C2110 *Operační systém UNIX a základy programování*

7. lekce

Skriptování v jazyce bash

Petr Kulhánek

kulhanek@chemi.muni.cz

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

Obsah

- **Skriptování v bashi**
  - **základní syntaxe, srovnání s vývojovými diagramy, spouštění skriptů**
- **Proměnné**
  - **nastavení a získaní hodnoty, proměnné a procesy, interpretace řetězců**
- **Vstup a výstup**
  - **read, echo, printf**
- **Aritmetické operace**
  - **operace s celými čísly**

Skriptování v bashi

Skript v Bashi

```
#!/bin/bash
# toto je komentar
echo 'Toto je skript v interpretu Bash!'
echo "Obsah adresare `pwd` je:"
ls     # vypise obsah adresare
A=6    # nastaveni hodnoty promenne A
echo "Hodnota promenne A je $A"
echo "jeden prikaz"; echo "druhy prikaz"
./mujprikaz prvni_argument druhy_argument \
            treti_argument
```

ihned následuje nový řádek

- prázdné řádky se ignorují
- text uvozený znakem **#** se ignoruje (používá se ke komentovaní funkčnosti skriptu)
- na jeden řádek lze uvést více příkazů, příkazy se oddělují středníkem **;**
- jeden příkaz lze napsat na více řádků pomocí zpětného lomítka ****

V čem psát skripty (a programy)

Jelikož jsou **skripty** a zdrojové kódy programů **textové soubory**, lze použít libovolný textový editor umožňující uložení textu v čisté formě (bez formátovacích metadat).

Textové editory:

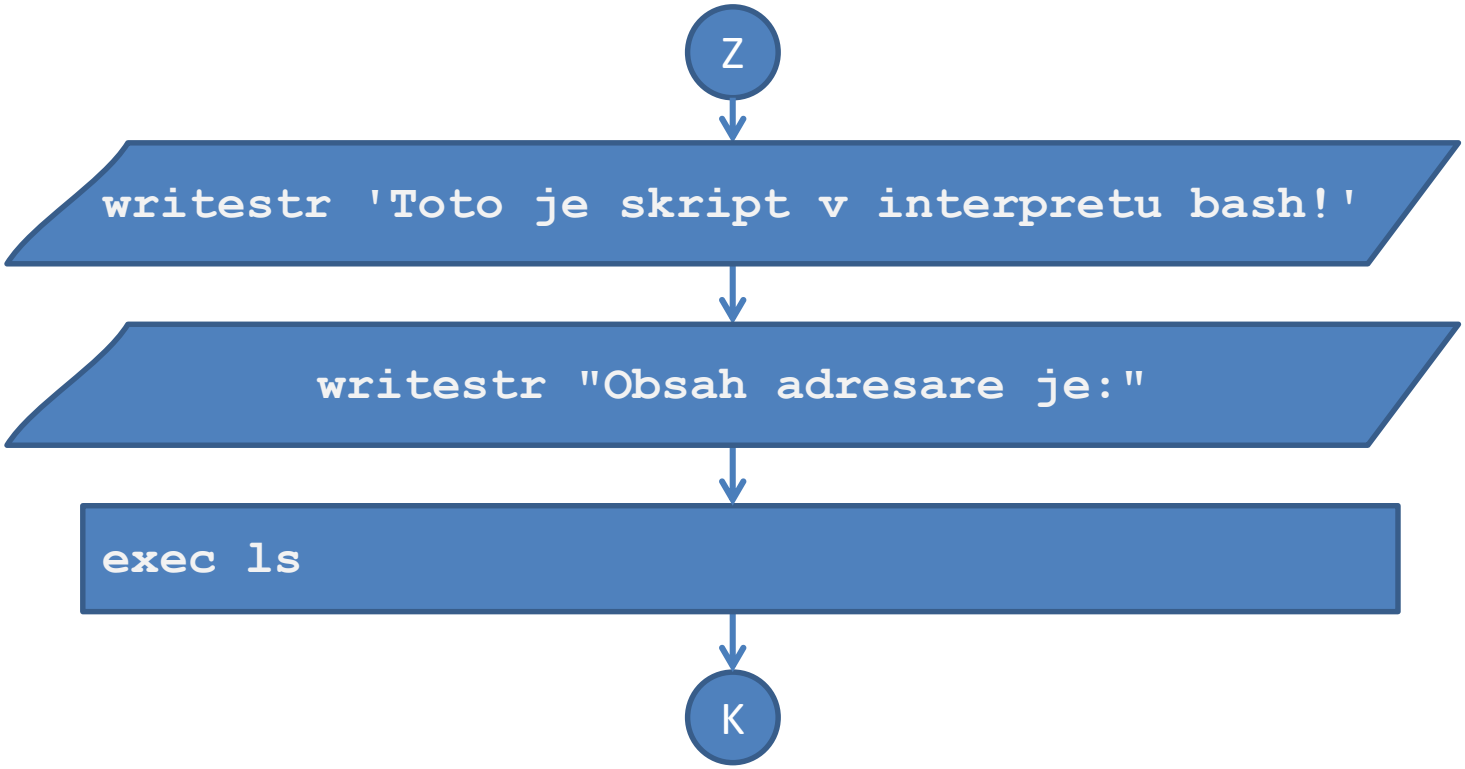
- vi
- kwrite
- kate
- **gedit**

K psaní skriptů a zdrojových kódů programů lze používat i specializované vývojové prostředí – **IDE** (Integrated Development Enviroment). IDE obsahuje kromě editoru i správce projektu, ladící nástroje (debugger) a další. Většinou dostupné pro komplexnější jazyky: *C, C++, Fortran, JavaScript*, *Python*, *PHP*, atd.

Vývojové prostředí:

- Kdevelop
- qtcreator
- NetBeans
- Eclipse

Vývojový diagram a skript v bashi



pořadí vykonávání příkazů

```
#!/bin/bash
echo 'Toto je skript v interpretu bash!'
echo "Obsah adresare je:"
ls
```

Spouštění skriptů

1) Nepřímé spouštění

Spouštíme interpreter jazyka a jako argument uvádíme jméno skriptu.

```
$ bash muj_skript_v_bashi
```

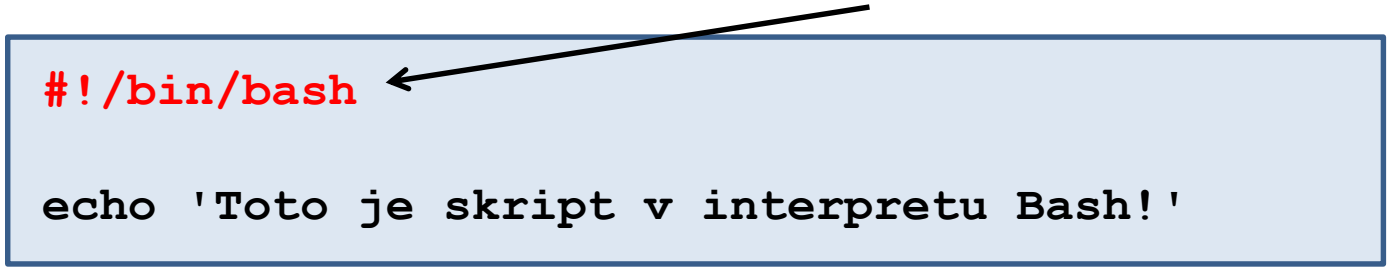
Skripty **nemusí** mít nastaven příznak x (executable).

2) Přímé spouštění

Spouštíme přímo skript (shell automaticky spustí interpreter).

```
$ chmod u+x muj_skript_v_bashi
$./muj_skript_v_bashi
```

Skripty **musí** mít nastaven příznak **x** (executable) a **interpreter** (součást skriptu).



