# C2110
# *Operační systém UNIX a základy programování*

## 8. lekce

Petr Kulhánek

kulhanek@chemi.muni.cz

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

# Obsah

- **Skriptování v Bashi**
  **podmínky, cyklus pomoci while, vnořování řídících struktur**

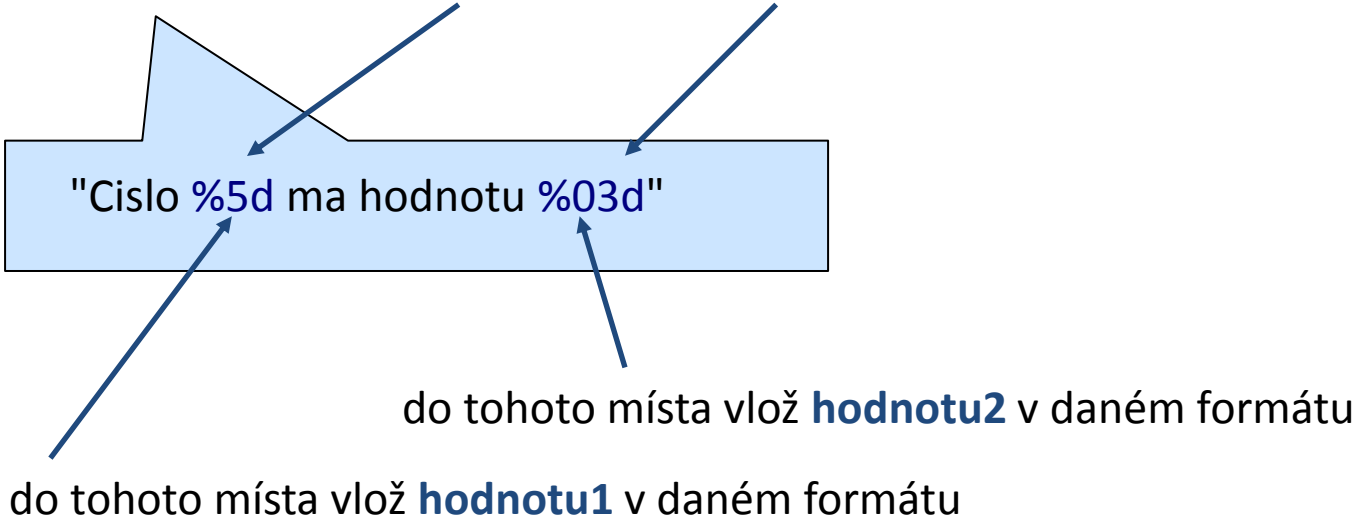
- **Nové příkazy**
  **printf, read, expr, test, exit**

# Příkaz printf

Příkaz **printf** slouží k vypisování formátovaných textů a čísel.

**Syntaxe:**

```
printf [format] [hodnota1] [hodnota2] ...
```



# Příkaz printf, příklady

```
$ I=10
$ B=12.345

$ printf "Hodnota promenne I je %d\n" $I
Hodnota promenne I je 10

$ printf "Zadane cislo B je %10.4f\n" $B
Zadane cislo B je    12.3450

$ printf "Zadane cislo B je %010.4f\n" $B
Zadane cislo B je 00012.3450

$ printf "Zadane cislo B je %+010.4f\n" $B
Zadane cislo B je +0012.3450

$ printf "Cislo I je %-5d a cislo B je %.1f\n" $I $B
Cislo I je 10    a cislo B je 12.3
```

# Příkaz printf, formát

[] – volitelná část

**%[priznak][delka][.presnost]typ**

počet míst za desetinou tečkou (reálná čísla)

celková délka pole

**Příznak:**

| | |
|---|---|
| **-** | zarovnat doleva |
| **0** | prázdné místo zaplnit nulami |
| **+** | vždy uvést znaménko |

**Typ:**

| | |
|---|---|
| **d** | celé číslo |
| **s** | řetezec (text) |
| **f** | reálné číslo |

**Speciální znaky:**

| | |
|---|---|
| **\n** | konec řádku |
| **\r** | vrať se na začátek řádku |
| **%%** | znak % |

Další informace: man bash, man printf

# Příkaz read

Příkaz **read** slouží k čtení textu ze standardního vstupu a jeho uložení do proměnných. Příkaz načte vždy celý řádek, do první proměnné se uloží první slovo, ..., do poslední proměnné se uloží zbytek řádku.

