

# ***C2110 Operační systém UNIX a základy programování***

## **8. lekce**

**bash – řídicí struktury (podmínky, cykly)**

**Petr Kulhánek**

[kulhanek@chemi.muni.cz](mailto:kulhanek@chemi.muni.cz)

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

## ➤ Blok rozhodování

- podmínky, cykly

## ➤ Návratová hodnota procesu

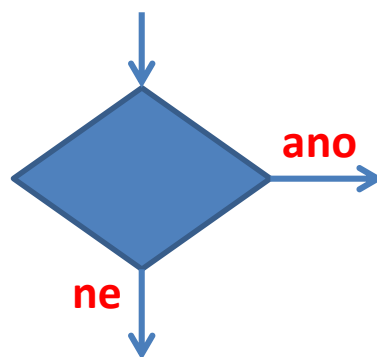
- příkaz exit

## ➤ Příkaz test

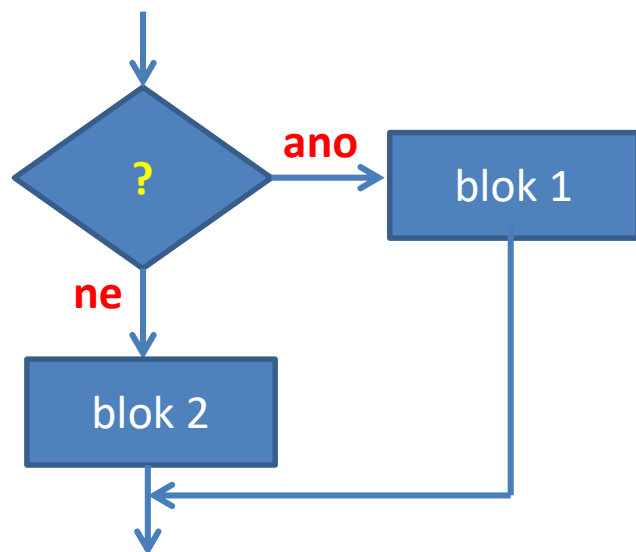
- operátory porovnání, logické operátory, zjednodušený zápis

# Blok rozhodování

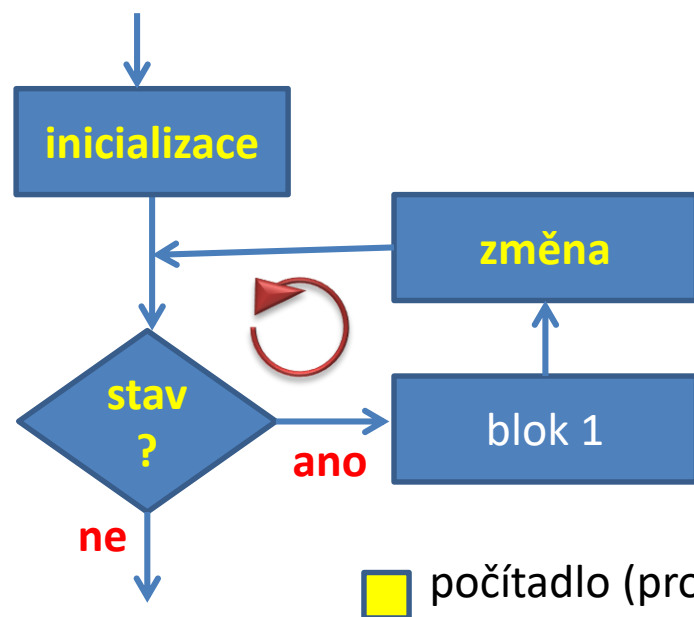
Typické použití bloku rozhodování



Podmíněné vykonání bloku (podmínky)



Cyklické vykonávání bloku (cykly)



■ počítadlo (proměnná)

# Podmínky

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

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

# Cyklus pomocí while/until

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

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

cyklus probíhá **zatímco** prikaz1 **vrací** v návratové hodnotě 0 (bez chyby)

