srovnání technologií digitální sazby z uživatelského hlediska

latex3, luatex a markdown

vítek starý novotný

ossconf

3. 7. 2026

úvod o autorovi

vítek starý novotný (witiko)[1] ***1992**, český krumlov

2004–2012 jazykové gymnázium j.n.n v č. budějovicích

2012–2022 bc–phd studium na fi mu,[2] šablony, výuka
2016– výbor a technická redakce cstugu[3]
2016– vývoj texového balíčku markdown[4]
2018–2022 vývoj pythonové knihovny gensim[5]

2023– markdown šablony pro istqb[6]
2023– zpracování dokumentů pro pii tools[7]
2024– vývoj statického analyzátoru expltools[8]
2025– šéfredaktor zpravodaje cstugu[9]

1 github.com/witiko 2 muni.cz/lide/409729-vit-stary-novotny 3 cstug.cz 4 ctan.org/pkg/markdown
5 radimrehurek.com/gensim 6 github.com/istqborg 7 pii-tools.com 8 ctan.org/pkg/expltools 9 bulletin.cstug.cz

úvod o přednášce *představení technologií*

tex typografický stroj a programovací jazyk
latex nadstavba texu (formát), většina současné publikační činnosti v matematice a fyzice[10]
latex3 současná vývojová verze latexu; nejviditelnější rozdíl oproti latexu 2e: jazyk expl3
lua vysokoúrovňový skriptovací jazyk **markdown** odlehčený znač. jazyk pro zápis html
pdftex rozšíření typografického stroje texu, umí generovat výstup ve formátu pdf
xetex rozšíření typografického stroje pdftexu, které zahrnuje mj. podporu opentype fontů
lua(meta)tex rozšíření typografického stroje pdftexu, které zahrnuje mj. interpretr jazyka lua
regulární výraz předpis pro konečný automat, slouží pro vyhledávání informací v textu
peg bezkontextová gramatika, kde operátor volby (|) není komutativní, silnější než regulární
lpeg standardní modul jazyka lua, který umožňuje konstruovat pe gramatiky

10 brischoux, françois, and pierre legagneux. „don't format manuscripts.“ *the scientist* 23.7 (2009): 24.

úvod o přednášce *struktura přednášky, motivace*

struktura přednášky: **while true do**
1 zvolím dvojici technologií
2 popíšu kritéria, pomocí kterých technologie porovnám
3 vyhodnotím jednotlivá kritéria
end

během přednášky se pokusíme zjistit odpovědi na následující otázky:
1 je markdown lepší než latex?
2 jak lze uplatnit princip nedestruktivních úprav při sazbě dokumentů v jazyce markdown?
3 je lepší programovat stroj texu skrz jazyk lua, nebo skrz expl3?
4 jsou pegy lepší než regulární výrazy?
5 je luatex lepší než xetex a pdftex?

autorský pohled i markdown vs. latex *kritéria*

1 čitelnost zdrojového textu
2 oddělení obsahu od formy
3 nutnost programovat
4 sazba matematiky a tabulek
5 analýza, víceformátové publikování

markdown vs. latex kritéria *čitelnost zdrojového kódu*

```
# nadpis první úrovně
## nadpis druhé úrovně
_zdůrazněný text_
[odkaz](https://ossconf.fri.uniza.sk/)

1. číslovaný seznam
2. …
```

```
\documentclass{article}
\usepackage{hyperref}
\begin{document}
\section{nadpis první úrovně}
\subsection{nadpis druhé úrovně}
\emph{zdůrazněný text}
\href{https://ossconf.fri.uniza.sk/}{odkaz}
\begin{enumerate}
\item číslovaný seznam
\item …
\end{enumerate}
\end{document}
```

je markdown čitelnější než latex?

ano, ale především pro jednoduché dokumenty, vizte dále

markdown vs. latex kritéria *oddělení obsahu od formy*

```
# nadpis první úrovně {#identifikátor}
```

```
![alt](img.jpg "popis"){.center, #img,
                        width=100%}
```

```
\documentclass{article}
\begin{document}
\section{nadpis první úrovně}
\label{identifikátor}
\begin{figure}
\centering
\includegraphics[alt=alt,width=1\linewidth]{img.jpg}
\caption{popis}
\label{img}
\end{figure}
\end{document}
```

umožňují html atributy čistší oddělení obsahu od formy než makra latexu?