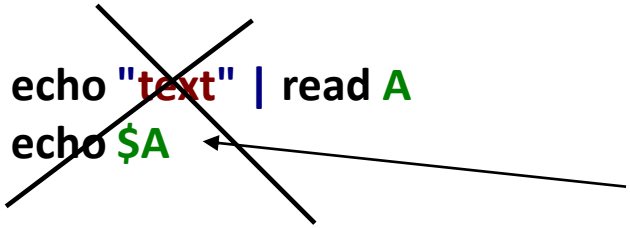
**Syntaxe:**

```
read A      # celý řádek se uloží do proměnné A
read A B    # první slovo se uloží do proměnné A
            # zbytek řádku do proměnné B
```

**Příklad:**

```
echo -n "Zadej hodnotu: "
read A
echo "Zadana hodnota je : $A"
```

**Pozor: nepoužívejte přikaz read ve spojení s rourami**



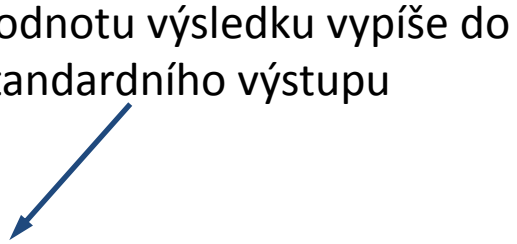

**Nebude** obsahovat hodnotu "text"

# Aritmetické operace

Aritmetické operace s celými čísly lze vykonat v bloku **(( ... ))**.

**Možné zápisy:**

```
(( I = I + 1 ))
(( I++ ))
I=$(( $I + 1 ))
echo "Hodnota I zvetsena o jedna : $(( I + 1 ))"
```




**Operátory:**

| | |
|---|---|
| `=` | přiřazení |
| `+` | sčítání |
| `-` | odčítání |
| `*` | násobení |
| `/` | dělení |
| `%` | zbytek po dělení |
| `++` | inkrementace (zvýšení hodnoty o 1) |
| `--` | dekrementace (snížení hodnoty o 1) |

# Cvičení

1. Napište skript, který vypíše počet argumentů, které jste zadali při jeho spuštění.
2. Napište skript, který vypíše první zadaný argument skriptu ve formátu **%4d**.
3. Napište skript, který načte ze standardního vstupu číslo a to vypíše následujícím způsobem (bude uvedeno znaménko, pro výpis se použije pět míst, prázdné místa budou vyplněny nulami):

   Zadane cislo je :  +0003
4. Co se stane, pokud skriptu ze cvičení 3, předložíte číslo: 123456?
5. Napište skript, kterému se budou předkládat dvě čísla jako argumenty. Skript tyto čísla vypíše a dále vypíše jejich součet.

# Aritmetické operace, příkaz expr

Příkaz **expr** vyhodnocuje matematické výrazy, výsledky se tisknou do standardního výstupu.

