C2110 *Operační systém UNIX a základy programování*

7. lekce / modul 1

PS/2020 Distanční forma výuky: Rev3

Petr Kulhánek

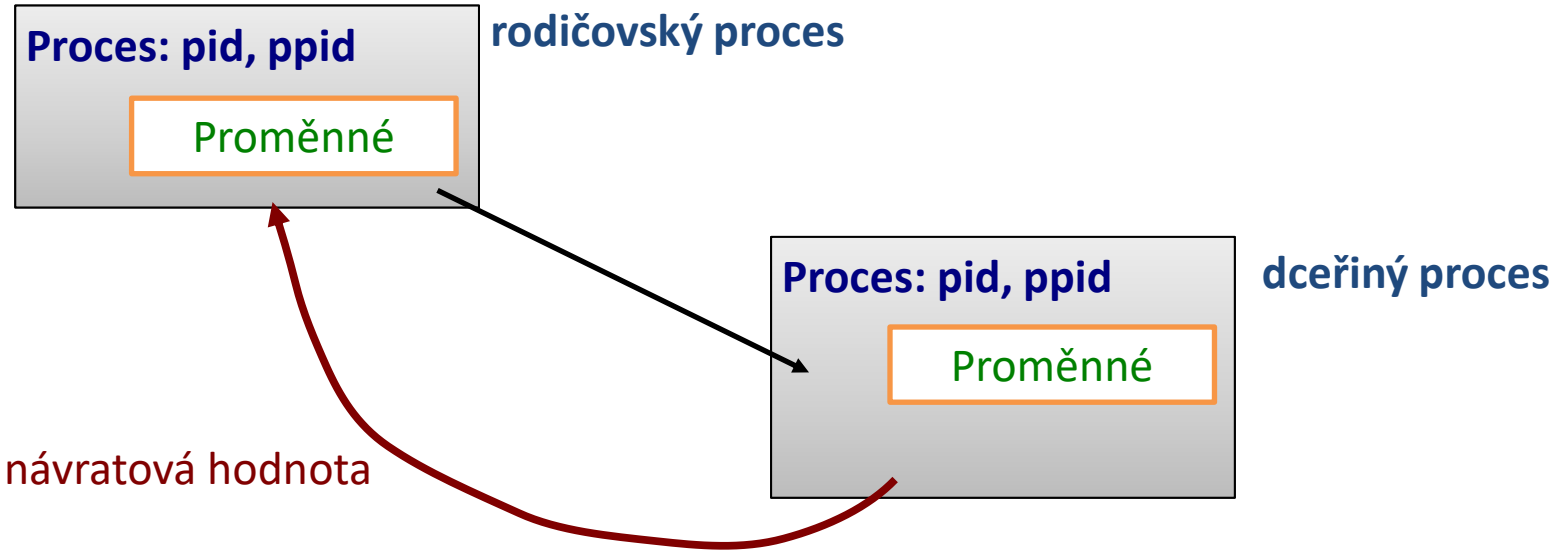
kulhanek@chemi.muni.cz

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

Rozhodování

I. Návratová hodnota procesu

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 (pravda)

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

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 exit

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

```
#!/bin/bash
if test "$1" -lt 0; then
      echo "Cislo je mensi nez nula!"
      exit 1
fi
echo "Cislo je vetsi nebo rovno nule."
exit 0
```

```
$ ./muj_skript 5
Cislo je vetsi nebo rovno nule.
$ echo $?
0
```

```
$ ./muj_skript -10
Cislo je mensi nez nula!
$ echo $?
1
```

II. Příkaz test

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).

Binární operátory (vyžadující dva argumenty):

```
test argument1 operator argument2
```

musí být mezery

Preferovaný „alternativní“ zápis:

```
[[ argument1 operator argument2 ]]
```

musí být mezery

Unární operátory (vyžadující jeden argumenty):

```
test operator argument1
```

musí být mezery

Preferovaný „alternativní“ zápis:

```
[[ operator argument1 ]]
```

musí být mezery

Příkaz test, celá čísla

Porovnávání celých čísel:

```
[[ 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)

Příklady:

```
[[ I -eq 5 ]]
```
je hodnota proměnné I rovna 5?

```
[[ J -le K ]]
```
je hodnota proměnné K menší nebo rovna hodnotě proměnné K?

při použití [[…]] a operátorů porovnávající celá čísla není nutné použít operátor $ pro získaní hodnoty proměnné

Příkaz test, řetězce

Porovnávání řetězců

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

Operátor :

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

Příklady:

```
[[ $A == "ahoj" ]]
```

obsahuje proměnná A řetězec "ahoj"?

