

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 2014/2015

15. října 2014

- 1 Co jsme se naučili minule
- 2 Hrabání listí dělá pořádek (Rake)
- 3 Zpracování textu
 - Obecný rozbor
 - Načítání výstupního souboru
- 4 Automatizace tvorby vstupů
 - Zápis všech výsledků do tabulky
 - Co dál?

Obsah

- 1 Co jsme se naučili minule
- 2 Hrabání listí dělá pořádek (Rake)
- 3 Zpracování textu
- 4 Automatizace tvorby vstupů

- procvičili jsme si kreslení grafů gnuplotem a vstup/výstup v jazyce Ruby
- naučili jsme se zpracovat větší počet souborů
- generování hromady grafů ani procházení ještě větší hromady tabulek nás nezaskočí

Obsah

- 1 Co jsme se naučili minule
- 2 Hrabání listí dělá pořádek (Rake)**
- 3 Zpracování textu
- 4 Automatizace tvorby vstupů

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

- 1 Co jsme se naučili minule
- 2 Hrabání listů dělá pořádek (Rake)
- 3 Zpracování textu**
 - Obecný rozbor
 - Načítání výstupního souboru
- 4 Automatizace tvorby vstupů

Problém č. 3: mnoho výpočtů, inženýrova smrt

Zadání

Při přípravě základního kritického experimentu je pomocí MCNP potřeba najít kritickou polohu regulační tyče R2. Jak se tato poloha změní při změně polohy tyče R1?

Co máme k dispozici?

MCNP

Pokud připravíme vstupní soubor (v netriviální formě obsahující polohy regulačních tyčí R1 a R2), spočítá nám keff.

Potřebovali bychom ale něco na:

- 1 vytvoření velkého množství vstupních souborů
- 2 extrakci keff z výstupních souborů
- 3 popřípadě na vyhodnocení získaných poloh tyčí a keff

Pracovní postup

- 1 načíst keff z výstupního souboru MCNP

Pracovní postup

- 1 načíst keff z výstupního souboru MCNP
- 2 vygenerovat potřebné vstupní soubory

Pracovní postup

- 1 načíst keff z výstupního souboru MCNP
- 2 vygenerovat potřebné vstupní soubory
- 3 vyrobit BAT soubor na spuštění výpočtů

Pracovní postup

- 1 načíst keff z výstupního souboru MCNP
- 2 vygenerovat potřebné vstupní soubory
- 3 vyrobit BAT soubor na spuštění výpočtů
- 4 načíst výsledky ze všech výstupních souborů do jedné tabulky

Pracovní postup

- 1 načíst keff z výstupního souboru MCNP
- 2 vygenerovat potřebné vstupní soubory
- 3 vyrobit BAT soubor na spuštění výpočtů
- 4 načíst výsledky ze všech výstupních souborů do jedné tabulky
- 5 buď zpracovat ručně (Excel), nebo být Myšpulín a vyrobit skript (úkol s hvězdičkou)

Algoritmus

- 1 najít řádek s keff

Algoritmus

- 1 najít řádek s keff
- 2 vytáhnout z něj keff, takže například:

Algoritmus

- 1 najít řádek s keff
- 2 vytáhnout z něj keff, takže například:
- 3 rozdělit podle rovnítka

Algoritmus

- 1 najít řádek s keff
- 2 vytáhnout z něj keff, takže například:
- 3 rozdělit podle rovnítka
- 4 druhou část rozdělit podle mezer

Algoritmus

- 1 najít řádek s keff
- 2 vytáhnout z něj keff, takže například:
- 3 rozdělit podle rovnítka
- 4 druhou část rozdělit podle mezer
- 5 vzít první prvek

Realizace (1/5)

```
keff = nil

IO.foreach("cl_lo") do |line|

end

puts keff
```


Realizace (2/5)

```
keff = nil

IO.foreach("cl_1o") do |line|
  if line.include?("final estimated combined")

    end
end

puts keff
```

Realizace (3/5)

```
keff = nil

IO.foreach("c1_1o") do |line|
  if line.include?("final estimated combined")
    a = line.split("=")

    end
  end

puts keff
```

Realizace (4/5)

```
keff = nil

IO.foreach("cl_1o") do |line|
  if line.include?("final estimated combined")
    a = line.split("=")
    b = a[1].split

    end
  end

puts keff
```

Realizace (5/5)

```
keff = nil

IO.foreach("c1_lo") do |line|
  if line.include?("final estimated combined")
    a = line.split("=")
    b = a[1].split
    keff = b[0]
  end
end

puts keff
```

Obsah

- 1 Co jsme se naučili minule
- 2 Hrabání listí dělá pořádek (Rake)
- 3 Zpracování textu
- 4 Automatizace tvorby vstupů**
 - Zápis všech výsledků do tabulky
 - Co dál?

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
```

Výroba šablon

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

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

Výroba šablon

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

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

nějakou značkou (*placeholder*):

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


Chytáky a zádrhele

- kromě samotné plochy konce absorbéru je nutno správně umístit i z-plochu konce hlavice o 7,102 cm níže
- obecně je na místě ohlídat si, že placeholder nebude kolidovat s ničím jiným

Doporučené nástroje jsou:

- již známá funkce `sub` pro nahrazení jednoho řetězce jiným
- pro pragmatické lenochy funkce `IO.read` načítající celý soubor do řetězce (na což nelze v Pascalu ani pomyslet)
- možno ovšem použít i `IO.readlines` (v čem je to lepší?)

Realizace

```
DELTA = 44.8000 - 37.6980
```

```
template = IO.read('template')
(0..10).each do |i1|
  (0..10).each do |i2|
    r1 = i1 * 50
    r2 = i2 * 50
    File.open("inputs/c_#{i1}_#{i2}", "w") do |f|
      s = template.sub("%r1%", r1.to_s)
      s = s.sub("%r1_%", (r1 - DELTA).to_s)
      s = s.sub("%r2%", r2.to_s)
      s = s.sub("%r2_%", (r2 - DELTA).to_s)
      f.puts template
    end
  end
end
end
```

Jak na to

Máme všechno, co potřebujeme:

- načtení keff z jednoho výstupního souboru (`IO.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  
  
  IO.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.

Chytrá horákyně

... by jistě vyrobila toto:

```
0 0 0.94800
0 10 0.99800
0 1 0.94850
0 2 0.95000
0 3 0.95250
0 4 0.95600
...
```

Nápovědou je funkce `split` (podle podtržítka) a funkce `to_i` (co asi dělá?)

Realizace chytré horákyne

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

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

  s = filename.split("_")
  puts "#{s[1].to_i} #{s[2].to_i} #{keff}"
end
```

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)

Co jsme se naučili minule
Hrabání listů dělá pořádek (Rake)
Zpracování textu
Automatizace tvorby vstupů

Zápis všech výsledků do tabulky
Co dál?

A to je vše, přátelé!

