Informatika pro moderní fyziky (4) vstupní a výstupní soubory pro výpočetní programy

František HAVLŮJ

e-mail: haf@ujv.cz

ÚJV Řež
oddělení Reaktorové fyziky a podpory palivového cyklu

akademický rok 2019/2020
16. října 2019

Obsah

Ladění programů

- v každém programu je aspoň jedna chyba
- není důležité nedělat chyby, ale je nutné je umět najít
- když si program/Ruby na něco stěžuje, tak si to přečtěte, jinak se nic nedozvíte
- pokud nepoznám, v čem je chyba, jsem bezezbytku ztracen
- následují tři úlohy, kde je úkolem najít všechny chyby
- z didaktických důvodů postupujte metodou tupého spouštění a postupného opravování

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 "find something somewhere"
task :find do
...
end
```

spuštění

```
rake find
rake -T
```

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/gnuplot.exe\" plot1.gp")
end
```

Obsah

Jak na to

Máme všechno, co potřebujeme:

- načtení keff z jednoho výstupního souboru (`File.foreach`, `include` a `split`)
- procházení adresáře (`Dir.each`)
- zápis do souboru (`File.open` s parametrem `w`)

Takže už to stačí jen vhodným způsobem spojit dohromady!

Realizace

```
Dir["*o"].each do |filename|
  keff = nil

  File.foreach(filename) do |line|
    if line.include?("final estimated combined")
      a = line.split("=")
      b = a[1].split
      keff = b[0]
    end
  end

  puts "#{filename} #{keff}"
end
```

Výstup

Výsledkem je perfektní tabulka:

```
outputs/c_0_0o 0.94800
outputs/c_0_10o 0.99800
outputs/c_0_1o 0.94850
outputs/c_0_2o 0.95000
outputs/c_0_3o 0.95250
outputs/c_0_4o 0.95600
...
```

Hloupé je, že nikde nemáme tu polohu tyčí.

Víc by se nám hodilo

něco jak toto:

```
0   0   0.94800
0 640   0.99800
0  64   0.94850
0 128   0.95000
0 192   0.95250
0 256   0.95600
...
```

(rozsah je 0 – 640, my máme kroky 0 – 10)

A protože přehlednost je nade vše

něco jak toto:

keff	0	64	128	...
0	0.94800	0.94900	0.95000	...
64	0.94850	0.94950	0.95050	...
128	0.95000	0.95100	0.95200	...
192	0.95250	0.95350	0.95450	...
256	0.95600	0.95700	0.95800	...
320	0.96050	0.96150	0.96250	...
384	0.96600	0.96700	0.96800	...
448	0.97250	0.97350	0.97450	...
512	0.98000	0.98100	0.98200	...
576	0.98850	0.98950	0.99050	...
640	0.99800	0.99900	1.00000	...
...				

– alternativně můžete vyrobit tabulku v excelu!

Navážeme na úspěchy z minulých týdnů

- 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)

Obsah

HELIOS

Tabulka výstupů:

```
List name         : list
List Title(s)  1) This is a table
               2) of some data
               3) in many columns
               4) and has a long title!
```

	bup	kinf	ab	ab	u235	
0001	0.00E+00	1.16949	9.7053E-03	7.6469E-02	1.8806E-04	7.
0002	0.00E+00	1.13213	9.7478E-03	7.9058E-02	1.8806E-04	7.
0003	1.00E+01	1.13149	9.7488E-03	7.9070E-02	1.8797E-04	7.
0004	5.00E+01	1.13004	9.7521E-03	7.9093E-02	1.8760E-04	7.
0005	1.00E+02	1.12826	9.7559E-03	7.9218E-02	1.8714E-04	7.
0006	1.50E+02	1.12664	9.7594E-03	7.9407E-02	1.8668E-04	7.
0007	2.50E+02	1.12399	9.7657E-03	7.9869E-02	1.8577E-04	7.
0008	5.00E+02	1.12007	9.7812E-03	8.1065E-02	1.8351E-04	7.
0009	1.00E+03	1.11561	9.8203E-03	8.3169E-02	1.7914E-04	7.
0010	2.00E+03	1.10542	9.9329E-03	8.6731E-02	1.7088E-04	7.
0011	3.00E+03	1.09354	1.0067E-02	8.9717E-02	1.6316E-04	7.
0012	4.00E+03	1.08126	1.0207E-02	9.2299E-02	1.5591E-04	7.