Určení interpretru

Specifikace interpretru (první řádek skriptu):

```
#!/absolutní/cesta/k/interpretru/skriptu
```

Skript v bashi

```
#!/bin/bash

echo "Toto je skript v bashi!"
```

Skript v gnuplotu

```
#!/usr/bin/gnuplot

set xrange[0:6]

plot sin(x)

pause -1
```

- Pokud není interpreter skriptu při jeho přímém spuštění uveden, použije se interpreter systémového shellu (bash).
- Interpreter uvedený ve skriptu se ignoruje při nepřímém spuštění.
- Interpretr je vhodné do skriptu vždy uvádět, protože je použit textovými editory pro zvýrazňování syntaxe. (Název souboru uvádíme bez přípony, popř. s příponou **.sh**)

Cvičení I

1. Napište skript v jazyce bash, který vypíše text "Aktualni adresar je:" následovaný výpisem absolutní cesty k aktuálnímu adresáři. Skript spusťte nepřímo pomocí interpretu bash.
2. Rozšiřte funkčnost předchozího skriptu tak, aby kromě výše uvedeného vypsal "Obsah adresare je:" následovaný výpisem seznamu souborů a adresářů v dlouhém formátu. Skript spusťte přímo.

Proměnné

Proměnné

V jazyce Bash se proměnnou rozumí **pojmenované umístění** v paměti, které obsahuje hodnotu. Hodnota proměnné v jazyce Bash je vždy **typu řetězec (text)**.

Nastavení proměnné:

nesmí být mezera mezi **jménem proměnné** a **=**

```
$ JMENO_PROMENNE=hodnota
$ JMENO_PROMENNE="hodnota s mezerami"
```

Přístup k hodnotě proměnné:

```
$ echo $JMENO_PROMENNE
```

"TEXT ${PROMENNA}TEXT"

pokud má být hodnota součásti textu, tak se název proměnné uvádí do složených závorek

Zrušení proměnné:

```
$ unset JMENO_PROMENNE
```

Přehled všech definovaných proměnných:

```
$ set
```

Nastavení proměnné

```
$ JMENO_PROMENNE="hodnota s mezerami"
```

```
$ JMENO_PROMENNE ="hodnota s mezerami"
```

interpretuje se jako název programu — mezera — interpretuje se jako argument programu

```
$ JMENO_PROMENNE= "hodnota s mezerami"
```

JMENO_PROMENNE se nastaví na prázdný řetězec, hodnota proměnná je dostupná pouze spouštěnému programu — mezera — interpretuje se jako název programu

```
$ JMENO_PROMENNE="hodnota s mezerami" program [arg1...]
```

lze uvést několik proměnných a jejich hodnot (v zápisu se dvojice JMÉNO=HODNOTA oddělují mezerou), které jsou dostupné jen pro spuštěný program

pokud název programu obsahuje rovnítko, musí být název uveden v uvozovkách

Řetězce

V jazyce Bash lze použít čtyři typy řetězců:

- **bez uvozovek**

```
A=pokus
B=*
C=$A
```

může dojít k expanzi, vše záleží na kontextu zápisu a obsahu aktuálního adresáře

nahradí se hodnotou proměnné A

- **s uvozovkami**

```
A="pokus hokus"
B="* $A"
```

hodnota proměnně obsahuje dvě slova oddělené mezerou

nahradí se hodnotou proměnné A, hvězdička se neexpanduje (je uvedena v uvozovkách)