jsou deklarativnější, pro dosažení speciálních efektů netřeba předefinovávat např. `\label`

markdown vs. latex kritéria *nutnost programovat*

```
# nadpis první úrovně {#identifikátor}
```

```
\documentclass{article}
\begin{document}
\section{nadpis první úrovně}
\label{identifikátor}
```

```
![alt](img.jpg "popis"){.center, #img,
                        width=100%}
```

```
\begin{figure}
\centering
\includegraphics[alt=alt,width=1\linewidth]{img.jpg}
\caption{popis}
\label{img}
\end{figure}
\end{document}
```

autor programovat nemusí, ale kdo zajistí překlad atributů na koncový výstup?
programátorská práce se přesouvá na grafika/programátora, dává smysl především u šablon

markdown vs. latex kritéria *sazba matematiky i*

jednoduchou matematiku lze zapsat snadno:

```
 $ a^2 + b^2 = c^2 $

$$ E = mc^2 $$
```

```
\documentclass{article}
\begin{document}
 $ a^2 + b^2 = c^2 $

$$ E = mc^2 $$
\end{document}
```

co ale se strukturním obsahem jako definice, lemmy, věty a důkazy?

13 witiko.github.io/markdown/#option-texmathdollars 14 pandoc.org/MANUAL.html#extension-tex_math_dollars

markdown vs. latex kritéria *sazba matematiky ii*

nejsnáze přes nadpisy:

```
### definice
funkce $f: X \to Y$ je prostá, když \[
  f(x_1) = f(x_2) \implies x_1 = x_2
\] pro všechna $x_1, x_2 \in X$.
```

```
\documentclass{article}
\usepackage{amsmath,amssymb,amsthm}
\theoremstyle{definition}
\newtheorem{definice}[veta]{Definice}
\theoremstyle{remark}
\newtheorem{poznamka}[veta]{Poznámka}
\begin{document}
\begin{definice}
funkce $f: X \to Y$ je prostá, když \[
  f(x_1) = f(x_2) \implies x_1 = x_2
\] pro všechna $x_1, x_2 \in X$.
\end{definice}
\end{document}
```

markdown vs. latex kritéria *sazba matematiky iii*

lépe přes `<div class="definice">`, ale horší podpora náhledu v textových editorech:

```
::: definice
funkce $f: X \to Y$ je prostá, když \[
  f(x_1) = f(x_2) \implies x_1 = x_2
\] pro všechna $x_1, x_2 \in X$.
:::
```

```
\documentclass{article}
\usepackage{amsmath,amssymb,amsthm}
\theoremstyle{definition}
\newtheorem{definice}[veta]{Definice}
\theoremstyle{remark}
\newtheorem{poznamka}[veta]{Poznámka}
\begin{document}
\begin{definice}
funkce $f: X \to Y$ je prostá, když \[
  f(x_1) = f(x_2) \implies x_1 = x_2
\] pro všechna $x_1, x_2 \in X$.
\end{definice}
\end{document}
```

15 witiko.github.io/markdown/#fenced-divs 16 pandoc.org/MANUAL.html#extension-fenced_divs

markdown vs. latex kritéria *sazba matematiky iv*

přes surové `{=latex}` bloky (čitelnost?):

````
``` {=latex}
\begin{definice}
funkce $f: X \to Y$ je prostá, když \[
  f(x_1) = f(x_2) \implies x_1 = x_2
\] pro všechna $x_1, x_2 \in X$.
\end{definice}
```
````

```
\documentclass{article}
\usepackage{amsmath,amssymb,amsthm}
\theoremstyle{definition}
\newtheorem{definice}[veta]{Definice}
\theoremstyle{remark}
\newtheorem{poznamka}[veta]{Poznámka}
\begin{document}
\begin{definice}
funkce $f: X \to Y$ je prostá, když \[
  f(x_1) = f(x_2) \implies x_1 = x_2
\] pro všechna $x_1, x_2 \in X$.
\end{definice}
\end{document}
```

