ScanCSV – Lua knihovna pro zpracování CSV souborů ConTEXtem a LuaLATEXem

Jaroslav Hajtmar

Tento článek popisuje možnosti použití jazyka Lua pro vytvoření knihovny Lua funkcí, které mohou zajímavým způsobem zpřístupnit ConTEXtu, LuaTEXu a LuaLATEXu textové databázové údaje, uložené v CSV souborech. Prioritou při tvorbě popisované luaknihovny bylo, aby mohla být používána i uživateli bez sebemenších znalostí jazyka Lua. Kdo zvažuje, že si „něco začne“ s Lua, má příležitost zjistit, jak Lua funguje. Kdo chce zůstat „ryzím TEXistou“, má možnost používat popisovanou knihovnu formou „blackboxu“, tj. do zdrojového textu ConTEXtu (PlainTEXu, LATEXu) zapsat několik řádků Lua kódu, zkompilovat zdroják odpovídajícím formátem a koukat, jak to celé krásně funguje. Pro vážnější zájemce jsem připravil ke stažení řadu pokročilých ukázek, demonstrujících zajímavé možnosti praktického použití knihovny. Ačkoliv je Lua knihovna primárně určena pro použití v ConTEXtu, přichystal jsem řadu ukázek použití v LATEXu, který bude zřejmě čtenáři tohoto článku preferován.

1. Úvod

V tomto příspěvku se pokusím popsat svůj začátečnický pokus o vytvoření luaknihovny, s jejíž pomocí lze snadným způsobem zpřístupnit ConTEXtu, LuaTEXu (Plainu) a LuaLATEXu textové databázové údaje, uložené v CSV souborech.
ConTEXtem budu v celém příspěvku mínit verzi MKIV, která je postavena na jazyce Lua (narozdíl od původní verze MKII). Aplikace uvedených Lua funkcí umožňuje snadné vytváření dokumentů, obsahujících hromadná data z jednoduchých CSV databází, která jsou vhodným způsobem naformátována. Použití je velmi pestré, např. tisk hromadných dopisů, různých formulářů, vysvědčení, pozvánek, průkazek, vizitek, kartiček atd.
Při programování knihovny byl kladen velký důraz na to, aby funkce byly široce použitelné nejen v ConTEXtu (pro který byly primárně napsány), ale i v LuaTEXu a LuaLATEXu. Hlavním cílem bylo také to, aby uživatel (Plainista, LATEXista či ConTEXtista) těchto Lua funkcí nemusel mít vůbec žádné znalosti jazyka Lua, a přitom mohl velmi dobře prakticky tyto funkce používat. Ve skutečnosti se ve zdrojovém TEXovém kódu objevuje volání Lua kódu jen na začátku dokumentu, samotné zpřístupnění CSV dat je realizováno prostřednictvím TEXových maker.

2. CSV formát

Většina čtenářů jistě zná formát CSV (comma-separated values, hodnoty oddělené čárkami). Jde se o velmi jednoduchý souborový formát určený pro výměnu tabulkových dat. Obecně CSV soubor sestává z řádků (záznamů), v nichž jsou jednotlivé položky (pole) odděleny oddělovačem (většinou čárkou nebo středníkem). Hodnoty takto oddělených položek v řádku mohou být navíc vymezeny dalšími znaky (standardně jsou uzavírány do uvozovek). To umožňuje, aby text položek obsahoval i oddělovač polí (tedy např. čárku či středník). Standardně CSV formát umožňuje vymezovačem (např. uvozovkami) vymezit pouze ty položky, které obsahují oddělovač položek, navíc umožňuje i to, aby jednotlivé položky obsahovaly i znak vymezovače (tj. uvozovky) atd. Jednoduchý algoritmus, který jsem navrhnul pro rozparsování položek ovšem předpokládá, že pokud již začneme ve svých CSV souborech používat vymezovač, musí jím být vymezeny úplně všechny položky. Navíc nesmí být znak pro vymezovač obsažen uvnitř žádné ze sloupcových položek. Parsovací funkci `ParseCSVdata` lze samozřejmě zobecnit. Za hojné používání CSV souborů se středníkem (byť pod názvem CSV) vděčí tento formát Microsoft Excelu v české verzi Microsoft Windows. Standardní oddělovač (středník) lze změnit (v místním a jazykovém nastavení).

Ukázky CSV tabulek na něž se budu následně v textu i zdrojovém kódu ukázkového dokumentu odvolávat:

Příklad 1 (zaci.csv) *CSV tabulka bez záhlaví. Položky oddělené středníkem, vymezovač není:*

```
1;Petr;Novák;19.5.1989;m;Nymburk;U Brány 7
2;Jan;Novotný;5.7.1991;m;Praha;Uhlířská 178
...
```

Příklad 2 (studenti.csv) *CSV tabulka se záhlavím. Položky oddělené středníkem, vymezovač není:*

```
Prijmeni;Jmeno;DatumNarozeni;Pohlavi;Mesto;PSC;Ulice
Novák;Jan;14.10.1997;m;Zbečno;27024;Farní 21
Pospíšilová;Hana;4.1.1996;ž;Zábřeh;78901;Studénky 420
...
```

Příklad 3 (zamestnanci.csv) *CSV tabulka se záhlavím. Oddělovačem položek je tentokrát čárka, vymezovačem jsou uvozovky, uvnitř nichž se může vyskytovat separátor polí (čárka):*

```
"PorCis","Prijmeni","Jmeno","Pohlavi","DatumNarozeni","Bydliste"
"1","Májková","Barbora","ž","2.4.1995","Mírov 96, 789 53 Mírov"
"2","Nová","Zuzana","ž","13.1.1995","U Dráhy 8, 789 01 Zábřeh"
...
```