**Příklady:**

```
$ expr 1 + 2
3

$ expr 2 \* 3
6

I=`expr $I + 1`
```

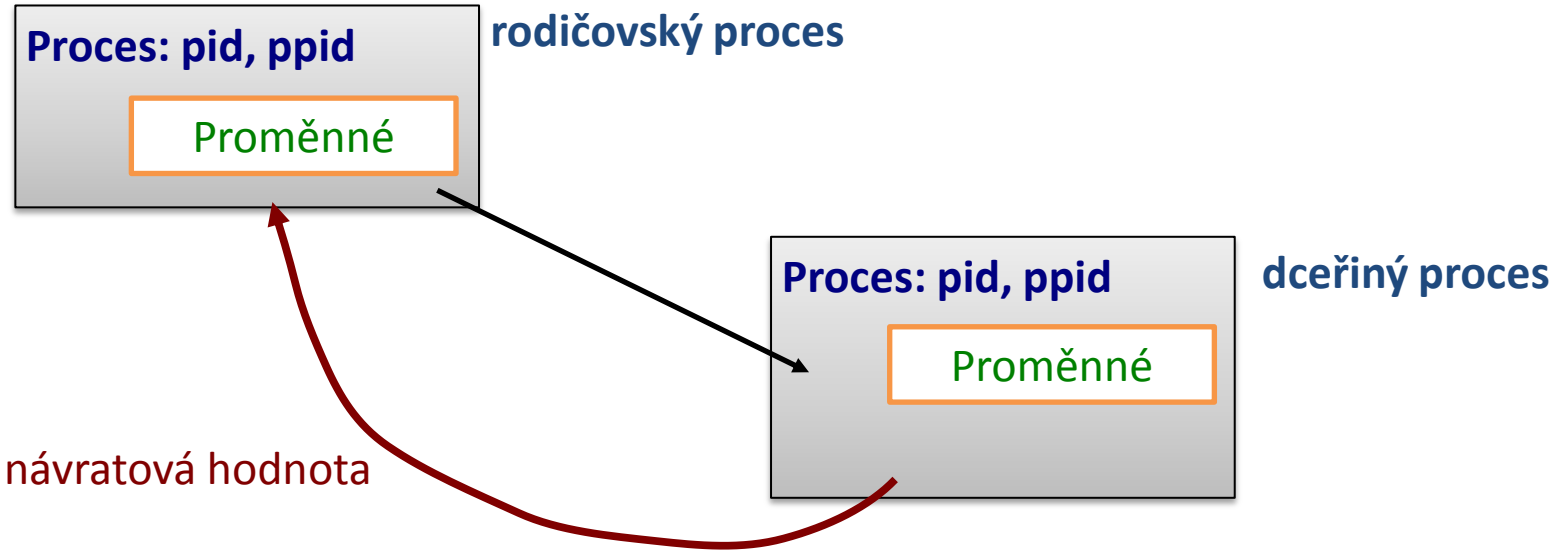
**** zabrání expanzi speciálního znaku * na jména souborů a adresářů nacházejících se v aktuálním adresáři

výsledek vložíme do proměnné I

Další informace: man expr

# Návratová hodnota příkazů

**Končící proces** může rodičovskému procesu sdělit informaci o svém průběhu pomocí **návratové hodnoty**. Návratová hodnota je celé číslo nabývající hodnot 0-255.




**Návratová hodnota:**

**0** = **vše proběhlo úspěšně**

**> 0** = **došlo k chybě**, vrácená hodnota pak zpravidla identifikuje chybu

**Návratovou hodnotu** posledně provedeného příkazu lze zjistit pomocí proměnné **?**.

# Návratová hodnota, příklady

```
$ mkdir test
$ echo $?
0
```

```
$ mkdir test
mkdir: cannot create directory `test1': File exists
$ echo $?
1
```

```
$ expr 4 + 1
5
$ echo $?
0
```

```
$ expr a + 1
expr: non-integer argument
$ echo $?
1
```

# Příkaz test, celá čísla

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

**Porovnávání celých čísel:**

```
test cislo1 operand cislo2
```

**Operand:**

| | |
|---|---|
| **-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) |

# Příkaz test, řetězce

**Porovnávání řetězců**

```
test retezec1 operand řetezec2
```

**Operand:**

| | |
|---|---|
| **==** | řetezce jsou identické |
| **!=** | řetězce se liší |

**Testovaní řetězců**

```
test operand retezec1
```

**Operand:**

| | |
|---|---|
| **-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** |

# Podmínky

```
if prikaz1
      then
            prikaz2
            ...
      fi
```

Pokud **prikaz1** skončí s návratovou hodnotou **0**, vykoná se **prikaz2**. V opačném případě se vykoná **prikaz3**.

```
if prikaz1
      then
            prikaz2
            ...
      else
            prikaz3
            ...
      fi
```

