOS a chvíli na to s grafickým uživatelským rozhraním. Tehdy to byl Microsoft Windows 3.11. Éra šestnáctibiťáků. Zde jsem se poprvé setkal s editorem T602, AmiPro a Mapou znaků. Srdíčko žáka zaplesalo, že by bylo hezké vidět všechny znaky z nabízených písem. Po dvaceti letech se sen stal skutečností, viz další strana.

Autorský záměr bylo vypsat všechny glyfy (kresby znaků) z vybraných písem, odstranění opakujících se znaků, optimalizovat rozložení do obdélníku, a samozřejmě, vše automatizovaně s vektorovými výstupy. Navíc s nějakou zajímavou úvodní stránkou a rejstříkem.

2 TEX

Typografický systém TEX je v mnoha ohledech komplexní nástroj. Je to dílo z pohledu typografie, programování, matematiky i možnosti automatizace. Základem práce u písem jsou formáty PFB (kresba znaků) a TFM (metriky). Po instalaci TEXové distribuce (např. pomocí TEX Live) je řada písem připravená k ostrému použití. Zde naše bádání začneme.



Tři designy z prvním tří knih. Výsledek snah a řady příjemných experimentů.

2.1 Výpis znaku

Vezměme si písmeno »P« od slova Pavel. To můžeme zapsat přímo z klávesnice. Také můžeme využít informaci o pozici znaku v písmu, konkrétně »P« je 80_{10}. Přeloženo do lidské řeči: v ASCII tabulce najdeme písmeno »P« na pozici 80, což je číslo zapsané v desítkové soustavě. Relativně snadno můžeme získat osmičkovou a šestnáctkovou reprezentaci takového znaku. V TEXu použijeme příkaz `\char` či stříškovou (bohužel ne střížkovou) konvenci.

```
\char'120 \char80 \char"50 ^^50
```

Získáváme: PPPP. Pokud tedy nevíme jak nebo neumíme vygenerovat znaky přímo, můžeme použít tuto zkratku. Při použití cyklu místo konstanty, nyní hodnota 80_{10}, použijeme proměnnou, jak si ukážeme na této a další straně.

2.2 Výpis všech znaků či glyfů

Dáme si první náročnější úkol. Budeme chtít vypsat všechny dostupné znaky používaného písma v TEXu. Vezmeme si například TEXové písmo `csr10`. Rychlý náhled v podobě tabulky v PDF získáme makrem `testfont`, tedy zapsáním:

```
pdftex testfont ←┘
csr10 ←┘
\table\bye ←┘
```

Ze začátku je to nezvyk, ale budeme-li chvíli výstup zkoumat, najdeme tam osmičkovou i šestnáctkovou reprezentaci znaků. Také vidíme, že maximum je 256 znaků. Nevidíme ani kerningy. Jedná se o přibližování a oddalování párů znaků, k tomu je potřeba zobrazit například souvislý text. Také není těžké si představit, že při tisících fontů (to je průměrná galerie) by byl při tisku náhled jednoho písma na stránku extrémně finančně náročný. Zkusíme si proto vlastní výpis.

2.3 Definovaný výpis

Výhodou definovaného výpisu je, že máme plnou kontrolou nad rozsahem vypisovaných znaků. Lze si přizpůsobit velikost písma, řádkový proklad, případně si navrhnout dosti specifický design. Můžeme si kontrolovaně vypisovat páry znaků kvůli kerningu. Na ukázku zvolíme netradiční vektorové písmo s japonskou slabičnou abecedou hiragana.

```
\font\hira=dmjhira at 9.245pt % Načtení písma a jeho aktivace.
\newcount\citac \citac=-1     % Inicializace čítače.
{\sloppy                      % Nepřetékání řádků.
\loop                         % Zahájení cyklu.
\advance\citac by 1           % Navýšení čítače o jedničku.
{\hira\char\citac}%           % Výpis specifického znaku nebo glyfu.
```

```
\discretionary{}{}{}%          % Lze zalomit řádek za každým glyfem.
\ifnum\citac<255\repeat        % Výpis celého rozsahu 0-255.
\par}                          % Konec skupiny pro \sloppy.
\normalfont\selectfont         % Zpět do základního písma.
```

Výstupem získáváme:

ぁあぃいぅうぇえぉおかがきぎくぐけげこごさざしじすずせぜそぞただちぢっつづてでとどなにぬねのはばぱひびぴふぶぷへべぺほぼぽまみむめもゃやゅゆょよらりるれろゎわゐゑをん

Můžeme si před či za glyfem vypsat jeho číselnou reprezentaci užitím `\the\citac`. V té chvíli však zjistíme, že u většiny písem není všech 256 pozic obsazených. To při jisté jednotě zobrazování činí nemalé potíže. Jednou máme před sebou základní sadu 26 znaků, vedle toho písma s diakritickými a speciálními znaky, jednou s a jednou bez číslic atd.

Naši ukázku si rozšíříme. Abychom nemuseli měnit rozsah ručně (u hiragany by byl vhodný od 1 po 83), lze si změřit a otestovat délku nebo výšku znaku. Pokud je délka nulová, čítač a neexistující glyf se nezobrazí.

```
\fontsize{6.6}{8}\selectfont % Úprava velikosti písma a prokladu.
\font\kata=dmjkata at 7.15pt % Načtení písma a jeho aktivace.
\newcount\citac \citac=-1    % Inicializace čítače.
{\sloppy \hfuzz=0pt          % Formátování odstavce.
\pretolerance=50 \tolerance=50 % Tolerance přetečení u řádků.
\fontdimen3\font=0pt \fontdimen4\font=0pt \fontdimen7\font=0pt
\loop                        % Rozpal mezery a zahájení cyklu.
\advance\citac by 1          % Navýšení čítače o jedničku.
\setbox0=\hbox{\kata\char\citac} % Změříme glyf, \wd0, \ht0 a \dp0.
\setbox1=\hbox{}%            % Změříme si prázdný box.
\ifnum\wd0=\wd1\else         % Je délka znaku nulová?
\texttt{\ifnum\citac<10 0\fi % Přidání nuly před cifry 0-9.
\the\citac.\copy0 }%         % Výpis specifického znaku nebo glyfu.
\discretionary{}{}{} %       % Lze zalomit řádek za každým glyfem.
\fi%                         % Konec podmínky \ifnum.
\ifnum\citac<255\repeat      % Výpis celého rozsahu 0-255.
\par}    % Ukončení odstavce a uzavření lokální skupiny.
\normalsize\selectfont       % Zpět do základního písma.
```

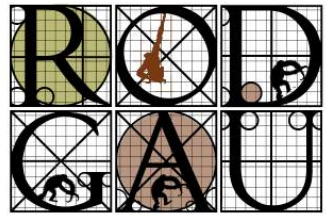
Výstupem dostáváme očíslovaný seznam glyfů katakany bez prázdných pozic.

01.ァ 02.ア 03.ィ 04.イ 05.ゥ 06.ウ 07.ェ 08.エ 09.ォ 10.オ 11.カ 12.ガ 13.キ 14.ギ 15.ク 16.グ 17.ケ 18.ゲ
19.コ 20.ゴ 21.サ 22.ザ 23.シ 24.ジ 25.ス 26.ズ 27.セ 28.ゼ 29.ソ 30.ゾ 31.タ 32.ダ 33.チ 34.ヂ 35.ッ 36.ツ
37.ヅ 38.テ 39.デ 40.ト 41.ド 42.ナ 43.ニ 44.ヌ 45.ネ 46.ノ 47.ハ 48.バ 49.パ 50.ヒ 51.ビ 52.ピ 53.フ 54.ブ
55.ブ 56.ヘ 57.ベ 58.ペ 59.ホ 60.ボ 61.ボ 62.マ 63.ミ 64.ム 65.メ 66.モ 67.ャ 68.ヤ 69.ュ 70.ユ 71.ョ 72.ヨ
73.ラ 74.リ 75.ル 76.レ 77.ロ 78.ヮ 79.ワ 80.ヰ 81.ヱ 82.ヲ 83.ン 84.ヴ 85.ヵ 86.ヶ

Nyní vidíme, že kdybychom chtěli zajistit design do cca definovaného poměru obdélníku pro více písem, tak to není triviální úloha. První čtyři řádky jsou v pořádku, pátý je příliš krátký. Jedna z možností je postupné zvětšování/zmenšování velikosti písma, např. metodou půlení intervalu, ovšem i tak zůstává nevýhodou nedotažení posledního řádku.

2.4 Prezentování písem

Grafici a designéři si prezentují svá písma nejrůznějšími způsoby. Zde je několik náhledů na Kleinova písma. Jedná se o TTF DoubleFaces, RodGauApes Initials a FilledABC z roku 2004. U prvního zmíněného náhledu je potřeba podotknout, že Manfred Klein byl mezi úplně prvními, kteří začali vyvíjet 3D písma, chceme-li písma s 3D efekty.


2.5 Pangram

Řada písmolijen prezentuje svá písma za použití pangramů (kontrola existence znaků, náhled písma při různých velikostech) a souvislých textů (pohled na ligatury a kerning párů).

```
\def\pCZ{Vřeštící příšery se dožadovaly úplně čerstvých řízečků.}
\def\pSK{Päťtýždňové vĺčatá nervózne štekajú na môjho ďatľa v tŕní.}
```

Nastavit specifickou velikost písma a řádkového prokladu lze příkazem `\fontsize`, konkrétně např. `\fontsize{13.5}{15}\selectfont`. Jedná se o body (pt).

Ukážeme si výpis pangramu se změnou velikosti písma s/bez změny řádkového prokladu. Nejjednodušší ukázka se změnou velikosti písma je:

```
\newcount\vyska \vyska=2
\loop
\advance\vyska by 1
%\fontsize{\vyska}{\vyska}\selectfont % Ke srovnání.
\font\mujfont=csr10 at \vyska pt \mujfont
\par{\pCZ\ Písmo csr10 ve velikosti \the\vyska pt.}
\ifnum\vyska<10\repeat
\normalsize\selectfont
```

Výstupem získáváme:

Vřeštící příšery se dožadovaly úplně čerstvých řízečků. Písmo csr10 ve velikosti 3pt.

Vřeštící příšery se dožadovaly úplně čerstvých řízečků. Písmo csr10 ve velikosti 4pt.

Vřeštící příšery se dožadovaly úplně čerstvých řízečků. Písmo csr10 ve velikosti 5pt.

Vřeštící příšery se dožadovaly úplně čerstvých řízečků. Písmo csr10 ve velikosti 6pt.

Vřeštící příšery se dožadovaly úplně čerstvých řízečků. Písmo csr10 ve velikosti 7pt.

Vřeštící příšery se dožadovaly úplně čerstvých řízečků. Písmo csr10 ve velikosti 8pt.

Vřeštící příšery se dožadovaly úplně čerstvých řízečků. Písmo csr10 ve velikosti 9pt.
Vřeštící příšery se dožadovaly úplně čerstvých řízečků. Písmo csr10 ve velikosti 10pt.

Následuje ukázka se zásahem do řádkového prokladu:

```
\newdimen\vyska   \vyska=2pt
\newdimen\proklad \proklad=0pt
\loop
\advance\vyska by 1pt
\proklad=1.22\vyska
\fontsize{\vyska}{\proklad}\selectfont % Dochází k substituci.
\font\mujfont=csr10 at \vyska \mujfont % Přepsání výšky.
\par\makebox[3cm][l]{\the\vyska/\the\proklad}\pSK
\ifdim\vyska<10pt\repeat
\normalsize\selectfont
```

Výstupem dostáváme:

3.0pt/3.66pt Päťtýždňové vĺčatá nervózne štekajú na môjho ďatľa v tŕní.
4.0pt/4.88pt Päťtýždňové vĺčatá nervózne štekajú na môjho ďatľa v tŕní.
5.0pt/6.1pt Päťtýždňové vĺčatá nervózne štekajú na môjho ďatľa v tŕní.
6.0pt/7.32pt Päťtýždňové vĺčatá nervózne štekajú na môjho ďatľa v tŕní.
7.0pt/8.54001pt Päťtýždňové vĺčatá nervózne štekajú na môjho ďatľa v tŕní.
8.0pt/9.76001pt Päťtýždňové vĺčatá nervózne štekajú na môjho ďatľa v tŕní.
9.0pt/10.98001pt Päťtýždňové vĺčatá nervózne štekajú na môjho ďatľa v tŕní.
10.0pt/12.20001pt Päťtýždňové vĺčatá nervózne štekajú na môjho ďatľa v tŕní.

2.6 Balíček shapepar

Pokud chceme vysázet výpis znaků do definovaného obdélníku, můžeme použít TEXový balíček shapepar. Do preambule přidáme `\usepackage{shapepar}`, dokumentaci nejrychleji získáme zapsáním v příkazové řádce: `texdoc shapepar` ↵.

```
\newcount\znak
\font\mujfont=wncyr10 at 7.5pt
\def\vypis{
  \znak=-1
  \loop
  \advance\znak by 1
  {\mujfont\char\znak}%
  \discretionary{}{}{}%
  \ifnum\znak<255\repeat
  } % Konec příkazu \vypis.
\begin{minipage}{0.3\textwidth}  \heartpar{\vypis}\end{minipage}
\begin{minipage}{0.4\textwidth}   \starpar{\vypis}\end{minipage}
\begin{minipage}{0.3\textwidth}\hexagonpar{\vypis}\end{minipage}
```

Výstupem získáváme všechny znaky (a srdíčko navíc) písma `wncyr10` ve třech verzích:

ЊЉЏЭІ ЄЂЋњљ
џэіѥђћЮЖЙЁѴѲЅЯюжйёѵѳѕя
¨!”Ђ˘%´’()*Ђ,-./0123456789:;«
ı»?˘АБЦДЕФГХИЈКЛМНОП
ЧРСТУВЩШЫЗ[“]ЬЪ‘аб
цдефгхијклмнопчрсту
вщшыз–—№ьъ
♡

Њ
ЉЏ
ЭІЄЂ
Ћњљџэіє
ђћЮЖЙЁѴѲЅЯюжйёѵѳѕя¨!”Ђ˘%´’()*Ђ,
-./0123456789:;«ı»?˘АБЦДЕФГ
ХИЈКЛМНОПЧРСТ
УВЩШЫЗ[“]ЬЪ
‘абцдефгхијклмнопч
рстувщ шыз–—
№ ьъ

ЊЉЏЭІЄЂЋњ
љџэіѥђћЮЖЙЁѴѲ
ЅЯюжйёѵѳѕя¨!”Ђ˘%´’()
*Ђ,-./0123456789:;«ı»?˘АБ
ЦДЕФГХИЈКЛМНОПЧРС
ТУВЩШЫЗ[“]ЬЪ‘абц
дефгхијклмнопчрст
увщшыз–—№ьъ

Zjistit automatizovaně zalomení řádků je zdánlivě náročný úkol. Autor použil přístup, že si výsledné PDF převedl na text a z něj potřebné detaily zjistil. Speciální znaky byly opět převedeny na TEXové sekvence s příkazem `\char`.

`pdftotext -nopgbrk -raw -eol unix vstup.pdf vystup.txt` ←┘

Řešení zdánlivě nejtěžšího problému, jak zajistit zaplnění celého definovaného obdélníku, spočívalo jen v nalezení největší velikosti písma. Stačilo kontrolovat, zda-li je výstupní PDF na jedné straně, nebo již přeteklo na dvě strany. Mezi každým znakem se nastaví příkaz, který pruží. Tím se zajistí optimální rozložení po délce. Podobně se nechají pružit řádkové proklady. V běžné sazečské praxi se používají příkaz `\hfill` a `\vfill`. Výstupní PDF lze ořezat o ochranný prostor nástrojem `pdfcrop` nebo zjištěním bounding boxu, např. přes GhostScript. Ukázky zdrojových kódů jsou bohužel nad rámec tohoto článku.

2.7 Rotace

Praktickým problémem především u rozsáhlých písmových projektů Manfreda Kleina byla jiná situace. Byly to opakující se kresby a kresby otočené zrcadlením. U náhledů totiž stačí jedna vizuální varianta a uvolněný prostor lze využít na zvětšení písma. Připomeneme, že rotace a zrcadlení nejsou již náročné operace na úrovni TEXu samotného. Následující kód zpracuje znak »Ž« z příjmení Stříž a »y« kvůli testování hloubky znaků, to vše za pomoci balíčku graphicx (dokumentaci lze získat opět přes `texdoc graphicx` ←┘):

```
\setbox0=\hbox{Žy} \unhcopy0\ --\
\reflectbox{\unhcopy0}\ --\
\rotatebox[origin=c]{180}{\unhcopy0}\ --\
\reflectbox{\rotatebox[origin=c]{180}{\unhcopy0}}
```

Existuje nespočet možností rotace, ale s těmito třemi se lze setkat u písem na úrovni TTF/OTF nejčastěji. Zásadní je samozřejmě zrcadlové překlopení (2. možnost). Ukázky jsou následující: Žy – yŽ – ʎZ – Zʎ . Občas bylo možné se setkat i s inverzí, zešikmením a dalšími transformacemi kreseb znaků.

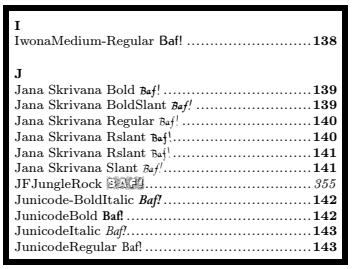
2.8 Identita a podobnost znaků

V našem projektu bylo možné vyřadit stejné nebo velmi podobné znaky. To by TEX uměl (především LuaTEX), ale není to triviální problém. Na vyhledávání duplicitních a otočených kreseb lze použít program FontForge, `http://fontforge.sourceforge.net/`.

FontForge je poměrně unikátní projekt, který umí pracovat s písmy v grafickém režimu, dávkově i jako knihovna Pythonu. Má svůj vlastní programovací jazyk a automatizovat s ním operace s písmy není výrazný problém.

V článku nám na ukázky již prostor nezůstal, proto autor srdečně zve na přednášku na konferenci OSSConf2010 (začátek července), a také na čtvrté setkání CONTEXTistů a konferenci TEXperience 2010 (září; mlýn Brejlov u Prahy; konference jsou pod záštitou Ministerstva školství, mládeže a tělovýchovy České republiky). Akce společně organizují $\mathcal{C}_S$TUG a FaME UTB ve Zlíně.

3 Zajímavosti kolem knih



3.1 Úvodní listy

S úvodními listy jsme si „vyhráli“. Svět typografie má rád hladkost a uhlazenost, matematici a statistici mají rádi naopak náhodu a chaos. Jak to spojit dohromady?

Vyzkoušeli jsme tuto cestu. Sazba náhodně volených znaků byla klasicky v řádcích, ale jednotlivé znaky byly náhodně otáčeny a posouvány tam či zpět z aktuální pozice. Výsledkem se podařilo rozumné rozložení šedi na papíru se zahrnutým efektem náhodnosti.

Ukážeme si jeden z možných principů, jak z dostupné sady znaků zkontrolovaných a vypsaných FontForgem náhodně (s vracením i bez vracení) vybírat. Zdrojový kód jednoduché ukázky může vypadat takto:

```
#!/bin/bash
glyfy=znaky.txt glyfydoc=ztemp.txt # Pomocné soubory.
echo -e "A\nB\nC\nD\nE\nF" >$glyfy # Pracovní sada.
RANDOM=100 tahusop=15 tahubez=7
pocet=$(cat $glyfy | sed -n '$=')  # Počet řádků.
if [ $pocet -lt $tahubez ]; then
```

```
  echo -e "Opravuji počet tahů bez navracení na $pocet.\n"
  let tahubez=$pocet
  fi
function volba {
  let "vyber=$RANDOM % $pocet + 1" # Jedna až počet řádků.
  radek=$(cat $glyfy | sed -n "$vyber p") # Vypiš řádek.
  echo $i. volba je $vyber. řádek. Obsahem je $radek.
  }
# Výběr s navracením.
for i in $(seq 1 1 $tahusop); do   # Tahy s opakováním.
  volba
done; echo # Konec for cyklu.
# Výběr bez vracení.
for i in $(seq 1 1 $tahubez); do   # Tahy bez opakování.
  volba
  cat  $glyfy | sed "$vyber d" >$glyfydoc   # Odebere daný řádek.
  # cat $glyfy | sed "/$radek/d" >$glyfydoc # Alternativa.
  cp $glyfydoc $glyfy
  echo "Výpis zbytku souboru:"; cat $glyfy; echo
  let pocet=$pocet-1
done
rm -f $glyfy $glygydoc
```

Prezentovaný princip je prostý.

V textovém souboru máme výpisy znaků, vždy jeden na každém řádku (písmeno nebo to může být příkaz). Spočteme celkový počet řádků takového souboru a z něj volíme řádek. Poté buď následuje zásah do souboru v podobě vymazání voleného řádku (výběr bez navracení), nebo nikoliv (výběr s vracením).