- **s jednoduchými uvozovkami (apostrof)**

```
A='pokus hokus'
B='* $A'
```

text je uveden přesně, bez žádné expanze či transformace

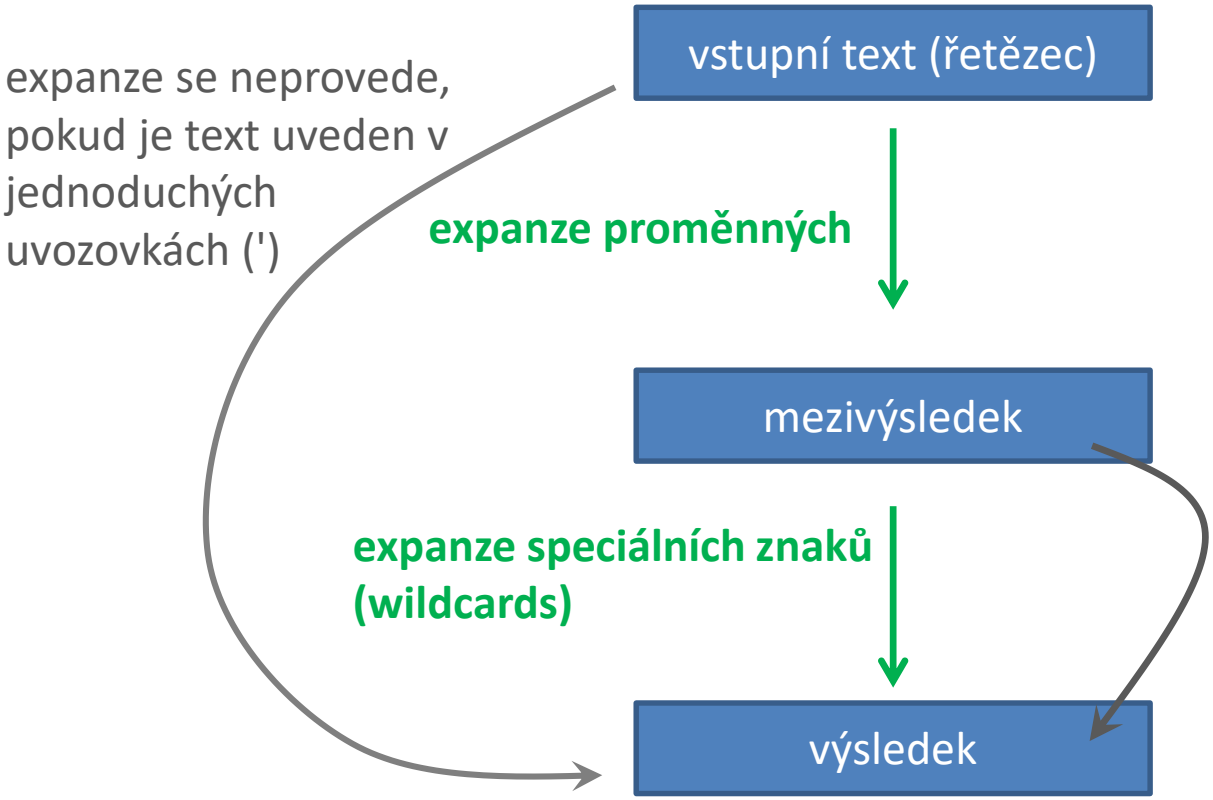
- **s obrácenými jednoduchými uvozovkami (obrácený apostrof)**

```
A="`ls -d`"
B="pocet : `ls | wc -l`"
```

do **místa** obrácených uvozovek se vloží výstup **příkazu** uvedeného v uvozovkách

