C2110 *Operační systém UNIX a základy programování*

8. 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

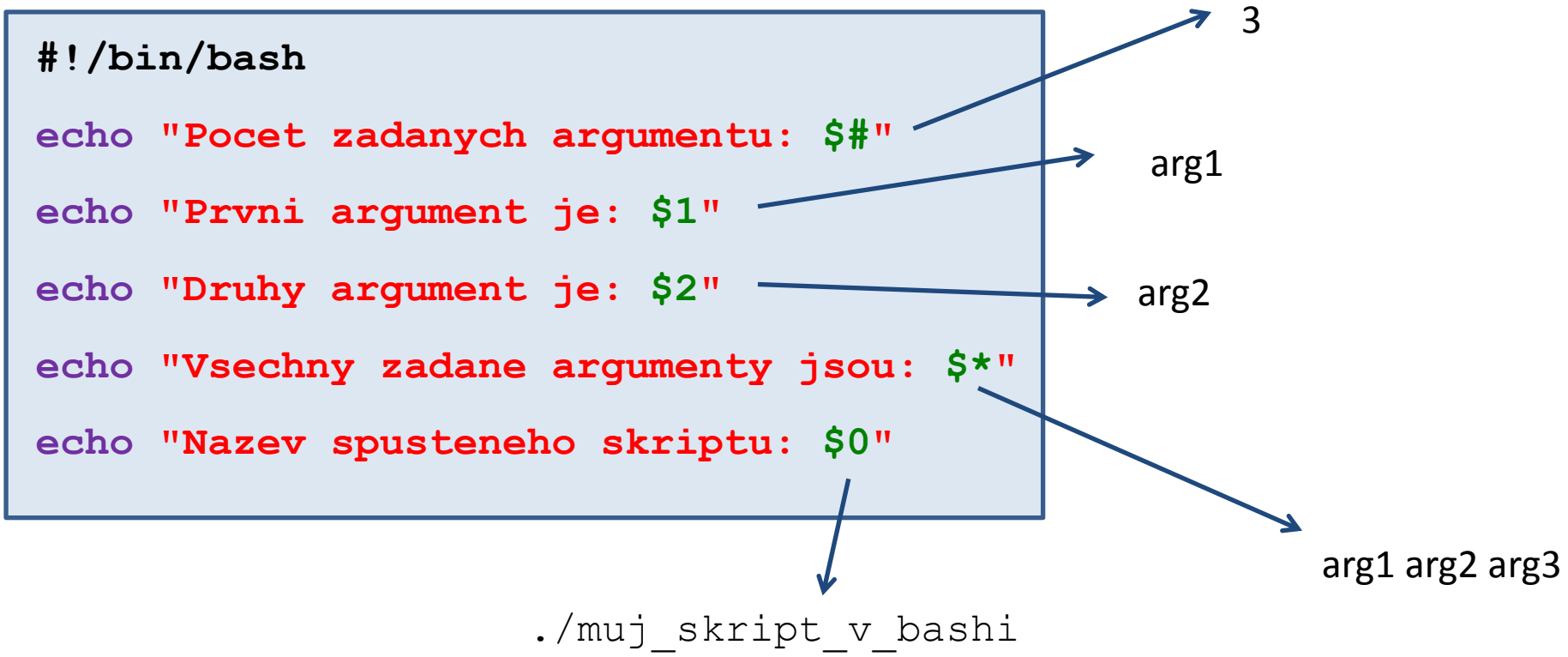
- Speciální proměnné
  - argumenty skriptu
- Vstup/Formátovaný výstup
  - příkazy printf, read
- Aritmetické operace
  - příkaz expr
- Návratová hodnota příkazů
- Podmínky
  - příkazy test, exit

Speciální proměnné

- **Argumenty skriptu**

Argumenty skriptu

```
$ bash muj_skript_v_bashi arg1 arg2 arg3
$./muj_skript_v_bashi arg1 arg2 arg3
```



Použití a význam argumentů si určuje autor skriptu.

Přehled

Argumenty skriptu:

#	počet argumentů, se kterými byl skript spuštěn
0	název spuštěného skriptu
1 … 9	hodnoty argumentů 1 až 9, se kterými byl skript spuštěn
*****	všechny argumenty, se kterými byl skript spuštěn

Procesy:

?	návratová hodnota posledně vykonaného příkazu (procesu)
$	číslo procesu (PID)

Pokročilá práce s argumenty:

Pokud potřebujeme předat více jak devět argumentů, je nutné použít příkaz **shift**. Příkaz odstraní první argument ze seznamu argumentů.

