C2110 *Operační systém UNIX a základy programování*

8. lekce

bash – dokončení (skoro)

Petr Kulhánek, Jakub Štěpán

kulhanek@chemi.muni.cz

Národní centrum pro výzkum biomolekul, Přírodovědecká fakulta
Masarykova univerzita, Kamenice 5, CZ-62500 Brno

Domácí úkoly

Pokyny:

1. Uvedené úkoly jsou pro **pokročilé studenty.**
2. **Cílem úkolů je rozvinout vaši schopnost řešit problémy, který jsou zdánlivě neřešitelné z pohledu možností a zdrojů, které máte k dispozici**. V případě jazyka bash se jedná převážně o možnost pracovat pouze s celočíselnou aritmetikou a omezený způsob vykreslování na terminál.

Zadání:

1. Vykreslete kruh z písmen "X". Poloměr kruhu zadá uživatel po spuštění skriptu.
2. Vykreslete kružnici z písmen "X". Poloměr kružnice zadá uživatel jako první argument skriptu.

Obsah

- **Příkaz test**
  - **porovnávání celých čísel a řetězců**
- **Cyklus**
  - **for vs while, for in, přesměrování a roury**

Příkaz test, celá čísla

Příkaz **test** slouží k porovnávání hodnot a testovaní typů souborů a adresářů (man bash, man test). V případě, že je test splněn, je návratová hodnota příkazu nastavena na 0 (pravda).

Porovnávání celých čísel:

```
test cislo1 operator cislo2
```

!=	nerovná se
==	rovná se
<	menší
<=	menší nebo rovno
>	větší
>=	větší nebo rovno

Operátor:

```
-eq   rovná se (equal)
-ne   nerovná se (not equal)
-lt   menší než (less than)
-le   menší než nebo rovno (less or equal)
-gt   větší než (greater  than)
-ge   větší než nebo rovno (greater or equal)
```

Alternativní zápis:

```
[[ cislo1 operator cislo2 ]]
```

musí být mezery

Příkaz test, řetězce

Porovnávání řetězců

```
test retezec1 operator retezec2
[[ retezec1 operator retezec2 ]]
```

Operátor :

`==`	řetezce jsou identické (lze použít i `=`)
`!=`	řetězce se liší

Testovaní řetězců

```
test operator retezec1
[[operator retezec1 ]]
```

Operátor :

`-n`	testuje zda-li řetězec **nemá** nulovou délku
`-z`	testuje zda-li řetězec **má** nulovou délku
`-f`	testuje zda-li je řetězec název existujícího **souboru**
`-d`	testuje zda-li je řetězec název existujícího **adresáře**

Příkaz test, logické operátory

Logické operátory:

```
||    logické nebo
&&    logické ano
!     negace
```

- Pomocí logických operátorů, lze vytvářet složitější podmínky.
- Pokud neznáme prioritu operátorů nebo si nejsme jisti, tak používáme kulaté závorky.
- Bash používá **líné vyhodnocování** podmínek, které spočívá ve vyhodnocování pouze té části složené logické podmínky, kterou je nutné vyhodnotit pro zjištění výsledné logické hodnoty.

Příkaz test, příklady

```
[[ (I -ge 5) && (I -le 10) ]]
```
Je hodnota proměnné I v intervalu <5;10>?

```
[[ (I -lt 5)||(I -gt 10) ]] nebo [[ !((I -ge 5)&&(I -le 10)) ]]
```
Je hodnota proměnné I mimo interval <5;10>?

```
[[ I -ne 0 ]]
```
Je hodnota proměnné I menší nebo rovna nule?

```
[[ "$A" == "test" ]]
```
Obsahuje proměnná A řetězec "test"?

```
[[ "$A" != "test" ]]
```
Obsahuje proměnná A jiný řetězec než "test"?

```
[[ -z "$A" ]]
```
Obsahuje proměnná A prázdný řetězec?

```
[[ -f "$NAME" ]]
```
Existuje soubor, jehož jméno je v proměnné NAME?

```
[[ ! (-d "$NAME") ]]
```
Neexistuje adresář, jehož jméno je v proměnné NAME?

