C2110 *Operační systém UNIX a základy programování*

6. lekce

Petr Kulhánek, Jakub Štěpán

kulhanek@chemi.muni.cz

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

evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
UNIVERSITAS MASARYKIANA BRUNENSIS
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

CZ.1.07/2.2.00/15.0233

Obsah

- **Skripty**
  - Skripty vs. programy
  - Kompilace programu
  - Spouštění programu a ukázkového skriptu
- **Proměnné**
  - Nastavování a rušení proměnných
  - Proměnné a procesy
  - Typy řetězců

Skripty

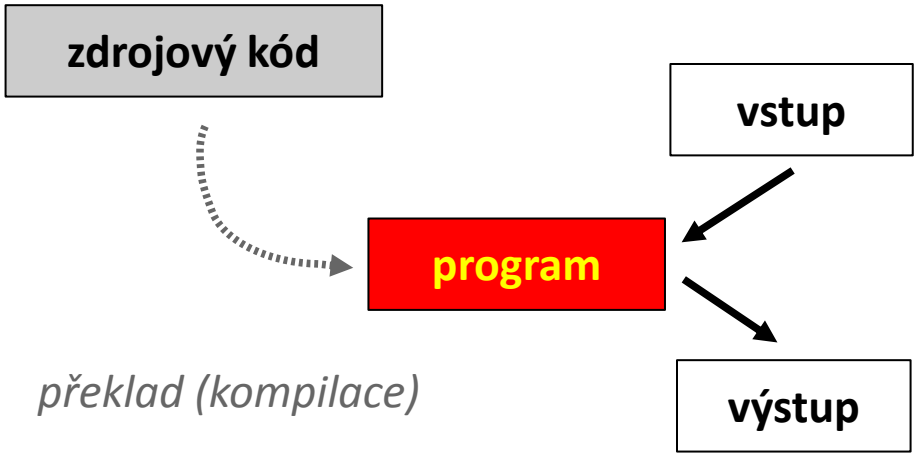
- Skripty vs. programy
- Kompilace programu
- Spouštění programu a ukázkového skriptu

Programy *vs* Skripty

Program je soubor strojových instrukcí zpracovávaných přímo procesorem. Program vzniká překladem zdrojového kódu programovacího jazyka.

Překládané jazyky:

C/C++
Fortran

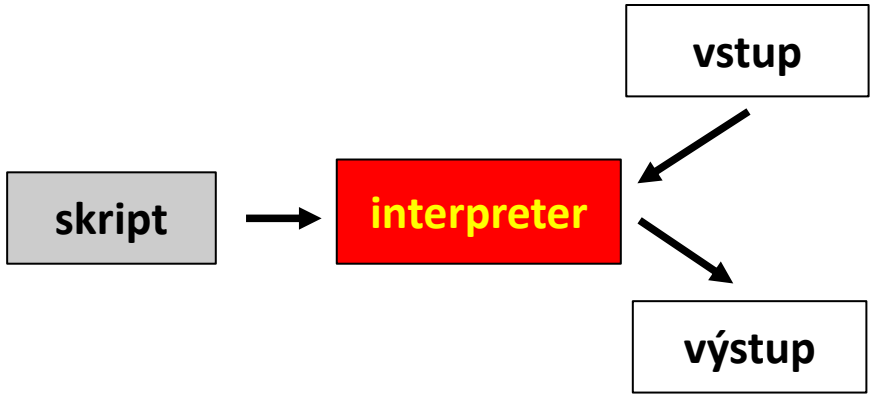


překlad (kompilace)

Skript je textový soubor obsahující příkazy a řídící sekvence, které jsou vykonávány interpreterem použitého skriptovacího jazyka.

Skriptovací jazyky:

bash
gnuplot
awk
JavaScript
PHP



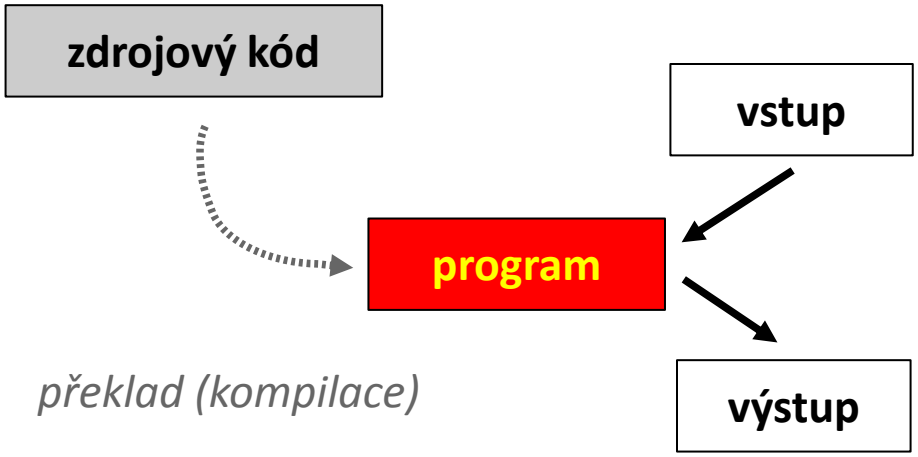
Programy *vs* Skripty, ...