3.2 Rejstřík

Tohle byla jedna ze zapeklitějších situací, které autor na sebe ušil, ale i to se podařilo vyřešit. Charakteristické je, že u každého písma je několik znaků písmem vysázených (Baf!), ty charakterizují malá písmena (»af«; mohlo by být i »á«, zástupce diakritických znaků), verzálky (»B«) a zástupce speciálních znaků (»!«). Takto čtenář hned vidí, co lze v písmu přibližně očekávat. Ještě obecnější by bylo »Báf2×«, kde je i číslice a matematický symbol. Inspirace vznikla z rejstříku prezentovaného v závěru dokumentace `texdoc symbols` ↵.

Další charakteristiky rejstříku jsou, že je generovaný strana po straně kvůli velkému počtu písem v každé knize, poté jsou strany spojeny při kompletaci. Každá strana začíná třídícím znakem. Kapitoly knihy jsou odlišeny řezem písma čísla stránky.

U kandžíků jsou rejstříky navíc předgenerovány za pomoci nástroje Bash.

JLPT Level 1 (by freq.)

3.3 Kompletace

U PostScriptových souborů se používaly nástroje `psselect` a `psbook`. U zpracování PDF lze použít LaTeXový balíček pdfpages. Máme-li například PDF o osmi stranách, můžeme jej přesázet na dvě strany v režimu 2×2 takto:

```
\documentclass[a4paper]{article}
\usepackage{pdfpages}
\begin{document}
\includepdf[pages={-},nup=2x2,frame]{vstup.pdf}
%\includepdf[pages={-},pagecommand={Hello World!}]{vstup.pdf}
\end{document}
```

Výhodou zůstává, že se lze vrátit zpátky do sazby vkládané stránky nepovinným parametrem `pagecommand`. V ukázce si znak procenta (komentování řádku) z pátého řádku přesuňte na začátek řádku čtvrtého.

3.4 XeTeX

Písma pro tři knihy fontíků byla získána konverzí do TFM (soubor metrik) a PFB (kresba glyfů) nástroji `afm2tfm`. V rámci testů byly využity nástroje `tftopl`, `pltotf` a `vptovf`. Samozřejmě nástroje Bash, Sed, Cat či Tr byly na denním pořádku, především u zpracování `kanjidic` Jima Breena, `http://www.csse.monash.edu.au/~jwb/kanjidic.html`.

Jakmile jsme čelili více než 256 znakům v jednom písmu, vyzkoušeli jsme XeTeX. Ten umí přímou práci s kódováním UTF-8. Umí zobrazit jeden znak několika způsoby, viz článek pravděpodobně v openMagazinu 5/2010.

Můžeme pracovat s názvem glyfu, včetně názvu `.notdef` a glyfů s indexem nula. Takto lze zjistit počet definovaných glyfů přes různá písma a dle toho upravit sazbu. V navrženém rozvržení nevypadalo hezky, když byly například přítomny tři kandžíky v celém okraji.

Rozmístění objektů bylo zrealizováno prostředím `picture` a načtení více než devíti parametrů v jednom TeXovém příkazu bylo zrealizováno standardně přes vnořování příkazů. Jeden z následovníků TeXu, tzv. LuaTeX, umí tuto možnost ještě elegantněji, ale o tomto rozšíření TeXu snad až někdy jindy.

3.5 Další zajímavosti

Za zmínku stojí několik maličkostí.

- První kniha obsahuje 703, druhá 2632 a třetí 2741 různých písem a jejich řezů.
- Na stahování dostupných písem z Internetu byl na všechny možné i nemožné způsoby použit program `wget`.
- Bitmapou byla vložena jen písma, která byla záměrně autorsky vytvořena tak, aby přesáhla technické možnosti písmových formátů.
- Při výrobě knih se zpracovávají TTF, OTF, PFB i DFONT formáty v jednom konverzním běhu díky programu FontForge.
- V knize kandžíků bylo představeno 68 znaků a znamének latinky, 197 speciální glyfů spolu s hiraganou a katakanou a 6376 kandžíků.
- Jako doplněk poslední knihy vznikla postupnými výběry a natažením 2230 kandžíků přes balíček pdfpages metrová housenka.
- Na tisk byly předány PDF ve velikostech 3 MB (housenka), 101 MB (kandžíci), 179 MB (fontíci 1), 255 MB (fontíci 2) a 706 MB (fontíci 3).
- Váha knih je 1,6 kg (kandžíci), 2,2 kg (fontíci 1), 2,3 kg (fontíci 2) a 3,7 kg (fontíci 3).

4 Závěrem snění do budoucna

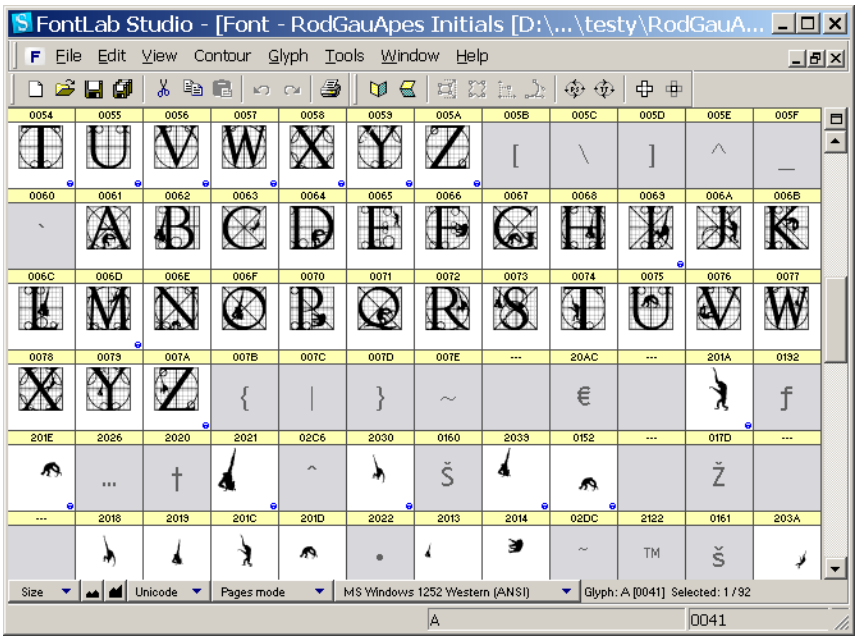
Na knihy by snad mohly navázat další dva díly, opět s trochu jiným designem. První představující projekt `www.ceskefonty.cz` s různými transformacemi písem a druhý díl s náhledy 10.000 písem a jejich řezů galerie `www.ultimatefontdownload.com`, která je dostupná za cenově přijatelných 19,99 USD.

Lze se těšit na knihu čínských znaků, kterých je víc než sto tisíc? Možná, možná až bude UTF-16 na denním pořádku.

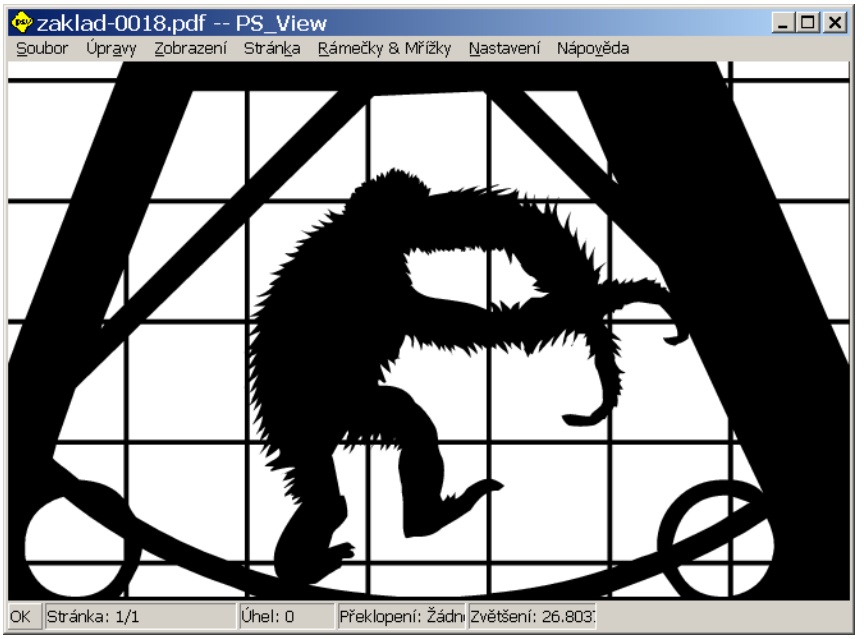
Autor se s vámi rozloučí náhledem na jedno z Kleinových písem, kde je vidět, že pomocné a pracovní nákresy (většinou nad rozsahem úvodních 256 znaků) musely být z náhledů odstraněny. Spodní obrázek je zvětšenina znaku »A« ze zmíněného písma z horního obrázku na další straně.

Kontaktní adresa

Ing. Pavel STŘÍŽ, Ph.D.
Ústav statistiky a kvantitativních metod
Fakulta managementu a ekonomiky
Univerzita Tomáše Bati ve Zlíně
Mostní 5139, CZ-760 01 Zlín
`striz@fame.utb.cz`



Náhled na písmo RodGauApes Initials od Manfreda Kleina, 2004. První reakce autora článku byla: „Sakra! Co je to tam za fleky?“



Oblíbený náhled autora v programu PS_View při výrazném zvětšení (nyní 26,8×). Glyf U+0061 v plné kráse.

JAK JSEM SE SKAMARÁDIL S LaTeXOVÝM BALÍČKEM ANIMATE

STŘÍŽ, Pavel, (CZ)

Abstrakt. Bez mučení se autor čtenáři přiznává, že není příznivcem animací. Je zvyklý na konečné podklady určené k tisku. To však neznamená, že se k takto výjimečným typografickým partiím člověk nedostane nebo že nedozná jejich významu. Článek poukazuje na vybrané úlohy z oblasti animování v PDF přes animaci textu, grafiky až po různé drobné vychytávky a pomůcky při přípravě vstupu. Na vektorovou grafiku je nejčastěji použit sourozenec TeXu přezdívaný METAPOST.

1 Úvod

S animacemi v PDF jsme se postupně setkali jako čtenáři článků autorů Bittó [1], Holeček [2], Holeček a Sojka [3] a Plch a Šarmanová [4]. Několikrát jsme v minulých letech pracovali s balíčkem movie15. Autorem balíčku je Alexander Grahn. K balíčku animate od stejného autora jsme se dostali náhodou, když jsme recenzovali článek o šachách pro Zpravodaj CSTUG a procházeli jsme možnosti balíčku chessboard, kdy vás v dokumentaci nemůže nezaujmout úvodní strana a strana 75. Později jsme experimentovali s vrstvením na úrovni OCG s balíčky fancytooltips a ocgtools.

2 Animace textu

Elementární ukázku, tzv. Hello World!, pracovně nazýváme Pozor! Pozor! Jedná se o upozornění na důležitou partii ve studijních pomůckách. Je to identicky vypsaný text, u kterého se mění barva písma. Chce to vyzkoušet různé prohlížeče, Adobe Reader 8.0+ náhled zvládá. V samostatném souboru by ukázka s minimální LaTeXovou strukturou vypadala:

```
% pdflatex animace.tex
\documentclass{article}
```

```
\usepackage{animate,color}
\begin{document}
\def\text{Pozor!}   % Co se bude vypisovat.
\begin{animateinline}[autoplay,loop]{1} % Jeden snímek za vteřinu.
{\color{cyan}\text} % První vrstva.
\newframe           % Přechod na novou vrstvu.
{\color{blue}\text} % Druhá vrstva.
\end{animateinline}
\end{document}
```

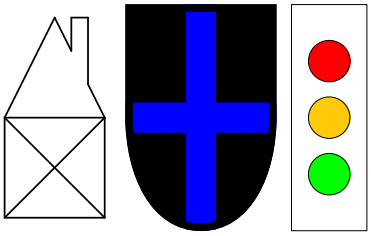
Záležitost, kterou se nám zatím nepodařilo odhalit je, že se oříznou dotahy křivek znaků. Pracovně se to dá vyřešit obalením do ochranné zóny užitím příkazu `\fbox`, např. konkrétně o jednom bodu: `{\fboxsep=1pt \fboxrule=0pt \fbox{Pozor!}}`.

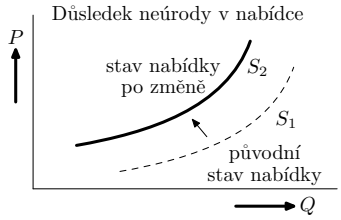
Místo textu lze dodat libovolný sazební podklad. Příkaz `\newframe*` místo `\newframe` by byl pokyn k vyčkání na kliknutí tlačítka myši.

3 Animace grafiky

3.1 Typické ukázky

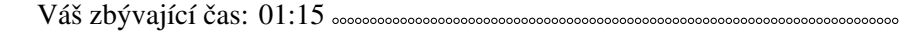
Na tomto místě si bez zdrojových kódů představíme několik jednodušších animací.





Připravíme si cvičný papír.
Nakreslíme a popíšeme osu *x*. Jedná se o množství.
Poté nakreslíme a popíšeme osu *y*. Jedná se o cenu.
Nakreslíme původní nabídku.
Půjde o posun nabídky doleva (pokles nabízeného množství).
Vidíme novou nabídku (plná čára).

Grafy vznikly pomocí METAPOSTu a nyní jedna ukázka v programu Ti*k*Z:



3.2 Podpěra

Při práci s grafikou je potřeba dát si pozor na několik maličkostí, stojí to však za to, poněvadž animace i tiskový výstup zachovává vektorovou formu.

Ať už při rýsování nákres rozšiřující či přírůstkový, vždy čelíme problému jednoty délky a výšky objektů. S tím se při vrstvení čtenář pravděpodobně již setkal. Nakreslíme-li si objekt v pravém horním rohu bez podpěry (anglicky strut), tak jej nemůžeme vrstvit s mnohem větším objektem.

Jedna ze začátečnických možností je kolem dokola všech objektů nakreslit bílý obdélník. Tím si vynutíme jednotu. Kresba má řadu nevýhod. Především na začátku rýsování nevíme

celkové rozměry obrázku a bílý objekt je sice očím neviditelný, ale častokrát pracujeme s barevným pozadím či texturou, a to by nepůsobilo vábně.

Další možnost je upravit bounding box po vytvoření obrázků za ten největší. Zjistit rozměry levého dolního a pravého horního rohu umí například GhostScript.

```
gs -dNOPAUSE -q -dBATCH -sDEVICE=bbox obrazek.mps
```

Pod Microsoft Windows užijte `gswin32c`. Tento program však nemusíte mít nainstalovaný. Výstupem získáváte nejčastěji druhý a třetí řádek PostScriptového souboru.

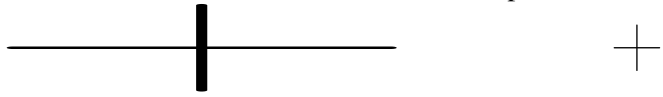
```
%%BoundingBox: -3 -3 144 59
%%HiResBoundingBox: -2.25 -2.25 143.98225 58.9429
```

Programem `epsffit` lze zrealizovat editaci stávajícího bounding boxu. Takovou výměnu řádků lze však zrealizovat nástroji jako Bash, Sed a dalšími. Nechávám čtenáři na procvičení, invencím se meze nekladou.

Na úrovni METAPOSTu na to šli šibalsky pánové Mařík a Grahn, viz soubor `exp.mp` stáhnutelný ze serveru CTAN.ORG. Není nad to se naučit METAPOST.

```
def orezat =
  setbounds currentpicture to bounds;
  clip currentpicture to bounds;
enddef;
filenametemplate "%j_%c.mps";
path bounds; u:=5cm; v:=1cm; k:=3mm;
def obd = draw (0,0)--(u,0)--(u,v)--(0,v)--cycle; enddef;
def kriz= draw (u/2-k,v/2)--(u/2+k,v/2);
          draw (u/2,v/2-k)--(u/2,v/2+k);          enddef;
beginfig(1) obd;  bounds:=bbox currentpicture; orezat; endfig;
beginfig(2) kriz; orezat; endfig;
beginfig(3) obd;  orezat; endfig;
beginfig(4) kriz; endfig;
bye
```

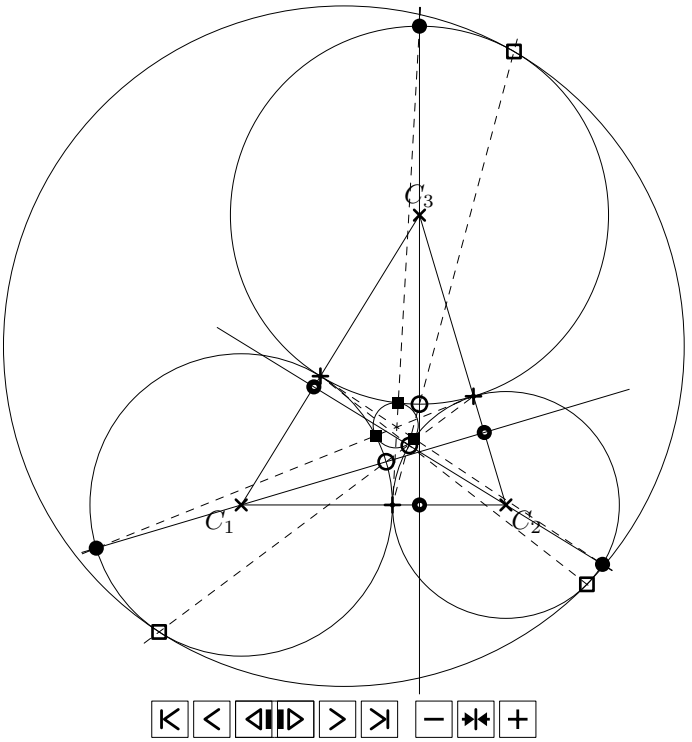
Zde můžeme vidět deformaci kříže s a bez potřebného nastavení a oříznutí.



3.3 Navigace

Pokud máme připravenou sérii obrázků `krok_00.mps` až `krok_16.mps`, tak lze v TeXu zapsat: `\animategraphics[controls,poster=last]{3}{krok_}{00}{16}`

Tímto zápisem se nám zobrazí navigace a při případném tisku půjde na výstupní zařízení poslední snímek, chceme-li vrstva. Náročným čtenářům přikládáme animaci Eppsteinovy konstrukce vedoucí k nalezení vnitřní a vnější Soddyho kružnice [5].



4 Partie pro náročnější

Práce na úrovni METAPOSTu však může být napínavá, pokud s animacemi začínáte. Jakákoliv drobná změna či nepřesnost předchozí ukázky by se musela relativně komplikovaně zapracovat, neb původní zdrojový kód na vrstvy nepamatoval. To je o cviku.

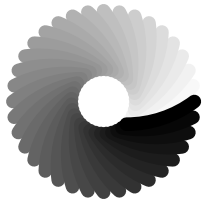
4.1 Parametr timeline balíčku animate

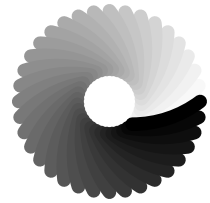
Výhodnější je následující postup. Připravit si všechny objekty v různých a požadovaných verzích (barva, síla, popisky) a na úrovni balíčku animate je pomocí časování, vrstvení, čištění a seskupování volat. Úprava pak spočívá jen v nastavení textového souboru bez zásahů do zdrojových kódů METAPOSTu.

Tento přístup si na přednášce na OSSConf2010 (začátek července), čtvrtém setkání CONTEXTistů a TEXperience 2010 (září; Česká republika; Brejlov u Prahy; konference jsou pod záštitou MŠMT ČR) podrobněji představíme. Autor tohoto článku srdečně zve!

4.2 Série obrázků přímo z METAPOSTu

Použijeme nové poznatky a vygenerujeme si dílčí části růžence. Načteme celý obrázek, díl po dílku, a do třetice přírůstkově.





Příprava růžence je na další straně. Prvně se zrealizuje sazba do souboru `ruze.400` poté se generují dílčí partie přes `beginfig(citac)`. Navíc se generuje soubor `ruze.txt`, což je časování a vlastnosti pro balíček animate. Načtení na úrovni TeXu lze zrealizovat:

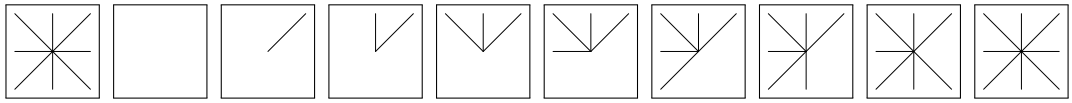
```
\convertMPtoPDF{ruze_400.mps}{0.5}{0.5}
\animategraphics[scale=0.5,autoplay,loop]{3}{ruze_}{0}{36}
{\animategraphics[poster=last,autoplay,loop,
   scale=0.5,timeline=ruze.txt]{3}{ruze_}{0}{36}}
```

Během přípravy tohoto článku nám `timeline` ovlivnil další sazbu (volání po `^^M`), tak jsme celý příkaz uzavřeli do lokální skupiny. Pomohlo to!