Expanze řetězce/příkazové řádky

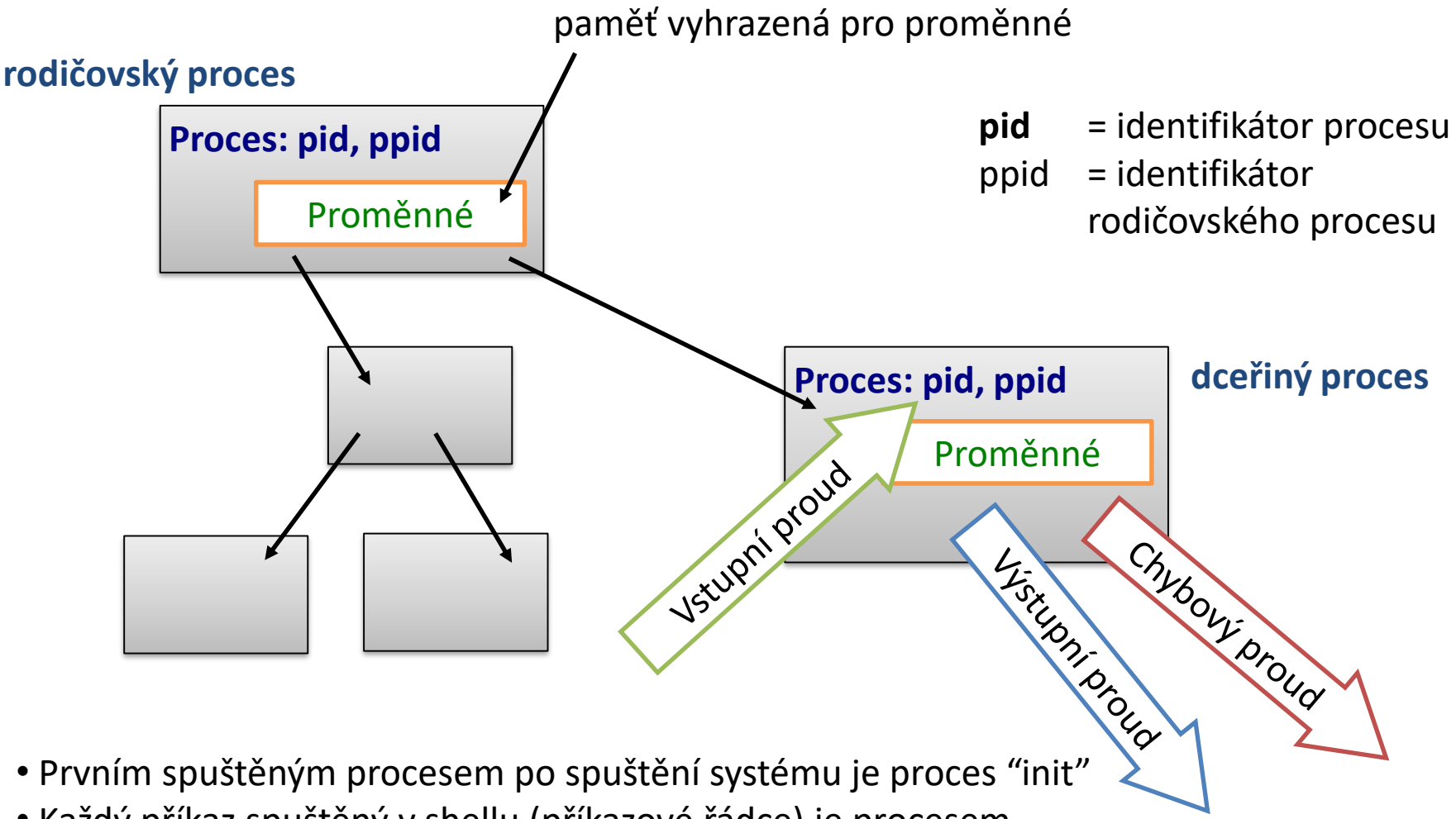
Pořadí expanze řetězce/příkazové řádky:



expanze se neprovede, pokud je text uveden ve standardních uvozovkách ("), nebo není obsažen ve slově, které by bylo možné expandovat, netýká se tedy přiřazení hodnoty do proměnných