**Kompaktní zápisy:**

```
if prikaz1; then
      prikaz2
      ...
fi
```

```
if prikaz1; then
      prikaz2
      ...
else
      prikaz3
      ...
fi
```

# Příkaz exit

Příkaz **exit** slouží k ukončení běhu skriptu nebo interaktivního sezení. Nepovinným argumentem příkazu je návratová hodnota.

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

```
$ ./muj_skript 5
Cislo je vetsi nez nula.
$ echo $?
0
```

```
$ ./muj_skript -10
Cislo neni vetsi nez nula!"
$ echo $?
1
```

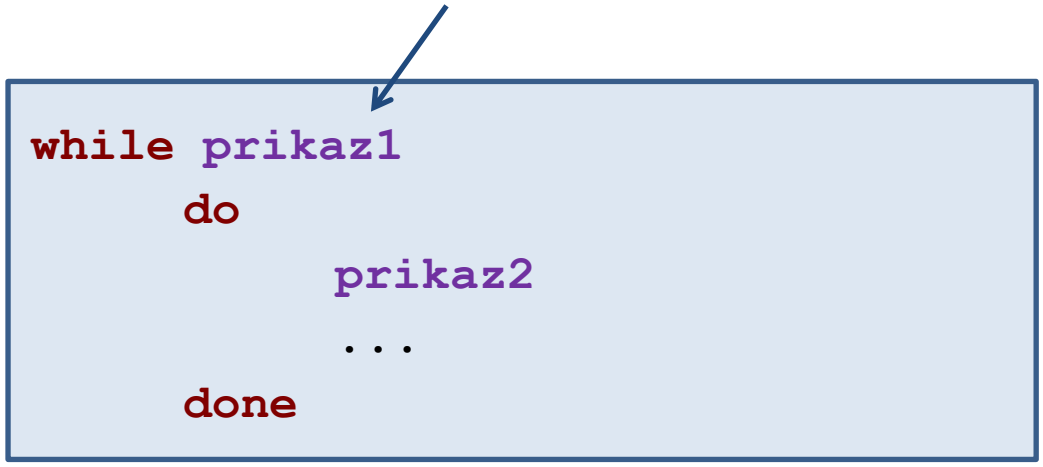
# Cvičení

1. Napište skript, který ze standardního vstupu přečte dvě čísla. Skript tyto čísla vypíše a dále vypíše informaci, zda-li je první číslo vetší nebo menší než druhé (formát výpisu je ponechán na autorovi skriptu).
2. Napište skript, kterému se budou předkládat dvě čísla jako argumenty. Skript tyto čísla vypíše a dále vypíše jejich podíl. Pomocí podmínky ošetřete situaci zamezující dělení nulou.
3. Seznam souborů a adresářů, které se vyskytují ve vašem domovském adresáři uložte do souboru **list.txt**
4. Napište skript, kterému předložíte název souboru, jako argument. Skript otestuje, zda-li soubor existuje a pokud ano, tak vypíše jeho obsah a počet řádků, které soubor obsahuje.  Funkčnost skriptu ověřte na souboru **list.txt**.