4.3 Generování zdrojového kódu TeXu

Dalším silným nástrojem programátora je generování části nebo celého požadovaného zdrojového kódu. S procesem generování se setkáváme dennodenně, nebude tomu jinak ve světě TeXu. Zde je jedna ukázka, kdy počet obrázků (9) je TeXu předpřipraven. Výhodou však je, že máme plnou kontrolu nad sazbou grafiky, ochranné zóny, rámování, popisků atd.

```
\begin{animateinline}[autoplay,palindrome,loop]{3}
\multiframe{9}{iobr=1+1}{\convertMPtoPDF{obr.\iobr}{1}{1}}
\end{animateinline}
```



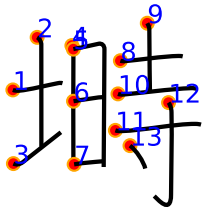
Na tomto příkladu si během přednášky ukážeme možnost generování série obrázků z jednoho zdrojového kódu METAPOSTu či jiných dat po nastavení si pravidel generování.

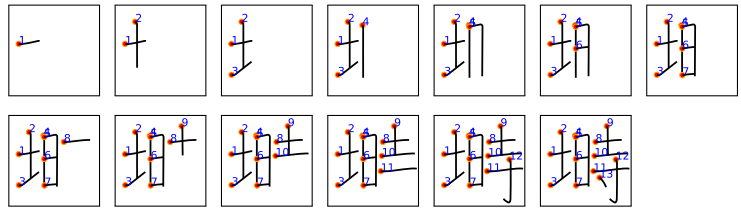
4.4 Zpracování externích dat – kandži

Animace a krokování může být užitečnou studijní pomůckou studentek a studentů japonštiny či čínštiny. Při využití znalostí nabytých v předchozí ukázce můžeme vzít projekt Ulricha

```
% mpost ruze.mp
filenametemplate "%j_%c.mps"; % Styl generovaných souborů.

path bounds; path p;
p:=(5mm,-5mm){right}..(2cm,0); % Rotovaná křivka.
numeric j, barva;

def obrazek(expr p, j) = % Příprava dílku celé animace.
  barva:=1-j/360;
  draw p rotated j withpen pencircle scaled 3mm withcolor barva;
enddef;

def orezat = % Z kompletního nárysu nastavení rozměrů.
  setbounds currentpicture to bounds;
  clip currentpicture to bounds;
enddef;

beginfig(400) % Vykreslení celé kresby.
for j=0 step 10 until 360:
  obrazek(p,j);
endfor;
bounds:=bbox currentpicture; % Zjištění rozměrů bounding boxu.
orezat;
endfig;

numeric citac; citac:=-1; % Pomocný čítač.
for j=0 step 10 until 360:
  citac:=citac+1;
  beginfig(citac) % Generování souborů.
    obrazek(p,j);
    orezat;
  endfig;
  % Data pro balíček animate:
  write "::"&decimal(citac)&"x0" to "ruze.txt";
endfor;

% Zde by mohlo následovat generování TeXového souboru s příkazy
% načítající animaci atp. Následné použití v TeXu např. přes
% příkaz: \input soubor.tex

bye.
```

Apela KanjiVG, `http://kanjivg.tagaini.net/`, a vstupní XML data si připravit tak, abychom na výstupu získali animovanou sekvenci a její rozkreslení dle tahů.





4.5 Poznámky k formátu SVG

Kdo je příznivcem SVG formátu, tak vězte, že převod z a do PDF není náročným úkolem. Inkscape umí pracovat z příkazové řádky a program `pstoedit` umí výstup z PDF a (E)PS do mnoha formátů, viz `pstoedit -help` ↵. K ověření práce zkuste:

```
inkscape --export-pdf=vystup.pdf vstup.svg ↵
pstoedit -f svg vstup.pdf vystup.svg ↵
```

Automatizovat si celý proces bez spuštění konzole (`-z`) lze na úrovni nástroje Bash:

```
#!/bin/bash
pripona=.svg
for i in ‘ls *$pripona‘; do
nazev=${i%$pripona}
echo "Zpracovávám: "$nazev
inkscape -z --export-pdf=$nazev.pdf $i
done
```

Avizovaná zpráva od vývojáře knihovny METAPOSTu `mplib` Taco Hoekwatera je, že se plánuje přímý výstup z METAPOSTu do SVG. To k budoucnosti.

Pro běžného uživatele je dostatečné a lze již testovat `outputformat:="svg";` zapsáním ve zdrojovém kódu vznikajících obrázků METAPOSTu.

5 Závěr

Jako studenti jsme se setkali s celou řadou videosekvencí (formáty AVI, MPEG, OGG, FLV), Flash animací (SWF) i animací na webových stránkách (Java). Každému sedí něco jiného, pravda. Výhoda představeného přístupu je, že se poslední nebo první snímek animace stává součástí tištěné publikace a zůstává zachována její vektorová podoba. Zadání a řešení nemusíme mít na více snímcích či stranách, můžeme oživit studijní text a také můžeme jednotlivé kroky zvýraznit a komentovat v textu.

Zaujatému čtenáři doporučujeme dokumentaci balíčku `animate`, kde je několik dosti poučných ukázek ze světa matematiky a deskriptivní geometrie, viz `texdoc animate` ↵.

Literatura

[1] BITTÓ, Ladislav: TEX a PostScript v grafike programovacích jazykov. In *Zpravodaj Československého sdružení uživatelů TEXu,* ročník 14, číslo 1, str. 28–38. Editor Zdeněk Wagner, Česká republika, Praha, 2004. ISSN 1211-6661, e-ISSN 1213-8185. URL: `http://bulletin.cstug.cz/gpvtv.zip` (ukázky a doprovodné soubory) `http://bulletin.cstug.cz/pdf/bul_041.pdf` (příspěvek)

[2] HOLEČEK, Jan: TEXem generované animace. In *Sborník čtvrtého ročníku semináře o Linuxu a TEXu – SLT 2004.* Znojmo, 24. – 27. června 2004, str. 103–112. 1. vydání, 224 stran. Konvoj, CSTUG, CZLUG, 2004. ISBN 978-80-7302-068-8. URL: `http://www.fi.muni.cz/~xholecek/tex/files/slt04paper.pdf` `http://www.fi.muni.cz/~xholecek/tex/pdfanim.xhtml`

[3] HOLEČEK, Jan – SOJKA, Petr: Animations in pdfTEX-Generated PDF: A New Method for Directly Embedding Animations into PDF. In: Apostolos Syropoulos et al. (Eds): *TEX, XML, and Digital Typography.* Proceedings of TUG 2004 conference, Springer, Lecture Notes in Computer Science 3130, pp. 179–191, August/September 2004, Xanthi, Greece, ISBN 978-3-540-22801-2. URL: `http://www.springerlink.com/content/88k8bbvvg6klfhg9/fulltext.pdf` `http://www.fi.muni.cz/usr/sojka/papers/tugboat04anim.pdf`

[4] PLCH, Roman – ŠARMANOVÁ, Petra: Interaktivní 3D grafika v HTML a PDF dokumentech. In *Zpravodaj Československého sdružení uživatelů TEXu,* ročník 18, číslo 1–2, str. 76–92. Editor Zdeněk Wagner, Česká republika, Praha, 2008. ISSN 1211-6661, e-ISSN 1213-8185. URL: `http://bulletin.cstug.cz/pdf/bul_0812.pdf`

[5] ROEGEL, Denis: Kissing circles: A French romance in METAPOST. In *TUGboat*, ročník 26, číslo 1, str. 10–17. USA, Portland, 2005. ISSN 0896-3207. Překlad vyjde v některém budoucím Zpravodaji CSTUG. URL: `http://tug.org/TUGboat/Articles/tb26-1/tb82roegel.pdf`

Kontaktní adresa

Ing. Pavel STŘÍŽ, Ph.D.
Ústav statistiky a kvantitativních metod
Fakulta managementu a ekonomiky
Univerzita Tomáše Bati ve Zlíně
Mostní 5139, CZ-760 01 Zlín
`striz@fame.utb.cz`

GRAFICKÝ SOFTWARE VE VÝUCE A PRO VÝUKU

TALANDOVÁ, Petra (CZ)

Abstrakt. Otevřený software v oblasti počítačové grafiky nachází využití i ve výuce na vysoké škole, a to jak přímo při vyučování, tak i pro přípravu studijních materiálů. V praktických cvičeních těch předmětů, kde se využívá počítačová grafika, se studenti seznamují s různými grafickými programy, mj. GIMP a Inkscape. Tyto nástroje jsou rovněž využívány i pro přípravu výukových materiálů. Zde se ale navíc objevuje potřeba animovaných prvků, které jsou vhodným doplňkem zejména při přípravě e-learningových materiálů. Jednou z oblastí, kde lze tyto animace využít pro velmi názorné vysvětlení problematiky, je například programování, kde animace pomohou studentům lépe se orientovat v obvyklých postupech řešení. Tento příspěvek se tedy zabývá využitím otevřených rastrových i vektorových grafických programů, včetně animačních programů, pro výuku a přípravu studijních materiálů.

1 Úvod

Jedním z důležitých bodů rozhodování při přípravě projektu (kterým může být pro naše účely i výuka nebo tvorba výukových materiálů) je volba používaného softwaru. Je třeba zohlednit účel softwaru, požadavky na něj kladené a jeho funkčnost, znalosti uživatelů a další. Především je však třeba zajistit fungující týmovou spolupráci a vzít v úvahu platformu softwaru, míru kompatibility programů (nebo více verzí jednoho programu), používané souborové formáty. Důležitým faktorem je i cena. Zatímco podnik může zvolit jednotné (a třeba i komerční) řešení, v prostředí školy se otevírá prostor pro diskuzi nad použitím otevřeného softwaru.

Školy, resp. vyučující nebo ICT koordinátoři, by měli vzít v úvahu následující faktory:

- Cena a licence. Školy si sice obvykle mohou pořídit levnější školní multilicence, nulová cena je však nepřekonatelná. Nejen pro studenty je důležitá i možnost pořídit si software legálně v několika kopiích současně.

- Kompatibilita. Komerční programy se nezřídka vyznačují nekompatibilitou verzí svých programů, je nutné pořizovat stále nové verze. U otevřených programů takové riziko nepozorujeme.
- Platforma. Jak studenti, tak učitelé mohou používat velmi rozdílné operační systémy. Přesto je nutné, aby bylo možné výsledky týmové práce (např. studentské seminární práce, e-learningové opory) bez problémů sdílet. To se opět snáze podaří s otevřeným multiplatformním softwarem.
- Podpora souborových formátů. Lze předpokládat, že otevřený software bude používat standardní souborové formáty, jejichž sdílení a zpracování je pro uživatele (organizačně) jednodušší. U komerčního softwaru bývá využíván vlastní (uzavřený) formát, který je v jiných aplikacích podporován obvykle jen částečně nebo vůbec.
- Účel. Při použití ve výuce je software často používán pro ukázky, představení principů, pro seznámení se s novou problematikou. Cílem není profesionální práce, využití všech detailních možností a precizní ovládání konkrétní verze programu. Pro výukové účely je otevřený software obvykle zcela postačující a má-li komerční software svůj otevřený ekvivalent, je možné jej velmi dobře využít.

Jednou z oblastí, která bývá náplní výuky na školách různých stupňů a zaměření, je počítačová grafika (vektorová a rastrová, popř. animace). Tento příspěvek se tedy zabývá možnostmi použití otevřeného grafického softwaru ve výuce a pro výuku.

2 Grafický software

Výuka počítačové grafiky je velmi rozšířená a na mnoha typech vysokých škol bývá povinnou, nebo alespoň volitelnou součástí výuky. Zároveň jsou grafické programy využívány tvůrci elektronických studijních materiálů, a to napříč všemi obory. Počítačová grafika má proto své nezastupitelné místo a zbývá rozhodnutí o volbě programů.

Ačkoli mnohé důvody hovoří pro využití otevřeného softwaru [5] a řadu otevřených programů využitelných ve výuce lze provozovat pod OS Linux i Windows [7], používá se i ve školním prostředí komerční software. Následující tabulky proto nabízejí stručné srovnání několika parametrů pro vektorové a pro rastrové programy. Výběr parametrů byl proveden s ohledem na předpokládané využití, tj. pro práci ve výuce a pro přípravu výukových materiálů. Proto je největší pozornost věnována podporovaným souborovým formátům, díky nimž je umožněna spolupráce uživatelů i na různých platformách.

V případě vektorových programů byly porovnávány otevřené programy Inkscape [6], OpenOffice.org Draw [3] a komerční program Corel DRAW [1].

Inkscape byl vybrán kvůli své oblibě nejen ve výuce (program se stává jakýmsi výukovým standardem). Jeho značnou výhodou je používání standardního formátu SVG. Program poskytuje rozsáhlé možnosti, které lze vyjádřit v řadě souborových formátů.

Tabulka 1: Srovnání vybraných vlastností vektorových grafických programů

Vlastnost	**Inkscape**	**OO Draw**	**Corel Draw**
Licence	GPL	LGPL	komerční
Cena vč. DPH	0 Kč	0 Kč	12 159 Kč
Operační systémy			
Linux	ano	ano	ne
Windows	ano	ano	ano
Mac OS X	ano	ano	ne
Vlastní formát	(SVG)	ODG	CDR
Vybrané vstupní formáty (vč. importu)			
SVG	ano	ne	ano
PDF	ano	ne	ano
PS, EPS	ne	ano	ano
CDR	ano	ne	ano
DXF	ano	ano	ano
JPG	ano	ano	ano
PNG	ano	ano	ano
GIF	ano	ano	ano
PSD	ne	ano	ano
Vybrané výstupní formáty (vč. exportu)			
SVG	ano	ano	ano
PDF	ano	ano	ne
EPS	ano	ano	ano
TEX	ano	ne	ne
ODG	ano	ano	ne
CDR	ne	ne	ano
DXF	ne	ne	ano
JPG	ne	ano	ano
PNG	ano	ano	ano
GIF	ne	ano	ano
PSD	ne	ne	ano

Podstatně jednodušší, ale pro výuku využitelnou alternativou je program OpenOffice.org Draw. Jeho výhodou je integrace do kancelářského balíku a tedy možnost mít vždy přístup ke kreslicímu programu. Podpora různorodých formátů je v tomto případě menší.

Jako zástupce komerčních programů byl zvolen CorelDRAW. Aplikace vyniká propra-

covanými možnostmi ovládání a použití i cenou. [1] Komerční verze pak s sebou nese všechny nevýhody popsané v úvodu.

Tabulka 2: Srovnání vybraných vlastností rastrových grafických programů

Vlastnost	**GIMP**	**Corel PhotoPaint**
Licence	GNU GPL	komerční
Cena vč. DPH	0 Kč	12 159 Kč
Operační systémy		
Linux	ano	ne
Windows	ano	ano
Mac	ano	ne
Vlastní formát	XCF	CPT
Vybrané vstupní formáty (vč. importu)		
JPG	ano	ano
PNG	ano	ano
GIF	ano	ano
PSD	ano	ano
SVG	ano	ne
PDF	ano	ne
PS, EPS	ano	ano
CDR	ne	ano
Vybrané výstupní formáty (vč. exportu)		
JPG	ano	ano
PNG	ano	ano
GIF	ano	ano
PSD	ano	ano
PS, EPS	ano	ano

V případě rastrových programů byl porovnáván otevřený program GIMP [4] s komerčním programem Corel Photopaint [1].

GIMP je v rastrové grafice, podobně jako Inkscape ve vektorové grafice, velmi oblíben nejen ve výuce, ale i pro jiné použití. Nabízí rozsáhlé možnosti a pro účely spolupráce je vybaven řadou vstupních a výstupních formátů. Komerční alternativou je program Corel Photopaint, který je také součástí balíku programů.

Vhodnou dvojicí pro výuku počítačové grafiky může být (a v mnoha případech je) právě sada CorelDRAW Graphics Suite. Programy v ní obsažené nabízejí rozsáhlé možnosti

[1] Jde o cenu za celou novou sadu CorelDRAW Graphics Suite X5 u prodejce svetsoftware.cz ze dne 8. 6. 2010.

využitelné na profesionální úrovni a díky množství podporovaných souborových formátů by snad bylo možné očekávat snadnou spolupráci s jinými programy.

Otevřenou alternativou je kombinace Inkscape + GIMP. Tato „sada“ nabízí srovnatelné možnosti a také umožňuje otevírat a uchovávat data v různých formátech. To při týmové práci znamená větší flexibilitu a lepší možnosti spolupráce. Nezanedbatelnou výhodou je ovšem i multiplatformnost a typ licence. Cena tohoto „balíku“ je nulová a při nákupu softwaru tak lze ušetřit až několik tisíc korun.

Dalším rysem, který obě skupiny programů rozděluje, je uživatelské rozhraní a způsob ovládání. Zatímco programy společnosti Corel používají jediné okno s pracovní plochou a panely nástrojů, GIMP a Inkscape se vyznačují uspořádáním složeným z pracovního okna a samostatně umístěných panelů nástrojů. Důležité však je, že spolupracující programy jsou si z hlediska ovládání podobné a nabízejí dostatek možností pro výměnu dat.

Z uvedeného vyplývá, že při rozhodování o použití softwaru je možné zvolit obě varianty, limitující v tomto případě však zřejmě budou finanční prostředky.

3 Grafický software a vysokoškolská výuka

Využití grafického softwaru ve výuce bude uvedeno na příkladu studijních programů akreditovaných na Provozně ekonomické fakulty Mendelovy univerzity v Brně. Ekonomická fakulta nabízí studijní programy Ekonomika a management, Hospodářská politika a správa, Ekonomická informatika a Inženýrská informatika. Žádný z těchto studijních programů není primárně zaměřen na grafiku, grafický design, návrhy, projektování nebo konstruování, přesto má počítačová grafika ve výuce své místo.

Studenti všech oborů studijních programů Ekonomika a management a Hospodářská politika a správa (celkem cca 650 studentů ročně) absolvují povinný předmět *Informatika pro ekonomy I*, který je věnován prezentaci informací. Jeho součástí je i úvod do principů počítačové grafiky (vektorové i rastrové). Pro výuku se většinou používá dvojice programů CorelDRAW a Corel PhotoPaint, které jsou na učebnách nainstalovány. Vzhledem k počtu studentů a organizaci výuky je tento software nainstalován v pěti učebnách, v různých verzích a s příležitostnými funkčními problémy. Také obnova licencí je finančně náročná; v souvislosti s budováním nové velkokapacitní počítačové učebny to znamená značný růst výdajů za softwarové licence.

Na tento předmět navazuje volitelný kurz *Moderní počítačové aplikace*, kde je kromě jiných témat věnován prostor rozšíření znalostí z počítačové grafiky. Vzhledem k odlišnému charakteru předmětu zde nachází větší využití otevřený software, tedy Inkscape pro vektorovou grafiku a GIMP pro rastrovou grafiku a animace. Studenti tedy pracují s jinými programy než v předmětu Informatika pro ekonomy I. Díky znalosti principů počítačové grafiky (nikoli ovládání konkrétního komerčního softwaru) to však nedělá problémy a studenti si na otevřený software rychle zvykají.

Studenti všech oborů mohou studovat předmět *Počítačová grafika*, který je zaměřen na podrobnější práci s rastrovou i vektorovou grafikou a na základy 3D modelování. Také zde se používá mj. sada programů firmy Corel, ačkoli i využití otevřeného softwaru by přicházelo

v úvahu. Dále si studenti v rámci studijních specializací mohou zvolit směr zaměřený na počítačovou grafiku, jehož součástí jsou i předměty Digitální fotografie nebo Animace a geoprostor. Tyto specializované předměty však vyžadují i specializované (komerční) vybavení.

Grafický software se dále okrajově využívá i v ostatních předmětech (např. Zpracování dat pro manažery) a při zpracování seminárních prací nebo tvorbě dokumentace. Rozhodnutí o výběru softwaru je ponecháno na studentech, bývají využívány otevřené i komerční programy. (Bylo pozorováno, že uživatelé MS Windows si častěji vybírají komerční programy, i když je k dispozici jejich otevřená alternativa.)

4 Grafický software pro e-learning

Nedílnou součástí moderní výuky jsou i elektronické studijní opory (e-opory), jejichž častými prvky jsou obrázky, kresby a animace. Příkladem může být nově vytvářená e-opora pro předmět Programovací techniky, který je vyučován jako povinný pro studenty programů Ekonomická informatika a Inženýrská informatika na PEF Mendelovy univerzity v Brně. Programovací techniky navazují na základní kurz Algoritmizace, předpokládá se tedy, že studenti mají základní znalosti programování, na kterých je možno dále stavět. Hlavní (a pro studenty velmi složitá) část předmětu je věnována dynamickým datovým strukturám a jejich zpracování. To se odráží i ve struktuře e-opory, která je věnována především dynamickým datovým strukturám a vyplňuje prázdné místo mezi ostatními dostupnými studijními materiály.