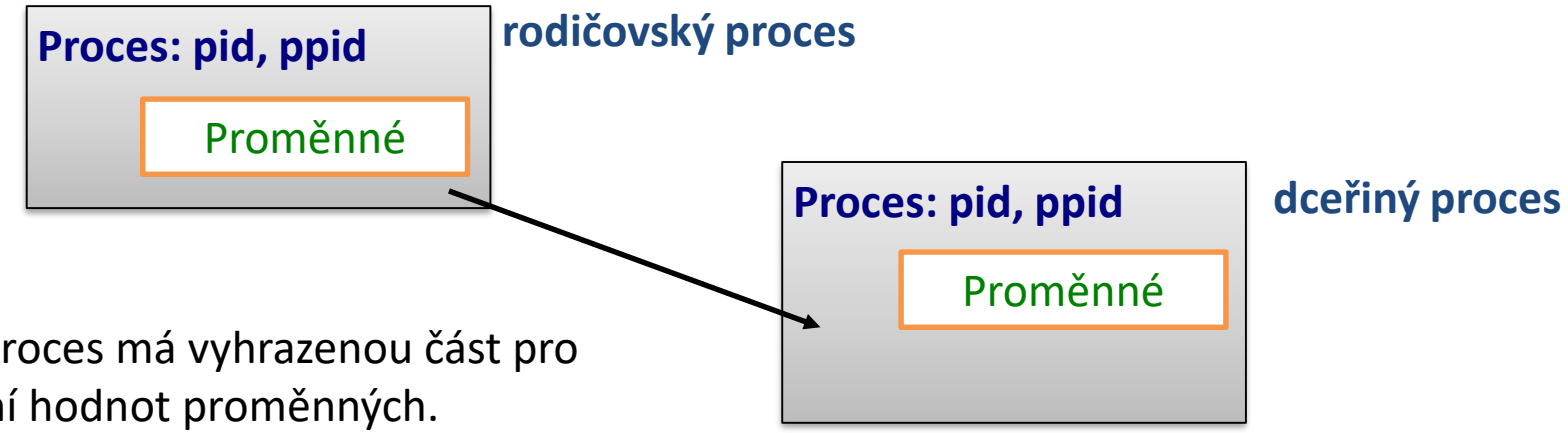
Procesy

Proces je instance běžícího **programu**.



- Prvním spuštěným procesem po spuštění systému je proces “init”
- Každý příkaz spuštěný v shellu (příkazové řádce) je procesem

Proměnné a procesy



Každý proces má vyhrazenou část pro ukládání hodnot proměnných.

Dceřiný proces v okamžiku svého spuštění **získá kopii** proměnných (exportovaných) a jejich hodnot od rodičovského procesu. Tyto proměnné může dle potřeby měnit nebo mazat. Dále může nastavovat nebo mazat nové proměnné. **Všechny tyto změny však po skončení dceřiného procesu zaniknou.** Změny se **neprojeví** na hodnotách **původních proměnných** rodičovského procesu.

Export proměnné:

```
$ export JMENO_PROMENNE
$ export JMENO_PROMENNE="hodnota"
```

export

export s přiřazením

Cvičení II

Pracujte v interaktivním režimu shellu.

1. Nastavte proměnnou A na hodnotu 55.
2. Vypište hodnotu proměnné A (příkazem echo)
3. Vylistujte všechny proměnné nastavené v daném terminálu. Je mezi nimi proměnná A? Použijte příkaz less nebo more k zpřehlednění výpisu.
4. Použijte příkaz grep a vypište pouze řádek obsahující záznam o proměnné A. Vyhledávací vzor zvolte tak, aby byl nezávislý na hodnotě proměnné.
5. Vypište všechny nastavené proměnné, jejichž jména začínají písmenem A (grep ^TEXT).
6. Změňte hodnotu proměnné na "tohle je dlouhy retezec".
7. Vypište hodnotu proměnné A.
8. Zrušte proměnnou A.
9. Ověřte, že jste proměnnou zrušili (postupem řešeným v bodě 4).
10. Postupně nastavujte proměnné A, B a C podle příkladů uvedených na straně 13. Postupně ověřujte jejich hodnotu příkazy set a echo. Analyzujte případné rozpory.

Cvičení III

Pracujte v novém terminálu.

1. Zrušte proměnnou PATH. Jak se změna projeví na funkcionalitě příkazové řádky? Zkuste spustit příkaz ls a pwd. Chovaní vysvětlete.
2. Kdy se provede expanze divokého znaku * v následujícím příkladu:

```
$ B="Obsah adresare je *"
$ echo $B
```

3. Napište skript s názvem print_C, který vypíše hodnotu proměnné C. Vysvětlete chování v následujících příkladech:

```
$ ./print_C
$ C="hodnota 1" ./print_C
$ echo $C
$ C="hodnota 2"
$ echo $C
$ ./print_C
$ export C
$ ./print_C
```

Vstup/výstup

Příkaz read

Příkaz **read** slouží k **čtení textu** ze standardního vstupu (**tj. pro interaktivní načítaní 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