3. Použití CSV tabulek v TEXu

V r. 2005 jsem náhodou objevil makro `scanbase.tex` Petra Olšáka, které mne velmi zaujalo svou praktičností. Petr Olšák jej následně zobecnil a zmodifikoval tak, aby se dalo použít i pro zpracování CSV souborů nejen v Plainu, ale i v LaTeXu (modifikaci v r. 2008 navrhl Jaromir Kuben) a ConTEXtu MKII. Pro svoji práci v Plainu jsem používal makro `scancsv.tex` hezkou řadu let, makro `scancsv-context.tex` občas použiju v ConTEXtu MKII i dnes.
Původní makro Petra Olšáka mi bylo tak zásadní inspirací pro tvorbu mých Lua funkcí, že jsem ve své Lua aplikaci ponechal i stejné názvy hlavních TEXových maker. Důležitým důvodem ovšem bylo také to, že jsem chtěl, aby se mi snáze předělávaly vlastní starší aplikace, založené na použití původního Olšákova makra. V původním zdrojovém textu pak stačilo udělat jen pár drobných úprav a zkompilovat jej ConTEXtem MKIV. Jistou nepříjemností při používání CSV souborů v ConTEXtu (LuaTEXu, LuaLaTEXu) je, že pro správné fungování aplikací je požadováno UTF-8 kódování, zatímco CSV tabulky vytvořené v Microsoft Excelu jsou defaultně kódované CP 1250. Překódování naštěstí snadno zvládne každý solidní textový editor.

4. Knihovna `scancsv.lua`

4.1. Princip fungování luaknihovny

Lua knihovna `scancsv.lua` je tvořena sérií funkcí, které zpřístupňují data v CSV tabulkách (kódovaných UTF-8). Ačkoliv se budu neustále odvolávat na použití knihovny `scancsv.lua` v ConTEXtu MKIV, musím poznamenat, že funkce byly navrženy vesměs tak, aby fungovaly i v LuaTEXu či LuaLaTEXu (bylo testováno pouze na jednoduchých příkladech).
Princip fungování knihovny vychází z již zmiňovaného TEXového algoritmu Petra Olšáka. Postupně jsou data obsažená v řádcích CSV tabulky prostřednictvím programových Lua cyklů rozparsována a zapracována TEXových maker. Makra jsou v průběhu zpracování automaticky nadefinována (vygenerována) a následně mohou být použita ve zdrojovém textu ConTEXtu (LuaLaTEXu či LuaTEXu). První verze knihovny vyžadovala jisté znalosti Lua a určitý způsob umísťování Lua kódu do zdrojového textu. Vzhledem k tomu, že hlavní prioritou bylo, aby mohl knihovnu používat kterýkoliv TEXista, aniž by měl sebemenší znalosti Lua, doznala knihovna nakonec takových změn, že ve zdrojovém dokumentu je umístěn Lua kód pouze na začátku dokumentu [1]. Kód slouží k načtení luaknihovny, což bude předvedeno v ukázkových příkladech pro ConTEXt a LuaLaTEX.

[1] Výjimkou může být volání knihovní funkce `filelineaction(csvfile,fromrow,torow)` v LuaLaTEXu, která zajistí, že se načte určitý rozsah řádků CSV tabulky (`fromrow` – `torow`). Pro zpracování celého CSV souboru je i v LuaLaTEX u připraveno samostatné TEXové makro.

Vzhled výsledného dokumentu (tiskové sestavy) je předurčen libovolně nadefinovaným TEXovým makrem `\lineaction`. V tomto makru mohou být samozřejmě použita všechna makra automaticky vygenerovaná funkcemi knihovny. Sestava může mít libovolný vzhled, nejčastěji tabulkový nebo formulářový. Po ukončení makra `\lineaction` se v cyklu automaticky načte další řádek CSV tabulky, znovu se naplní všechna vygenerovaná makra texty sloupcových položek z tohoto řádku a znovu se spustí `\lineaction`. To se opakuje tak dlouho, dokud není zpracována celá tabulka nebo její část (určená dvojicí parametrů `fromrow` a `torow`). Knihovna makro `\lineaction` implicitně nadefinuje, takže pokud uživatel vlastní definicí toto makro nevytvoří, použije se implicitní nastavení, které udělá to, že se vypíší všechny řádky CSV tabulky v jakémsi „zhuštěném“ formátu.
Já většinou nadefinuji makro `\printaction` (ve kterém se používají automaticky vygenerovaná makra), které určuje vzhled tiskové sestavy a toto makro následně vhodným způsobem zakomponuji do makra `\lineaction`. Volitelně si může uživatel nadefinovat také makra `\blinehook` (háčkové makro, které se vždy provede automaticky před provedením makra `\lineaction` tj. před zpracováním každého řádku CSV tabulky) a `\elinehook` (provede se vždy po `\lineaction`) a dále makra `\bfilehook` a `\efilehook`, což jsou makra, která se provedou před začátkem zpracování a na konci zpracování celého CSV souboru. Pomocí těchto „háčků“ (které jsou implicitně knihovnou nastaveny na `\relax`) lze nastavovat různá záhlaví a zápatí skupin, odstránkování atd. Rád bych upozornil na to, že nic nebrání zpracování i několika různých CSV souborů v jednom dokumentu atd.

4.2. TEXová makra pro zpřístupnění sloupcových dat CSV tabulky

Nejdůležitějšími makry pro použití jsou makra, která zpřístupňují sloupcová data v CSV tabulce. Názvy těchto maker jsou odvozeny od názvů sloupců v excelovské tabulce. Data ve sloupcích A, B, C, atd. jsou k dispozici v makerch s názvy `\cA`, `\cB`, `\cC`, atd.
K vytváření názvů sloupců je potřebná funkce `ar2xls(number)` (Arabic to XLS). Ta převádí pořadové číslo sloupce na název excelovského sloupce. Nepředpokládá se, že by někdo potřeboval více než 703 sloupců (tj. sloupce nad rozsah A – ZZ). Funkci `ar2xls` by bylo nutné v tom případě upravit.
Pokud by někomu místo excelovského formátu názvů sloupců spíše vyhovovaly názvy, založené na římských číslech, tj. `\cI`, `\cII`, `\cIII`, `\cIV`, atd., lze to snadno zařídit nastavením Lua proměnné `UserColumnNumbering` na jinou hodnotu než `XLS` (defaultní hodnota). V tomto případě se použije pro vytváření názvů sloupců funkce `ar2rom(number)`. Počet sloupců v CSV tabulce není nijak limitován (některé verze Excelu však mohou např. obsahovat maximálně 256 sloupců). Pro CSV tabulky s větším počtem sloupců může být ovšem používání excelovského či římského označování sloupců poněkud nepřehledné. Knihovna proto vygene-