Vysvětlení této problematiky vyžaduje značné množství obrázků a – pro snazší pochopení – také animací, které vizualizují průběh algoritmu a upozorňují na důležitá místa řešení. Vizualizace (a nejen u algoritmů) se při e-learningu využívá poměrně často, jak dokládají bakalářské a diplomové práce zpracovávané na toto téma (např. [2, 8]). Velmi často je pro přípravu animovaných materiálů použit program Flash, který se stal „de facto standardem“ pro použití animací na webu. Těchto animací vzniká velké množství a na různá témata.

Zpracovávaný projekt zaměřený na e-opory pro předmět Programovací techniky se vydá jinou cestou. Součástí e-opory budou nejen obecná vysvětlení, ale také komentované a ilustrované příklady s postupem a řešením a dále příklady k samostatnému procvičení, opět s řešením. Důkladné zpracování příkladů vyžaduje, aby příklad obsahoval nejen text, ale také obrázky nebo animace „na míru“.

Především díky požadavkům na sdílení a přenositelnost byly pro tvorbu grafických prvků e-opor vybrány otevřené programy. Sem patří již zmiňované aplikace Inkscape a GIMP, pro animace (byť jednoduché) přichází v úvahu vektorová animace (např. program Synfig) nebo trojrozměrná animace (Blender). S využitím těchto programů bude možné vytvořit animaci přímo ke konkrétnímu příkladu, podle potřeby. Elektronické studijní opory pro předmět Programovací techniky budou dostupné v e-learningovém subsystému Univerzitního informačního systému přes webové rozhraní. Další výhodou proto je uplatnění standardizovaných souborových formátů, které je možné použít přímo v prohlížeči.

5 Závěr

Příspěvek se zabýval možnostmi využití programů pro práci s počítačovou grafikou ve výuce i při přípravě studijních materiálů. Ze zkušeností s výukou vyplývá, že přechod na otevřený software by byl jistě možný, alespoň v základních kurzech zaměřených na grafiku. Prozatím se však stále využívá komerční software, který splňuje i náročné požadavky na vlastnosti grafických aplikací. Dalším aspektem je příprava studijních materiálů a vysvětlujících příkladů, které vyžadují nové vytvoření obrázků a animací. Zde je jeví jako vhodnější použití otevřených aplikací, které nabízejí mnohé výhody a vytvoření stejně kvalitních výsledků. Postupný vývoj by mohl směřovat k uplatnění „duálního přístupu“. Studenti by měli možnost seznámit se jak s profesionálním (ale komerčním) vybavením, se kterým mimo školu obvykle nepřijdou do kontaktu, tak také s otevřeným a stejně kvalitním softwarem. Jako studenti ekonomické fakulty pak mohou posoudit, které řešení je výhodnější.

Poděkování

Příspěvek vnikl s podporou grantové agentury FRVŠ v rámci řešení projektu 1719/2010/F1/d „Tvorba e-learningových opor pro předmět Programovací techniky“.

Literatura

[1] *Corel* 2010 [cit. 2010-06-01] URL: `http://apps.corel.com/int/cz/`

[2] DOSOUDIL, V.: *Software pro podporu výuky datových struktur*. Bakalářská práce. 2005 [cit. 2010-06-01] URL: `http://is.muni.cz/th/60914/fi_b/book.pdf`

[3] *Draw*. 2010 [cit. 2010-06-01] URL: `http://www.openoffice.cz/draw`

[4] GIMP.CZ: *GIMP*.2010 [cit. 2010-06-01] URL: `http://www.gimp.cz/`

[5] HULA, D.: *Open source software při výuce grafiky*. 2009 [cit. 2010-06-01] URL: `http://ui.pefka.mendelu.cz/files/hula.pdf`

[6] *Inkscape* 2010 [cit. 2010-06-01] URL: `http://inkscape.org/`

[7] PECH, J.: *Tvorba e-learningových materiálů pro využití volně šiřitelných programů ve výuce informatiky*. [cit. 2010-06-01]
URL: `http://everest.natur.cuni.cz/konference/2006/prispevek/pech.pdf`

[8] SKUTKA, O.: *Vizualizace geometrických algoritmů pro potřeby výuky*. Diplomová práce. 2006 [cit. 2010-06-01] URL: `http://is.muni.cz/th/39249/fi_m/diplomka.pdf`

Kontaktní adresa

Petra TALANDOVÁ (Ing.),
Ústav informatiky, PEF, Mendelova univerzita v Brně, Zemědělská 1
613 00 Brno, `petra.talandova@mendelu.cz`

NAVRHOVÁNÍ ŘÍZENÍ SVĚTELNÝCH KŘIŽOVATEK PETRIHO SÍTĚMI

TUREK, Michal, (CZ)

Abstrakt. Příspěvek se zabývá navrhováním logiky řízení světelných křižovatek Petriho sítěmi. V první části příspěvku je uvedeno definování problému a proveden teoretický rozbor Petriho sítí. V další části příspěvku je proveden návrh logiky řízení světelné křižovatky na základě Petriho sítí. V závěrečné části je provedeno vyhodnocení.

1 Úvod

Zvyšující se zájem o individuální automobilovou dopravu ve 20. století představuje vzrůstající počty vozidel na jednoho obyvatele, což však na přetížených komunikacích a křižovatkách přináší zpomalování, resp. zastavování vozidel vedoucího ke vzniku kongescí, které jsou příčinou enormního a nadbytečného znečišťování ovzduší a souvisejících negativních vlivů. Z hlediska vzniku kongescí jsou nejkritičtějšími místy ve městech křižovatky a jejich okolí, situace může být kritická především u neřízených křižovatek. Za účelem odstraňování uvedených problémů na křižovatkách se vypracovávají návrhy úprav ve více variantách s dopravně ekonomickým posouzením vzhledem k životnosti řešení. Na základě dopravního a ekonomického posouzení je nutné vyhledat z dlouhodobého hlediska co nejpříznivější řešení.

2 Motivace

Vzhledem k tomu, že Petriho sítě představují nástroj určený k modelování a analýze procesů, lze předpokládat jejich možné uplatnění i při návrhu řídící logiky světelných křižovatek. Řídící logika světelných křižovatek musí z důvodu zachování základního principu světelného řízení křižovatek umožnit současné jízdy pouze nekolizním, resp. podmíněně kolizním dopravním proudům, aby byla zajištěna bezpečnost a plynulost provozu.

V případě, že pro návrh řídící logiky využijeme kromě uznávaných konvenčních přístupů [1], příp. [2], také Petriho sítí, pak můžeme použít libovolnou Petriho síť, protože veškeré Petriho sítě obsahují prvky, jimiž lze modelovat všechny rozhodující aspekty světelně řízené křižovatky (návěstidla, světelné signály, změny světelných signálů).

Z široké skupiny Petriho sítí lze pro návrh logiky řízení světelných křižovatek využít především P/T Petriho sítí, protože obsahují pouze prvky, resp. vlastnosti těchto prvků, které pro návrh řídící logiky postačují. Další studium problematiky pak může poukázat na složitější typy Petriho sítí, kterých bude muset být využíváno. Při vytváření struktury P/T Petriho sítě se prvky sestavují tak, aby prostřednictvím pravidel pro uskutečňování přechodů byl splněn základní princip světelného řízení křižovatek, tj. zajištěno správné přepínání světelných signálů na návěstidlech světelného signalizačního zařízení a koordinace tohoto přepínání u všech světelných návěstidel pro jednotlivé směry.

Na základě výše uvedených informací bude v předloženém článku pozornost věnována popisu P/T Petriho sítí a pravidel pro uskutečňování přechodů v P/T Petriho sítích.

3 Petriho sítě

Petriho sítěmi je označována široká třída diskrétních matematických modelů, které umožňují popisovat specifickými prostředky řídící toky a informační závislosti uvnitř modelovaných systémů. Více než čtyřicetiletá historie Petriho sítí je charakterizována postupným rozvojem jednotlivých Petriho sítí, při kterém byly na základě jednoduchých Petriho sítí vytvářeny složitější Petriho sítě, jenž jsou definovány jako Petriho sítě vysoké úrovně.

Petriho sítě vysoké úrovně umožňují zohlednění mnoha aspektů, např. zahrnutí pojmu času, resp. trvání, vytváření hierarchické struktury nebo provázání různých částí systému (např. světelných signálů na světelných signalizačních zařízeních a intenzit jednotkových vozidel) prostřednictvím různých typů tokenů, čímž bude docíleno dokonalejší úrovně modelování reálných problémů.

Přehled Petriho sítí vysoké úrovně [5]:

- P/T Petriho sítě s inhibičními hranami,
- P/T Petriho sítě s prioritami,
- časované (Timed) Petriho sítě,
- barevné (Coloured) Petriho sítě,
- hierarchické (Hierarchical) Petriho sítě
- objektové (Objected Oriented) Petriho sítě.

V dalším textu budou podrobněji popsány P/T Petriho sítě, které mohou být použity pro návrh logiky řízení světelných křižovatek. Uvedený popis P/T Petriho sítí, grafické vyjádření P/T

Petriho sítí a definování pravidel pro uskutečňování přechodů v P/T Petriho sítích lze v určité míře vztáhnout na libovolnou Petriho síť.

3.1 P/T Petriho sítě

P/T Petriho sítě tvoří místa (places), přechody (transitions), orientované hrany (arcs) a značky (tokens). Každému místu p je přiřazena kapacita k udávající maximální počet tokenů, který se v místě p může současně nacházet a každé hraně a je přiřazena násobnost w udávající maximální počet tokenů, který se po hraně a může současně přesunovat. Kapacity míst a násobnosti hran jsou představovány ohodnocením míst, resp. ohodnocením hran. Místo p bez ohodnocení je považováno za místo s neomezenou kapacitou. Hrana a bez ohodnocení je považována za jednoduchou (s násobností 1).

Počáteční stav P/T Petriho sítě je dán výchozím značením, při kterém se zajišťuje splnění vstupních míst přechodů, které mají být provedeny. Uskutečněním proveditelných přechodů se přesouvají tokeny do výstupních míst uskutečněných přechodů, čímž dochází ke změně stavu P/T Petriho sítě. Výstupní místa uskutečněných přechodů jsou zároveň vstupními místy přechodů, které mají být teprve uskutečněny. Uskutečněním dalších přechodů dochází opět ke změně stavu P/T Petriho sítě. Uvedený princip se neustále opakuje a může být ukončen, jestliže z místa nevedou žádné orientované hrany.

Obrázek 1: Příklad změny stavu v P/T Petriho síti

Změny stavů P/T Petriho sítí se uskutečňují podle následujících zásad [5]:

- stav sítě je určen značením, tj. počtem tokenů v každém místě,
- místo p patří do vstupní množiny přechodu t, jestliže z místa p vede orientovaná hrana do přechodu t,
- místo p patří do výstupní množiny přechodu t, jestliže z přechodu t vede orientovaná hrana do místa p,
- přechod t je proveditelný, jestliže pro každé místo p vstupní množiny přechodu t platí, že obsahuje alespoň tolik tokenů, kolik činí násobnost hrany vedoucí z místa p do přechodu t,
- přechod t je proveditelný, jestliže pro každé místo p výstupní množiny přechodu t platí, že násobnost hrany vedoucí z přechodu t do místa p nepřevyšuje kapacitu místa p,

- při změně stavu se nejdříve počet tokenů v každém vstupním místě p přechodu t zmenší o násobnost hrany w spojující místo p s p řechodem t a následně se počet tokenů v každém výstupním místě p přechodu t zvětší o násobnost hrany w spojující přechod t s místem p .

4 Software pro sestavení Petriho sítí

Při návrhu světelného řízení byl využit pro sestavení a simulaci Petriho sítě software Snoopy a Pipe2. V následující části příspěvku bude provedeno seznámení s uvedenými software.

4.1 Software Snoopy

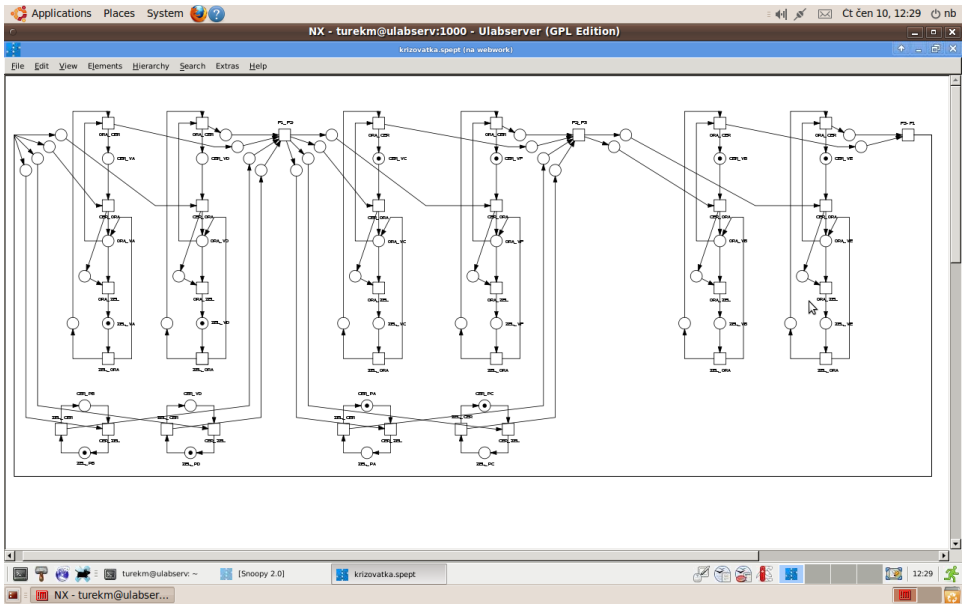
Při spuštění software se zadává v nabídce Templates (šablony) typ Petriho sítě, se kterou bude dále pracováno. Po zadání typu Petriho sítě se kromě základní obrazovky umožňující práci se souborem zobrazí také obrazovka umožňující sestavení Petriho sítě prostřednictvím přehledného menu a p racovní plochy. Výběrem příslušné položky z menu se sestavují jednotlivé prvky Petriho sítě na p racovní plochu, přičemž u každého místa je možné definovat Name (název), Marking (značení), Comment (komentář) a Graphic (grafickou podobu), u každého přechodu je možné definovat Name (název), Comment (komentář) a Graphic (grafickou podobu), pro každou hranu je možné definovat Multiplicity (násobnost), Comment (komentář) a Graphic (grafickou podobu). Po sestavení Petriho sítě se výběrem položky Start Anim-mode zadávají parametry simulace Referesh (obnovení), Duration (trvání) Stepping (posílení) a následně může dojít ke spuštění simulace. Simulace v software Snoopy je velmi přehledná, protože se tokeny přesouvání po jednotlivých hranách a jsou zvýrazněny červenou barvou.

Pro názornost bude nyní uvedeno pracovní prostředí software Snoopy (Obr. 2).

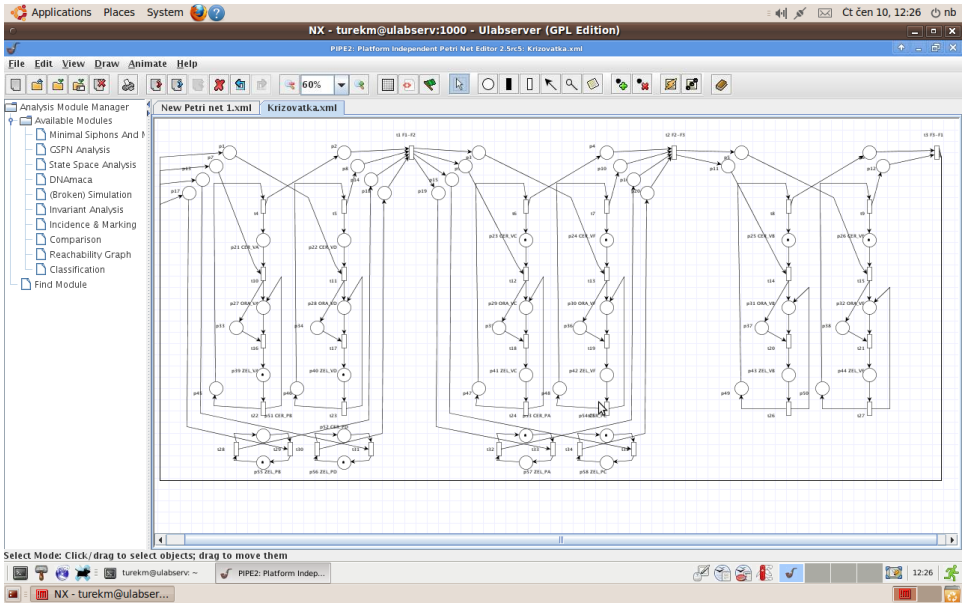
4.2 Software Pipe2

Při spuštění software se zobrazí menu a obrazovka rozdělená na dvě části, v jedné části se náchází Analysis Module Manager umožňující rozšířenou analýzu Petriho sítí a druhá část představující pracovní plochu, v níž se Petriho sítí sestavují. Výběrem příslušné položky z menu se sestavují odpovídající prvky Petriho sítě na pracovní plochu, přičemž u každého místa je možné definovat Name (název), Marking (značení), Capacity (kapacita), u každého přechodu je možné definovat Name (název), Rate (sazba) a Timing (časování), pro každou hranu je možné definovat Multiplicity (násobnost). Po sestavení Petriho sítě se definuje přechod, po němž dojde k ukončení simulace, a následně může dojít ke spuštění simulace. Při simulaci v software Pipe2 nedochází k přesunu tokenů po hranách, pouze se mění počty tokenů v jednotlivých místech, což snižuje přehlednost.

Pro názornost bude nyní uvedeno pracovní prostředí software Pipe2 (Obr. 3).



Obrázek 2: Pracovní prostředí software Snoopy



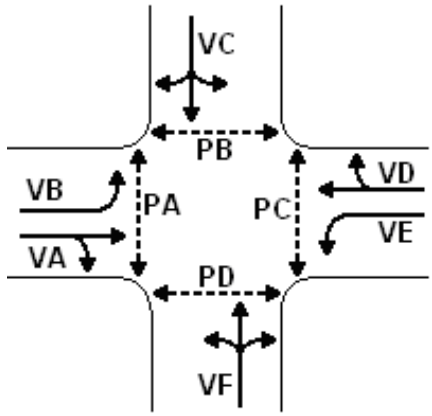
Obrázek 3: Pracovní prostředí software Pipe2

5 Návrh řízení světelné křižovatky Petriho sítěmi

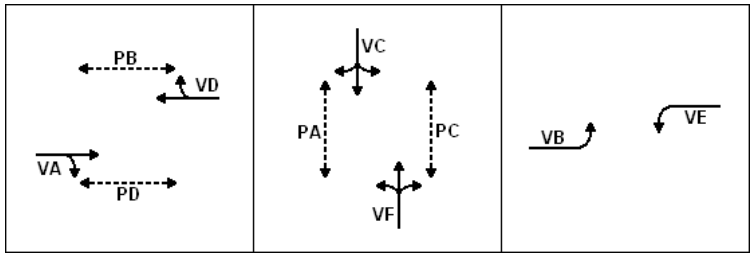
Za účelem dalšího zkoumání, kterým se budu zabývat v rámci disertační práce, byla navržena logika řízení světelné křižovatky prostřednictvím Petriho sítí. Kroky, které je nutné při návrhu logiky řízení světelných křižovatek provést, jsou popsány v následující části příspěvku.

5.1 Analýza vstupních podkladů

Pro návrh logiky řízení světelné křižovatky byly zohledněny následující vstupní podklady: schéma vzorové křižovatky (Obr. 4) a fázové schéma (Obr. 5), prostřednictvím kterého se zajišťují časový intervaly, v nichž mají současně volno určité, zpravidla nekolizní dopravní pohyby na křižovatce. Pro návrh světelného řízení je pochopitelně zapotřebí ještě celá řada údajů, ty však s návrhem řídící logiky nesouvisejí.



Obrázek 4: Schéma vzorové křižovatky



Obrázek 5: Fázové schéma vzorové křižovatky

5.2 Sestavení Petriho sítě

Před sestavením Petriho sítě byl stanoven význam prvků v zamýšlené Petriho síti. Místa v Petriho síti budou představovat světelné signály na návěstidlech SSZ, přechody v Petriho síti budou představovat okamžiky, při kterých dochází k přepínání světelných signálů, orientované hrany v Petriho síti umožňují realizovat změny stavů a aktuální poloha tokenu bude reprezentovat aktuální návěstní znak.