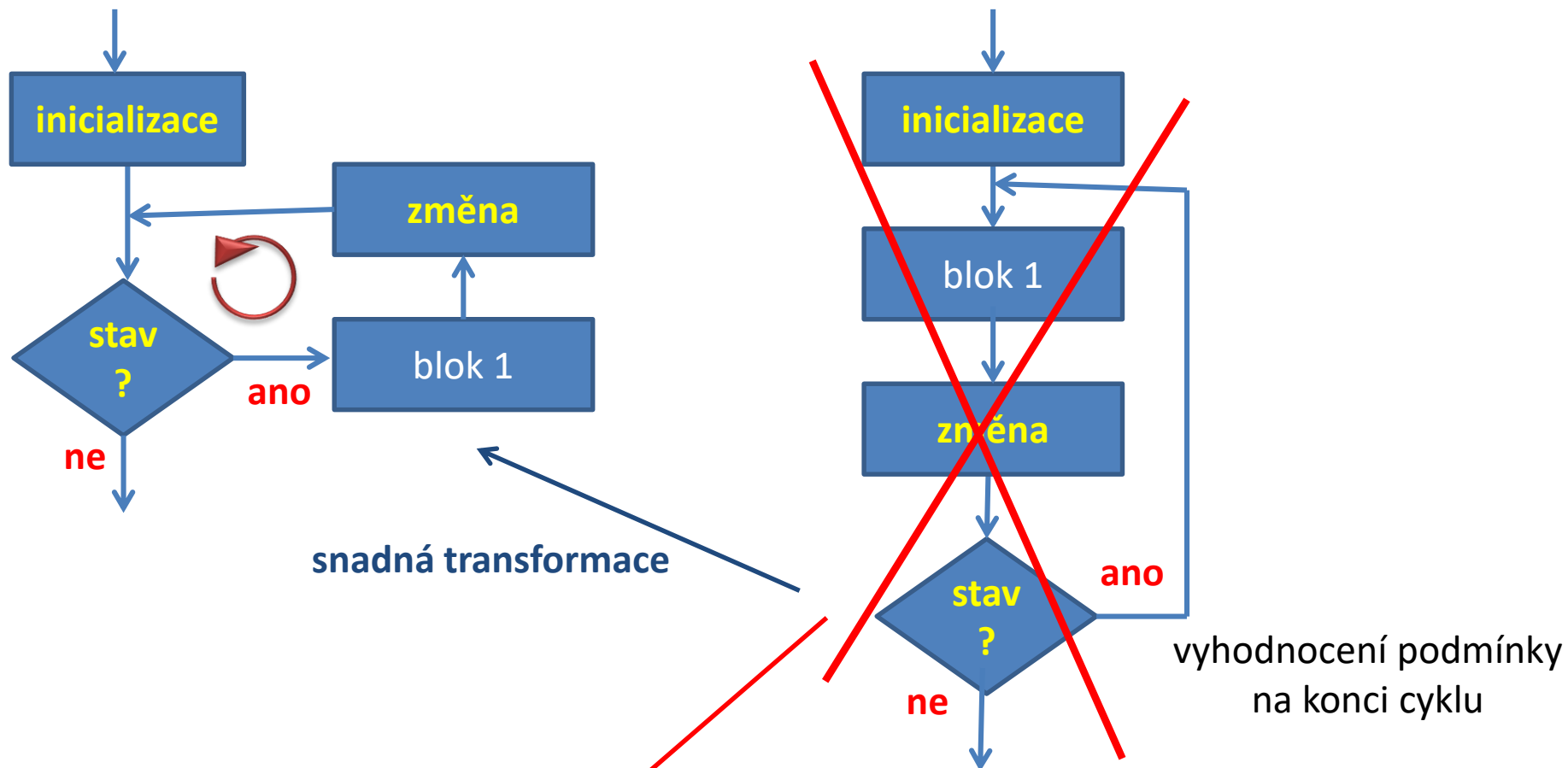
Kompaktní zápis:

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

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

```
until prikaz1; do
    prikaz2
    ...
done
```