ruje i TEXová makra se stejnými názvy, jako jsou názvy polí v prvním řádku CSV tabulky (záhlaví). Uživatel tak nemusí pracně „odpočítávat“ a identifikovat jednotlivé sloupce, postačí mu znát název sloupce a makro tohoto jména mu zpřístupní potřebná data. Je na místě dodat, že tato makra se vygenerují dokonce i v případě, že první řádek je „datový“ (tj. není tzv. záhlavím), přičemž tato volba je nastavena defaultně. Názvy maker v tomto případě budou nejspíš nesrozumitelné, neboť „nevhodné“ názvy sloupců jsou (z hlediska striktních požadavků na názvy TEXových maker) vždy ošetřovány funkcí `TMN(string)` (TEXMacro Name). Vstupním parametrem této funkce je textový řetězec ze záhlaví daného sloupce CSV tabulky. V tomto řetězci se nahradí všechny diakritické znaky české abecedy (pro jiné jazyky je nutná modifikace makra) znaky bez diakritiky, arabské číslice se nahradí římskými, nealfabetické znaky se se nahradí zástupným znakem `"x"`. Nakonec se zkrátí délka makra na 20 znaků (lze zvýšit). Výstupem funkce je název makra, který může být pro uživatele nesrozumitelný, ale TEXu by neměl vadit. Pokud tedy zamýšlíte využívat toho, že knihovna vygeneruje „pěkné“ názvy maker ze záhlaví CSV tabulky, měli byste se při vytváření záhlaví vyvarovat používání všech znaků, které se nemohou vyskytovat v názvech TEXových maker. Nutno dodat, že nastavením Lua proměnné `CSVHeader` na hodnotu `true` docílíme toho, že knihovna první řádek CSV tabulky nepovažuje za datový řádek. Data z tohoto řádku pak knihovna nezpracovává (nevypisuje je). Tento řádek se pak ani nezapočítává do počtu zpracovávaných řádků tabulky. Defaultní hodnota Lua proměnné `CSVHeader` je ovšem `false`, což nebrání knihovně vytvořit z prvního řádku názvy sloupcových maker. Všechny řádky jsou ovšem brány automaticky jako datové a zpracovávají se.

Pokud použijeme ke zpracování CSV soubor, uvedený v příkladu **??**, budou sloupcová data zpřístupněna jednak v makrech `\cA`, `\cB`, `\cC`, ..., `\cH`, ale také v makrech `\Prijmeni`, `\Jmeno`, `\DatumNarozeni` atd., zatímco pokud použijeme ke zpracování CSV soubor, uvedený v příkladu **??**, budou sloupcová data zpřístupněna v makrech `\I=\cA` (číslice 1 je nahrazena římskou číslicí), `\Petr=\cB`, `\Novak=\cC` (písmeno „á“ nahrazeno písmenem „a“), podivně vyhlížející makro `\IIXxVxIIXVIIIIIX=\cD` vzniklo nahrazením arabských cifer a teček. [2]

Všechna výše uvedená TEXová makra vznikají v průběhu cyklu zcela automaticky bez jakéhokoliv přispění uživatele, uživatel je dokonce nemusí ani v ConTEXtu definovat. Tím nastává paradoxní situace, neboť se ve zdrojovém ConTEXtovém textu se mohou objevovat např. makra `\Prijmeni`, `\Jmeno`, `\DatumNarozeni` atd., která ale nejsou nikde výše ve zdrojáku nadefinovaná.

Pokud má někdo strach, že si automaticky generované názvy TEXových maker

[2] Sloupcové údaje jsou v tomto režimu k dispozici i prostřednictvím globálních Lua proměnných CSV.Prijmeni, CSV.Jmeno, CSV.DatumNarozeni atd. Jejich použití však již vyžaduje jisté znalosti Lua.

nezapamatuje, mohu jej uklidnit. Pomocí Lua funkce `CSVReport()` resp. pomocí TEXového makra `\csvreport` si může uživatel kdykoliv ve zdrojovém textu ConTEXtu či LuaLATEXu nechat vypsat nejen řadu užitečných informací o otevřené CSV tabulce, ale mj. i názvy všech vygenerovaných maker.
Vzhledem k tomu, že jsem předpokládal, že většina CSV tabulek nebude obsahovat záhlaví s názvy sloupců, knihovna „předpokládá“, že záhlaví u tabulek nejsou. Implicitní hodnoda Lua proměnné `CSVHeader` je v knihovně určena hodnotou proměnné `UserCSVHeader` (defaultně `false`), což lze v knihovně vyeditovat. Pokud by měl někdo zájem používat snadno zapamatovatelná makra (označující sloupcové položky) a přitom nemít záhlaví CSV tabulek, může to zařídit tak, že si na začátku definice makra `\lineaction` (resp. `\printaction`) pomocí `\let` vytvoří potřebná jména, a jim přiřadí hodnoty standartních excelovských (či římských) názvů maker (např. `\let\PorCislo\cA`). Způsob nastavení vlastních názvů bude později předveden v ukázkovém příkladu.

4.3. TEXová makra pro získání „systémových“ informací

Knihovna vygeneruje automaticky i další TEXová makra, která může uživatel využít pro tvorbu tiskových sestav nebo pro vypsání důležitých informací o zpracovávané CSV tabulce:

`\csvfilename` : Jméno aktuálně otevřeného CSV souboru.