Při sestavování Petriho sítě se vychází z fázového schématu, v tomto případě se jednalo o fázové schéma uvedené na obrázku 5. Nejdříve byla sestavena část Petriho sítě představující první fázi, resp. světelné signály na návěstidlech pro dopravní proudy, které se v první fázi vyskytují, následně byla sestavena část Petriho sítě představující druhou fázi, resp. světelné signály na návěstidlech pro dopravní proudy, které se vyskytují ve druhé fázi a část Petriho sítě představující třetí fázi, resp. světelné signály na návěstidlech pro dopravní proudy, které se vyskytují ve třetí fázi. Na závěr byly části Petriho sítě propojeny tak, aby mohlo docházet ke střídání fází v rámci cyklu.

Sestavená Petriho síť, do které byla doplněna světelná signalizace příslušných světelných signálů v jednotlivých fázích prostřednictvím tokenů (počáteční značení), je uvedena na obrázku 6.

Návrh řídící logiky spočívá ve dvou etapách:

1. návrh správných posloupností jednotlivých světelných signálů na návěstidle,
2. návrh správných posloupností jednotlivých světelných signálů u jednotlivých návěstidel zajišťující správné přechody mezi fázemi.

V první etapě bylo zajištěno, že po světelném signálu zeleného světla bude zařazen světelný signál oranžového světla a světelný signál červeného světla. Dále bylo zajištěno, že po světelném signálu červeného světla bude zařazen světelný signál oranžového světla a světelný signál zeleného světla. Na obrázcích 7. a 8. je uvedeno, jak pomocí P/T Petriho sítě navrhnout správnou aktivaci návěstních znaků na návěstidle.

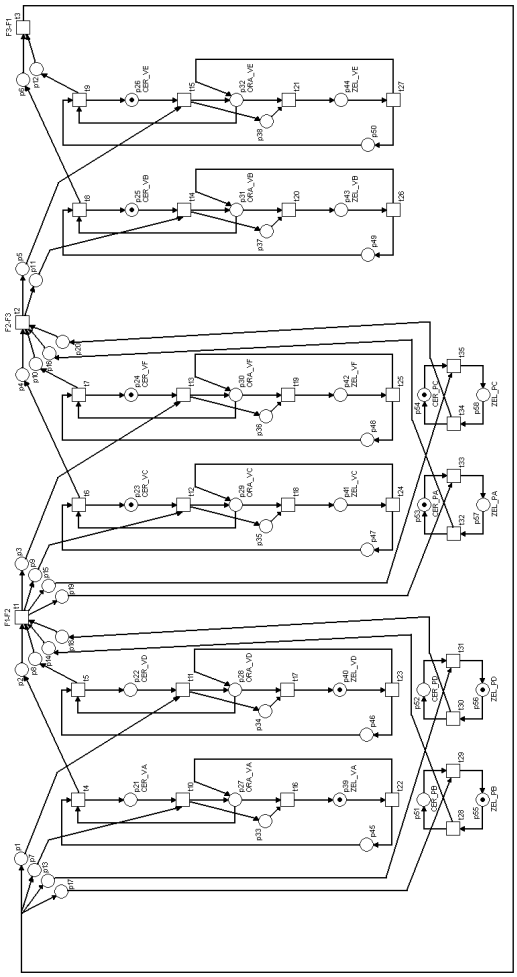
Komentář k principu uvedeném na obrázku 7

Po světelném signálu zeleného světla p_{39} se prostřednictvím přechodu t_{22} přesouvá token do místa představujícího světelný signál oranžového světla p_{27} a doplňujícího místa p_{45}, následně může dojít pouze k přechodu t_4, při kterém se token přesouvá do místa představujícího světelný signál červeného světla p_{21}.

Komentář k principu uvedeném na obrázku 8

Po světelném signálu červeného světla p_{21} se prostřednictvím přechodu t_{10} přesouvá token do místa představujícího světelný signál oranžového světla p_{27} a doplňujícího místa p_{33}, následně může dojít k přechodu t_{16}, při kterém se token přesouvá do místa představujícího světelný signál zeleného světla p_{39}.

V popsaných principech se vyskytují doplňující místa p_{33} a p_{45}, která slouží k tomu, aby v daném časovém okamžiku mohly nastat pouze očekávané přechody, resp. očekávané

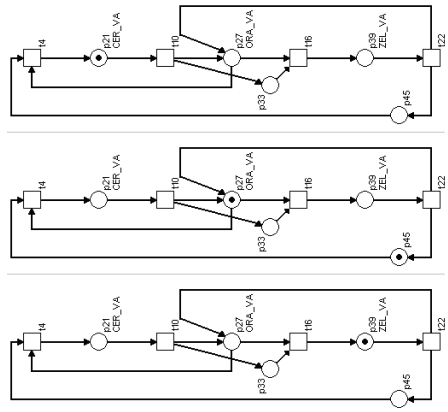


Obrázek 6: Návrh řídící logiky světelné křižovatky v P/T Petriho síti

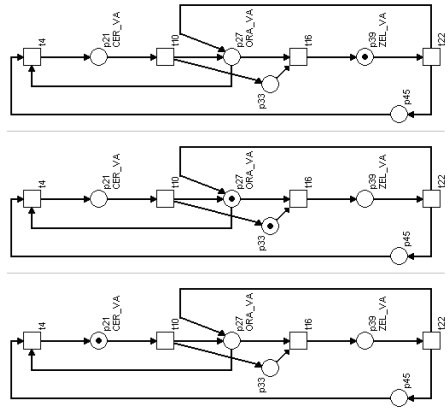
změny světelných signálů. Nemůže tedy dojít k situaci, že po světelném signálu zeleného světla bude zařazen světelný signál oranžového světla a světelný signál zeleného světla, namísto červeného, resp. k situaci, že po světelném signálu červeného světla bude zařazen světelný signál oranžového světla a světelný signál červeného světla, namísto zeleného.

Kromě částí Petriho sítě zajišťujících správnou řídící logiku týkající se jednotlivých návěstidel v první, druhé a třetí fázi, se v sestavené Petriho síti nacházejí doplňující místa, jež zajišťují správnou funkci logiky řízení světelné křižovatky při přechodech mezi jednotlivými fázemi.

Ve druhé etapě bude zajištěno, aby signály zeleného světla v následující fázi mohly



Obrázek 7: Změna světelného signálu zeleného světla

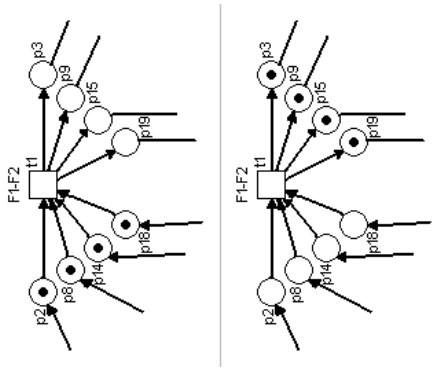


Obrázek 8: Změna světelného signálu červeného světla

nastat až poté, co v předchozí fázi nastaly signály červeného světla, prostřednictvím principu uvedeného na obrázku 9.

Komentář k principu uvedeném na obrázku 9

Současně s umístěním tokenů do míst představujících světelný signál červeného světla pro dopravní proudy, které vyskytují v předchozí fázi, se přesouvají tokeny také do doplňkových míst p_2, p_8, p_{14}, p_{18}, jež představují vstupní místa přechodu t_1. Uskutečněním přechodu t_1 je následně realizována změna fází. Po změně fází se nacházejí tokeny v doplňkových místech p_3, p_9, p_{15}, p_{19}, jež představují výstupní místa přechodu t_1 a mohou tak nastat světelné signály zeleného světla pro dopravní proudy, které se vyskytují v následující fázi.



Obrázek 9: Změna fází

V popsaném principu se vyskytují doplňující místa p_2, p_8, p_{14}, p_{18} a p_3, p_9, p_{15}, p_{19}, která slouží k tomu, aby v daném časovém okamžiku mohly nastat pouze očekávané přechody, resp. očekávané změny světelných signálů při přechodu mezi jednotlivými fázemi.

Závěr

Pro návrh logiky řízení světelných křižovatek splňující základní princip světelného řízení křižovatek a zajišťující správné přepínání světelných signálů na návěstidlech světelného signalizačního zařízení byla sestavena P/T Petriho síť v software Snoopy a Pipe2, následně byla spuštěna simulace sestavené P/T Petriho sítě.

Vybrané software mají z hlediska ovládání obdobné vlastnosti. Významný rozdíl mezi software se nachází v případě modelovacích schopností, protože software Pipe2 umožňuje stanovení kapacity jednotlivých míst a časování jednotlivých přechodů, což může být přínosné především při vytváření složitějších návrhů. Při simulaci vykazuje větší přehlednost a lepší názornost software Snoopy, protože se po hranách přesunují červeně označené tokeny. V případě software Pipe2 se mění pouze počet tokenů v jednotlivých místech Petriho sítě a nedochází k přesunu tokenů po hranách Petriho sítě.

Postupným prohlubováním a obohacováním návrhu logiky řízení světelných křižovatek prostřednictvím Petriho sítí, ve kterých bude zohledněna širší množina vstupních podkladů (např. intenzity jednotkových vozidel, trvání světelných signálů), bude možné docílit vytváření signálních plánů v software Pipe2. Petriho sítě tedy mohou nabízet další alternativní způsob, pomocí kterého je možné navrhovat světelné řízení křižovatek.

Literatura

[1] ČERNÝ, J. – KLUVÁNEK, P.: Základy matematickej teorie dopravy. Bratislava: VEDA, 1990. ISBN 80-224-0099-8

[2] ČERNÝ, J. – ČERNÁ, A.: Teorie řízení a rozhodování v dopravních systémech. Pardubice: Institut J. Pernera, 2004. ISBN 80-86530-15-9

[3] TUREK, M.: Návrh světelného řízení křižovatky Dolní–Kralická v Prostějově. Ostrava: VŠB-Technická univerzita Ostrava, 2009. (Diplomová práce)

[4] VONDRÁK, I.: Metody byznys modelování. Ostrava: VŠB – Technická univerzita Ostrava, 2004

[5] MARKL, J.: Petriho sítě I. Ostrava: VŠB – Technická univerzita Ostrava, 2009.

[6] KOCHANÍČKOVÁ, M.: Petriho sítě. Olomouc: Univerzita Palackého, 2008.

[7] Snoopy. Oficiální stránka. Dostupné na internetu: `http://www-dssz.informatik.tu-cottbus.de/index.html?/software/snoopy.html`

[8] Pipe2. Oficiální stránka. Dostupné na internetu: `http://pipe2.sourceforge.net`

Kontaktná adresa

Michal TUREK (Ing.),
Institut dopravy FS VŠB-TU Ostrava,
17. listopadu 15, 708 33 Ostrava – Poruba,
`michal.turek@atlas.cz`

KOORDINACE LINEK MHD S VYUŽITÍM PETRIHO SÍTÍ

TUREK, Richard, (CZ)

Abstrakt. Příspěvek se zabývá modelováním pohybu vozidel v síti linek MHD pomocí Petriho sítí. V úvodu je uveden význam problému tvorby sítě linek MHD a motivace využití nástroje Petriho sítí. Následuje teoretická analýza včetně základní koncepce a rozdělení Petriho sítí. Pomocí P/T Petriho sítí je nastíněno řešení konkrétního příkladu.

1 Úvod

Městská hromadná doprava zaujímá v dopravní soustavě každého státu nezastupitelnou funkci z hlediska každodenní mobility obyvatelstva. Při jejím plánování je žádoucí zabývat se kromě tvorby tras linek také modelováním pohybu vozidel v navržené síti, což může souviset s hledáním optimální varianty provozu.

V rámci procesu optimalizace MHD v tuzemsku i zahraničí existují matematické modely na tvorbu systémů linek MHD, které využívají pokročilé metody operačního výzkumu. K rozhodnutí, které linky vybrat tak, aby byla pokryta poptávka cestujících a určit jaký počet vozidel je třeba přiradit linkám, je využívána např. metoda PRIVOL (Přiřazení Vozidel Linkám), která patří mezi úlohy lineárního programování [1], [2].

K dalším nástrojům, které jsou předmětem zájmu široké vědecké komunity, patří také Petriho sítě. Petriho sítě se dostaly do popředí zájmu v souvislosti s aplikacemi pro modelování a teoretické zkoumání distribučních a paralelních systémů, jako jsou komunikační protokoly, počítačové sítě či databázové systémy. V rámci modelování v oblasti dopravy se hovoří především o barevných Petriho sítích.

2 Motivace

Petriho sítě představují významný formalismus pro modelování diskrétních systémů, který spojuje výhody srozumitelného grafického zápisu a možnosti simulace s dobrou formální

analyzovatelností. Srozumitelnost a analyzovatelnost Petriho sítí je dána jejich jednoduchostí. Model je popsán místy (places), která obsahují stavovou informaci ve formě značek (tokens), přechody (transmisions), které vyjadřují možné změny stavu a hranami (arcs), propojujícími místa a přechody navzájem. Existuje celá řada typů Petriho sítí a jejich speciálních podtřídy, až po vysokoúrovňové (High-Level Petri nets) a barevné sítě [5].

Existence různých variant Petriho sítí souvisí se snahou zvyšovat modelovací schopnosti a úroveň popisu modelu (přiblížit příslušný formalismus modelovaným skutečnostem) a přitom zachovat konceptuální jednoduchost, která je pro Petriho sítě příznačná. Obecně platí, že vyšší typy Petriho sítí jsou poněkud hůře analyzovatelné, poskytují však vyšší komfort při modelování.

Uplatnění Petriho sítí je značně široké, v oblasti dopravy však používání Petriho sítě doposud jako modelovacího, resp. optimalizačního nástroje příliš nerozšířilo. Smyslem je tedy zkoumat, zda Petriho sítě nemohou určitým způsobem přispět k řešení některých optimalizačních problémů v dopravě, případně odstranit některé nevýhody v současnosti používaných řešících nástrojů. V předloženém článku bude pomocí Petriho sítě prezentován pohyb vozidel v linkové síti MHD (obecně však jakékoliv síti linek).

3 Petriho sítě

Jak je uvedeno např. v [3], využívají Petriho sítě ke své výstavbě následující koncept.

Parciální stavy systému jsou modelovány místy a možné jevy, které jsou aktivátorem změny, jsou definovány přechody. Okamžitý stav systému je definován umístěním značek (tokens) v místech, což se v grafu Petriho sítě vyjadřuje tečkami v místech. Přítomnost tokenu v místě modeluje skutečnost, že daný stav je momentálně aktuální. Každý přechod má definována vstupní a výstupní místa, což je v grafu Petriho sítě vyjádřeno orientovanými hranami mezi místy a přechody. Tím je dáno, které aspekty stavu systému podmiňují výskyt odpovídající události (provedení přechodu), a které aspekty stavu jsou výskytem této události ovlivněny.

Pro každý přechod jsou definovány vstupní a výstupní podmínky. Přechod může být proveden pouze v případě, že všechna jeho vstupní místa obsahují značky, tj. má splněny všechny vstupní podmínky. Provedením přechodu se odstraní značky ze vstupních míst (vstupní podmínky přestanou platit) a umístí se nové značky do výstupních míst (uplatní se výstupní podmínky). Provedení přechodu je atomická operace, která odpovídá výskytu události.

K modelování provozu bude v předloženém článku využito P/T Petriho sítí, které patří k základním Petriho sítím. P/T Petriho síť je tvořena následujícími objekty:

- místy (places), graficky reprezentovanými kružnicemi,
- přechody (transitions), graficky reprezentovanými obdélníky, orientovanými hranami (arcs), graficky reprezentovanými šipkami směřujícími od míst k přechodům nebo od přechodů k místům,

- udáním kapacity (capacity indications) pro každé místo sítě, tj. přirozeného čísla udávajícího maximální počet tokenů, který se může v místě nacházet,
- udáním váhy (weights) pro každou hranu sítě, tj. přirozeného čísla udávajícího násobnost hrany,
- udáním počátečního značení (initial marking), udávajícího počet tokenů pro každé místo sítě.

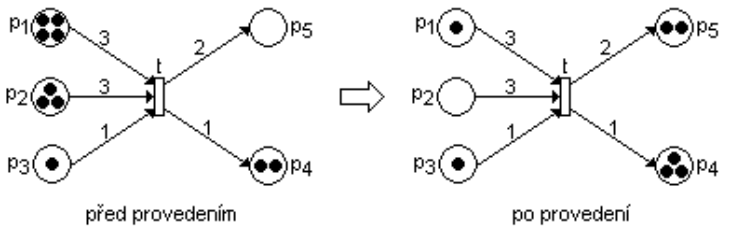
Změny stavů (značení) P/T Petriho sítě jsou charakterizovány následujícími pravidly:

- stav sítě je určen značením, tj. počtem tokenů v každém místě,
- místo p patří do vstupní množiny (pre-set) přechodu t, jestliže z místa p vede hrana do přechodu t a místo p patří do výstupní množiny (post-set) přechodu t, jestliže z přechodu t vede hrana do místa p,
- přechod t je proveditelný (enabled, activated), jestliže:
  - pro každé místo p vstupní množiny přechodu t platí, že obsahuje alespoň tolik tokenů, kolik činí násobnost hrany vedoucí z místa p do přechodu t,
  - pro každé místo p výstupní množiny přechodu t platí, že počet tokenů obsažených v místě p zvětšený o násobnost hrany, mířící z přechodu t do místa p, nepřevyšuje kapacitu místa p,
- při provedení (firing) proveditelného přechodu t se změní stav (značení, marking) sítě takto:
  - počet tokenů v každém vstupním místě p přechodu t se zmenší o násobnost hrany spojující toto místo s tímto přechodem
  - počet tokenů v každém výstupním místě p přechodu t se zvětší o násobnost hrany spojující toto místo s tímto přechodem

Vstupní a výstupní podmínky přechodů specifikují počty odebíraných/umísťovaných značek. V grafu Petriho sítě se to vyjádří ohodnocením orientovaných hran do/z přechodu. Příklad změny stavu je znázorněn na obr. 1. V P/T Petriho sítích podle obvyklých zásad místa označují stavy modelovaného systému a přechody změny stavu. Stav je charakterizován celým nezáporným číslem daným značením daného místa (počtem tokenů v daném místě). Při modelování počítačových dějů, jedná se např. o počty jednotek volné nebo obsazené paměti (např. bufferu), o počty jednotek disponibilních nebo využívaných zdrojů různého typu apod..

Implicitně předpokládáme násobnost hrany 1 a kapacitu místa nekonečnou. Násobnost jednoduchých hran ($w = 1$) a kapacitu kapacitně neomezených míst ($k = \infty$) na grafech Petriho sítí není nutné uvádět, především kvůli větší přehlednosti.

Speciálním případem P/T Petriho sítí jsou C/E Petriho sítě, což jsou P/T Petriho sítě ve kterých je kapacita každého místa a násobnost každé hrany rovna 1.



Obrázek 1: Příklad změny stavu po provedení přechodu v P/T Petriho síti

4 Petriho sítě vyšší úrovně

Základní typy Petriho sítí byly postupně obohacovány a zobecňovány tak, aby schopnost tohoto modelovacího nástroje vyhověla praktickým potřebám. Dalšími typy Petriho sítí vzniklými rozšířením P/T sítí jsou:

- Petriho sítě s inhibičními hranami (P/T PN with inhibitors),
- Petriho sítě s prioritami (P/T PN with priorities),
- Časované Petriho sítě (Timed PN),
- Barevné Petriho sítě (Coloured PN),
- Hierarchické Petriho sítě (Hierarchical PN),
- Objektové Petriho sítě (Object-Oriented PN).

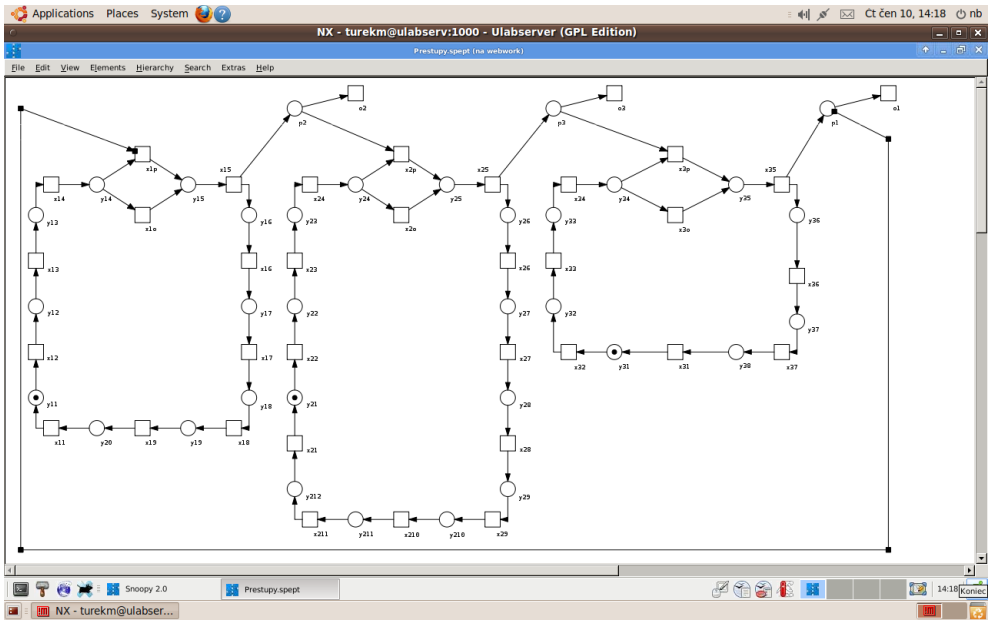
5 Specializovaný software pro sestavení Petriho sítě

Při vytváření Petriho sítě a následnou simulaci je nutné využít specializovaný software. V následující části budou popsány specializované software Snoopy a Pipe2, protože byly použity k řešení koordinace linek MHD s využitím Petriho sítí.