17 witiko.github.io/markdown/#option-rawattribute 18 pandoc.org/MANUAL.html#extension-raw_attribute

markdown vs. latex kritéria *sazba tabulek i*

jednoduché tabulky lze zapsat (čitelně):

```
| den     | předpověď |
|:--------|:----------|
| pondělí | slunečno  |
| úterý   | polojasno |
| středa  | oblačno   |
| čtvrtek | zataženo  |
| pátek   | přeháňky  |
| sobota  | kroupy    |
| neděle  | sníh      |

: popisek tabulky
```

```
\documentclass{article}
\begin{document}
\begin{table}
\begin{tabular}{ll}
den     & předpověď \\
\hline
pondělí & slunečno  \\
úterý   & polojasno \\
středa  & oblačno   \\
čtvrtek & zataženo  \\
pátek   & přeháňky  \\
sobota  & kroupy    \\
neděle  & sníh      \\
\end{tabular}
\caption{popisek tabulky}
\end{table}
\end{document}
```

ale co složitější tabulky?

markdown vs. latex kritéria *sazba tabulek ii*

```
clause | iso 25010:2023 | iso 25010:2011 | notes
:--|---|---|---
3.4.6 | inclusivity | [accessibility]{rows=2} | [split and renamed]{rows=2}
3.4.7 | user assistance
3.4.8 | [self-descriptiveness]{rows=2 cols=3}
3.4.9
```

```
\documentclass{article}
\usepackage{tabularray}
\begin{tblr}{colspec = llll, hlines, vlines}
clause & iso 25010:2023 & iso 25010:2011 & notes \\
3.4.6 & inclusivity & \SetCell[r=2]{l} accessibility & \SetCell[r=2]{l} split and renamed \\
3.4.7 & user assistance \\
3.4.8 & \SetCell[r=2,c=3]{l} self-descriptiveness \\
3.4.9 \\
\end{tblr}
```

clause	iso/iec 25010:2023	iso/iec 25010:2011	notes
3.4.6	inclusivity	accessibility	split and renamed
3.4.7	user assistance		
3.4.8	self-descriptiveness		
3.4.9			

markdown vs. latex kritéria *sazba tabulek iii*

```
``` {=latex}
\begin{tblr}{colspec = llll, hlines, vlines}
clause & iso 25010:2023 & iso 25010:2011 & notes \\
3.4.6 & inclusivity & \SetCell[r=2]{l} accessibility & \SetCell[r=2]{l} split and renamed \\
3.4.7 & user assistance \\
3.4.8 & \SetCell[r=2,c=3]{l} self-descriptiveness \\
3.4.9 \\
\end{tblr}
```

```
\documentclass{article}
\usepackage{tabularray}
\begin{tblr}{colspec = llll, hlines, vlines}
clause & iso 25010:2023 & iso 25010:2011 & notes \\
3.4.6 & inclusivity & \SetCell[r=2]{l} accessibility & \SetCell[r=2]{l} split and renamed \\
3.4.7 & user assistance \\
3.4.8 & \SetCell[r=2,c=3]{l} self-descriptiveness \\
3.4.9 \\
\end{tblr}
```

clause	iso/iec 25010:2023	iso/iec 25010:2011	notes
3.4.6	inclusivity	accessibility	split and renamed
3.4.7	user assistance		
3.4.8	self-descriptiveness		
3.4.9			
```

markdown vs. latex kritéria *analýza i*

markdown je obtížně parsovatelný (není bezkontextový, různé standardy), ale je rozhodnutelný
markdown nezná syntaktické chyby – co není prvek jazyka, je prostý text
navzdory tomu existují statické analyzátory jako **markdownlint**,[20] které detekují překlepy a chyby

překlep

```
[odkaz][]

 [odaz]: https://ossconf.fri.uniza.sk/
```

strukturní chyba

```
# nadpis první úrovně
### nadpis třetí úrovně
```

20 github.com/davidanson/markdownlint

markdown vs. latex kritéria *analýza ii*

latex je také obtížně parsovatelný, ale na rozdíl od markdownu není ani rozhodnutelný

existuje množství statických analyzátorů, které se zaměřují na rozhodnutelné části jazyka:*
chktex,[21] **lacheck**[22] obecná statická analýza latexových dokumentů
chklref[23] detekce nepoužitých identifikátorů z příkazů `\label`
match_parens[24] detekce nepárových složených závorek `{` a `}`
lezer_latex[25] formální gramatika latexových dokumentů používaná editorem overleaf

stejně tak existují dynamické analyzátory latexu, které získávají informace kompilací:
cmdtrack[26] detekce nepoužitých příkazů **nag**[27] detekce obsolescentních příkazů a balíčků