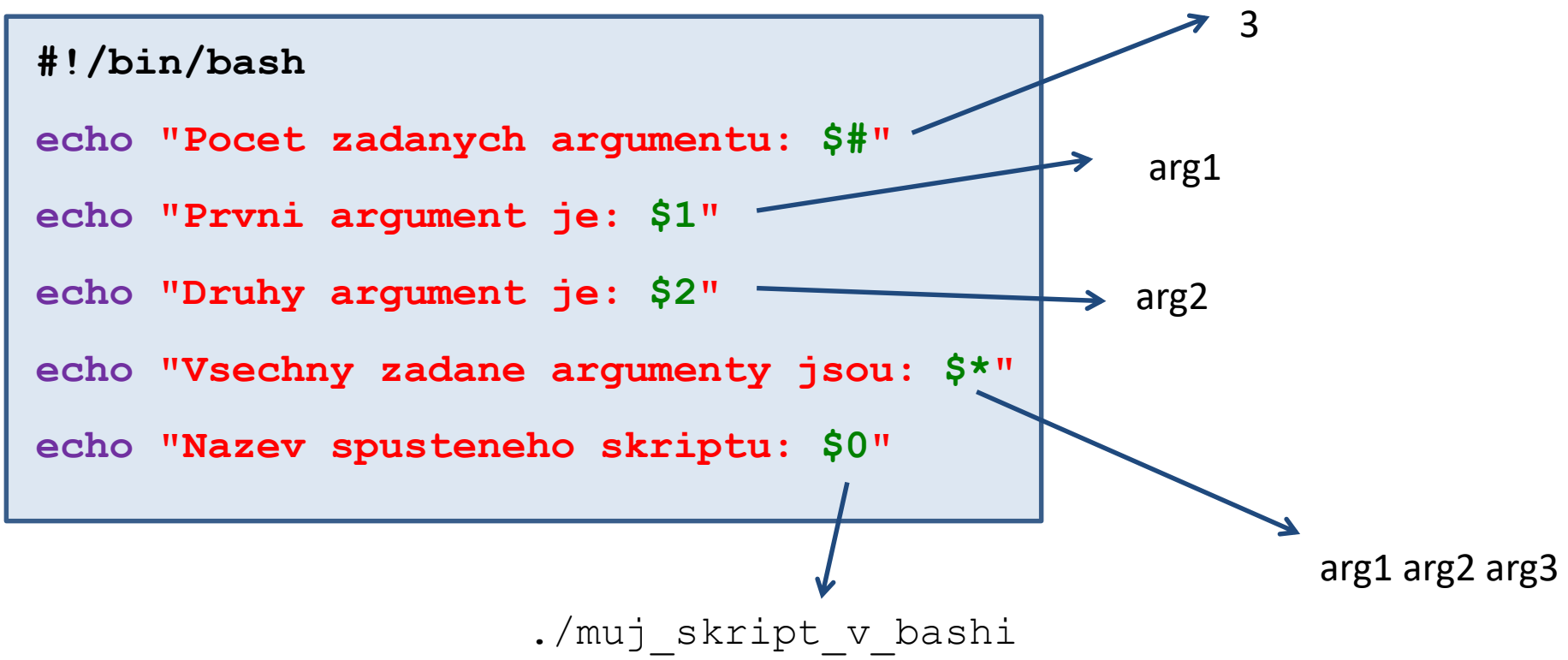
```
echo "text" | read A
echo $A
```

Nebude obsahovat hodnotu "text"

Argumenty skriptu

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

```
#!/bin/bash
echo "Pocet zadanych argumentu: $#"
echo "Prvni argument je: $1"
echo "Druhy argument je: $2"
echo "Vsechny zadane argumenty jsou: $*"
echo "Nazev spusteneho skriptu: $0"
```



Použití a význam argumentů si určuje autor skriptu.

Argumenty skriptu - proměnné

Argumenty skriptu (názvy speciálních proměnných):

#	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

Pokročilá práce s argumenty:

"$@"	všechny argumenty, se kterými byl skript spuštěn vložené do uvozovek (ošetřuje vstup, kde argumenty obsahují mezery) Rozdílné chování vůči **$*** nebo **"$*"** ! !

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 IV

1. Napište skript, který se dotáže uživatele na jeho oblíbenou barvu, kterou posléze vypíše na terminál.
2. Napište skript, který vypíše počet zadaných argumentů a hodnotu prvního argumentu.