`\numcols` : Počet sloupců CSV tabulky.

`\numrows` : Počet aktuálně zpracovaných (vypsaných) řádků. Může se lišit od celkového počtu řádků tabulky (pokud uživatel vypisuje pouze určitý rozsah řádků).

`\numline` : Pořadové číslo aktuálně načteného řádku. Vhodné pro použití v tiskových sestavách.

`\csvreport` : Reportní informace o otevřeném CSV souboru (mj. názvy všech upotřebitelných TEXových maker atd.).

`\printline` : Aktuální řádek CSV tabulky ve „zhuštěném“ tvaru.

`\printall` : CSV tabulka ve „zhuštěném“ tvaru.

4.4. Modifikace funkčnosti knihovny

Uživatel si může knihovnu do značné míry přizpůsobit svým potřebám. Prostřednictvím několika globálních proměnných lze nastavit řadu defaultních vlastností. Proměnné mohou být přenastaveny buď přímo ve vlastním souboru knihovny, což umožní uživateli „jednou provždy“ provést nastavení vyhovující jeho standartním potřebám (např. oddělovačem položek v používaných CSV tabulkách nebude středník ale čárka). Nastavení hodnot příslušných proměnných lze provést po načtení knihovny i ve zdrojovém TEXovém souboru.
Přímo v knihovně lze nastavit nastavit defaultní hodnoty těchto proměnných:

`UserCSVSeparator` : Používaný separátor polí. Defaultní je středník.

`UserCSVLeftDelimiter` : Pole mohou být vymezena vymezovačem. To umožní používat znak pro separátor polí i uvnitř textu pole. Defaultní je prázdný řetězec `UserCSVLeftDelimiter=''`.

`UserCSVRightDelimiter` : Lze nastavit i pravý vymezovač. Defaultní hodnota je stejná jako v předchozím případě.

`UserCSVHeader` : Načtený CSV soubor je defaultně posuzován jako bez hlavičky (údaje v hlavičce jsou brány jako názvy sloupcových maker). Defaultně je `UserCSVHeader=false`.

`UserColumnNumbering` : Něco jiného než XLS nebo nedefinovaná hodnota (např. zakomentováním řádku) nastaví římské číslování. Defaultní je `XLS`.

Ve zdrojovém textu lze „přebít“ defaultní hodnoty výše uvedených proměnných (v případě, že mimořádně zpracováváme jiný typ CSV souboru než je běžné) a změnit tak chování knihovny prostřednictvím nastavení těchto proměnných:

`File2Scan` : Jméno používaného CSV souboru. Proměnná se může nastavit manuálně nebo se nastaví automaticky při volání funkce, jejímž parametrem je název souboru. Její nastavení umožňuje volat funkce `OpenCSVFile()`, `CSVReport()`, `lineaction()` a `filelineaction()` bez parametrů. Nastavit proměnnou lze i makrem (např. `\setfiletoscan{zaci.csv}`).

`Sep` : Oddělovač sloupců v CSV souboru. Defaultní hodnota separátoru je v proměnné `UserCSVSeparator`. Některé znaky bohužel nelze použít jako separátor. Použít lze rovněž makro (např. `\setsep{,}`).

`Ld` : Left delimiter. Sloupcové položky mohou být ohraničeny jinak zleva a jinak zprava. Defaultní hodnota je v proměnné `UserCSVLeftDelimiter`. Některé znaky nelze použít. K dispozici je makro (např. `\setld{"}`).

`Rd` : Right delimiter. Defaultní hodnota je v proměnné `UserCSVRightDelimiter`. Nejčastěji bývají jednotlivé položky ohraničeny znakem `"` (tj. vloženy do uvozovek). Makrem např. `\setrd{"}`.

`CSVHeader` : Hodnota `true`, znamená, že 1. řádek obsahuje záhlaví se jmény sloupců (tj. není datový). Defaultní hodnota `false` – všechny řádky jsou uvažovány jako datové. Lze užít TEXové makro `\setheader`.

Další možnosti nastavení zvídavý čtenář vyčte přímo z knihovního souboru `scancsv.lua`. Ten obsahuje vysvětlující komentáře, z nichž lze vyčíst princip fungování.

4.5. Praktické ukázky užití knihovny

Po dohodě s redakcí jsme ustoupili od zveřejnění zdrojového textu knihovny. Zdrojový text obsahující množství komentářů si bude moci čtenář spolu s řadou funkčních příkladů stáhnout z úložiště a následně prostudovat, vyzkoušet a otestovat. V tuto chvíli pojďme ukázat pouze výpis krátkého programu. Z něj snad bude patrné, jak knihovna funguje.