- snadná optimalizace
- rychlé vykonávání

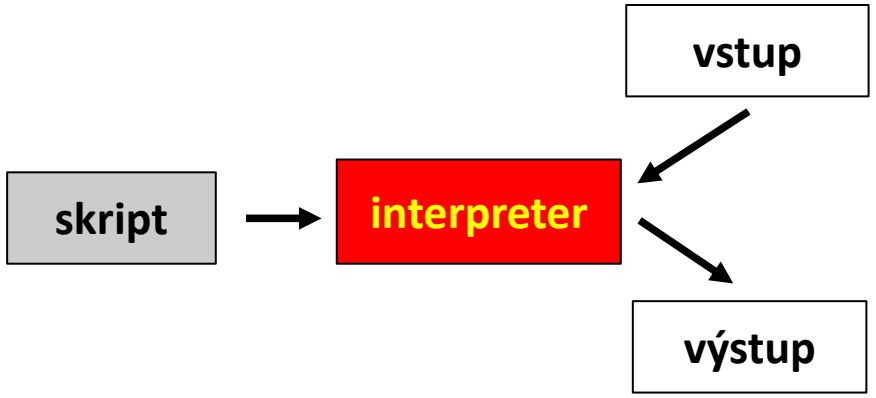
- nutnost rekompilace
- nelze vytvářet samospustitelný kód

- nevyžaduje rekompilaci
- vytváření samospustitelného kódu

- špatná optimalizovatelnost
- pomalejší vykonávání



překlad (kompilace)



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: *JavaScript*, *Python*, *PHP*, atd.

Vývojové prostředí:

- Kdevelop
- qtcreator
- NetBeans
- Eclipse

Program v jazyce C

Zdrojový kód

```
#include <stdio.h>

int main(int argc,char* argv[])
{
  printf("Tohle je program v jazyce C! \n");
  return(0);
}
```

Kompilace

```
$ gcc program.c -o program
```

kompiler jazyka C

název souboru s vytvořeným programem

Spuštění programu

```
$ ./program
```

soubor **program** musí mít práva **pro spuštění**

Program ve Fortranu

Zdrojový kód

```
program Hello

  write(*,*) 'Toto je program ve Fortranu!'

end program
```

Kompilace

```
$ gfortran program.f90 -o program
```

kompiler jazyka Fortran

název souboru s vytvořeným programem

Spuštění programu

```
$ ./program
```

soubor **program** musí mít práva **pro spuštění**

Skript v Bashi

Skript

```
#!/bin/bash

echo 'Toto je skript v interpretu Bash!'
```

Spuštění skriptu

```
$ bash skript.bash
```

soubor **skript.bash** **nemusí** mít práva **pro spuštění**

interpret Bash

Skript v GNUPlotu

Skript

```
#!/usr/bin/gnuplot

set title "Toto je skript v GNUPlotu!"
plot sin(x)

pause -1
```

Spuštění skriptu

```
$ gnuplot skript.gnuplot
```

interpret GNUPlot

soubor **skript.gnuplot** **nemusí** mít práva **pro spuštění**

Cvičení

1. Vytvořte čtyři adresáře s názvy **ukol01, ukol02, ukol03, ukol04**
2. Do jednotlivých adresářů uložte postupně soubory **program.c , program.f90, skript.bash, a skript.gnuplot** z adresáře **/home/kulhanek/Data/programs**
3. Zkompilujte zdrojové kódy programů napsaných v jazyce C a Fortran. Ověřte, že vzniklé programy lze spustit.
4. Jaká je velikost souboru obsahující výsledný program vzniklý kompilací zdrojového kódu v jazyce C. Otevřete vzniklý soubor v textovém editoru. Co soubor obsahuje?
5. Ověřte funkčnost skriptů **skript.bash a skript.gnuplot** jejich spuštěním.