5.1 Software Snoopy

Spuštěním software Snoopy se zobrazí kromě základního okna také nabídka Petriho sítí a po zvolení zamýšlené Petriho sítě se zobrazí obrazovka obsahující menu a pracovní prostor. V pracovním prostoru (viz. obr. 2) se postupně vytváří Petriho síť prostřednictvím položek, které se nacházejí v menu. Kromě sestavení prvků Petriho sítě je možné definovat příslušné parametry, u každého místa lze definovat název, značení, komentář a grafickou podobu, u každého přechodu lze definovat název, komentář a grafickou podobu, pro každou hranu

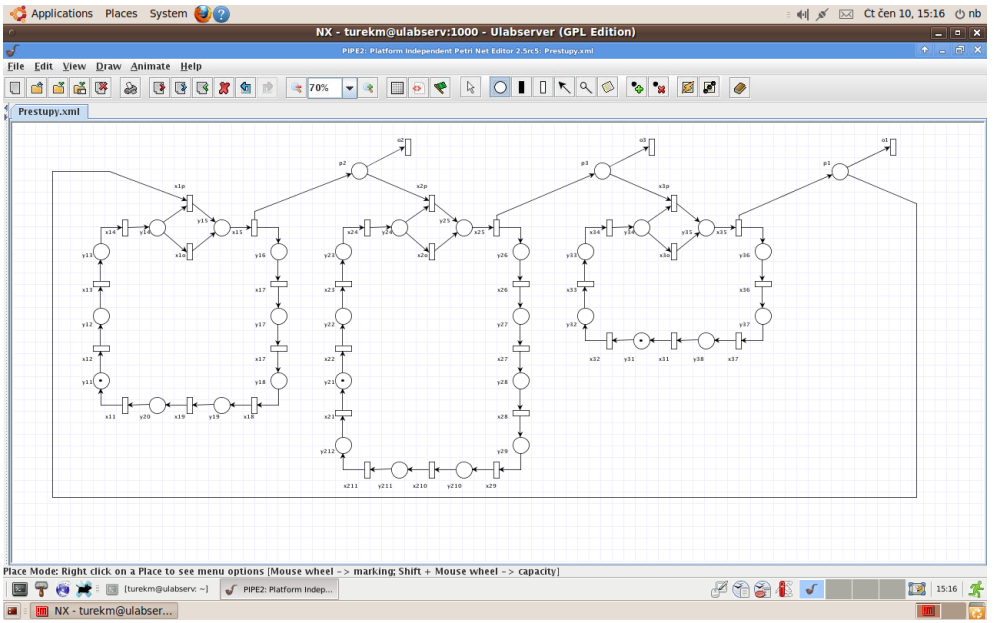
lze definovat násobnost, komentář a grafickou podobu. Prostřednictvím položky Start Animmode, která je součástí menu, se v pracovním prostoru spouští simulace sestavené Petriho sítě. V rámci simulace se po hranách přesunují čeveně zbarvené tokeny a mění se počty tokenů v příslušných místech.



Obrázek 2: Pracovní prostor v software Snoopy

5.2 Software Pipe2

Spuštěním software Pipe2 se zobrazí obrazovka obsahující menu, pracovní prostor a rozšířenou analýzu Petriho sítí. K vytvoření Petriho sítě slouží pracovní prostor (viz. obr. 3), do kterého se prostřednictvím položek z menu vkládají jednotlivé prvky Petriho sítě. V případě, že se prvky Petriho sítě nacházejí v pracovním prostoru je možné definovat jejich parametry, u každého místa lze definovat název, značení a kapacitu, u každého přechodu lze definovat název, sazbu a časování, pro každou hranu lze definovat násobnost. Prostřednictvím položky Animate, která je součástí menu, se v pracovním prostoru spouští simulace sestavené Petriho sítě. Při simulaci se mění počty tokenů v příslušných místech a nedochází k viditelnému přesunu tokenů po hranách.



Obrázek 3: Pracovní prostor v software Pipe2

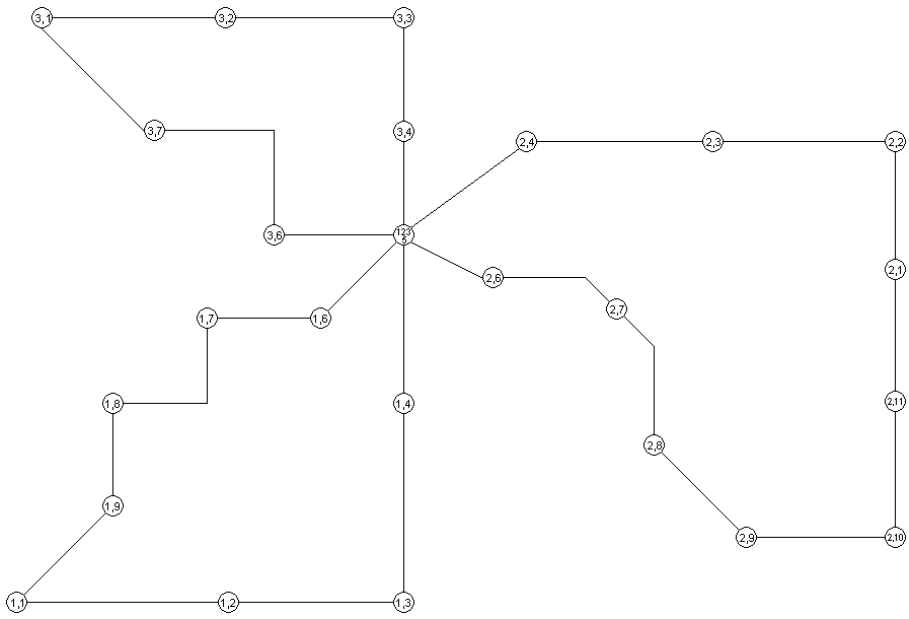
6 Aplikace P/T Petriho sítě na řešení úlohy o modelování koordinace linek MHD

Jedním ze způsobů řešení uvedeného problému je graficky založený model, kdy přechody představují zastávky, místa jízdu vozidla mezi zastávkami a tokeny představují vozidla.

Níže uvedený graficky založený model vychází z předlohy fiktivní dopravní sítě s okružním systémem tras linek MHD, které mají společnou přestupní zastávku v centru. Schéma sítě linek MHD modelové sítě je znázorněno na obr. 4. Celkový počet zastávek v síti činí 25, linka č. 1 obsluhuje 9 zastávek, linka č. 2 obsahuje 11 zastávek a linka č. 3 zahrnuje 7 zastávek. Čísla přiřazená jednotlivým hranám reprezentují číslo linky, která příslušnou trasu obsluhuje.

Pro potřeby modelování problému nástroji Petriho sítí se bude každá linka L_i skládat z posloupnosti přechodů $(x_{i1}, x_{i2}, \ldots, x_{ij})$ a míst, která jsou spojeny hranami, jak je uvedeno na obr. 5. Jednotlivé přechody reprezentují zastávky, přechod x_{i1} je výchozí zastávkou, x_{i5} představuje přestupní zastávku. Přechody x_{ip} a x_{io} představují propojení přestupních zastávek. Místa yij mezi těmito přechody představují pohyb vozidel mezi zastávkami. Místa označená p_i spojují linky L_i a L_{i+1}. Jejich prostřednictvím je možné modelovat pohyb cestujících, kteří chtějí přestoupit mezi dvěma spoji dvou linek.

Tokeny v Petriho síti budou mít dva významy. Jednak budou modelovat pohybující se vozidlo a potom také skupiny cestujících, kteří budou v přestupním uzlu přestupovat. V pří-



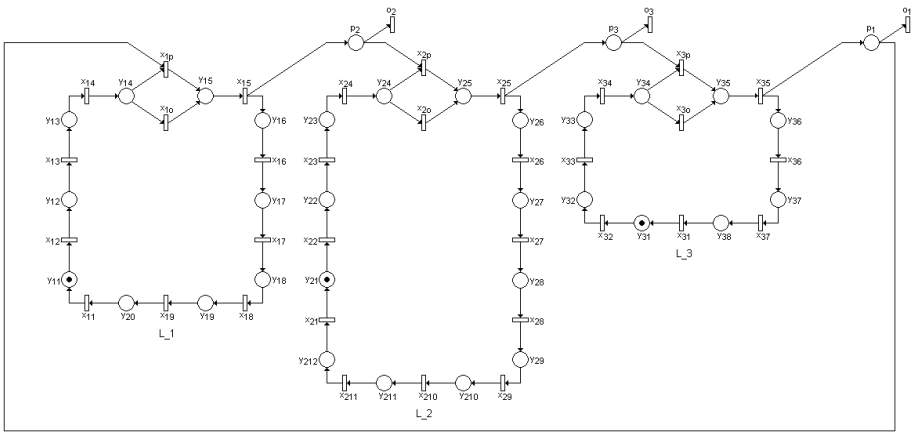
Obrázek 4: Předlohová dopravní síť MHD

padě místa pi představuje token všechny cestující čekající na přestupní zastávce. V modelu dopravní sítě MHD se mohou vyskytovat různé varianty, buď je aktivován přechod reprezentující přestup cestujícího na spoj jiné linky nebo je aktivován přechod, který představuje že cestující nepřestupuje na spoj druhé linky.

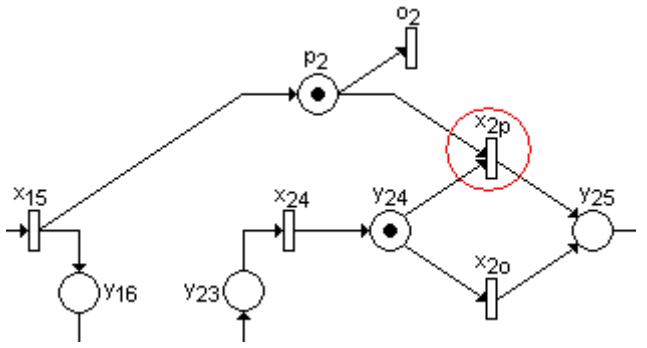
Vzhledem k tomu, že v Petriho síti přechody x_{ip} a x_{io} představují propojení přestupních zastávek a nereprezentují zastávky bude počet přechodů na jednotlivých linkách v modelu dopravní sítě MHD o 2 přechody vyšší než je počet zastávek na jednotlivých linkách. To znamená, že celkový počet přechodů na jednotlivých linkách bude činit 33. Přestupy cestujících modelují v navržené Petrino síti specifické komponenty. Příklad takové komponenty je uveden na obr. 6.

V případě, že cestující budou požadovat přestoupit mezi dvěma spoji dvou linek, bude tento požadavek znázorněn setrváním tokenu ve vstupním místě pi představujícím všechny cestující, kteří chtějí přestoupit. To znamená, že v případě výskytu tokenu ve druhém vstupním místě y_{i4} bude splněna vstupní podmínka pro provedení přechodu x_{ip} a dojde k jeho aktivaci. V případě požadavku na přestup bude tedy zajištěna provázanost s odjezdem autobusu na přestupní zastávce prostřednictvím přechodu x_{ip}, který bude aktivován pouze vpřípadě splnění vstupních podmínek.

Jestliže cestující nebudou potřebovat přestoupit, bude aktivován přechod o_i a v místě p_i se nebude vyskytovat token reprezentující přestupující cestující. V případě výskytu tokenu v místě y_{i4} znázorňující vozidlo tak bude splněna jediná vstupní podmínka přechodu x_{io}



Obrázek 5: Grafické znázornění Petriho sítě pro řešený problém

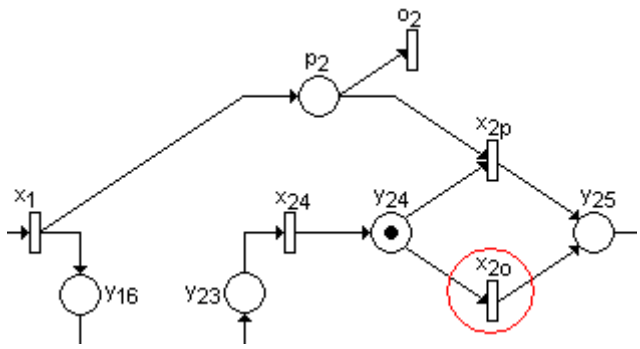


Obrázek 6: Fragment modelu Petriho sítě znázorňující požadavek přestup

a dojde k jeho aktivaci. Tato situace je znázorněna na fragmentu dopravní sítě MHD na obr. 7.

K modelování Petriho sítí se obecně využívá specializovaných software, např. Snoopy, Pipe2. Provedení simulace umožňuje pozorovat chování navrženého modelu a možné konflikty. Na základě zjištěných skutečností je pak možné změnit konfiguraci modelu. V případě modelu dopravní sítě MHD v Petriho síti se může jednat o modelování návaznosti na přestupních zastávkách.

Postup při potřebě využití Petriho sítí je tedy následující. V první řadě je třeba sestavit podle zvolené koncepce Petriho sítí model v Petriho síti. Každá Petriho síť se skládá z posloupnosti přechodů a míst spojených orientovanými hranami. Prostřednictvím orientovaných hran se znázorní logické vazby mezi objekty, které jsou vzájemně ovlivněny. K tomu, aby bylo možno modelovat pohyb vozidel, je nutné před začátkem simulace do některého místa na každé lince umístit token představující vozidlo. Token reprezentující cestující, kteří



Obrázek 7: Fragment modelu Petriho sítě bez požadavku na přestup

potřebují přestoupit vznikne rozdělením tokenu představujícího vozidlo, který vstoupí do přechodu x_{i5} – přestupní zastávky ze kterého vycházejí dvě orientované hrany.

7 Závěr

Předmětem článku je modelování koordinace linek MHD s využitím Petriho sítí. V první části je seznámení s nástrojem Petriho sítí včetně koncepce a popsány software umožňující sestavení Petriho sítě. Sestavování Petriho sítě v software Snoopy a Pipe2 je shodné, protože oba software využívají pro sestavení Petriho sítě pracovní prostor, do kterého se prostřednictvím položek z menu vkládají jednotlivé prvky Petriho sítě. Rozdíl mezi software nastává v případě definování vlastností prvků, protože definování kapacity míst a časování přechodů umožňuje pouze software Pipe2. Z hlediska simulace dochází ke změnám počtu tokenů v příslušných místech při použití obou software, přičemž při použití software Snoopy dochází navíc k viditelnému přesunování zvýrazněných tokenů po hranách.

Ve druhé části je vytvořen zjednodušený model dopravní sítě MHD v Petriho síti, ve kterém přechody reprezentují zastávky, místa jízdu vozidla mezi zastávkami a tokeny představují autobusy, případně skupiny přestupujících cestujících. V rámci řešené problematiky se jedná např. o modelování pohybu vozidel v síti linek MHD a zajištění koordinace spojů na přestupních zastávkách. Petriho sítě by tak v budoucnu mohly být využívány při hodnocení návrhu dopravní sítě MHD.

V případě výše uvedeného modelu došlo k určitému zjednodušení, v reálném provozu je totiž třeba zohlednit intenzity přepravního proudu, počty a kapacity vozidel, doby oběhů vozidel na jednotlivých linkách. Při návrhu odpovídajícímu reálnému provozu bude vhodné uplatnit pro sestavení a následnou simulaci software Pipe2, protože vykazuje lepší modelovací schopnosti.

Literatura

[1] ČERNÝ, J. – KLUVÁNEK, P.: Základy matematickej teorie dopravy. Bratislava: VEDA, 1990. ISBN 80-224-0099-8

[2] ČERNÝ, J. – ČERNÁ, A.: Teorie řízení a rozhodování v dopravních systémech. Pardubice: Institut J. Pernera, 2004. 1. vydání, 150 s. ISBN 80-86530-15-9.

[3] MARKL, J.: Petriho sítě I. Ostrava: VŠB-Technická univerzita Ostrava, 2009.

[4] KOCHANÍČKOVÁ, M.: Petriho sítě. Olomouc: Univerzita Palackého, 2008.

[5] Snoopy. Oficiální stránka. Dostupné na internetu: `http://www-dssz.informatik.tu-cottbus.de/index.html?/software/snoopy.html`

[6] Pipe2. Oficiální stránka. Dostupné na internetu: `http://pipe2.sourceforge.net`

Kontaktní adresa

Richard TUREK (Ing.),
Institut dopravy FS VŠB – TU Ostrava,
17. listopadu 15, 708 33 Ostrava-Poruba,
`richardturek@seznam.cz`

POUŽITIE MYŠLIENKOVEJ MAPY NA ZÁZNAM INFORMÁCIÍ (FreeMind)

ŽARNAY, Michal, (SK)

Abstrakt. Myšlienková mapa (mind map) je graficky usporiadaný text doplnený obrázkami s vyznačením súvislostí. Jej elektronická verzia navyše umožňuje rýchlejšiu orientáciu vo veľkom množstve informácií, čím sa výrazne zvyšuje jej použiteľnosť. V príspevku opíšeme nasadenie tejto metódy použitím nástroja s otvoreným zdrojovým kódom FreeMind v práci učiteľa, kde ho úspešne používame na záznam informácií k výučbe predmetov, výskumu, príprave článkov, prejavov, ale aj k evidencii pracovných úloh a plánovaniu pracovného času.

1 Prečo myšlienková mapa a tento príspevok?

Myšlienková mapa (niekedy zvaná tiež mentálna mapa, anglicky *mind map*, MM) je graficky usporiadaný text doplnený obrázkami s vyznačením súvislostí. Stáročia sa využívala na zaznamenávanie, učenie sa, grafické zobrazenie alebo riešenie problémov [1].

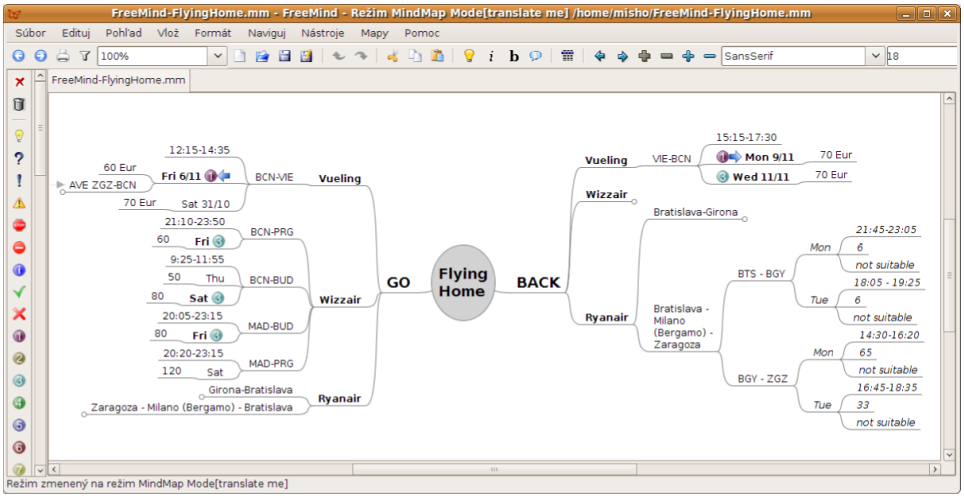
Príchodom výpočtovej techniky do kancelárskych činností a vznikom aplikácií na používanie MM pribudla ešte jedna užitočná dimenzia tomuto menej bežnému spôsobu elektronického zaznamenávania informácií: možnosť pružne skrývať a odkrývať časti MM podľa potreby, a tým sa sústrediť na detaily časti mapy, ktorá ako celok môže obsahovať obrovské množstvo informácií. To výrazne zvyšuje použiteľnosť MM v každodennej kancelárskej práci pri zapisovaní vlastných poznámok, nejako spolu súvisiacich. Zvýšenie efektívnosti v tejto oblasti a viacročné skúsenosti s používaním MM nás motivovali k tomuto príspevku na inšpiráciu pre každého, komu záleží na starostlivom uchovávaní a neskoršom využití svojich nápadov a cenných informácií a komu sa jeho doterajší systém zdá nedostatočný.