V následujícím zdrojovém textu budeme předpokládat, že používáme při práci UTF-8 kódované CSV soubory `zaci.csv`, `studenti.csv` a `zamestnanci.csv` z příkladů **??**, **??** a **??**. Zdrojový text příkladu je pro pochopení fungování dostatečně okomentován. Ukázka výstupu následujícího příkladu je na obr. 1 a obr. 2. Na nich jsou předvedeny poslední dvě stránky výsledného PDF souboru.

```
% Ukázka použití. Kompilace: lualatex examples-latex.tex
\documentclass{article}
\usepackage{luatextra}
\usepackage[utf8]{luainputenc}
\parindent0pt

\begin{document}
\pagestyle{empty}

% Načtení knihovny
\directlua{dofile("../scancsv-general/scancsv.lua")}

% Volitelné definice "háčků" (defaultně jsou háčky "zarelaxovány")
% Hooks před a po zpracování celé tabulky:
\def\bfilehook{Seznam účastníků zájezdu {\bf\csvfilename}:
  \par\bigskip}
\def\efilehook{\par Sestavou bylo vypsáno {\bf\numrows} účastníků.
\bigskip}

% Hooks před a po zpracování řádkové informace:
\def\blinehook{Účastník č. \numline :\par} % akce před výpisem ř.
\def\elinehook{\par\smallskip\hrulefill\medskip\par} % po výpisu ř.

% Definice pomocných vyhodnocujících maker použitých v sestavách
\def\priponaa{\if m\pohl\else a\fi}
\def\priponai{\if m\pohl\else í\fi}

% Definice tiskové sestavy:
\def\printaction{
pan\priponai\ \Jmeno\ \Prijmeni\par
narozen\priponaa\ \DatumNarozeni\par
bytem \bydliste
}

% Pro zpracování nyní stačí definice makra \lineaction např. takto:
% \def\lineaction{\printaction}
```

```
% Vzhledem k ukázkovému zpracování více CSV souborů příklad
% poněkud zobecníme (byť za cenu menší přehlednosti)

% Vlastní tiskové makro pro soubor zaci.csv si připravíme např. takto:
% Uvědomme si, že tento CSV soubor je "bez hlavičky" tj. není známa
% informace o názvech sloupců

\def\lineaction{
  % Kvůli přehlednosti si můžeme nastavit potřebná makra
\let\Jmeno\cB % ve 2. sloupci tj. excelovsky sloupec B
\let\Prijmeni\cC
\let\DatumNarozeni\cD
\let\pohl\cE% pro fungování pomocných maker \priponaa a \priponai
\def\bydliste{\cG, \cF} % můžeme i definovat nová makra
% Cílem těchto "tanečků" je docílit toho, aby byla všechna makra
% používaná v tiskové sestavě \printaction nadefinována dříve, než
\printaction  % zavoláme.
}

% Zpracování sestavy pro tabulku zaci.csv zařídí Lua tento kód:
%\directlua{filelineaction("zaci.csv")}

\filelineaction{zaci.csv} % nebo příslušné TeXové makro

% Závěrečná ukázka použití některých "systémových" maker:

Poslední zpracovaný řádek tabulky \csvfilename:\par\smallskip

\printline\bigskip

CSV tabulka \csvfilename\ ve "zhuštěném tvaru":\par\smallskip

\printall\pagebreak

Výpis reportních informací z CSV souboru \csvfilename:\par\bigskip

\csvreport

% Ukázka toho, jak naráz zpracovat více různých CSV souborů:

% Tiskové makro pro soubor studenti.csv si připravíme jinak.
```

```
% Tabulka má záhlaví s názvy sloupců. Lze se tak odvolávat na názvy
% sloupcových maker, případně si vytvořit další potřebná makra

\def\lineaction{
\let\pohl\Pohlavi% fungování pomocných maker \priponaa a \priponai
\def\bydliste{\Ulice, \PSC\ \Mesto} % bydliště trochu jinak
\printaction % standartní tisková sestava (lze užít i jinou)
}

% Vlastní zpracování sestavy pro tabulku studenti.csv
\setheader % nastaví CSVHeader=true - příznak existence záhlaví
% \filelineaction{studenti.csv} % V LaTeXu zpracuje jen celý soubor
% nebo pomocí Lua i souvislé skupiny řádků
% filelineaction("studenti.csv",3,5) -- vypíše jen od ř.3 do ř.5

\directlua{filelineaction("studenti.csv",3,5)}
% \directlua{filelineaction("studenti.csv",8,17)} % další řádky

Poslední zpracovaný řádek tabulky \csvfilename:\par\smallskip

\printline\bigskip

CSV tabulka \csvfilename\ ve "zhuštěném tvaru":\par\smallskip

\printall\pagebreak

Výpis reportních informací z CSV souboru \csvfilename:\par\bigskip

\csvreport\pagebreak

% Zveřejněná ukázka výstupu programu bude provedena od tohoto místa:

% Ukázka toho, že lze také zpracovat i CSV soubor s jiným formátem:
% Příprava tiskového makra pro soubor zamestnanci.csv
% Tabulka má také záhlaví s názvy sloupců, takže postačí jen:
\def\lineaction{
\let\pohl\Pohlavi% Pro pomocná makra
\def\bydliste{\Bydliste} % jinak - dle CSV tabulky zamestnanci.csv
\printaction % standartní tisková sestava (lze ale použít i jinou)
}

% Vlastní zpracování sestavy pro tabulku zamestnanci.csv
```

```
% Předem nastavíme potřebné proměnné modifikující chování knihovny
\setheader % tj. soubor má hlavičku s názvy polí
\setsep{,} % oddělovač polí
\setld{"} % levý vymezovač
\setrd{"} % pravý vymezovač
\filelineaction{zamestnanci.csv}

Poslední zpracovaný řádek tabulky \csvfilename:\par\smallskip

\printline\bigskip

CSV tabulka \csvfilename\ ve "zhuštěném tvaru":\par\smallskip

\printall\pagebreak

Výpis reportních informací z CSV souboru \csvfilename:\par\bigskip

\csvreport

\end{document}
```

5. Netriviální použití luaknihovny

Jednoduchá ukázka nemůže plně vystihnout možnosti popisované luaknihovny, proto jsem pro zájemce připravil i netriviální ukázky jejího praktického použití. Na několika příkladech si můžete systém vyzkoušet a nechat se inspirovat možnostmi. Na mém datovém úložišti jsou uloženy dokumenty, které byly vytvořeny pro účely administrativní agendy ConTEXtmeetingu 2010 a TEXperience 2010. Jedná se o seznamy účastníků, identifikačních kartiček, časových rozvrhů a programů konferencí (viz ukázka na obr. 3 a obr. 4). Připravena je i ukázka současného zpracování tabulek `ucitele.csv` a `zaci.csv` (s fiktivními údaji), které jsou během zpracování propojeny přes název třídy pomocí relace `1:N`. Výsledkem je dokument se seznamy tříd s jejich třídními učiteli. Všechny ukázky jsou připraveny v mém oblíbeném ConTEXtu, stejně tak jako ukázka tisku maturitních protokolů (s fiktivními údaji), které byly na našem gymnáziu několik let prakticky využívány. Aby nepřišli zkrátka ani zájemci z řad LATEXistů, přiložil jsem dvě ukázky pro tisk obálek a faktur. Modifikací zdrojových souborů a jejich následnou kompilací můžete proniknout do možností luaknihovny. Jednotlivé zdroje lze kompilovat aktuálními verzemi LuaLATEXu a ConTEXtu.
Všechny dokumenty potřebné pro vyzkoušení fungování knihovny naleznete na adrese `http://cstugbull2012.hajtmar.com`.