Cvičení

1. Napište skript, který se uživatele postupně zeptá na dvě čísla. Po jejich zadání vypíše jejich podíl. Ve skriptu ošetřete možnou chybu při dělení nulou.
2. Napište skript, který vytvoří adresář, jehož jméno zadá uživatel po spuštění skriptu. Ošetřete chybovou situaci způsobenou tím, že vytvářený adresář již existuje.
3. Napište skript, který se dotáže na celé číslo. Skript pak otestuje, zda-li se skutečně jedná o celé číslo.

Cyklus pomocí for

Cyklus (smyčka) je řídicí struktura, která opakovaně provádí posloupnost příkazů. Opakování i ukončení cyklu je řízeno podmínkou.

provede se před spuštěním cyklu (inicializace počítadla)

pokud je podmínka splněna, vykonají se příkazy prikaz1 a další

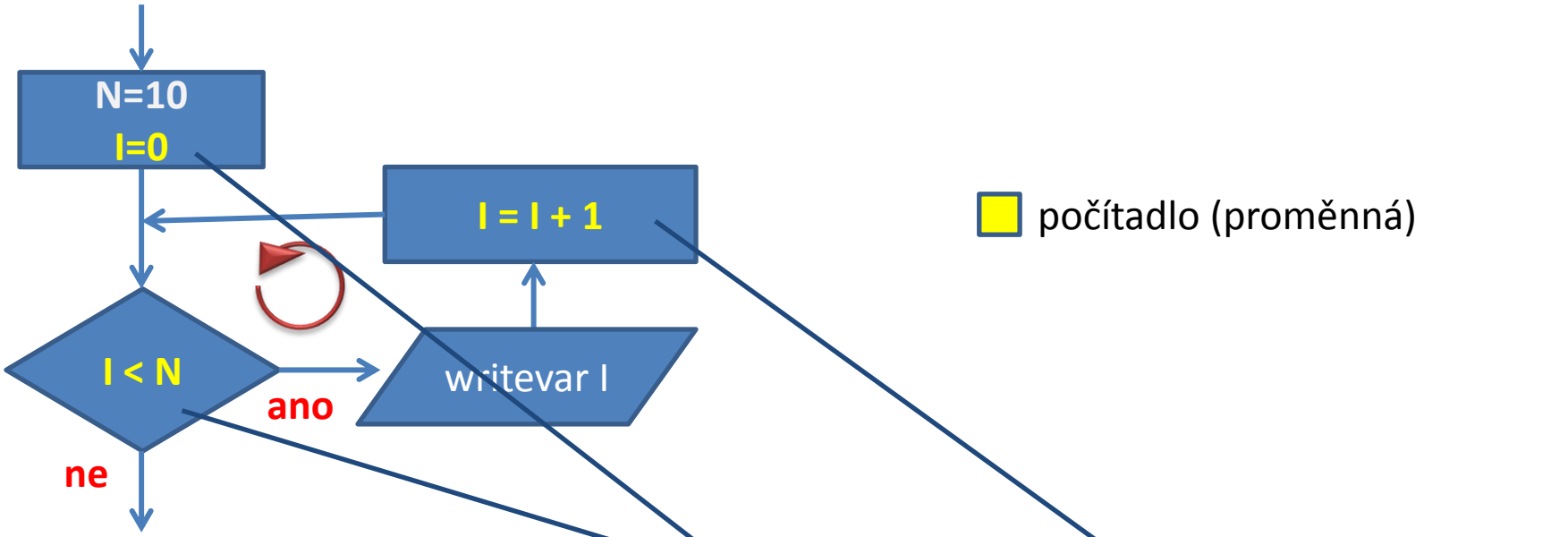
```
for((inicializace;podminka;zmena))
      do
            prikaz1
            ...
      done
```

aktualizace počítadla po vykonání příkazů

Kompaktní zápis:

```
for((inicializace;podminka;zmena)); do
      prikaz1
      ...
done
```