Hlavným cieľom tohto príspevku je tak opis využitia myšlienkovej mapy s pomocou aplikácie s otvoreným zdrojovým kódom (open source, OS) *FreeMind* v práci učiteľa na vysokej škole. Po stručnom uvedení aplikácie uvedieme príklady použitia MM v našej práci. Vymenujeme aj ďalšie vlastnosti nástroja, ktoré sú takisto k dispozícii, no doposiaľ sme ich príliš nevyužili.

2 FreeMind

FreeMind je rozšírený OS editor MM napísaným v jazyku Java. Aktuálne je k dispozícii jeho verzia 0.9.0 [2], ktorá priniesla výrazné doplnenie jeho schopností.

Ovládanie nástroja je intuitívne a po zvládnutí klávesových skratiek veľmi rýchle a efektívne. Kľúčovými operáciami sú editovanie, pohyb v mape a pre elektronickú verziu MM špecifické zabalenie a spätné rozvíjanie podsietí z nadradeného uzla (na jeden klik myšou či stlačenie klávesu), ktoré sa dejú na pracovnej ploche (obr. 1). Dopĺňajú ich operácie, tradičné v editoroch, cez tlačidlá hornej lišty a ponuku. Zo zvislej lišty vľavo možno do mapy pridávať ikony, ktoré graficky spestria usporiadaný text a uzlom siete môžu dodať ďalší význam.



Obrázok 1: Aplikácia FreeMind s MM o dopravnom spojení zo Zaragozy smerom na Slovensko a späť

Medzi štandardné schopnosti FreeMind-u patria:

- inteligentné presúvanie (kopírovanie) uzlov a ich skupín v sieti metódou *Uchop a polož*, aj z cudzích aplikácií,
- inteligentné kopírovanie textu cez schránku,
- export mapy do HTML,
- textové vyhľadávanie v podsieti aktuálneho uzla,
- linky HTML v uzloch vedúce na web alebo k miestnym súborom,
- editovanie viacriadkových uzlov, včítane vloženia tabuľky definovanej vo formáte XML,

- grafické odlíšenie uzlov – okrem ikon aj vlastnosťami písma,
- mapy sú uložené v súboroch formátu XML s možnosťou prenositeľnosti údajov (cez externé aplikácie) do iných editorov, napr. komerčnej aplikácie MindManager,
- vloženie súborovej štruktúry vo forme MM,
- export do formátov štruktúrovaných/editovateľných (HTML, XHTML, Java Applet, Flash, OpenOffice document) a obrázkových (PNG, JPEG, PDF, SVG).

Slabšími miestami v tejto verzii je okrem iného podpora obrázkov v uzloch a zamykací mechanizmus pri súčasnom editovaní tej istej mapy viacerými používateľmi naraz. V našej praxi nás však viac ako uvedené nedostatky postihli dlhšie reakčné doby u MM s veľkým objemom informácií zobrazeným naraz, ak boli súčasne spustené ďalšie pamäťovo náročné aplikácie – ak teda FreeMind nemá dostatok pamäti, jeho správanie je ťažkopádnejšie.

3 Použitie myšlienkových máp

V práci učiteľa na vysokej škole sú typickými oblasťami výučba a výskum. Pozrieme sa preto na tieto oblasti podrobnejšie. Za tým pridáme ďalšie príklady pre ľubovoľné zamestnanie: sebavzdelávanie a plánovanie času.

3.1 MM pre učiteľov na VŠ

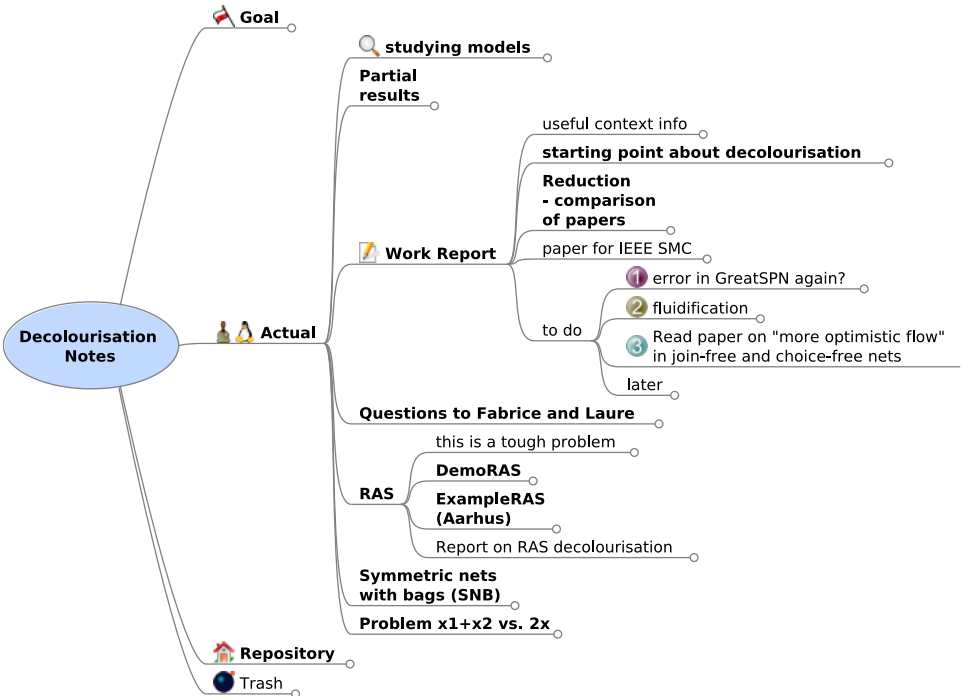
V rámci *výučby* máme MM zavedené pre každý vyučovaný predmet. Obsahujú všetky poznámky k predmetu:

- *rámcové informácie* ako ciele predmetu, cieľová skupina, témy prednášok a cvičení, hlavné informačné zdroje,
- *návrhy na vylepšenia* vlastné a od poslucháčov prevedené do formy konkrétnych úloh na vykonanie.

V rámci *výskumu* sú množstvo a rozmanitosť poznámok tým väčšie, čím je výskumník tvorivejší alebo čím viac sa stretáva s inšpiráciou. Zvlášť v takom prípade je MM užitočná a vymedziť jej približnú štruktúru je zložitejšie. V zásade môže MM obsahovať:

- *rámcové informácie* ako ciele výskumnej témy, prípadne termíny,
- *aktuálne poznámky* – čo je práve v popredí pozornosti,
- *depozitár* – staršie poznámky, vyriešené problémy, otvorené otázky pre neskoršie riešenie.

Niekedy potrebujeme zapísať dlhší súvislý text alebo prehľadné informácie v tabuľke. Hoci sa to môže tiež nachádzať v MM, odporúčame na to použiť externé dokumenty pripojené cez spojovací odkaz v mapke. Príklad takej mapky je na obr. 2. Tá bola vytvorená pri výskume formalizmu Petriho sietí počas zahraničného pobytu v jazyku výskumnej skupiny.

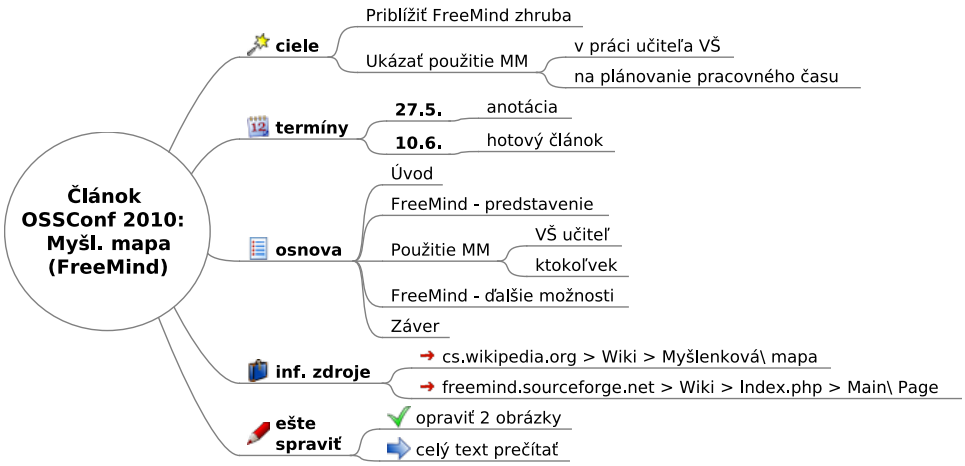


Obrázok 2: MM k výskumnej téme odfarbovania farbenej Petriho siete

S výskumom súvisí aj publikovanie jeho výsledkov vo forme *článkov* a *prezentácií*. Pri ich tvorbe si opäť treba usporiadať myšlienky, na jedno miesto zapísať podstatné štartovacie informácie (kde sa publikuje, termíny, rozsah a pod.) a neskôr počas prípravy aj aktualizovať svoju predstavu. Príklad mapky pre tento článok je na obr. 3. Takto zapísané detaily nám neskôr môžu pomôcť pri príprave prezentácie článku.

3.2 MM pre kohokoľvek

MM je užitočná pre záznam informácií k čomukoľvek. Príkladom je aj obsah mapky na obr. 1, ktorý poslúžil na zmapovanie dopravných spojení pre slovenského výskumníka v španielskej Zaragoze na návštevu domova. Tých sa ponúkalo niekoľko a bez zhromaždenia informácií do prehľadnej podoby bolo zložité určiť, ktoré spojenie sa javí ako najvýhodnejšie za daných podmienok. Navyše sú tieto poznámky užitočné aj s odstupom niekoľkých mesiacov, keď chce niekto opäť uskutočniť podobnú cestu. MM s krátkou aktualizáciou informácií mu poslúži na rýchle zorientovanie. A vďaka zvolenému jazyku aj zahraničným kolegom.



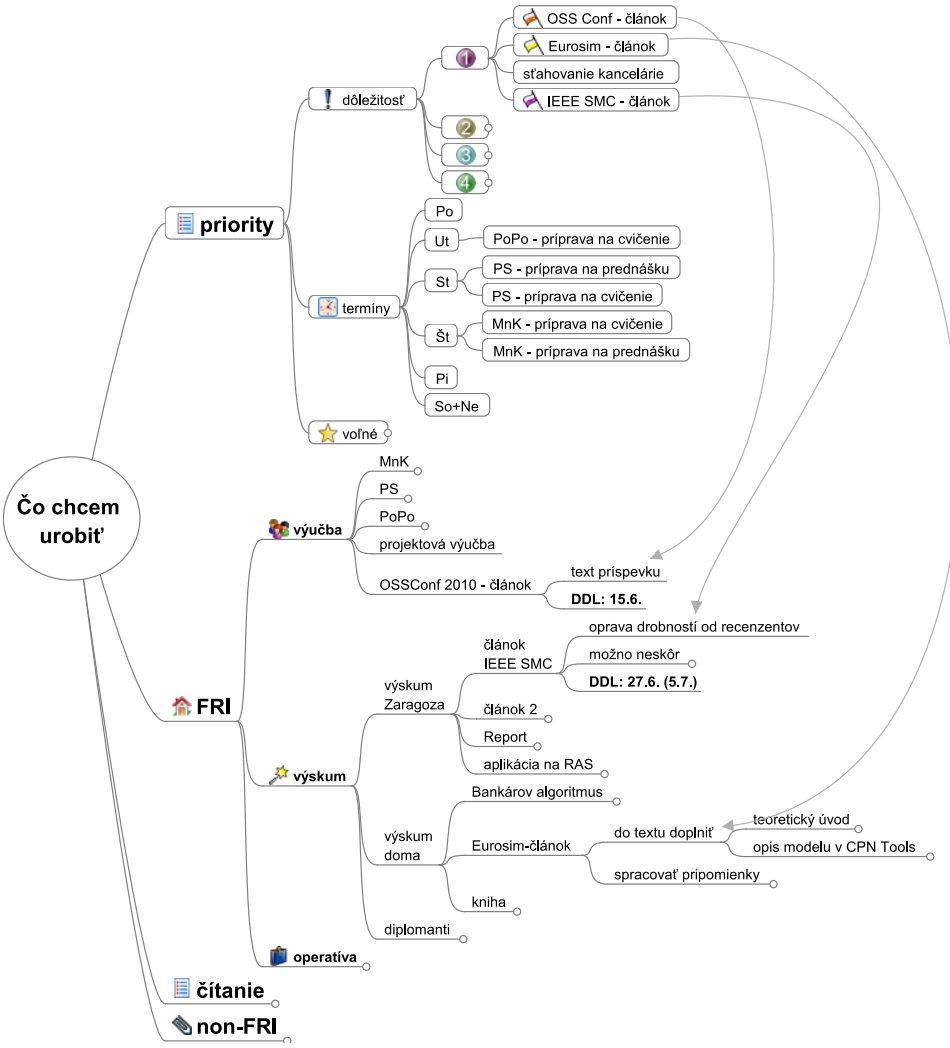
Obrázok 3: MM pre článok o myšlienkovej mape na konferencii OSSConf 2010

Súčasťou práce v ľubovoľnej profesii (učiteľa zvlášť) je aj *štúdium* – v ľubovoľnej forme, či už je to čítanie odborných článkov, inšpirácia od študentov a kolegov alebo snaha o ovládnutie novej softvérovej aplikácie. Zvlášť, ak ide o zložitejšiu aplikáciu alebo celý nový systém, môžu osobné poznámky vo vyhovujúcej štruktúre začínajúcemu používateľovi pomôcť. Takáto MM, ktorá nám už vyše roka pomáha napríklad pri udomácňovaní sa v prostrediach Linuxu i LaTeX-u, obsahuje najmä:

- príklady občas používaných príkazov, ktoré si ešte nepamätáme,
- zápis o občasných problémoch a ich riešení, ktoré nie je triviálne,
- odkazy na ďalšie zdroje informácií na webe alebo vlastnom disku,
- ľubovoľné informácie, ktoré sa môžu v budúcnosti zísť.

Na prvý pohľad vzdialenou oblasťou od mapovania myšlienok je *organizácia pracovného času*, kde tradičné metódy presadzujú skôr lineárne zoznamy a štrukturované tabuľky (napr. týždenný plán). V pozadí týchto nástrojov však vždy stojí zmapovanie a utriedenie zámerov v jednotlivých oblastiach snaženia, prípadne ich rozpracovanie do konkrétnych úloh aj s približnými termínmi v stredno- a dlhodobom horizonte. Na to je elektronická MM ako stvorená. Keď sa k tomu pridá aj pokus o (čiastočnú) náhradu menovaných plánovacích nástrojov v popredí, vznikne ucelený nástroj na plánovanie pracovného času, ktorý obidva pohľady integruje.

V príklade na obr. 4 pozadie reprezentujú podstromy pod uzlami *FRI*, *non-FRI* a *čítanie*, členené do podrobností podľa potreby. Popredím je podstrom *priority*, kde sú úlohy zoradené *podľa poradia dôležitosti*: poduzly s číslami vo farebných kruhoch a v rámci nich ďalšie členenie, kde priorita klesá v smere zhora dolu. K poradiu dôležitosti možno pripojiť aj časové



Obrázok 4: MM pre plánovanie činností v pracovnom čase

rozdelenie *podľa dní týždňa* pre úlohy, ktoré sa vykonávajú pravidelne v určité dni a ich zaradenie do priorít by mohlo ohroziť ich vykonanie v pravý čas. Aby sa opis úloh nemusel opakovať opäť v tejto časti, používajú sa odkazy k podrobnostiam do podstromov v pozadí, ako napr. odkaz od uzla *OSS Conf – článok* k uzlu *text príspevku*. Podstrom *voľné* združuje aktuálne voľné odkazovacie uzly.

Časové plánovanie pri použití tohto nástroja potom pozostáva z občasnej aktualizácie odkazov a poradia uzlov, aby zodpovedali aktuálnej situácii používateľa. Pri realizácii plánu je veľmi užitočnou operáciou skrývanie a odkrývanie častí MM podľa toho, čo práve potrebujeme. Možno skryť väčšinu informácií a ponechať zobrazené len najdôležitejšie a termínovo

najsúrnejšie úlohy – tým sa používateľ vyhne rušivým vplyvom a môže sa koncentrovať iba na najbližšie priority. Toto považujeme za najväčší prínos oproti klasickému prístupu, kde preplnené tabuľky sťažujú orientáciu a sústredenie sa na aktuálne úlohy.

3.3 Poznámky k používaniu MM

Použitie MM odráža štýl práce používateľa. Naznačená schéma sa doposiaľ osvedčila nám, no každý si ju môže upraviť podľa svojich potrieb. Schéma sa taktiež časom vyvíja v súlade s vývojom použitého nástroja a dôkladnejšou znalosťou jeho ovládania.

MM možno kresliť aj ručne na papier a využiť hierarchickú štruktúru a ikony. Elektronická verzia však výrazne zosilňuje efekt vlastností MM v možnosti (ro)zbalenia časti siete podľa aktuálnej potreby, čím prispieva k zachovaniu veľkého množstva informácií pri súčasnom sústredení sa len na potrebnú časť z nich.

Raz za čas si MM vyžaduje určité upratovanie – preveriť obsah a štruktúru dlhšie nenavštívených častí, aby obsah celej MM bol konzistentný, súvisiace info boli prepojené a nepotrebné informácie odstránené. Je dôležité udržiavať MM v takom stave, aby človek mal prehľad o potrebe všetkých častí, inak sa nám jej organizácia môže vymknúť spod kontroly a budeme používať iba časť informácií vo veľkom súbore. Toto je však všeobecný problém údržby bázy znalostí nielen v MM, ale aj v iných formách.

MM možno využívať individuálne i skupinovo – vtedy je dôležité sa dohodnúť na konvenciách používania MM – štrukturovanie informácií a význam použitých ikon (legenda). Existujú nástroje aj na webe, no nemáme informácie o existencii OS on-line nástroja.

4 Ďalšie možnosti FreeMind-u

Posledná uvoľnená verzia FreeMindu 0.9.0 priniesla viacero významných noviniek. Sami autori vyzývajú na testovanie nových vlastností a varujú pred ich používaním na „vážne“ účely. S novinkami máme zatiaľ minimum skúseností, no dovolíme si tu vymenovať aspoň najväčšie z nich. Možno poslúžia ako inšpirácia pre zaradenie do bežného používania nielen pre čitateľov, ale aj tvorcu tohto príspevku.

1. Atribúty poskytujú rôzne pohľady na usporiadanie a prezeranie MM, odlišné od pohľadu daného štruktúrou mapy.

2. Filtrovanie uzlov mapky – ďalšia možnosť koncentrácie na časti mapy.

3. Editor WYSIWYG pre uzly a poznámky.

4. Skriptovanie prostredníctvom objektovo orientovaného programovacieho jazyka Groovy – skripty sa môžu pripojiť k jednotlivým uzlom mapy a následne spustiť.

5. Integrácia OS nástroja na projektový manažment pre Linux TaskJuggler, čo umožní vytváranie uzlov v jeho formáte v rámci MM.

6. Možnosť pridávania vlastných používateľských ikon (toto nám doposiaľ asi najviac chýbalo) vo forme súborov PNG, ktorých veľkosť nie je limitovaná na 16×16 px.
7. Vďaka pluginu LaTeX možno do uzlov vkladať matematické vzorce v tomto formáte.
8. Väzba na uzly v inom súbore formátu `.mm` – na prepájanie máp vo viacerých súboroch.

5 Záver

Myšlienková mapa je skvelý nástroj na záznam informácií v atraktívnej forme. Jej elektronická verzia vďaka operáciám ukrývania a zobrazovania podstromov pod uzlami značne uľahčuje a zefektívňuje prácu s informáciami, nech už je užívateľom učiteľ na vysokej škole alebo ktokoľvek.

OS nástroj FreeMind sa doposiaľ osvedčil ako šikovný nástroj a jeho pokračujúci vývoj sľubuje ďalšie zefektívnenie práce s vlastnými poznámkami.

Literatúra

[1] *Myšlenková mapa*. Dostupné na: `http://cs.wikipedia.org/wiki/Myšlenková_mapa`. Online.

[2] *FreeMind*. Dostupné na: `http://freemind.sourceforge.net/wiki/index.php/Main_Page`. Online.