Spouštění skriptů

1) Nepřímé spouštění

Spouštíme interpreter jazyka a jako argument uvádíme jméno skriptu.

```
$ bash muj_skript_v_bashi

$ gnuplot muj_skript_v_gnuplotu
```

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

2) Přímé spouštění

Spouštíme přímo skript (shell automaticky spustí interpreter).

```
$ ./muj_skript_v_bashi

$ ./muj_skript_v_gnuplotu
```

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.
- Interpreter uvedený ve skriptu se ignoruje při nepřímém spuštění.

Určení interpretru, II

Pokud se absolutní cesta k interpretru mění (např. při použití softwareových modulů), lze použít následující konstrukci:

```
#!/usr/bin/env interpreter
```

Interpreter musí být v některém adresáři určeném systémovou proměnnou PATH.

Skript v bashi

```
#!/usr/bin/env bash

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

Skript v gnuplotu

```
#!/usr/bin/env gnuplot

set xrange[0:6]

plot sin(x)

pause -1
```

Cvičení

1. Změňte přístupová práva u souborů **skript.bash a skript.gnuplot** (příkaz **chmod**).
2. Ověřte, že lze skripty spustit přímo bez uvedení interpretru.
3. Co se stane, pokud k interpretaci **skriptu skript.gnuplot** použijete interpretr **bash**?

Proměnné

- Nastavování a rušení proměnných
- Proměnné a procesy
- Typy řetězců

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
```

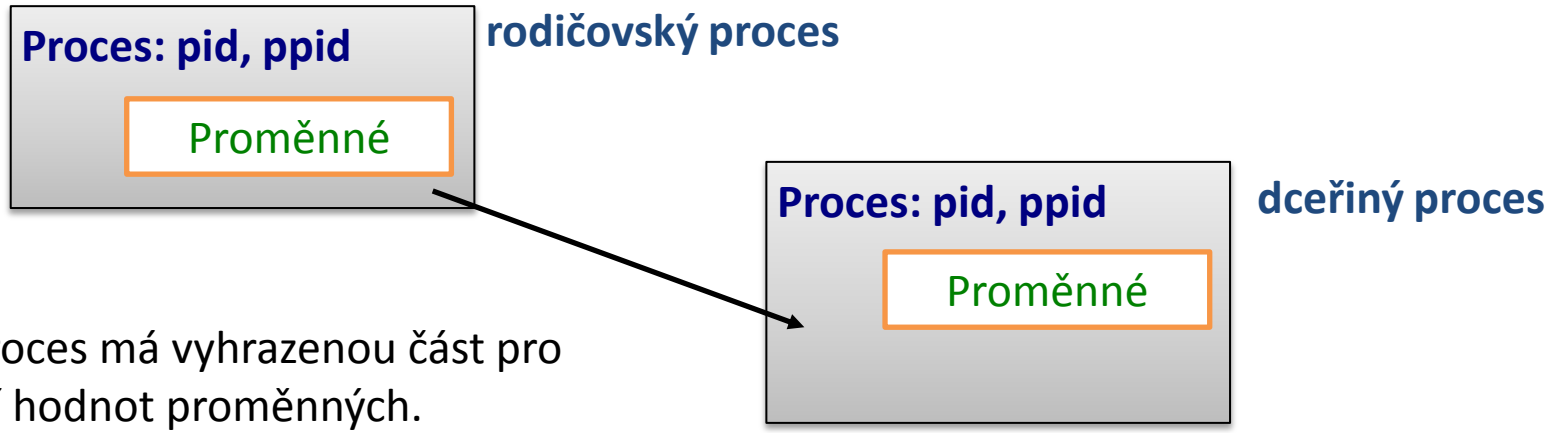
Zrušení proměnné:

```
$ unset JMENO_PROMENNE
```

Přehled všech proměnných:

```
$ set
```

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

Řetězce

V jazyce Bash lze použít čtyři typy řetězců:

- **bez uvozovek**

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

nahradí se seznamem souborů a adresářů, které jsou v aktuálním adresáři (lze použít složitější kontrukce)

