Informatika pro moderní fyziky (4) komplexnější projekt, generování dokumentů

František HAVLŮJ

e-mail: haf@ujv.cz

ÚJV Řež
oddělení Reaktorové fyziky a podpory palivového cyklu

akademický rok 2013/2014
5. listopadu 2013

Obsah

- čtení ze souboru, automatické vyhledání výrazu
- jak dostat keff z výstupu MCNP
- výroba vstupních souborů pomocí šablon

Obsah

Spousta skriptů, spousta zmatku

- mám jeden projekt/práci a potřebuju udělat víc věcí
- zatím jsme měli jeden skript na jednu věc
- což skončí hromadou .rb souborů, kde nebudu vědět co dělá který a budu v tom mít trochu zmatek
- nehledě na to, že bych mohl chtít sdílet nějakou konfiguraci (jména souborů atd.)

Nástroj Rake

- alternativa k unixovému MAKE, ale v Ruby (Ruby MAKE = Rake)
- nejjednodušší – nastrkám si do jednoho Rakefile víc úloh (`task`) a ty pak snadno spustím
- složitější – můžu specifikovat závislosti

Rakefile - příklad

obsah Rakefile

```
desc 'rearrange keff into a nice table'
task :rearrange do
...
end

desc ''
task :find do
...
end
```

spuštění

```
rake find
rake -T
```

Obsah

Co dál

- máme soubor s daty pro polohy tyčí (`keff.csv`)
- vykreslit graf! pro každou z 11 poloh R1 jedna čára (závislost keff na R2)
- (= csv soubor, gnuplot, znáte to)
- najít automaticky kritickou polohu R2 pro každou z 11 poloh R1
- a zase graf... (kritická poloha R2 v závislosti na R1)

Spouštění programů z Ruby

Je otrava psát pořád cestu ke gnuplotu a vůbec, takže lze samozřejmě vyrobit rake task:

```
task :plot do
  system("\"C:/Program Files/gnuplot/bin/wgnuplot.exe\" plot1.p")
end
```

Gnuplot - připomenutí

```
set terminal png
set output "r1_50.png"
plot "r1_50.csv" with linespoints
```

V souboru `r1_50.csv` mám ve dvou sloupcích R2 a keff pro R1=50 cm.

Definice funkce

```
def load
  keff = {}
  IO.foreach("keff.csv") do |line|
    ary = line.strip.split
    r1, r2, k = ary[0].to_i, ary[1].to_i, ary[2].to_f
    keff[[r1,r2]] = k
  end
  keff
end

task :rearrange do
  keff = load
  ...
end

task :plot do
  keff = load
  ...
end
```

Ruční hledání

- v takových datech se hrozně špatně hledá
- pokud bych měl na 11 řádcích (pro každou polohu R1) 11 hodnot (pro každou polohu R2), tak už “kouknu a vidím”
- na to se ale snadno napíše skript ...

Hezká tabulka

```
keff = {}
IO.foreach("keff.csv") do |line|
  ary = line.strip.split
  r1, r2, k = ary[0].to_i, ary[1].to_i, ary[2].to_f
  keff[[r1,r2]] = k
end

(0..10).each do |i|
  (0..10).each do |j|
    print "%8.5f" % keff[[i*10, j*10]]
  end
  puts
end
```