Kontaktná adresa

Michal ŽARNAY (Ing. PhD.),
Katedra dopravných sietí FRI ŽU v Žiline, Univerzitná 1,
010 26 Žilina, `michal.zarnay@fri.uniza.sk`

ABSTRAKTY
UKÁŽOK A PREZENTÁCIÍ

PROCESSING.ORG: PROGRAMOVANIE A DIGITÁLNE UMENIE, ARDUINO

BEDNAR, Juraj, (SK)

Processing je open source nástroj postavený na jazyku Java, určený pre umelcov alebo záujemcov o digitálne umenie. Nástroj obsahuje knižnice na prácu so zvukom, obrazom, videom a komunikáciu s externými zariadeniami. Základnou filozofiou je „sketch“ (náčrt) – autor nemusí vytvárať okná a strácať čas inicializáciou aplikácie a môže začať rovno programovať, čím je tento nástroj ideálny pre výučbu programovania. V krátkosti predstavíme aj sesterský projekt Arduino, pomocou ktorého je možné programovať otvorené hardvérové zariadenia podobným princípom.

PRAKTICKÉ VYUŽITÍ FOSS4G PROJEKTŮ PRO MONITOROVÁNÍ ÚZEMÍ POMOCÍ DRUŽICOVÝCH DAT

BRODSKÝ, Lukáš, KOLOMAZNÍK, Jan, ORLITOVÁ, Erika, VOBORA, Václav, (CZ)

Firma GISAT byla v letech 2007–2009 zapojena do mezinárodního projektu CASCADOSS, jehož hlavním cílem byla podpora uživatelů, kteří pracují s prostorovými daty v používání počítačových programů s otevřeným zdrojovým kódem (Open Source, OSS). Náplní projektu bylo provést průzkum a analýzu dostupného OSS pro oblast geografických informačních systémů a dálkového průzkumu Země (Free and Open Source Software for Geomatics, FOSS4G). Nasazení těchto programů umožňuje významně redukovat náklady na licence programů a přispívá tak k rychlejšímu rozvoji technologií.

GISAT využívá OSS v různých tématických aplikacích. Jako příklad uvádíme 3 projekty, ve kterých jsou dílčí úkoly řešeny s využitím FOSS4G.

Projekt FLOREO, celým názvem „Demonstration of ESA Environments in support to FLOod Risk Earth Observation monitoring“, je řešen v rámci programu PECS, který je společnou aktivitou České republiky a Evropské kosmické agentury (ESA) orientovanou na podporu zapojení českých firem a institucí do programů ESA.

Projekt je zaměřen na vybudování systému včasné předpovědi rizika vzniku povodňového jevu na regionální a národní úrovni ČR a vizuální prezentaci výsledků pomocí

mapového serveru. Metodika je založena na kombinaci družicového snímání povrchu Země s vyšší časovou periodou měření (optická a radarová data středního rozlišení) a in-situ dat v diskrétních bodech meteorologických stanic provozovaných ČHMÚ. Spojení obou zdrojů dat v jediném systému přináší více detailních informací pro hydrologický monitoring a včasnou předpověď nadcházejících povodňových jevů.

Cílem projektu SOSI (Spatial Observation Services and Infrastructure) je předvést použití technologie SSE (Service Support Environment) pro vývoj služeb nabízejících přístup k distribuovaným datovým zdrojům a k jejich vzájemnému propojení.

SOSI se skládá ze tří subprojektů řešených ve třech zemích EU – České republice (SOSI-CZ), Rakousku (SOSI-A) a Maďarsku (SOSI-H).

Jednou z částí SOSI-CZ, kterou řeší Gisat, je implementace národního portálu založeného na technologii SSE, který umožňuje přístup k družicovým datům MERIS z místní přijímací stanice.

RESPOND je projekt financovaný Evropskou kosmickou agenturou v rámci programu GMES (Globální monitoring životního prostředí a bezpečnosti) – společné iniciativy ESA a Evropské komise.

RESPOND je aliance evropských a mezinárodních organizací a firem, pohybujících se v oblasti humanitární pomoci, pracujících na zlepšení dostupnosti mapových podkladů, družicových dat a geografických informací obecně pro tyto potřeby. Hlavním cílem projektu RESPOND je přispět ke zvýšení včasnosti a efektivity evropské a mezinárodní humanitární pomoci díky využití vhodně přizpůsobených a spolehlivých geografických informací.

OPEN SOURCE SOFTVÉR V INFRAŠTRUKTÚRE PRE PRIESTOROVÉ INFORMÁCIE

CIBULKA, Dušan, (SK)

Pojem IPI (Infraštruktúra pre priestorové informácie) je používaný na označenie príslušnej základnej kolekcie technológií, politík a inštitucionálneho usporiadania, ktoré uľahčujú dostupnosť a prístup k priestorovým dátam. IPI musí byť viac ako jeden dátový súbor, alebo databáza. Zahŕňa geografické dáta a atribúty, dostatočnú dokumentáciu (metadáta), prostriedky na objavovanie, vizualizáciu a vyhodnocovanie dát (katalógy a webové mapovanie) a niektoré metódy k poskytovaniu prístupu ku geografickým dátam. Infraštruktúra poskytuje ideálne prostredie k prepojeniu aplikácií a dát. Aby mohla byť realizovaná, musí zahŕňať organizačné dohody potrebné k jej koordinácií na lokálnej, národnej a nadnárodnej úrovni.

Príkladom nadnárodnej infraštruktúry je INSPIRE (Infrastructure for Spatial Information in Europe) v Európe. Smernica INSPIRE nadobudla účinnosť 15. mája 2007 a bude realizované v rôznych fázach s plnou implementáciou požadovanou do roku 2019. Transpozíciou tejto smernice (2007/2/EC) vznikol zákon č. 3/2010. INSPIRE je založená na sérii nasledovných princípov; dáta by mali byť zhromažďované len raz a tam, kde môžu byť udržiavané najefektívnejšie, malo by byť možné kombinovať bezšvové priestorové informácie z rôznych

zdrojov naprieč Európou a zdieľať ich s mnohými užívateľmi a aplikáciami, geografické informácie potrebné pre dobrú správu verejných záležitostí na všetkých úrovniach by mali byť ľahko dostupné a transparentné. Takáto infraštruktúra má pomáhať pri vytváraní politiky cez hranice.

Článok sa zameriava na stručné priblíženie problematiky infraštruktúry pre priestorové informácie INSPIRE a na kategorizáciu, popis a možnosti využívania open source softvéru pri jej realizácií, testovaní a používaní. Príspevok postupne rozoberá nasledovné kategórie softvéru; databázové systémy, serverové aplikácie umožňujúce implementáciu webových služieb, desktopové GIS (geografické informačné systémy) a tiež využiteľné knižnice. Popisuje konkrétne softvérové riešenia, ich funkcionalitu, možnosti konfigurácie, podporované formáty, technológie a operačné systémy. Je zameraný aj na možnosti kompatibility a prepojenia popisovaných programov. V rámci tejto problematiky sú v článku spomínané jednotlivé štandardy a špecifikácie, od organizácií ako OGC (Open Geospatial Consortium) a ISO (International Organization for Standardization), ktoré tieto programy podporujú, implementujú a vedia s nimi pracovať. Práve štandardy od týchto organizácií sú využívané v INSPIRE ako základ zdieľania geografických dát v infraštruktúre pre priestorové informácie.

TERMINÁLOVÁ SIEŤ PO ROKU

FEDORIK, Slavko, (SK)

Je to rok, čo som v našej škole vytvoril laboratórium výpočtovej techniky, založené na technológii tenkých klientov, prihlasujúcich sa k terminálovému serveru. Základom technológie je projekt Linux Terminal Server (Project) – skrátene LTSP. Pre zavedenie tejto technológie som použil distribúciu Ubuntu 9.04.

V prezentácii v krátkosti zhrniem ročné skúsenosti s používania tejto technológie, jej klady a zápory, aj to čo sa podarilo a čo nie. Ťažiskom prezentácie je však zverejnenie výsledku prieskumu, ktorý som urobil medzi svojimi žiakmi. Ťažiskom prieskumu boli ich postoje a názory na používanie operačného systému GNU/Linux a FOSS pri vyučovaní. Ich pohľad na používanie tenkých klientov ako takých, ako aj ich názory a skúsenosti s používaním slobodného a otvoreného softvéru mimo vyučovacieho procesu.

CLOUD COMPUTING – DELTACLOUD

FOJTIK, Michal, (SK)

Cieľom prednášky je oboznámiť účastníkov konferencie s pojmom Cloud Computing, ktorý je v dnešnej dobe zrejme najskloňovanejším pojmom používaným v IT. Od definící prejsť ku praktickému využitiu pomocou API programovania. Následne poukázať na problém, ktorým sú rozdiely medzi jednotlivými API k týmto službám a ponúknuť riešenie v podobe open

source projektu Deltacloud, ktorého cieľom je zjednotiť tieto prístupy do jedného a tým rozšíriť možnosti pri využívaní Cloud Computingu.

APLIKÁCIE GEOPRIESTOROVÉHO SOFTVÉRU S OTVORENÝM KÓDOM VO VZDELÁVANÍ A VÝSKUME

HOFIERKA, Jaroslav, (SK)

GRASS GIS je jedným z hlavných reprezentantov geopriestorového softvéru s otvoreným kódom, ktorý si našiel svojich používateľov a aj vývojárov na Slovensku. Prešovská univerzita patrí medzi medzi vzdelávacie inštitúcie, ktoré GRASS GIS využívajú aktívne, vo vzdelávaní v rámci geoinformatických predmetov na bakalárskom a aj magisterskom stupni vzdelávania a zároveň tento softvér využíva vo svojich výskumných projektoch v oblasti modelovania krajiny, potenciálu obnoviteľných zdrojov energie a pri vedeckej vizualizácii krajinných procesov. V predloženom príspevku sumarizujeme spôsoby využívania softvéru GRASS GIS a Quantum GIS vo výuke a zároveň prezentujeme najzaujímavejšie výsledky niektorých výskumných projektov, kde GRASS GIS predstavoval hlavnú softvérovú platformu riešenia. Stručne prezentujeme naše aktivity v oblasti vývoja GRASS-u a tiež súčasný stav vo vývoji GRASS-u.

OpenSUSE 11.3

HRUŠECKÝ, Michal, (CZ)

V červenci vyjde nová verze distribuce openSUSE s pořadovým číslem 11.3. Přednáška ve stručnosti představí novinky na které se uživatelé v nové verzi mohou těšit. Součástí bude i živá ukázka některých z nich.

SUSE Studio

HRUŠECKÝ, Michal, (CZ)

Prezentace představí webový nástroj SUSE Studio. Tento nástroj slouží pro snadnou a rychlou tvorbu appliance. Umožňuje tak administrátorům vytvářet obrazy pro snadnou reinstalaci stanic včetně všech základních nastavení a programů. Učitelům umožňuje vytvořit pro žáky LiveCD s předinstalovaným Open Source softwarem pro výuku. A studentům nabízí základ na kterém mohou stavět specializované distribuce se svými projekty.

Hlavní předností SUSE Studia je, že se vše odehrává přímo ve webovém prohlížeči. Od výběru distribuce, na které chceme stavět, přes volbu dalších programů, vzhledu, či nastavení, až po spuštění virtualního stroje, kde je možné appliance otestovat před závěrečným stažením.

Další velkou výhodou SUSE Studia je jeho provázanost s openSUSE Build Service, která nabízí velké množství již zabaleného software.

OSS VE ZDRAVOTNICTVÍ

JELÍNEK, Lukáš, (CZ)

Zdravotnictví je oblast, která se trvale potýká s nedostatkem finančních prostředků. Využití open source softwaru je cesta, která umožňuje výrazně snížit náklady na software včetně jeho správy a omezit závislost na dodavatelích. Cílem prezentace je popsat možnosti nasazení OSS ve zdravotnictví s ohledem na právní, technologická a jiná omezení, zhodnotit aktuální dostupnost vhodných produktů a specifikovat konkrétní řešení v této oblasti.

AGENT-BASED MODEL DEVELOPMENT WITH REPAST SIMPHONY

KANIK, Tomasz, (PL)

In this paper, we describe Repast Simphony (Repast S) which is widely used, free, and Open source agent-based modeling (ABM) and simulation toolkit. To better understand the capabilities of this environment, real world situations are modelled with visual language and hard-coded examples. Visual language focuses on the design of agent-based simulation models, considering a simulation environment where agent modeling is mainly concerned with the spatial aspects and constraints of the scenario. Such environment, based on Grovy language is used to test and validate specific action plans designed for execution in specific situations. Hard-coded example building with Eclipse, Open source development environment, covers implementation of a system capable of harnessing the computational power of a wide simulation infrastructure with the design efficiency of an agent-toolkit. The proposed solutions help people like domain experts, generally with no programing skills, to start their work with this powerful modeling toolkit.

OpenWRT - PREMENIŤ WiFi ROUTER NA RAKETOPLÁN?

KEVICKÝ, Michal, (SK)

Stručné predstavenie OpenWRT – distribúcie určenej pre WiFi routery umožňujúcej wifi router obohatiť o funkcie, ktoré od malej krabičky pre sieťovú komunikáciu človek vonkoncom neočakáva.

LINUXOVÁ DISTRIBÚCIA SLACKWARE

KREHEĽ, Dušan, (SK)

Táto prezentácia popisuje spôsoby nainštalovania linuxovej distribúcie Slackware, jej konfiguráciu a jej charakteristické vlastnosti v porovnaní s inými Linuxovými distribúciami.

NOVÉ TRENDY ZRANITEĽNOSTÍ VO VEREJNE NASADZOVANÝCH TECHNOLÓGIÁCH

LUPTÁK, Pavol, (SK)

S rastúcou komplexnosťou informačných technológii, ich rozšírením, ako aj neustále rastúcim výpočtovým výkonom, sa na relatívne bezpečné technológie objavujú teoretické útoky a zo známych teoretických útokov sa stávajú prakticky realizovateľné hrozby.

Cieľom prezentácie bude poukázať na reálne zraniteľnosti masovo používaných technológií, ako sú GSM/3G, čipové RFID karty, či SMS lístky. Autor demonštruje, ako jednoducho je možné prelomiť najpoužívanejšie slovenské čipové karty Mifare Classic, či načítať základné informácie z nového biometrického pasu. Súčasne poukáže na nevyhnutnosť otvorenosti verejne používaných technológií, dôležitosť spätnej väzby zo strany technickej verejnosti a nezávislých bezpečnostných špecialistov.

OTVORENÝ PRÍSTUP KU GEOINFORMAČNÉMU OBSAHU NA SPRAVODAJSKOM PORTÁLI SLOVENSKEJ A ČESKEJ GEOKOMUNITY

OFÚKANÝ, Miloslav, (SK)

Prezentácia predstaví spravodajský portál `Geoinformatika.sk`, ktorý prináša aktuálne domáce a zahraničné dianie, týkajúce sa slovenskej a českej geokomunity (geoinformačnej komunity) – skupiny používateľov geografických informácií.

Našim vystúpením účastníci získajú prehľad, kde hľadať geografické informácie, geoinformačné systémy a technológie z rôznych akcií, odvetví a vedných disciplín, novinky v dátach, softvéri, hardvéri, službách a pracovných ponukách, ukážeme im ako pristupovať k fotkám, dokumentom (napr. študentkým prácam a prezentáciam) a diskusiám na rôzne témy.

Podporujeme odborné podujatia, časopisy, internetové stránky a organizácie občianskeho, podnikateľského, verejného, vzdelávacieho a výskumného sektora. Ponúkame spoluprácu a partnerstvá ďalším jednotlivcom a kolektívom. Geoinformačný obsah propagujeme aj na sociálnych sieťach.

Portál `Geoinformatika.sk` bol založený v septembri 2006 a od začiatku svojej existencie beží na slobodnom a voľne šíriteľnom redakčnom systéme. Do konca apríla 2010 sme spravodajstvo poskytovali cez CMS Joomla a v júni sme prešli na CMS Drupal.

OpenSUSE BUILD SERVICE

RUSNÁK, Pavol, (SK)

Build Service je vývojová platforma, ktorá poskytuje infraštruktúru potrebnú pre vytváranie softvérových balíčkov pre všetky väčšie Linuxové distribúcie (CentOS, Debian, Fedora, Mandriva, openSUSE, Red Hat Enterprise Linux a SUSE Linux Enterprise). Tento nástroj podporuje okrem vytvárania jednotlivých balíčkov aj vytvorenie CD prípadne DVD obrazu s celou distribúciou.

POKRYTÍ FUNKCÍ GIS S VYUŽITÍM OPEN SOURCE NÁSTROJŮ

RŮŽIČKA, Jan, (CZ)

Cílem prezentace je přiblížit stav nástrojů pro budování geografických informačních systémů (GIS) na bázi open source. V současné době je nabídka těchto nástrojů široká a je poměrně obtížné se v ní dobře zorientovat. Prezentace se pokusí shrnout možnosti nástrojů takovým způsobem aby se ukázalo, kde již je možné tyto nástroje plně nasadit do komerčního prostředí a kde naopak jsou nedostatky natolik závažné, že je nasazení prakticky vyloučeno. Zmíněny budou také oblasti vývoje těchto nástrojů s výhledem jak by mohla situace vypadat v horizontu pěti let.

VIRTUALIZOVANÉ PROSTŘEDÍ VE VÝUCE GIS

RŮŽIČKA, Jan, (CZ)

V rámci studia oborů využívajících výpočetní techniku se často setkáváme s problémem, kdy studenti nemohou využívat software dostupný v rámci konzultačního střediska např. z kapacitních důvodů. Tento, často specifický, software jsou pak nuceni si instalovat na své vlastní počítače. Instalace je spojena se dvěma problémy, které se mohou vyskytovat samostatně nebo i zároveň. V některých případech není instalace triviální a po samotné instalaci je nutné provést ještě dodatečnou konfiguraci tak, aby software pracoval správně. Instalace může být natolik komplikovaná, že naprosto znemožní samostudium mimo laboratoře konzultačního střediska. V jiných případech je instalace vázána na zakoupení licence k produktu, což může být pro mnoho studentů limitující. Tato situace pak často vede k nelegálnímu užívání software.

Řešení tohoto problému může být trojího druhu. Buďto má universita dostatek prostředků k zakoupení licence takového druhu, aby ji mohli využívat legálně i studenti. Druhou možností je existence tzv. studentských licencí, které však musí poskytovat distributor software. Třetí možností je přehodnocení nutnosti výuky na komerčních nástrojích a volba jiné alternativy. Pro většinu komerčních nástrojů v současné době existuje volně dostupná alternativa

nástroje, často s plnohodnotnou sadou funkcí jako má komerční produkt. Některé z těchto nástrojů jsou navíc šířeny pod licencí, která umožňuje šíření, čtení a případně i úpravu zdrojových kódů aplikace. Možnost čtení zdrojových kódů může být velice zajímavým doplňkem k pochopení studované problematiky.

Příspěvek popisuje možnosti využití zmiňovaných nástrojů a to ze dvou základních variant distribuce ke studentům. První varianta je založena na technologii LiveCD. Druhá varianta je založena na využití virtualizace. V příspěvku bude popsáno praktické využití těchto technologií pro výuku v oblasti orchestrace geowebových služeb, kde se plně ukazuje síla nasazení těchto technologií ve výuce.

VÝPOČTOVÉ SIETE

SÁRENÍK, Ján, (SK)

V prednáške zhrniem možnosti využitia programu Condor na prevádzkovanie vysoko priepustných výpočtových sietí zložených z obyčajných stolných počítačov. Poviem tiež v krátkosti niečo o základných konceptoch tohto odvetvia IT.

ON-LINE LOKALIZÁCIA SOFTVÉRU POMOCOU NÁSTROJA POOTLE

ŠRÁMEK, Miloš, (SK)

V príspevku predstavime výsledky ankety o stave lokalizácie otvoreného softvéru, ktorá prebiehala v máji a júni 2010 na serveri `sospreskoly.org`. Predstavíme viaceré prekladacie nástroje a porovnáme ich funkcionalitu. Napokon, naživo predstavíme Pootle a možnosti, ktoré poskytuje prekladateľom. Počas prednášky, rovnako ako aj pred ňou a po nej, budú mať záujemci možnosť si Pootle vyskúšať na adrese `http://pootle.soit.sk` a priamo sa tak sa zúčastniť jeho skúšobnej prevádzky.

PROGRAMOVACÍ JAZYK SCRATCH

ŠTRBA, Peter, (SK)

Cieľom prezentácie bude ukázať základné prvky programovacieho jazyka SCRATCH a pár námetov aj s ukážkami, ako je možné tento jazyk využiť pri rozvíjaní algoritmického myslenia na strednej škole s osemročnou formou štúdia.

Jak se daní stádo laní?

Myslivec už dobře ví,
problém řeší …

FlexiBee

WWW.FLEXIBEE.EU — ÚČETNÍ A EKONOMICKÉ SYSTÉMY

OTVORENÝ SOFTVÉR VO VZDELÁVANÍ, VÝSKUME A V IT RIEŠENIACH

Zborník príspevkov medzinárodnej konferencie OSSConf 2010

Prvé vydanie 2010
Počet strán 234
Elektronická sadzba programom pdfTEX
Vytlačili C–PRESS.sk, Košice

ISBN 978-80-970457-0-8

ISBN 978-80-970457-0-8