```
[[ $J != $K ]]
```

obsahuje proměnná K různý řetězec než proměnná K?

Pro získání hodnoty proměnné je NUTNÉ použít operátor $.

Příkaz test, řetězce II

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**

Příklady:

```
[[ -n $I ]]
```

obsahuje proměnná I hodnotu?

```
[[ -f $K ]]
```

obsahuje proměnná K název existujícího souboru?

Pro získání hodnoty proměnné je NUTNÉ použít operátor $.

Manipulace s názvy souborů/adresářů

Příkazy:

basename - vytiskne název souboru, eventuálně odstraní příponu z názvu
dirname - vytiskne název adresáře

Příkazy pracují s prostým textem, názvy nemusí odkazovat na existující soubory.

Příklady:

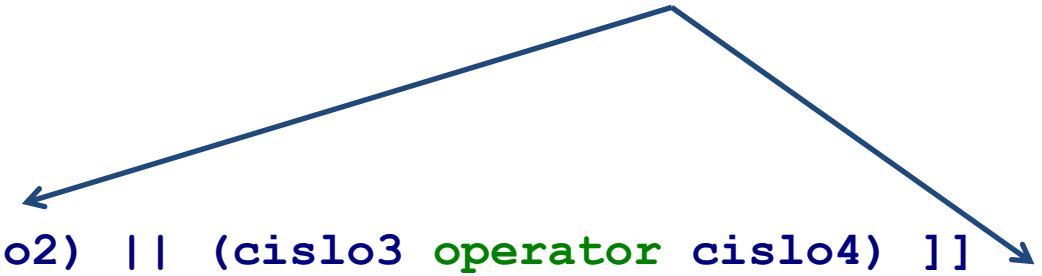
`basename pokus.txt .txt`	vypíše "pokus"
`basename adresar/pokus.txt`	vypíše "pokus.txt"
`NAME=`basename "$FILE" .doc``	do proměnné NAME vloží název souboru bez přípony .doc z proměnné FILE
`dirname adresar/pokus.txt`	vypíše "adresar"
`DIR=`dirname "$FILE"``	do proměnné DIR vloží název adresáře z proměnné FILE

Příkaz test, logické operátory

Logické operátory:

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

stejný výsledek, jiný způsob interpretace!



Příklady:

```
[[ (cislo1 operator cislo2) || (cislo3 operator cislo4) ]]
[[ (cislo1 operator cislo2) ]] || [[ (cislo3 operator cislo4) ]]
```

nedoporučuji používat

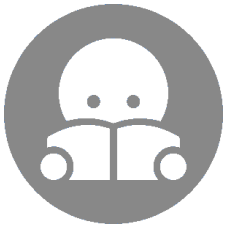
- Pomocí logických operátorů, lze vytvářet složitější podmínky.
- Pokud neznáme prioritu operátorů nebo si nejsme jisti, tak používáme kulaté závorky.
- Bash používá **líné vyhodnocování** podmínek, které spočívá ve vyhodnocování pouze té části složené logické podmínky, kterou je nutné vyhodnotit pro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

Líné vyhodnocování

`[[ výraz1 || výraz2 ]]` ~~**`<-> [[ vyraz1 ]] || [[ vyraz2 ]]`**~~

```
F || F = F
F || T = T
T || F = T
T || T = T
```

Pokud je první výraz pravda (**T**), tak je výsledek vždy pravda. Proto se výraz2 vyhodnocuje jen tehdy, pokud není první výraz pravda.



Trik:

```
mkdir adresar || exit 1
```

pokud příkaz mkdir selže (**F**), zavolá se příkaz exit a skript se ukončí

`[[ výraz1 && výraz2 ]]` ~~**`<-> [[ vyraz1 ]] && [[ vyraz2 ]]`**~~

```
F && F = F
F && T = F
T && F = F
T && T = T
```

Pokud je první výraz nepravda (**F**), tak je výsledek vždy nepravda. Proto se výraz2 vyhodnocuje jen tehdy, pokud je první výraz pravda.

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 ruzná od nuly?
[[ $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?
```

[[...]], test, [...]

```
[[ (I -ge 5) && (I -le 10) ]]
```

preferovaný zápis

```
test $I -ge 5 && test $I -le 10
[ ($I -ge 5) && ($I -le 10) ]
```