Seznam účastníků zájezdu **zamestnanci.csv**:

Účastník č. 1:
paní Barbora Májková
narozena 2.4.1995
bytem Mírov 96, 789 53 Mírov

Účastník č. 2:
paní Zuzana Nová
narozena 13.1.1995
bytem U Dráhy 8, 789 01 Zábřeh

Účastník č. 3:
pan Petr Dyk
narozen 10.4.1995
bytem Tůmova 207, 789 01 Zábřeh

Účastník č. 4:
paní Anežka Ferová
narozena 1.1.1995
bytem Revoluční 56, 787 01 Šumperk

Účastník č. 5:
pan Jan Neděla
narozen 2.4.1995
bytem Bratrušovská 31, 787 01 Šumperk

Sestavou bylo vypsáno **5** účastníků.

Poslední zpracovaný řádek tabulky zamestnanci.csv:

6,Neděla,Jan,m,2.4.1995,Bratrušovská 31, 787 01 Šumperk,

CSV tabulka zamestnanci.csv ve "zhuštěném tvaru":

1, Májková, Barbora, ž, 2.4.1995, Mírov 96, 789 53 Mírov,
2, Nová, Zuzana, ž, 13.1.1995, U Dráhy 8, 789 01 Zábřeh,
3, Dyk, Petr, m, 10.4.1995, Tůmova 207, 789 01 Zábřeh,
4, Ferová, Anežka, ž, 1.1.1995, Revoluční 56, 787 01 Šumperk,
6, Neděla, Jan, m, 2.4.1995, Bratrušovská 31, 787 01 Šumperk,

Obr. 1 Ukázka výstupu vygenerovaného triviálním ukázkovým příkladem. Zdrojový text příkladu je na předchozích stránkách.

Výpis reportních informací z CSV souboru zamestnanci.csv:

Informace o používaném CSV souboru
Vstupní CSV soubor: **zamestnanci.csv**
Vymezovače a oddělovače sloupců viz. Lua proměnné `Sep`, `Ld` a `Rd`
Nastavení vymezovačů a oddělovačů sloupců: `"pole1","pole2","pole3",` …
Počet sloupců v tabulce: **6**
Počet řádků v tabulce : **5**
Makra zpřístupňující řádková data v tabulce:
\cA=\PorCis, **\cB=\Prijmeni**, **\cC=\Jmeno**, **\cD=\Pohlavi**, **\cE=\DatumNarozeni**, **\cF=\Bydliste**,
Další předdefinovaná makra:
\csvfilename – otevřený soubor s CSV tabulkou (**zamestnanci.csv**)
\numcols – počet sloupců tabulky (**6**)
\numrows – počet aktuálně zpracovaných (vypsaných) řádků (**5**)
\numline – číslo aktuálně načteného řádku (pro použití v tiskových sestavách)
\csvreport – vypíše se report o otevřeném souboru
\printline – vypíše aktuální řádek CSV tabulky ve zhuštěném tvaru
\printall – vypíše CSV tabulku ve zhuštěném tvaru
\setfiletoscan{*filename*} – nastaví jméno zpracovávaného CSV souboru
\opencsvfile{*filename*} – otevření CSV souboru
\openheadercsvfile{*filename*} – otevření CSV souboru s hlavičkou
\setheader – nastavení příznaku existence hlavičky
\resetheader – nastavení příznaku neexistence hlavičky
\readrow – načtení řádku (+ zůstane na něm)
\nextrow – načtení řádku (+ skok na další)
\setsep{*separator*} – nastavení oddělovače polí na příslušný znak
\resetsep – nastavení oddělovače polí na defaultní hodnotu
\setld{*delimiter*} – nastavení levého vymezovacího znaku
\resetld – nastavení levého vymezovače na defaultní hodnotu
\setrd{*delimiter*} – nastavení pravého vymezovacího znaku
\resetrd – nastavení pravého vymezovače na defaultní hodnotu
\blinehook – nastavení háčku (vykonaného před načtením řádku)
\elinehook – nastavení háčku (vykonaného po zpracování řádku)
\bfilehook – nastavení háčku (vykonaného před načtením CSV souboru)
\efilehook – nastavení háčku (vykonaného po zpracování CSV souboru)

Obr. 2 Výpis reportních informací spolu se seznamem maker, které má uživatel knihovny `scancsv.lua` k dispozici.

Preliminary list of participants of ConTEXt meeting and TEXperience
Brejlov 13. – 19. 9. 2010
(to date 6. 9. 2010)