# Cyklus pomocí while/until ...

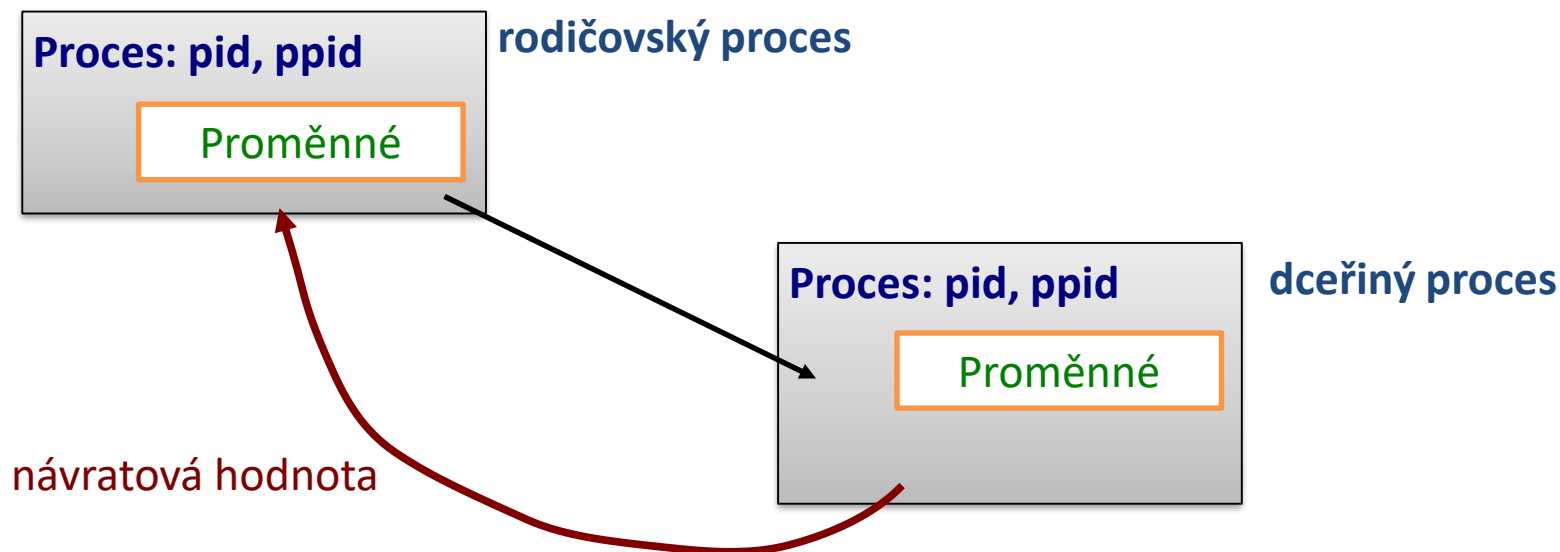


**komplikovaná implementace**

Tento algoritmus nemá přímou podporu v řídicích strukturách jazyka bash, jeho přepis je možný, ale za cenu horší čitelnosti výsledného kódu.