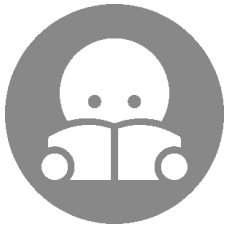
vyžaduje komplikovanější zápis, použití operátoru $ a eventuálně uvozovek

```
[[ -f $A ]]
```

preferovaný zápis

```
test -f "$A"
[ -f "$A" ]
```

vyžaduje komplikovanější zápis, použití operátoru $ a eventuálně uvozovek

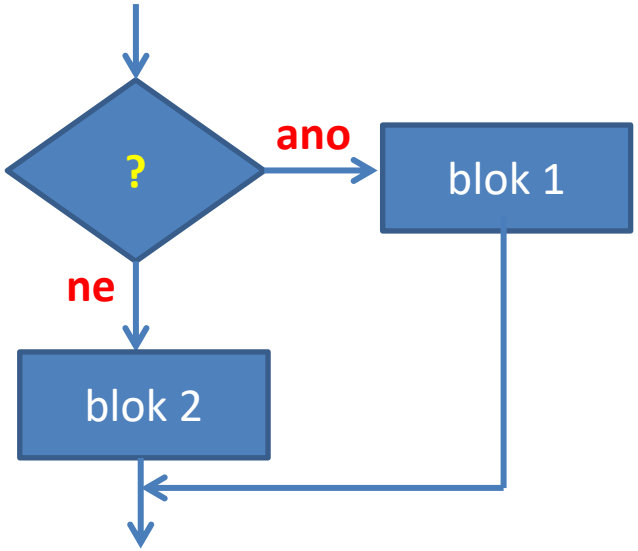


Podrobnosti:

- man test
- man bash (CONDITIONAL EXPRESSIONS)

Podmínky

Podmíněné vykonání bloku



Podmínky

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

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

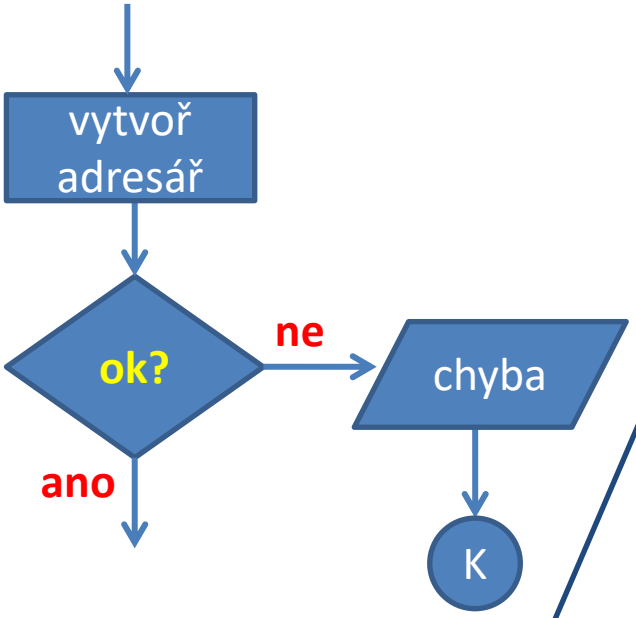
Pokud **prikaz1** skončí s návratovou hodnotou **0**, vykoná se **prikaz2**. V opačném případě se vykoná **prikaz3**.

Kompaktní zápisy:

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

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

Praktický příklad - podmínka



funkčně identické zápisy

```
mkdir adresar 2> /dev/null
if [[ $? -ne 0 ]]; then
      echo "Nemohu vytvorit adresar!"
      exit 1
fi
```

```
mkdir adresar 2> /dev/null
if test $? -ne 0; then
      echo "Nemohu vytvorit adresar!"
      exit 1
fi
```

```
if ! mkdir adresar 2> /dev/null; then
      echo "Nemohu vytvorit adresar!"
      exit 1
fi
```

Cvičení I

1. Vyzkoušejte si příklady uvedené na předchozí straně. Existenci adresáře monitorujte příkazem ls a měňte příkazy mkdir a rmdir.
2. Napište skript, který vypíše výsledek podílu dvou čísel. Hodnoty uživatel zadá interaktivně po spuštění skriptu. Skript ošetří možné dělení nulou.
3. Napište skript, který se dotáže na jméno souboru. Skript vypíše chybové oznámení, pokud soubor neexistuje. V opačném případě jej vypíše do standardního výstupu.