Id	Name	Nationality	Monday – Thursday	Thursday p.m.	Friday	Saturday	Sunday	E-mail
1	Bašić Nino		1	1	1	1	1	nino.basic@gmail.com
2	Braslau Alan		1	1	1	1	1	alan.braslau@cea.fr
3	Brejcha Miloš		1	1	1	1	1	brejcha@vydavatelskyservis.cz
4	Březina David		0	1	1	1	1	brezina@davi.cz
5	Egger Willi		1	1	1	1	1	w.egger@boede.nl
6	Giedryś Anna		0	0	0	0	0	brezina@davi.cz
7	Goddijn Frans		1	1	1	1	1	frans@goddijn.com
8	Gundlach Patrick		1	1	1	1	0	gundlach@speedata.de
9	Guravage Michael		1	1	1	1	1	guravage@literatesolutions.com
10	Habrovanská Pavlína		0	1	1	1	1	jan.viktorin@gmail.com
11	Hagen Hans		1	1	1	1	1	pragma@wxs.nl
12	Hajtmar Jaroslav		1	1	1	1	1	hajtmar@gyza.cz
13	Hála Tomáš		0	1	1	1	1	thala@pef.mendelu.cz
14	Haltiwanger John		1	1	1	1	1	john.haltiwanger@gmail.com
15	Hoekwater Taco		1	1	1	1	1	taco@bittext.nl
16	Homolka Lubor		1	1	1	1	1	homolkalubor@gmail.com
17	Horák Karel		0	1	1	1	1	horakk@math.cas.cz
18	König Harald		1	1	1	1	0	koenig@linux.de
19	Kula Jano		1	1	1	1	1	jano.kula@tiscali.cz
20	Miklavec Mojca		1	1	1	1	1	mojca.miklavec.lista@gmail.com
21	Novotný Lukáš		0	1	1	1	1	l.novotny@t-email.cz
22	Oliák Mirek		0	0	0	0	0	mirek@olsak.net
23	Oliák Petr		0	1	1	1	1	petr@olsak.net
24	Pavneet Arora		1	1	1	1	1	pavneet_arora@yaroc.com
25	Píška Karel		1	1	1	1	1	piska@fzu.cz
26	Procházka Lukáš		1	1	1	0	0	lPr@pontex.cz
27	Přichystal Jan		0	1	1	1	1	jprich@pef.mendelu.cz
28	Ramm Henning Hraban		1	1	1	1	1	hraban@ice.net
29	Reutenauer Arthur		1	1	1	1	1	arthur.reutenauer@normalesup.org
30	Rybička Jiří		0	1	1	1	1	rybicka@mendelu.cz
31	Scarso Luigi		1	1	1	1	1	luigi.scarso@gmail.com
32	Schuster Wolfgang		1	1	1	1	1	wolfgang.schuster@gmail.com
33	Strzelczyk Piotr		1	1	1	1	0	piotr@eps.gda.pl
34	Stříž Martin		1	1	1	1	1	martin@striz.cz
35	Stříž Milan		1	1	1	1	1	mstriz@tor.cz
36	Stříž Pavel		1	1	1	1	1	striz@fame.utb.cz
37	Střížová Yvona		1	1	1	1	1	yvonastrizova@seznam.cz
38	Štěpnička Jan		0	1	1	1	1	Jan.Stepnicka@osu.cz
39	Štěpničková Iva		0	1	1	1	1	Jan.Stepnicka@osu.cz
40	Štourač Milan		0	1	1	1	1	stourac.milan@seznam.cz
41	Šustek Jan		0	1	1	1	1	Jan.Sustek@osu.cz
42	Talandová Petra		0	1	1	1	1	PTalandova@seznam.cz
43	Tolarová Leona		0	0	0	0	0	leona.tolarova@czso.cz
44	Trušník Roman		0	1	1	1	1	trusnik@fhs.utb.cz
45	Van Deventer Eva		1	1	1	1	1	Vdeven@unisa.ac.za
46	Vieth Ulrik		1	1	1	1	0	ulrik.vieth@arcor.de
47	Viktorin Jan		0	1	1	1	1	jan.viktorin@gmail.com
48	Voipio Mari		1	1	1	1	0	mari.voipio@xpatents.com
49	Wagner Zdeněk		0	1	1	1	1	zdenek.wagner@gmail.com
50	Wolfrum Steffen		1	1	1	1	0	context@post.werksatz.com
–	Number of participants:	–	31	47	47	46	40	–

This document was created (generated) using scancsv.lua library by Jaroslav Hajtmar.
Scancsv.lua presentation to be held on Saturday, 18. 9. 2010 on 4OH+3TE in Brejlov.

Obr. 3 Ukázka netriviálního použití knihovny `scancsv.lua`. Využito pro administrativní agendu dvojkonference v Brejlově.

Programme of ConTEXt meeting and TEXperience
Brejlov 13. – 19. 9. 2010

Monday 13.9.

18:30	**Dinner**			90	Notice
20:00	Taco Hoekwater / Hans Hagen / Mojca Miklavec	ConTeXt installation clinic	workshop	120	Notice

Tuesday 14.9.

8:00	**Breakfast**			60	Notice
9:00		Conference opening		15	Notice
9:15	Arthur Reutenauer	Keynote talk	talk	15	Notice
9:30	Taco Hoekwater	Reference manual update	talk	15	Notice
9:45	Patrick Gundlach	Wiki context reference	talk	15	Notice
10:00	Taco Hoekwater / Hans Hagen	Other documentation news	talk	15	Notice
10:15	Peter Münster (presented by Taco)	Reference syntax in lua	talk	15	Notice
10:30	**Coffee**			30	Notice
11:00	Hans Hagen	Why structure matters	talk	30	Notice
11:30	Wolfgang Schuster	Module documentation	talk	30	Notice
12:00	Luigi Scarso	modules_mklv	talk	30	Notice
12:30	**Lunch**			90	Notice
14:00	Mari Voipio	A Different Philosophy – Thoughts on teaching ConTeXt to non-techies	discussion	30	Notice
14:30	Mari Voipio	Observations on approaches to learning ConTeXt and writing documentation to match	discussion	60	Notice
15:30	**Coffee**			30	Notice
16:00	Wolfgang Schuster	xml interface files and internal module doku (%D, %M etc.)	talk	30	Notice
16:30	Hans Hagen	Requirements For Documentation	talk	30	Notice
17:00	Hans Hagen	My Slowly Growing Test Suite	talk	30	Notice
17:30		Documentation discussion	discussion	60	Notice
18:30	**Dinner**			90	Notice
20:00	Idris Samawi Hamid	Towards the First ConTeXt Book	talk	30	Notice
20:30	Hans Hagen	Document workflow	tutorial	30	Notice
21:00	Hans Hagen	Whatever You Want To Know	q&a	90	Notice

Wednesday 15.9.