# Cyklus pomocí while

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

cyklus probíhá **dokud** prikaz1 vrací v návratové hodnotě 0

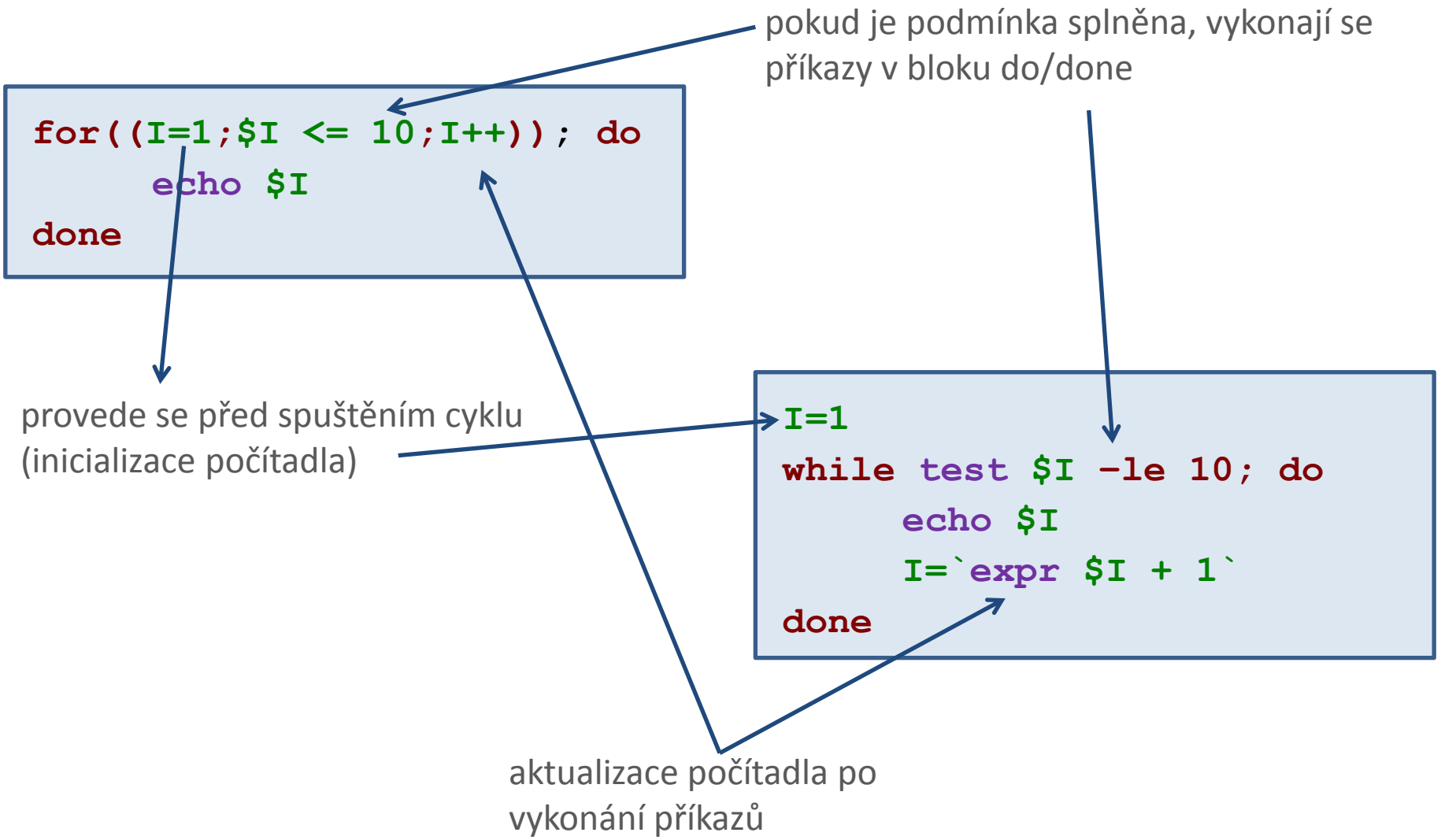


```
while prikaz1
      do
            prikaz2
            ...
      done
```

**Kompaktní zápis:**

```
while prikaz1; do
      prikaz2
      ...
done
```

# Cyklus pomocí for versus while



# 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**.

# Vnořování

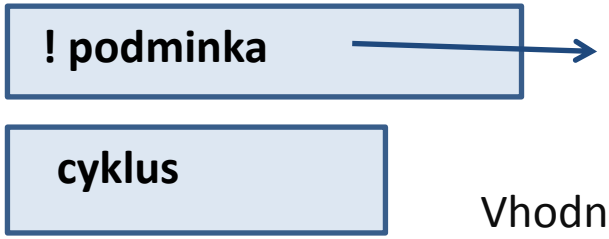
Řídící skupiny lze libovolně do sebe vnořovat.