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

Vypíše postupně zadané argumenty skriptu.

Cvičení

1. Napište skript, který vypíše počet argumentů, které jste zadali při jeho spuštění.
2. Napište skript, který vypíše znaky **A** vedle sebe. Počet vytisknutých znaků zadá uživatel jako první argument skriptu.
3. Napište skript, který vypíše číslo **PID** procesu, ve kterém je skript interpretován. Za výpis vložte příkaz **sleep**, kterým vykonávání skriptu pozastavíte na 5 minut. Použijte příkaz **kill** v jiném terminálu k předčasnému ukončení vašeho skriptu.

Vstup/výstup

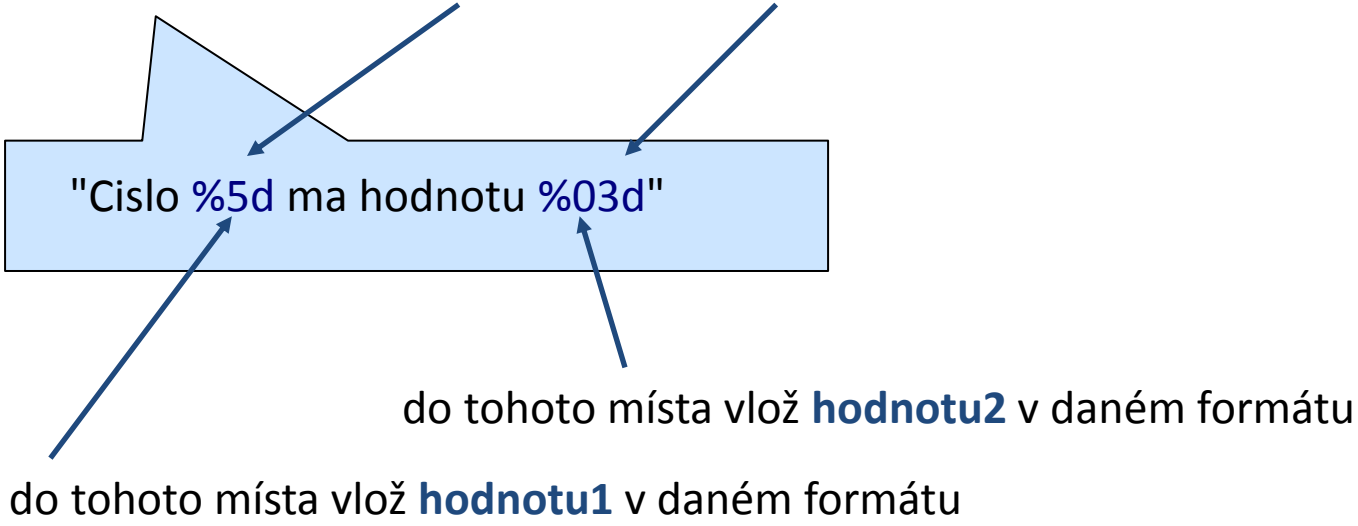
- Formátovaný výstup - printf
- Vstup - read

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.

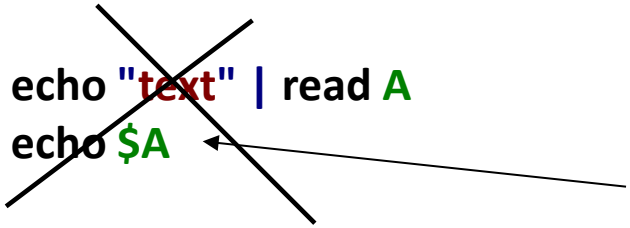
Syntaxe:

```
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

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

Možné zápisy:

hodnotu výsledku vypíše do standardního výstupu

```
(( 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)

Další informace: man bash

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

předáváme hodnotu proměnné

výsledek vložíme do proměnné I

Další informace: man expr

Cvičení

1. Napište skript, který vypíše první zadaný argument skriptu ve formátu **%4d**.
2. 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:

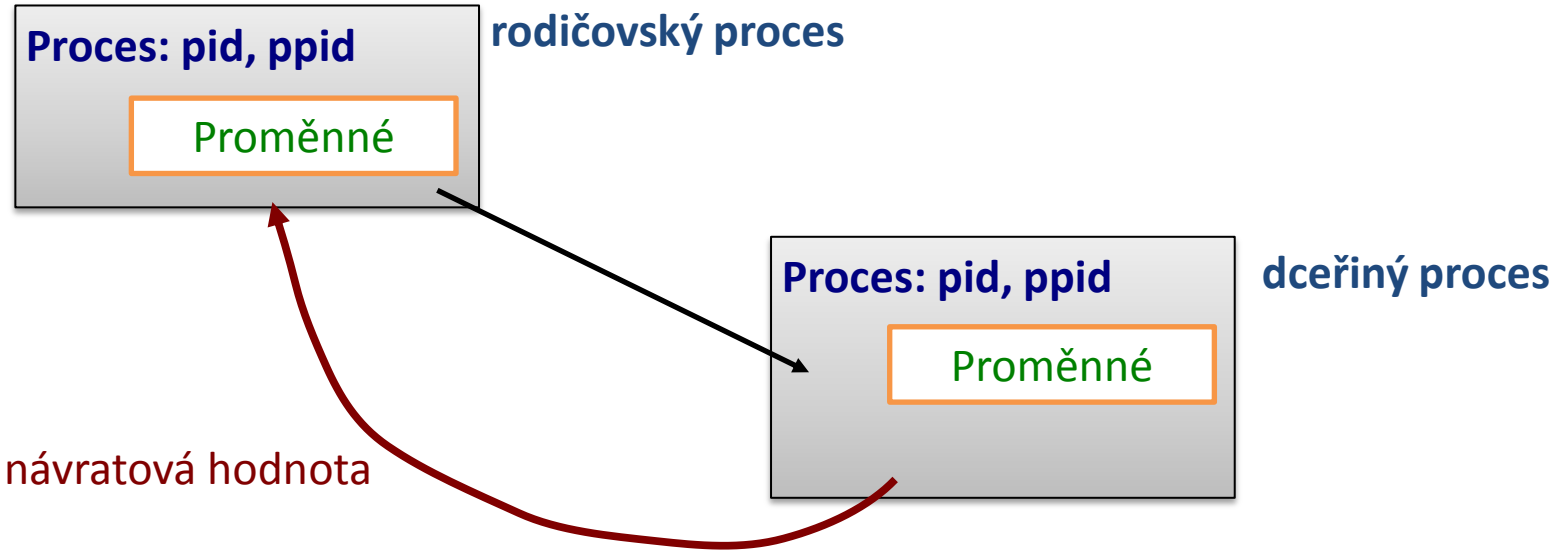
   Zadane cislo je : +0003
3. Co se stane, pokud skriptu ze cvičení 2, předložíte číslo: 123456?
4. Napište skript, kterému se budou předkládat dvě čísla jako argumenty. Skript tyto čísla vypíše a dále vypíše jejich součet.

Podmínky

- Návratová hodnota příkazů
- Podmínky

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 `test': 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 operator cislo2
```

Operator:

-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)

Další informace: man bash, man test

Příkaz test, řetězce

Porovnávání řetězců

```
test retezec1 operator řetezec2
```

Operator:

==	řetezce jsou identické
!=	řetězce se liší

Testovaní řetězců

```
test operator retezec1
```

Operator:

-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
```

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

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

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 "$1" -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
```

Vnořování

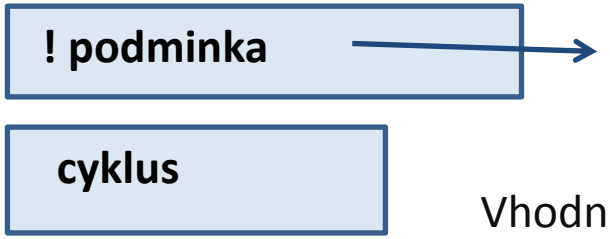
Řídící skupiny lze libovolně do sebe vnořovat.

```
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.

Cvičení

1. Napište skript, který ze **standardního vstupu** přečte **dvě čísla**. Skript tyto čísla **vypíše** a dále vypíše informaci, zda-li je první číslo **větší** nebo **menší** než druhé (způsob 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**.

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.