# 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

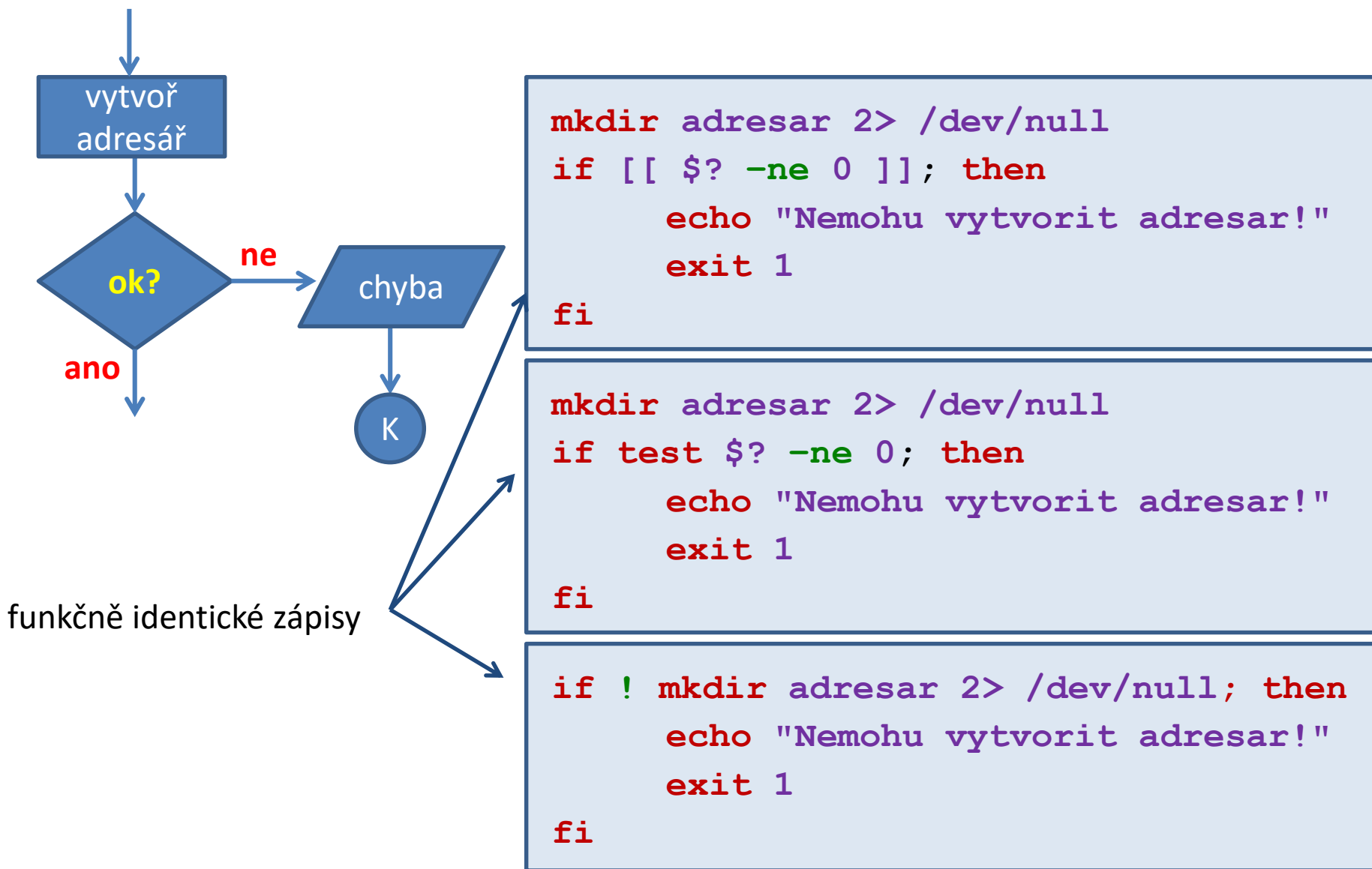
Příkaz **exit** slouží k ukončení běhu skriptu nebo interaktivního sezení. Nepovinným argumentem příkazu je **návratová hodnota**.

```
#!/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
```

# Praktický příklad - podmínka



# Příkaz test, operátory porovnávání

Příkaz **test** slouží k porovnávání hodnot a testování typů souborů a adresářů (man bash, man test). V případě, že je test splněn, je návratová hodnota příkazu nastavena na 0 (pravda).