* zde je výhodou „ukecanost“ latexu, např. implic. seznamy v `op-slides.tex` by se analyzovaly hůře a plain je nerozhodnutelný
21 ctan.org/pkg/chktex 22 ctan.org/pkg/lacheck 23 ctan.org/pkg/chklref 24 ctan.org/pkg/match_parens
25 doi.org/10.47397/tb/46-2/tb143jakobsen-parsing-editing 26 ctan.org/pkg/cmdtrack 27 ctan.org/pkg/nag

markdown vs. latex kritéria *víceformátové publikování*

markdown lze staticky konverzovat do desítek různých formátů nástroji jako **pandoc**[28]
surové `{=latex}` bloky je třeba vyčlenit a převést samostatně

„pěkné“ latexové dokumenty lze rovněž staticky konvertovat pandocem,[29] ale v obecnosti je třeba dynamická konverze nástroji jako **tex4ht**,[30] **latexml**,[31] nebo překladem se zapnutou podporou pdf/ua-2[32] a extrakcí struktury do formátu xml nástrojem **show-pdf-tags**[33]

28 pandoc.org 29 youtu.be/T9uZJFO54iM 30 tug.org/tex4ht 31 math.nist.gov/~BMiller/LaTeXML
32 latex3.github.io/tagging-project 33 ctan.org/pkg/show-pdf-tags

autorský pohled i markdown vs. latex *vyhodnocení*

čitelnost zdrojového textu markdown vítězí u dokumentů bez surových `{=latex}` bloků
oddělení obsahu od formy markdown umožňuje čistší oddělení, ale je třeba víc programování
nutnost programovat s markdownem autor neprogramuje, ale celkem je třeba víc programování
sazba matematiky a tabulek jednoduchá matematika/tabulky jsou s markdownem snazší, složitější obsah vyžaduje vlastní definice nebo `{=latex}` bloky
analýza markdown vede v rozhodnutelnosti, latex vede v množství odhalitelných chyb
víceformátové publikování markdown vítězí díky rozhodnutelnosti, zvlášť bez `{=latex}` bloků

> *vlastní definice*
balíček markdown přímo podporuje uživatelské definice významu prvků (nedestruktivní úpravy)
v konverzních nástrojích typu pandoc lze také psát vlastní filtry v jazyce lua pro dosažení téhož

ukázka šablony github.com/istqborg/istqb_product_base a testů v /istqb_product_template

programátorský pohled expl3 vs. lua(tex) *kritéria*

1 obecné programování
2 definice uživatelských maker
3 interakce s texovým strojem
4 interakce s texovými formáty
5 analýza
6 zpracování textu

expl3 vs. lua(tex) kritéria *obecné programování*

expl3 i lua nabízí základní datové typy a operace nad nimi:

```
\int_set:Nn \l_tmpa_int { 1 + 2 }
\tl_set:Nn \l_tmpa_tl { foo~bar }
\clist_set:Nn \l_tmpa_clist { foo, bar }
\prop_set_from_keyval:Nn \l_tmpa_prop { foo = bar }
\regex_if_match:nVT { f.* } \l_tmpa_tl {
  \int_compare:nNnT { \l_tmpa_int } = { 3 } {
    \clist_map_inline:Nn \l_tmpa_clist {
      \tl_show:n { #1 }
    }
  }
}
```

```
local a = 1 + 2
local b = "foo bar"
local c = {"foo", "bar"}
local d = {["foo"] = "bar"}
if b:find("f.*") and a == 3
  then
  for _, text in ipairs(c) do
    print(text)
  end
end
```