Cyklus pomocí for a vývojový diagram




```
N=10
for((I = 0; I < N; I = I + 1)); do
      echo $I
done
```

Cyklus pomocí for versus while

```
for((I=1;$I <= 10;I++)); do
      echo $I
done
```

pokud je podmínka splněna, vykonají se příkazy v bloku do/done

provede se před spuštěním cyklu (inicializace počítadla)

```
I=1
while [[ $I -le 10 ]]; do
      echo $I
      (( I = I + 1 ))
done
```

Kromě změny počítadla na konci cyklu, je možné provádět i další změny v těle cyklu. Toto se však NEDOPORUČUJE, protože to snižuje čitelnost kódu.

aktualizace počítadla po vykonání příkazů

Změnu počítadla je možné provádět kdekoliv v těle cyklu (i na více místech).

Cyklus pomocí for, použití

Vypíše čísla 1 až 10

```
for((I=1;I <= 10;I++)); do
      echo $I
done
```

Vypíše čísla 10 až 1

```
for((I=10;I >= 1;I--)); do
      echo $I
done
```

Proměnná **I** má roli **počítadla**.

Inicializace se řídí volnými pravidly, jelikož je výraz uveden v (( )) bloku.

Změna:
Lze použít libovolný výraz, který je možné interpretovat v (( )) bloku, např.

- `++` hodnotu proměnné zvýší o jedničku
- `--` hodnotu proměnné sníží o jedničku

další ....

Podmínka:
Lze použít následující porovnávací operátory:

`!=`	nerovná se
`==`	rovná se
`<`	menší
`<=`	menší nebo rovno
`>`	větší
`>=`	větší nebo rovno

Lze použít pouze na celá čísla v (( )).

Cyklus pomocí for, změna počítadla

Pokud lze proměnnou interpretovat jako celé číslo, lze použít následující aritmetické operátory:

++ hodnotu proměnné zvýší o jedničku

```
A++
```

-- hodnotu proměnné sníží o jedničku

```
A--
```

+ sečte dvě hodnoty

```
A = 5 + 6
A = A + 1
```

- odečte dvě hodnoty

```
A = 5 - 6
A = A - 1
```

***** vynásobí dvě hodnoty

```
A = 5 * 6
A = A * 1
```

/ vydělí dvě hodnoty (celočíselné dělení)

```
A = 5 / 6
A = A / 1
```

```
A=A+3
```

+= k proměnné přičte hodnotu

```
A += 3
A += B
```

-= od proměnné odečte hodnotu

```
A -= 3
A -= B
```

***=** proměnnou vynásobí hodnotou

```
A *= 3
A *= B
```

/= proměnnou podělí hodnotou

```
A /= 3
A /= B
```

Vnořování cyklů

Řídící skupiny cyklů lze do sebe libovolně vnořovat.

vnější cyklus

```
for((I=1;I <= 10;I++)); do
      for((J=1;J <= 10;J++)); do
            echo "$I $J"
      done
done
```

vnitřní cyklus

počítadlo vnějšího cyklu může ovlivňovat chování vnitřního cyklu

```
for((I=1;I <= 10;I++)); do
      for((J=1;J <= I;J++)); do
            echo "$I $J"
      done
done
```

Počet vnoření není omezen. Lze kombinovat s jinými cykly (while, until, for in) nebo podmínkami.

Cvičení

1. Napište skripty v jazyce bash pro Úkol 1 a 2, místo cyklu pomocí while použijte cyklus pomocí for. Rozměr vykreslovaného obrazce nechť uživatel zadá jako první argument skriptu. Skript otestuje, zda-li je zadán správný počet argumentů a zda-li je první argument celé číslo větší než nula.

Úkol 1

Do terminálu vytiskněte čtverec se znaků **X**. Délku strany čtverce zadá uživatel.

```
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
```

To, že se nejedná vzhledově o čtverec, ignorujte. Počet znaků **X** na řádku a počet řádků však musí být stejný.

