C2110 *Operační systém UNIX a základy programování*

12. lekce

Petr Kulhánek, Jakub Štěpán

kulhanek@chemi.muni.cz

Národní centrum pro výzkum biomolekul, Přírodovědecká fakulta
Masarykova univerzita, Kotlářská 2, CZ-61137 Brno

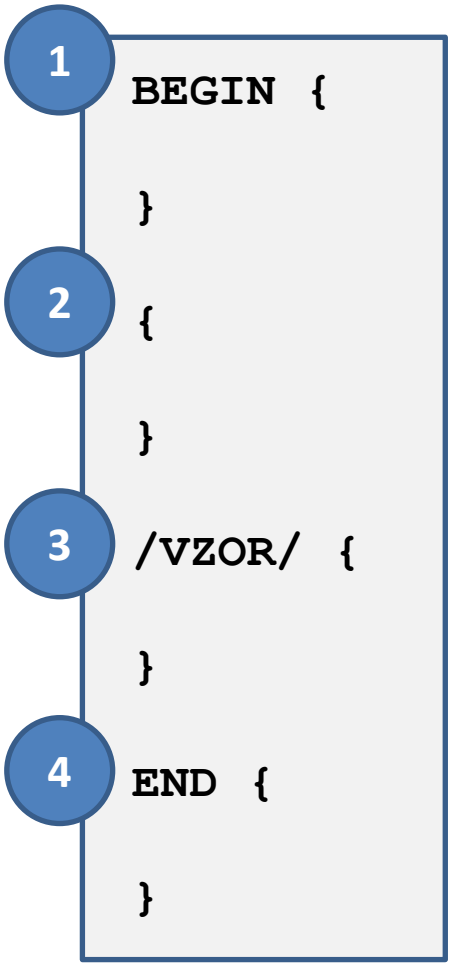
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

CZ.1.07/2.2.00/15.0233

Obsah

- **AWK**
  - Analýza textových souborů
  - Regulární výrazy
  - Pole
- **Kontrola vstupu v BASHi**
  - Ošetření chybových stavů v BASHi
  - Kontrola načtených hodnot

Průběh vykonávání skriptu



- Blok BEGIN (1) se vykoná (pokud je ve skriptu obsažen) před analýzou souboru.
  - Načte se záznam ze souboru. Ve výchozím nastavení je záznamem celý řádek analyzovaného souboru nebo proudu. Záznam se rozdělí na pole. Ve výchozím nastavení jsou pole jednotlivá slova v záznamu.
  - Pro daný záznam se vykoná blok (2).
  - Pokud záznamu vyhovuje VZOR, vykoná se blok (3).
  - .... vykonají se případně další bloky ....
- Blok END (4) se vykoná (pokud je ve skriptu obsažen) po analýze celého souboru.

Blok je uzavřen do složených závorek {}.
Výše uvedené programové bloky jsou volitelné.
Ve výchozím nastavení je záznamem řádek souboru.

Regulární výrazy

```
/VZOR/  {

}
```

Pokud záznamu vyhovuje VZOR, tak se blok vykoná.

Vzor je **regulární výraz**.

Regulární výraz je jazyk, který popisuje strukturu textového řetězce. Jazyk se využívá k vyhledávání textových řetězců, k nahrazování části řetězců.

Příklady jednoduchých regulárních výrazů:

TEXT - je splněno, pokud je v daném záznamu obsažen TEXT (může být kdekoliv)
^TEXT - je splněno, pokud je v daném záznamu obsažen TEXT na začátku
TEXT$ - je splněno, pokud je v daném záznamu obsažen TEXT na konci

Cvičení

1. Ze souboru **rst.out** vyextrahujte průběh teploty na čase. Průběh zobrazte v **gnuplotu**.

```
 NSTEP =     1000   TIME(PS) =       1.000  TEMP(K) =   305.69  PRESS =     0.0
  Etot   =       907.8481  EKtot   =       160.3711  EPtot      =       747.4770
  BOND   =        40.6154  ANGLE   =       273.9238  DIHED      =       164.5827
  1-4 NB =        14.6900  1-4 EEL =       973.2602  VDWAALS    =       -67.6091
  EELEC  =      -488.9232  EGB     =      -163.0629  RESTRAINT  =         0.3793
  EAMBER (non-restraint)  =       747.0977
```

2. Ze souboru **rst.out** vyextrahujte průběh celkové energie (**Etot**), kinetické energie (**EKtot**) a potenciální energie (**EPtot**) na čase. Průběh jednotlivých energií zobrazte v **gnuplotu**. Ověřte, že součet potenciální a kinetické energie se rovná celkové energii.