Co bychom chtěli

- mít načtené jednotlivé tabulky (zatím jen jednu, ale bude jich víc)
- asi po jednotlivých sloupcích, sloupec = pole (hodnot po řádcích)
- sloupce se jmenují, tedy použijeme `Hash`
- `table['kinf']`
- pozor na `ab`, asi budeme muset vyrobit něco jako `ab1`, `ab2` (ale to až za chvíli)

Nástrahy, chytáky a podobně

- tabulka skládající se z více bloků
- více tabulek
- tabulky mají jméno `list name` a popisek `list title(s)`

Jak uspořádat data?

- pole s tabulkami + pole s názvy + pole s titulky?

Jak uspořádat data?

- pole s tabulkami + pole s názvy + pole s titulky?
- co hashe tabulky[název] a titulky[název]?

Jak uspořádat data?

- pole s tabulkami + pole s názvy + pole s titulky?
- co hashe tabulky[název] a titulky[název]?
- nejchytřeji: `{ä=̈> {:title => T̈able title, data => {k̈inf=̈> ...}}}`

Jak uspořádat data?

- pole s tabulkami + pole s názvy + pole s titulky?
- co hashe tabulky[název] a titulky[název]?
- nejchytřeji: `{ä⁼⁼> {:title => T̈able title¸ data => {k̈inf⁼⁼> ...}}}`
- n̈ovášyntaxe: `{ä⁼⁼> {title: T̈able title¸ data: {k̈inf⁼⁼> ...}}}`

Z příkazové řádky

- a co takhle z toho udělat skript, který lze pustit s argumentem = univerzální
- `ruby read_helios.rb helios1.out`
- vypíše seznam všech tabulek, seznam jejich sloupců, počet řádků
- pole `ARGV`
- vylepšení – provede pro všechny zadané soubory: `ruby read_helios.rb helios1.out helios2.out`

Obsah

Určení poloh tyčí

Ve vstupním souboru si najdeme relevantní část:

```
c --------------------------------
c polohy tyci (z-plochy)
c --------------------------------
c
67 pz 47.6000    $ dolni hranice absoberu r1
68 pz 40.4980    $ dolni hranice hlavice r1
69 pz 44.8000    $ dolni hranice absoberu r2
70 pz 37.6980    $ dolni hranice hlavice r2
```

Určení poloh tyčí

příklad `c1_10_20`:

```
c -------------------------------
c polohy tyci (z-plochy)
c -------------------------------
c
67 pz 47.6000    $ dolni hranice absoberu r1
68 pz 40.4980    $ dolni hranice hlavice r1
69 pz 44.8000    $ dolni hranice absoberu r2
70 pz 37.6980    $ dolni hranice hlavice r2
```

Výroba šablon

Jak dostat polohy tyčí do vstupního souboru? Vyrobíme šablonu, tzn nahradíme

```
67 pz 47.6000     $ dolni hranice absoberu r1
```

Výroba šablon

Jak dostat polohy tyčí do vstupního souboru? Vyrobíme šablonu, tzn nahradíme

```
67 pz 47.6000     $ dolni hranice absoberu r1
```

nějakou značkou (*placeholder*):

```
67 pz %r1%     $ dolni hranice absoberu r1
```

A co takhle trocha zobecnění?

- když budu chtít přidat další tyče nebo jiné parametry, bude to děsně bobtnat
- funkce `process("template", "inputs/c_#{i1}_#{i2}", {'r1' => r1, 'r2' => r2, .....})`
- všechno víme, známe, umíme

A to je vše, přátelé!