## Porovnávání celých čísel:

```
test číslo1 operator číslo2
```

## 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)

## Alternativní zápis:

```
[[ číslo1 operator číslo2 ]]
```

musí být mezery



# 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) ]]
```

- 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 ]]`

<b>F</b>	<b>  </b>	<b>F = F</b>
<b>F</b>	<b>  </b>	<b>T = T</b>
<b>T</b>	<b>  </b>	<b>F = T</b>
<b>T</b>	<b>  </b>	<b>T = T</b>

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

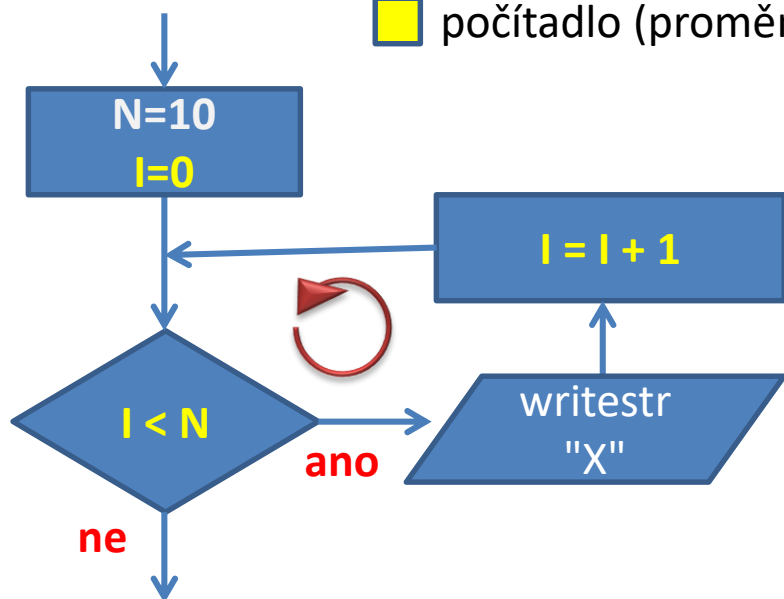
`[[ výraz1 && výraz2 ]] <-> [[ vyraz1 ]] && [[ vyraz2 ]]`

<b>F</b>	<b>&amp;&amp;</b>	<b>F = F</b>
<b>F</b>	<b>&amp;&amp;</b>	<b>T = F</b>
<b>T</b>	<b>&amp;&amp;</b>	<b>F = F</b>
<b>T</b>	<b>&amp;&amp;</b>	<b>T = T</b>

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

# Praktický příklad - cyklus

■ počítadlo (proměnná)



nutno použít \$

```
N=10
I=0
while test "$I" -lt "$N"; do
  echo "X"
  ((I = I + 1))
done
```

```
N=10
I=0
while [[ I -lt N ]]; do
  echo "X"
  ((I = I + 1))
done
```

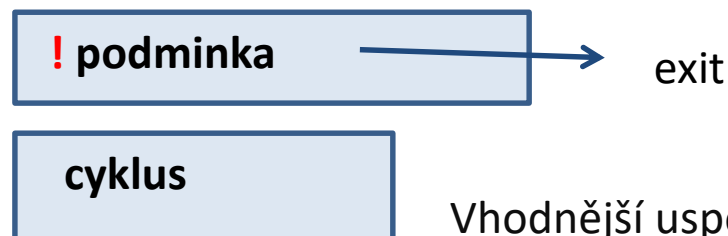
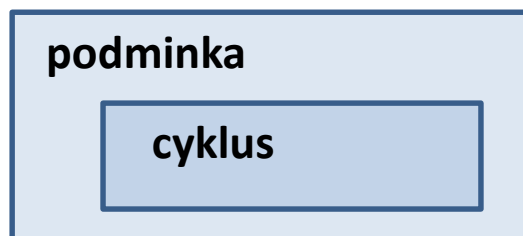
```
N=10
I=0
while [[ "$I" -lt "$N" ]]; do
  echo "X"
  ((I = I + 1))
done
```

volitelné \$, pokud je použit blok [[ ]] nebo (( ))

# Složitější konstrukce - vnořování

Jazyk bash **nemá návěští a příkaz goto**, či jeho obdobu. Komplexnějších konstrukcí lze tedy dosáhnout jen zanořováním cyklů a podmínek vzájemně do sebe. Úroveň zanoření není omezena.

Při návrhu algoritmu/skriptu se však snažíme o zamezení zbytečného vnořování (převážně z důvodu snadnější orientace ve skriptu).

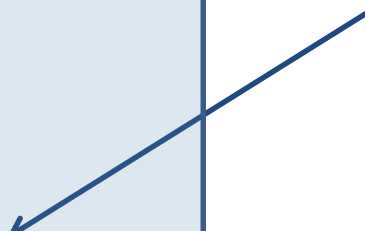


Vhodnější uspořádání např. pro testování vstupních dat od uživatelů.

# Vnořování cyklů - příklad