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

vnější cyklus

vnitřní cyklus

Při návrhu skriptu se snažíme o zamezení zbytečného vnořovaní (převážně z důvodu snadnější orientace ve skriptu).


Vhodnější uspořádání pro testovaní vstupních dat od uživatelů.

# Domácí úkol

1. Vykreslete do terminálu plný obdélník z písmen "X". Rozměry obdélníku zadá uživatel pomocí argumentů skriptu.
2. Upravte předchozí skript tak, že vykreslíte pouze obrys obdélníku.
3. Napište skript, který vykreslí dva pravoúhlé trojúhelníky v následujících orientacích. Délku odvěsny zadá uživatel po spuštění skriptu ze standardního vstupu.

```
X

X X

X X X


X X X

X X

X
```

4. Vykreslete kružnici nebo kruh z písmen X. Poloměr a to zda se má vykreslit kružnice či kruh zadá uživatel z klávesnice po spuštění skriptu.

# Domácí úkol, II

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
```

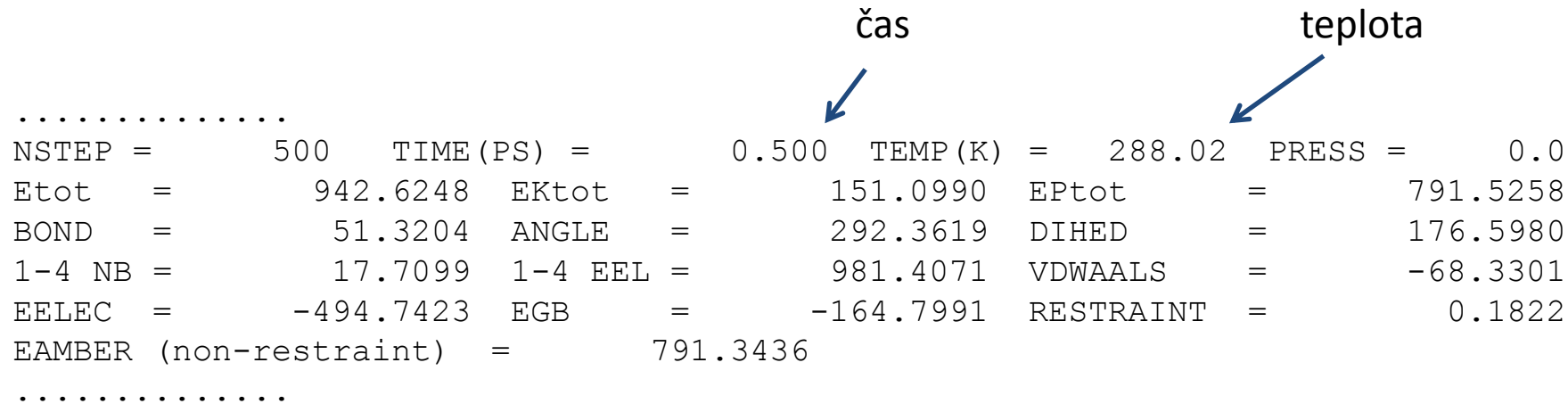
vypíše číslo 0

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

vypíše číslo 5

# Domácí úkol, III

Soubor rst.out (wolf.ncbr.muni.cz:/home/kulhanek/Data/rst.out) obsahuje výsledky z molekulové dynamiky. Úkolem je ze souboru vyextrahovat závislost teploty simulovaného systému na čase.




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