Příkaz echo

Příkaz **echo** slouží k neformátovanému výpisu do standardního výstupního proudu.

Syntaxe:

```
echo [volby] [retezec1] [retezec2] ...
```

Příkaz tiskne řetězce v zadaném pořadí oddělené mezerou.

Užitečné volby:

-n neodřádkuje výstup

Příklady:

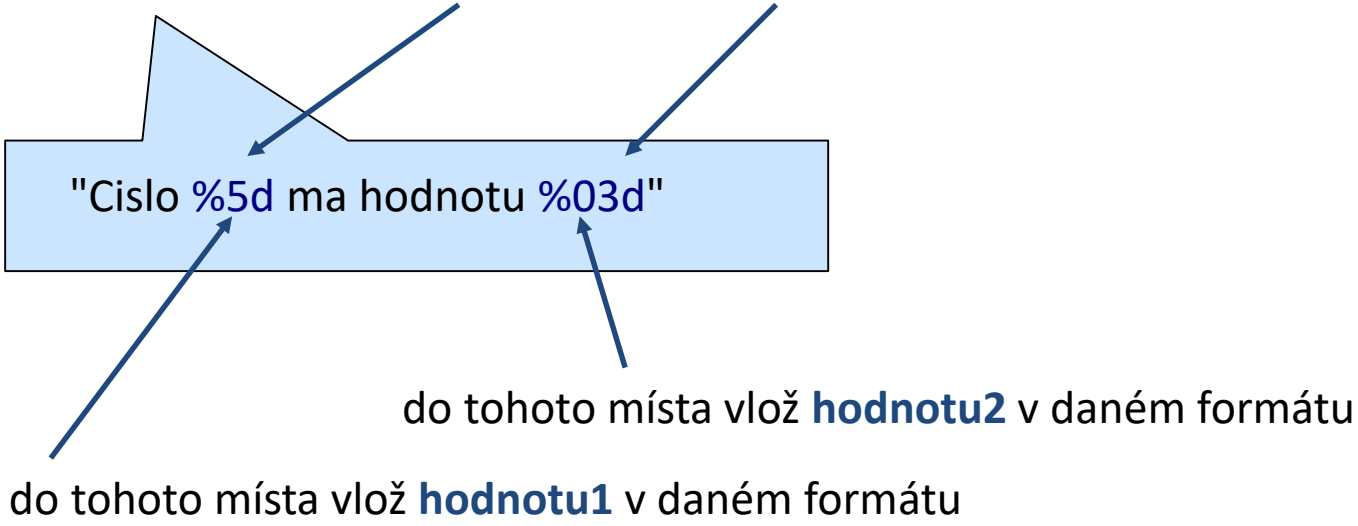
```
$ echo "Ema"       "mele" "maso."
Ema mele maso.
$ echo "Ema   mele   maso."
Ema   mele   maso.
$ echo -n "Zadejte barvu: "
$ A=5
$ echo "Hodnota promenne A je $A."
Hodnota promenne A je 5.
```

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

Cvičení V

1. Napište skript, který se dotáže uživatele na jeho oblíbenou barvu, kterou posléze vypíše na terminál. Dotaz vytiskněte tak, aby zadávaná barva byla na stejném řádku jako dotaz.
2. Procvičte si příkaz printf tak, že provedete příkazy uvedené v příkladech.
3. Napište skript, který vypíše první zadaný argument skriptu ve formátu %4d.
4. 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
5. Co se stane, pokud skriptu ze cvičení 4, předložíte číslo: 123456?

Domácí úkoly

Aritmetické operace

Aritmetické operace

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

Možné zápisy:

hodnotu výsledku vypíše do místa zápisu

```
(( 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

Další informace: man expr

výsledek vložíme do proměnné I

Další možností je použití příkazu **bc**, který umí pracovat i s reálnými čísly.

Domácí úkol

1. Napište skript, kterému se budou předkládat dvě čísla jako argumenty. Skript tyto čísla vypíše a dále vypíše jejich součet.
2. Napište skript, který se uživatele postupně zeptá na dvě čísla. Po jejich zadání vypíše jejich součin.