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

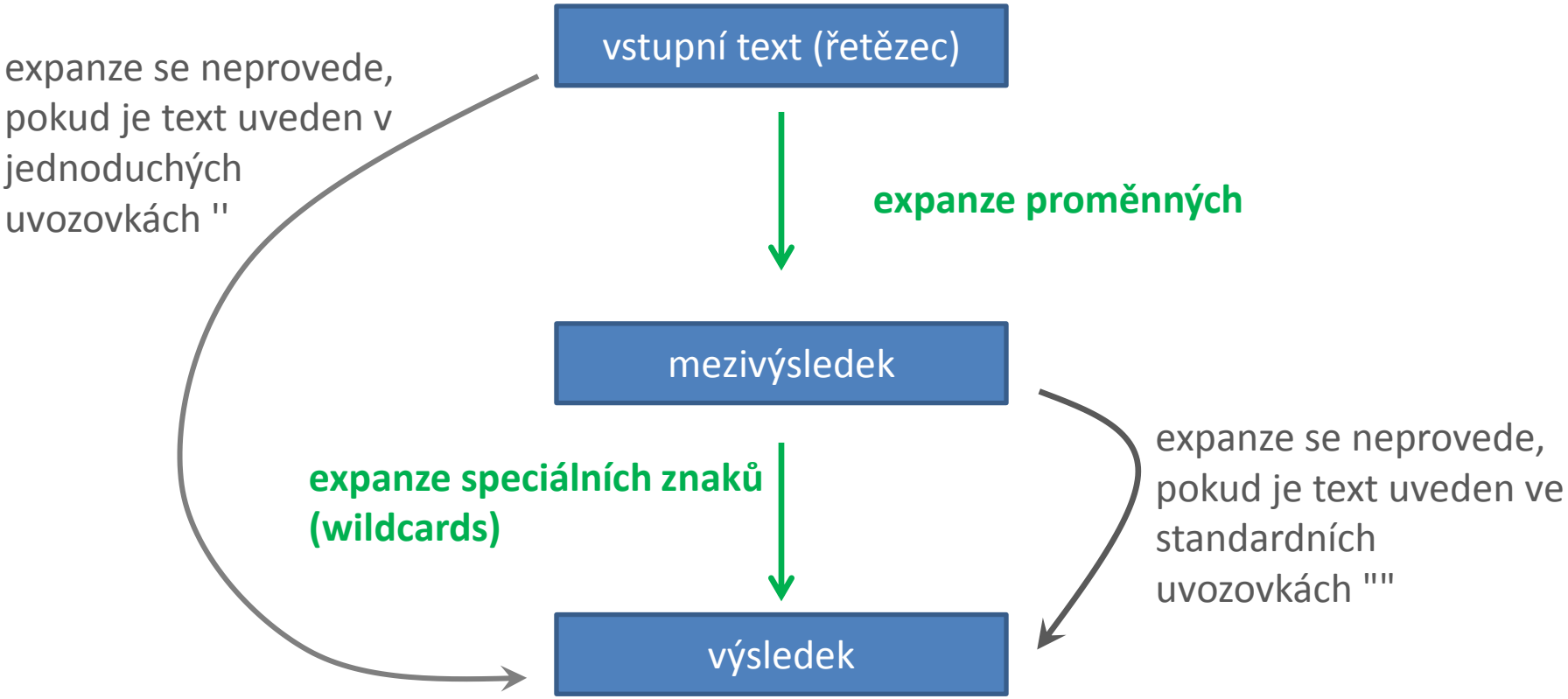
- **s obracenými jednoduchými uvozovkami (obrácený apostrof)**

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

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

Proměnné a speciálních znaky

Pořadí expanze řetězce:



Příkazy ke cvičení

more	vypíše obsah souboru nebo standardního vstupu po stránkách (vhodný pro zobrazení dlouhých souborů nebo výstupů příkazů)
less	podobná funkce jako **more** nicméně poskytuje větší funkcionalitu (např. posun v textu oběma směry)
xargs	spustí program s argumenty, které načte ze standardního vstupu, vhodné pro předávání velkého seznamu argumentů
grep	vypíše řádky ze souborů nebo standardního vstupu, které vyhovují vyhledávacímu vzoru

Příklady:

```
$ set | more
```

vypíše seznam nastavených proměnných a funkcí po stránkách

```
$ cat *.txt | less
```

vypíše obsah všech souborů se zakončením .txt po stránkách

```
$ cat directory_list.txt | xargs mkdir
```

vytvoří adresáře jejichž jména jsou uvedena v souboru directory_list.txt

```
$ grep AHOJ soubor.txt
```

vypíše řádky ze souboru soubor.txt, které obsahují text AHOJ

Cvičení

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 19. Postupně ověřujte jejich hodnotu příkazy **set** a **echo**. Analyzujte případné rozpory.
11. Vytvořte soubor **adresare.txt,** který bude obsahovat na každém řádku zvlášť slova **pokus1**, **pokus2**, **pokus3**. Použijte příkaz **xargs** k vytvoření adresářů, které jsou v tomto souboru uvedeny.