Hezká tabulka

0.94800	0.94850	0.95000	0.95250	0.95600	0.96050	0.96600	0.97250	0.98000	0.98850	0.99800
0.94900	0.94950	0.95100	0.95350	0.95700	0.96150	0.96700	0.97350	0.98100	0.98950	0.99900
0.95000	0.95050	0.95200	0.95450	0.95800	0.96250	0.96800	0.97450	0.98200	0.99050	1.00000
0.95100	0.95150	0.95300	0.95550	0.95900	0.96350	0.96900	0.97550	0.98300	0.99150	1.00100
0.95200	0.95250	0.95400	0.95650	0.96000	0.96450	0.97000	0.97650	0.98400	0.99250	1.00200
0.95300	0.95350	0.95500	0.95750	0.96100	0.96550	0.97100	0.97750	0.98500	0.99350	1.00300
0.95400	0.95450	0.95600	0.95850	0.96200	0.96650	0.97200	0.97850	0.98600	0.99450	1.00400
0.95500	0.95550	0.95700	0.95950	0.96300	0.96750	0.97300	0.97950	0.98700	0.99550	1.00500
0.95600	0.95650	0.95800	0.96050	0.96400	0.96850	0.97400	0.98050	0.98800	0.99650	1.00600
0.95700	0.95750	0.95900	0.96150	0.96500	0.96950	0.97500	0.98150	0.98900	0.99750	1.00700
0.95800	0.95850	0.96000	0.96250	0.96600	0.97050	0.97600	0.98250	0.99000	0.99850	1.00800

Najít kritickou polohu automaticky

- pro každou polohu R1 snadno najdu odpovídající R2
- jak? najdu nejdřív interval, ve kterém leží keff=1
- a potom už je to jen obyčejná lineární interpolace
- (nevzdávat, vymyslet!)
- a nakonec zase nakreslit graf

Obsah

Jak vyrobit zprávu?

- potřebujeme udělat hezké PDF shrnující výsledky našich výpočtů
- takže úvod, popis toho co jsme dělali a pak přehled výsledků
- tabulka s hodnotami, 11+1 graf
- co by znamenalo to dělat ve Wordu ..... ?
- hodilo by se to zautomatizovat!

Jak vygenerovat text?

- zase potřebujeme lepší nástroj na text, než jsou WYSIWYG (What You See Is What You Get) editory
- ideálně něco, co bude mít plain-text vstup (který můžeme s úspěchem generovat v Ruby) a co se pak
- odpověď zní LaTeX (LaTeX)
- nejvíc nejlepší text-processor na světě

Koncepce oddělení obsahu a formy

- když píšu, nechci říct, že je text tučně a o dva body větší, ale že je to nadpis kapitoly
- ideálně chci popsat někde, jak bude dokument
- styly ve Wordu se tomu vzdáleně blíží
- v LaTeXu vlastně píšu jen obsah a o formu se musím starat jen hodně málo
- je samozřejmostí zadarmo obsah, rejstřík atd.
- kdo píše diplomku v něčem jiném, kazí si život

Příklad jednoduchého dokumentu

- viz (document.tex)
- není potřeba úplně všemu rozumět, zatím to jen budeme upravovat v mezích zákona
- všechny příkazy začínají zpětným lomítkem, parametry jsou ve složených závorkách
- napíšu `pdflatex document.tex` a dostanu pdfko!

Text

- konce řádku nejsou důležité
- nový odstavec se dělá prázdným řádkem
- nové (pod)kapitoly pomocí příkazů `section`, `subsection`

Tabulka

- prostředí (= `begin` ... `end`)
- sloupce se

```
\begin{tabular}
a & b \\
c & d \\
\end{tabular}
```

Úkol

- vyrobit PDF s výsledky
- 11 grafů keff
- 1 graf závislosti kritické polohy R2 na poloze R1
- tabulka keff
- tabulka kritických poloh

Pozor na backslash

- v Ruby se v řetězci backslash `\` používá jako escape character
- např konec řádky je `\n`
- pokud chci vytiskout zpětné lomítko (což bude asi pro LaTeX potřeba), musím ho zdvojit: `\\`

Složitější práce

- LaTeX je ideální pro rozsáhlé texty
- výzkumák, diplomka, disertace se naformátuje sama a všechno funguje bez trápení a snažení
- odkazy, reference, citace .. všechno bez starostí

Prezentace

- balíček `beamer`
- bez nutnosti se uklikat to samo od sebe vypadá slušně
- viz tahle prezentace
- nevýhoda(?): obtížnost přizpůsobit rozmístění textu apod., ale na druhou stranu to aspoň drží jednotný styl

A to je vše, přátelé!