Úkol 2

Do terminálu vytiskněte pravoúhlý trojúhelník se znaků **X**, tak aby jedna odvěsna byla umístěna nahoře a druhá na levé straně. Délku odvěsny zadá uživatel.

```
x x x x x x x x x x
x x x x x x x x x
x x x x x x x x
x x x x x x x
x x x x x x
x x x x x
x x x x
x x x
x x
x
```

Cyklus pomocí for ... in ...

Příkazy v bloku **do/done** (**prikaz1, …**) se vykonají pro každý prvek v seznamu **LIST**. V daném běhu cyklu obsahuje proměnná **VAR** aktuální prvek ze seznamu **LIST**.

```
for VAR in LIST
      do
            prikaz1 $VAR
            ...
      done
```

Kompaktní zápis:

```
for VAR in LIST; do
      prikaz1 $VAR
      ...
done
```

Cyklus pomocí for ... in ..., seznamy

```
for A in a b c; do
      echo $A
done
```

Cyklus proběhne třikrát, během toho postupně vytiskne znaky **a**, **b**, **c**.

Seznamy položek je vhodné vytvářet programově (pomocí příkazů uvedených v obrácených apostrofech).

```
for A in `ls *.eps`; do
      ./process_file $A
done
```

Příkaz **process_file** se vykoná pro každý soubor s příponou **.eps**, který se nachází v aktuálním adresáři.

```
for A in `seq 1 0.25 10`; do
      printf "%8.3f\n" $A
done
```

Vypíše reálná čísla v intervalu od 1 do 10 s přírůstkem 0,25. Čísla budou uvedeny s přesností tří desetinných míst a zarovnány doprava v poli o délce 8 znaků.
Dokumentace: man seq

Přesměrování a roury

Čtení souboru po řádcích:

```
cat soubor.txt | while read A; do
      prikaz2
      ...
done
```

roura

```
while read A; do
      prikaz2
      ...
done < soubor.txt
```

přesměrování

Přesměrování do souboru:

```
for((I=1;I <= 10;I++)); do
      echo $I
done > soubor.txt
```

Výstup všech příkazů v cyklu je přesměrován do **soubor.txt**.

Přesměrování a roury - příklady

```
for((I=1;I <= 10;I++)); do
      echo $I
done > soubor.txt
```

stejná funkcionalita

```
rm -f soubor.txt
for((I=1;I <= 10;I++)); do
      echo $I >> soubor.txt
done
```

```
for((I=1;I <= 10;I++)); do
      echo $I
      printf "N=%10d\n" $I
done > soubor.txt
```

rozdílná funkcionalita

```
rm -f soubor.txt
for((I=1;I <= 10;I++)); do
      echo $I >> soubor.txt
      printf "N=%10d\n" $I
done
```

Cvičení

1. Upravte skripty z předchozího cvičení, tak aby se rozměr obrazce načítal ze standardního vstupu a výsledný obrazec se tisknul do souboru, jehož jméno zadá uživatel opět ze standardního vstupu.
2. Napište skript, který vypíše reálná čísla v intervalu od -10 do 10 s přírůstkem 0,5. Čísla budou uvedena včetně znaménka, zarovnány doprava v poli 10 znaků a uvedeny s přesností na jedno desetinné místo.

Domácí úkoly

Domácí úkol I

Vysvětlete rozdílné chování následujících skriptů. Soubor data.txt obsahuje pět řádků.

```
#!/bin/bash
I=0
cat data.txt | while read A; do
      I=$(($I+1))
done
echo $I
```