lua je ale rychlejší, čitelnější a poskytuje např. obecnější rekurzivní hašové tabulky

expl3 vs. lua(tex) kritéria *definice uživatelských maker*

možnost definice texových maker v jazyce lua je omezená:

```
\tl_set:Nn
  \l_tmpa_tl { text }
\cs_new:Nn
  \priklad_jedna:nn
  { foo~#1~#2 }
```

```
token.set_macro(
  "l_tmpa_tl", "text")
tex.print([[\cs_new:Nn
            \priklad_jedna:nn
            { foo~#1~#2 }]])
```

vhodnější je často definovat makro v texu/explu3 a jazyk lua použít pro implementaci:

```
\cs_new:Nn
  \priklad_dva:n
  { \lua_now:e
      { luova_funkce("\lua_escape:e { #1 }") } }
\priklad_dva:n { bar }
```

```
function luova_funkce(text)
  tex.print("foo~" .. text)
end
```

ale co kategorie tokenů v #1, umíme je zachovat? spíš neumíme

expl3 vs. lua(tex) kritéria *interakce s texovým strojem*

expl3 dokáže interagovat s texovým strojem do stejné míry jako (plain) tex

luatex nabízí mnohem širší možnosti interakce:
- zpětná volání během:
    - čtení souborů,
    - odstavcového zlomu,
    - dělení slov atd.
- manipulace se seznamy uzlů:
    - lepidla,
    - kerny,
    - penalty,
    - boxy atd.

expl3 vs. lua(tex) kritéria *interakce s tex. formáty*

expl3 dokáže interagovat s formátem latex3; ostatní formáty se volají přes standardní makra

luatex zahrnuje speciální moduly pro formáty:
- latex2e (`luatexbase`)
- latex3 (`ltx.utils`)
- context (`context`) – široké možnosti interakce
- optex (`optex`)

díky práci josepha wrighta ale expl3 funguje napříč formáty, vč. contextu lmtx (luametatex)

expl3 vs. lua(tex) kritéria *analýza*

podobně jako latex je expl3 obtížně parsovatelný a nerozhodnutelný
syntaktickou analýzu rozhodnutelných částí jazyka umožňuje nástroj **explcheck** z bal. **expltools**[8]

jazyk lua je snadno parsovatelný a rozhodnutelný
pro statickou analýzu poslouží nástroje **luacheck**[26] a **lua-language-server**.[27]

8 ctan.org/pkg/expltools 26 github.com/mpeterv/luacheck 27 github.com/LuaLS/lua-language-server

expl3 vs. lua(tex) kritéria *zpracování textu*

```
\regex_extract_all:nnN
  { \d+ }
  { 10 + 30 + 2 }
  \l_tmpa_seq
\int_zero:N
  \l_tmpa_int
\seq_map_inline:Nn
  \l_tmpa_seq
  { \int_add:Nn
      \l_tmpa_int
      { #1 } }
\int_use:N
  \l_tmpa_int
```

```
local text = "10 + 30 + 2"

local sum = 0
for n in text:gmatch("%d+") do
    sum = sum + tonumber(n)
end
print(sum)
```

lua kromě (okleštěných) reg. výrazů podporuje i pe gramatiky:

```
local num = lpeg.R("09")^1 / tonumber
local function add(acc, val) return acc + val end
local sum = lpeg.Cf(num * (" + " * num)^0, add)
print(sum:match(text))
```

dokážou regulární výrazy zpracovávat složitější jazyky, např. aritm. výrazy s `(`, `)` a `-`? spíš ne

programátorský pohled expl3 vs. lua(tex) *vyhodn.*

obecné programování lua je rychlejší a nabízí složitější struktury jako rekurzivní haš. tabulky
definice uživatelských maker předávání dat mezi texem a luou je ztrátové (tokeny -> text)
interakce s texovým strojem luatex nabízí mnohem širší možnosti interakce
interakce s texovými formáty luatex nabízí širší interakce u některých formátů (context)
analýza jazyk lua je rozhodnutelný (byť i expl3 má dnes ~25 % tex live staticky analyzovatelných)
zpracování textu expl3 podporuje regulární výrazy, lua podporuje pe gramatiky

doplňující kritérium: potřebujeme podporovat starší stroje (pdftex, xetex)?