```
N=10
I=0
while [[ I -lt N ]]; do
    J=0
    while [[ J -lt I ]]; do
        echo -n "X"
        ((J = J + 1))
    done
    echo ""
    ((I = I + 1))
done
```

počítadlo vnějšího cyklu může ovlivňovat chování vnitřního cyklu



U zanořených konstrukcí dbáme na **odsazování textových bloků**, které zvyšuje **přehlednost a čitelnost kódu**. V textových editorech je integrována podpora, která odsazování usnadňuje, např. v editoru **gedit**, lze odsazení označeného textového bloku dosáhnout klávesou TAB či Shift+TAB.



# Cvičení I

1. Napište skripty v jazyce bash pro následující úlohy. Rozměr vykreslovaného obrazce nechť uživatel zadá interaktivně po spuštění skriptu.

# Úkol 1

Do terminálu vytiskněte čtverec se znaků **X**. Délku strany čtverce zadá uživatel.

```
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
```

To, že se nejedná vzhledově o čtverec, ignorujte. Počet znaků **X** na řádku a počet řádků však musí být stejný.

# Úkol 2

Do terminálu vytiskněte pravoúhlý trojúhelník se znaků **X**, tak aby jedna odvěsna byla umístěna nahoře a druhá na levé straně. Délku odvěsny zadá uživatel.

```
X X X X X X X X X X
X X X X X X X X X
X X X X X X X X
X X X X X X X
X X X X X X
X X X X X
X X X X
X X X
X X X
X X
X
```

# Úkol 3

Do terminálu vytiskněte pravoúhlý trojúhelník se znaků **X**, tak aby jedna odvěsna byla umístěna dole a druhá na levé straně. Délku odvěsny zadá uživatel.

```
X
X X
X X X
X X X X
X X X X X
X X X X X X
X X X X X X X
X X X X X X X X
X X X X X X X X X
X X X X X X X X X X
```

# Úkol 4

Do terminálu vytiskněte obrys čtverce se znaků **X**. Délku strany čtverce zadá uživatel.

```
X X X X X X X X X X
X
X
X
X
X
X
X
X
X
X
X X X X X X X X X X
```

To, že se nejedná vzhledově o čtverec, ignorujte. Počet znaků **X** na řádku a počet řádků však musí být stejný.

# Úkol 5

Do terminálu vytiskněte obrys čtverce a jeho uhlopříčky se znaků **X**. Délku strany čtverce zadá uživatel.

```
X X X X X X X X X X
X X                X X
X  X              X  X
X    X          X    X
X      X X      X
X      X X      X
X    X          X    X
X  X              X  X
X X                X X
X X X X X X X X X X
```

To, že se nejedná vzhledově o čtverec, ignorujte. Počet znaků **X** na řádku a počet řádků však musí být stejný.

# Domácí úkoly

---



# Domácí úkoly

## Pokyny:

1. Uvedené úkoly jsou pro **pokročilé studenty**.
2. **Cílem úkolů je rozvinout vaši schopnost řešit problémy, který jsou zdánlivě neřešitelné z pohledu možností a zdrojů, které máte k dispozici.** V případě jazyka bash se jedná převážně o možnost pracovat pouze s celočíselnou aritmetikou a omezený způsob vykreslování na terminál.

## Zadání:

1. Vykreslete kruh z písmen "X". Poloměr kruhu zadá uživatel po spuštění skriptu.
2. Vykreslete kružnici z písmen "X". Poloměr kružnice zadá uživatel jako první argument skriptu.