8:00	**Breakfast**			60	Notice
9:00	Mari Voipio	Uses for ConTeXt in a standard office environment	talk	30	Notice
9:30	Hans Hagen / Mojca Miklavec	The database module, MkIV version	talk	30	Notice
10:00	Hans Hagen	XML processing news	talk+workshop	60	Notice
11:00	**Coffee**			30	Notice
11:30	Alan Braslau	Plotting data with Metafun/Metapost	talk+discussion	45	Notice
12:15	Alan Braslau	Drawing diagrams using the chart module	talk+discussion	45	Notice
13:00	**Lunch**			90	Notice
14:30	Hans Hagen	Font Goodies	talk	30	Notice
15:00	Mojca Miklavec	Some thoughts on typescripts	talk+discussion	60	Notice
16:00	**Coffee**			30	Notice
16:30	Aditya Mahajan (presented by Luigi)	Beg, borrow, and steal – running external filters in ConTeXt	talk	30	Notice
17:00	Alan Braslau	Drawing chemical structures using ppchTeX	talk	30	Notice
17:30	Wolfgang Schuster	The letter module	talk+workshop	90	Notice
19:00	**Dinner**			90	Notice
20:30	Taco Hoekwater	tlcontrib.metatex.org	talk	30	Notice
21:00	Mojca Miklavec	ConTeXt minimals, "Server edition"	talk+discussion	30	Notice
21:30	Patrick Gundlach	contextgarden.net	q&a	30	Notice

Obr. 4 Pro vygenerování výstupu byl použit zkonvertovaný nasdílený Google document s oficiálním programem dvojkonference (ConTEXt meeting a TEXperience 2010) v Brejlově.

6. `Scancsv.lua` a ConTEXtový modul `t-scancsv.lua`

Hlavní funkce luaknihovny `ParseCSVdata()` obsahuje jen jednoduchý parsovací algoritmus vycházející z formátu CSV tabulek, které používám. Algoritmus není schopen zpracovávat obecné CSV soubory, mající některé položky vymezeny vymezovačem a jiné nikoliv (když položky obsahují oddělovač polí). Stejně tak není schopen správně zpracovat CSV soubory, obsahující uvnitř vymezovačů samotné vymezovače (text pole vymezený uvozovkami obsahuje uvozovky). V ConTEXtovém modulu `t-scancsv.lua` [3], který vznikl kompletním přepracováním luaknihovny na základě inspirujících připomínek a oprav Hanse Hagena, je řada rozšíření a vylepšený parsovací algoritmus.
Skalní TEXisty jistě nepřekvapí, že zdrojový kód luaknihovny je oproti zmíněnému TEXovému zdrojáku makra p. Olšáka značně obsáhlý (a to nejen díky mým rozsáhlým komentářům). Funkčnost obou systémů je v podstatě stejná. Zmíněný ConTEXtový modul má navíc již několik zajímavých vylepšení mj. i možnost použití nepodmíněných i podmíněných cyklů, maker s proměnlivým počtem parametrů, možností relačního spojování tabulek (`1:N`) atd. Podstatným způsobem to rozšiřuje možnosti praktického použití (filtrování rozsáhlých CSV tabulek atd). Svoji původní luaknihovnu `scancsv.lua` (z r. 2010) tak mohu doporučit pro jednodušší praktické použití nebo alespoň jako studijní či inspirační materiál zájemcům o jazyk Lua a jeho používání ve vlastních TEXových dokumentech a makrech. S ConTEXtovým modulem `t-scancsv.lua` lze řešit i náročnější úlohy (viz ukázky na úložišti).

7. Závěr

Popisovaná luaknihovna i ConTEXtový modul vznikly díky laskavé pomoci členů mailové konference `ntg-context@ntg.nl`. Speciálně chci poděkovat Hansovi Hagenovi, Wolfgangu Schusterovi a Taco Hoekwaterovi. Členům mailové konference `csTeX@cs.felk.cvut.cz` chci poděkovat za rady a pomoc týkající se obecně TEXu. Speciální poděkování patří Zdeňku Wagnerovi a Petru Olšákovi. Pavlu Střížovi děkuji za testování knihovny, cenné rady a za to, že mne inspiroval luaknihovnu dopracovat do fáze, kdy ji snad může využít i někdo jiný.
Popisovaná luaknihovna by měla být považována za moji „luaprvotinu“. Nedokonalosti a neoptimálnosti kódu nechť jsou tomu přičítány. Lua jazykem jsem se začal zabývat především proto, aby se mohl oprostit zejména od častého používání jazyka Perl spolu s ConTEXtem (MKII). Pro možnosti jazyka Lua a jeho integraci

[3] V červnu 2011 jsem kompletně původní Lua knihovnu přepracoval a vytvořil regulérní luamodul pro ConTEXt MKIV. Tento modul (`t-scancsv.lua`) obsahuje řadu zajímavých vylepšení, nové funkce a je optimalizován pro ConTEXt, v němž byl také již mnohokrát v ostrém provozu otestován.

do TEXu jsem se nadchnul a všem TEXistům doporučuji Lua jazyk minimálně vzít na milost :-). Doufám, že se mezi čtenáři najdou i ti, kdo se mým příspěvkem nechají inspirovat a rozšíří řady příznivců jazyka Lua a LuaTEXu, LuaLATEXu či ConTEXtu.

Summary: ScanCSV – Lua Library for CSV file ConTEXt and LuaLATEX processing

Data stored in CSV (Comma Separated Values) files are often used in data processing. This article describes the author's `scancsv.lua` library, its origin and demonstrates practical examples of its usage in ConTEXt MKIV and LuaLATEX. Author shows how easily and quickly create print reports, letters, forms, certificates, invitations, cards, business cards, double-sided cards, tables, animations etc. using external CSV text databases.
Users of ConTEXt MKIV (but LuaLATEX and LuaTEX as well) can easily use data from external CSV tables in their own documents via the library, using the TEX macros built on the library and make this data available in an attractive and very simple and natural way.

Jaroslav Hajtmar, `hajtmar@gyza.cz`
Gymnázium Zábřeh
Nám. Osvobození 20
789 01 Zábřeh