autorský pohled ii lualatex vs xe/pdflatex *kritéria*

1. přístupnost pdf výstupu
2. podpora uživatelských balíčků
3. rychlost sazby
4. podpora a údržba

lualatex vs xe/pdflatex kritéria *přístupnost pdf výstupu*

lualatex při aktivaci `\DocumentMetadata{tagging=on}` taguje matematiku podle pdf/ua-2
xe/pdflatex tagují matematiku podle pouze pdf/ua-1 (nedokážou vyrobit mathml reprezentaci)

lualatex vs xe/pdflatex kritéria *podpora uživ. balíčků*

lualatex nad rámec standardních balíčků podporuje např. následující zajímavé:
linebreaker[29] automatická eliminace přetékajících řádků
lua-widow-control[30] automatická eliminace vdov a sirotků pomocí roztahování mezer
luavlna[31] automatická eliminace jednopísm. slov na konci řádku v č.s. sazbě (alt.: **xevlna**[32])
newpax[33] automatické zachování anotací (např. hypertext) u vkládaných pdf obrázků

29 ctan.org/pkg/linebreaker 30 ctan.org/pkg/lua-widow-control 31 ctan.org/pkg/luavlna 32 ctan.org/pkg/xevlna
33 ctan.org/pkg/newpax

lualatex vs xe/pdflatex kritéria *rychlost sazby*

lualatex je na závěrečných pracech fi muni ~3× pomalejší než pdflatex[34]
reddit[35] tvrdí, že lualatex je 3–5× pomalejší (při sazbě cyrilice)

github latexu[36] ukazuje, že lualatex je na vybraných dokumentech 4–8× pomalejší
předběžný závěr je, že jde o problémy s načítáním a vykreslováním opentype písem

existuje rozpracovaný projekt,[37] který řádově akceleruje datové struktury expl3 skrz jazyk lua

34 is.muni.cz/th/nghf5/thesis.pdf (tabulka 5.2) 35 reddit.com/r/LaTeX/comments/18g3elh/comment/kcysj5b
36 github.com/latex3/latex2e/issues/1877 37 github.com/latex3/latex3/pull/1557

lualatex vs xe/pdflatex kritéria *podpora a údržba*

pdftex je aktivně udržovaný, ale neočekává se budoucí vývoj a podpora balíčků klesá
xetex není aktivně udržovaný a má neopravené chyby[38]
luatex je aktivně udržovaný/vyvíjený a podpora balíčků stoupá

38 tex.stackexchange.com/a/593217/70941

autorský pohled ii lualatex vs xe/pdflatex *vyhod.*

přístupnost pdf výstupu pouze lualatex produkuje pdf/ua-2 výstup
podpora uživatelských balíčků výhradně lualatex podporuje rostoucí množství balíčků/šablon
rychlost sazby lualatex je na některých dokumentech řádově pomalejší
podpora a údržba pouze luatex je aktivně vyvíjený, pdftex je alespoň aktivně udržovaný

doplňující kritérium: potřebujeme podporovat vícepísmovou sazbu a opentype písma? (lua/xe)

závěr

je markdown lepší než latex? hlavně u prostých dokumentů, jinak mnohdy vyžaduje více práce

jak lze uplatnit princip nedestruktivních úprav při sazbě dokumentů v jazyce markdown? konkrétní vzhled se doplní během překladu/konverze bez nutnosti zásahů do zdrojového textu

je lepší programovat stroj texu skrz jazyk lua, nebo skrz expl3? pro obecné programování, složité zpracování textu a asynchronní zásahy do procesu sazby spíš lua. pro makro-programování a kód, který musí fungovat i mimo luatex spíš tex/expl3

jsou pegy lepší než regulární výrazy? pegy jsou silnější formalismus pro zpracování textu

je luatex lepší než xetex a pdftex? funkčně ano, ale ve spojení s latexem dnes řádově pomalejší

děkuji za pozornost

o novinkách v balíčcích markdown a expltools budu přednášet 18. 7. na tugu 2026 v kanadě bude se streamovat na youtube