Pole

AWK používá asociativní pole. Pole má název, k prvkům pole se přistupuje pomocí klíče. Klíč může mít libovolnou hodnotu a typ. Klíčem může být hodnota proměnné.

Přiřazení hodnoty:

```
moje_pole[klic] = hodnota;
```

Získání hodnoty:

```
hodnota = moje_pole[klic];
```

Příklady:

```
i = 5;
moje_pole[i] = 15;
print moje_pole[i];

a = "slovo";
moje_pole[a] = "hodnota";
print moje_pole["slovo"], moje_pole[5];
```

Pole, ...

Procházení seznamu klíčů:

```
for( promenna in pole) {
      print pole[promenna];
      ...
}
```

Vykoná tělo cyklu pro každý klíč, který byl použit pro uložení hodnoty do **pole**. Hodnota klíče je uložena do **proměnné** .

Mazání záznamů s klíčem:

```
delete pole[klic];
```

Cvičení

1. Ze souboru **rst.out** vyextrahujte **průběh teploty v čase**. Výsledný soubor nebude obsahovat dvě poslední hodnoty, které jsou průměrnou hodnotou a její fluktuací. **Průběh zobrazte** v gnuplotu.

2. Ze souboru **rst.out** vyextrahujte **průběh teploty** a spočítejte její **průměrnou hodnotu**. Vypočtenou hodnotu srovnejte s průměrnou hodnotou uvedenou v souboru rst.out. **Proč se hodnoty liší?**

Kontrola vstupu v BASHi

- Ošetření chybových stavů v BASHi
- Kontrola načtených hodnot

Chybný vstup v BASHi

Příklad z lekce 8. Skript vypíše chybu a **nechová se korektně** pokud je spuštěn **bez argumentu** nebo **s nečíselným argumentem**.

```
#!/bin/bash
if test "$1" -le 0; then
      echo "Cislo neni vetsi nez nula!"
      exit 1
fi
echo "Cislo je vetsi nez nula."
exit 0
```

```
$ ./muj_skript
muj_skript: line 2: test: -le: unary operator expected
Cislo je vetsi nez nula.
$ echo $?
0
```

```
$ ./muj_skript f
muj_skript: line 2: test: f: integer expression expected
Cislo je vetsi nez nula."
$ echo $?
0
```

Kontrola načtených hodnot

Hodnoty proměných načtených od uživatele je **nutno kontrolovat**.

Kontola

- Počet a typu argumentů
- Platnost číselných hodnot (dělení nulou, záporné množství prvků)
- Nenulová délka řetězců
- Přítomnost souborů před zpracováním

```
#!/bin/bash
echo "Zadej ciselnou hodnotu!"
read A
if ! expr $A + 0 &> /dev/null; then
            echo "Chyba! Nebylo nacteno cislo: $A"
            exit 1
fi
```

Cvičení

1. Změňte skript z domácího úkolu lekce 8 – úkol 1 (vykreslení čtverce) tak, že **ošetříte situaci**, kdy uživatel nezadá právě **dva** argumenty.
2. Do předchozího skriptu doplňte ošetření situace, kdy uživatel nezadá **rozměry přirozenými čísly**.
3. Doplňte předchozí skript tak, aby ve **třetím argumentu** byl uživatelem zadán znak nebo řetězec, který bude použit pro tisk obrazce (místo znaku "X"). Ošetřete situaci, kdy třetí argument bude prázdný řetězec.
4. Upravte předchozí skript tak, aby uživatel mohl zadávat hodnoty interaktivně na vyzvání **po spuštění skriptu**.
5. Změňte skript z domácího úkolu II lekce 9 tak, že skript bude přijímat **jméno analyzovaného souboru jako argument** a bude testovat **zda soubor existuje**.