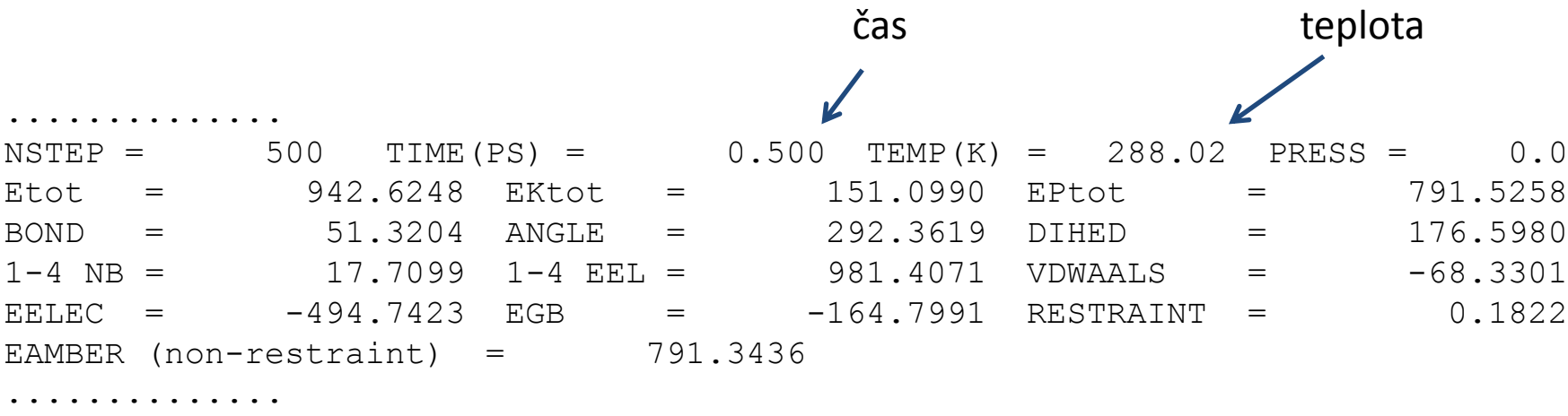
vypíše číslo 0

```
#!/bin/bash
I=0
while read A; do
      I=$(($I+1))
done < data.txt
echo $I
```

vypíše číslo 5

Domácí úkol II

Soubor rst.out (wolf.ncbr.muni.cz:/home/kulhanek/Documents/C2110/Lesson08/rst.out) obsahuje výsledky z molekulové dynamiky. Úkolem je ze souboru vyextrahovat závislost teploty simulovaného systému na čase a tyto data uložit do souboru temp.out, který bude obsahovat právě dva sloupce. První sloupec bude čas a druhý sloupec bude teplota.



POZOR: Skript nesmí obsahovat příkazy grep, awk a ani jejich varianty. Při řešení použijte příkaz read a while.

Samostudium

Funkce - pro **pokročilé studenty.**

Funkce - definice

Funkce je konstrukce, která umožňuje seskupit část kódu tak, aby jej bylo možné snadno použít na více místech skriptu. Funkce tedy usnadňuje a zpřehledňuje zápis opakovaných úkonů.

Definice:

```
function jmeno () {
        prikaz1
        prikaz2
        ...
}
```

```
function jmeno {
        prikaz1
        prikaz2
        ...
}
```

```
jmeno () {
        prikaz1
        prikaz2
        ...
}
```

alternativní zápisy

Argumenty funkce se nedeklarují, nedochází tedy k žádné kontrole v počtu argumentů, typové kontrole, funkce nelze přetěžovat. Zadané argumenty jsou dostupné přes speciální proměnné #, 1 až 9, *. Funkce se volají jako existující příkaz. **Proměnné ve funkci jsou globální** (lze změnit pomocí klíčového slova local). Dokumentace: man bash, sekce FUNCTIONS.

Funkce – použití

```
# print line - the length is in the first argument
function print_line () {
      N=$1
      for((J=1;J <= N;J++)); do
            echo -n " X"
      done
      echo ""
}

# use function
print_line 10    # print line 10 characters long
print_line  5    # print line 5 characters long
```

hodnota argumentu je dostupná ve speciální proměnné **1**

Cvičení

1. Napište jeden skript, který vytiskne čtverec a trojúhelník (podobně jako v úkolu 1 a 2) pro jednu zadanou délku za sebe do terminálu. Ve skriptu identifikujte část, která se opakuje a přepište ji za použití funkce.

```
X X X X
X X X X
X X X X
X X X X

X X X X
X X X
X X
X
```

To, že se nejedná vzhledově o čtverec, ignorujte. Počet znaků **X** na řádku a počet řádků však musí být stejný.