

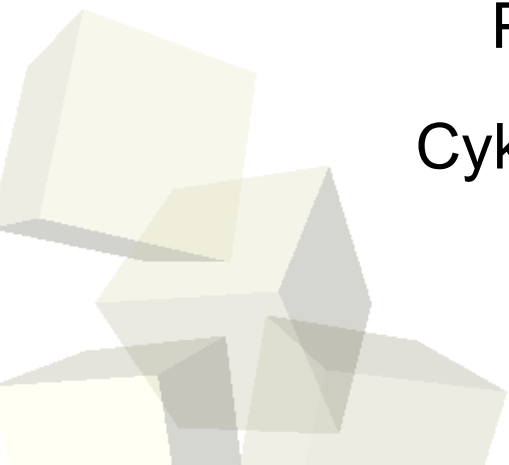


Přednáška 7

Celočíselná aritmetika. Návratový kód.

Příkazy pro větvení výpočtu.

Cykly. Předčasné ukončení cyklu.



Celočíselná aritmetika I

- **Příkaz** `expr` **výraz**

- Zašle na standardní výstup vyhodnocení výrazu uvedeného jako argumenty.
- Argumenty (operandy a operátory) musí být odděleny alespoň jednou mezerou.
- Pozor na kolizi se speciálními znaky shellu (musíme předřadit znak `\`).

Operátor	Význam	Příklad
<code>+</code>	sčítání	<code>N= `expr \$N1 + 3`</code>
<code>-</code>	odčítání	<code>N= `expr \$N1 - \$N2`</code>
<code>*</code>	násobení	<code>N= `expr 10 * 21`</code>
<code>/</code>	celočíselné dělení	<code>N= `expr \$N1 / \$N2`</code>
<code>%</code>	zbytek po celočíselné dělení	<code>N= `expr \$N1 % 5`</code>

Celočíselná aritmetika II

- **Výrazy se vyhodnocují podle priorit** (jako v matematice):
 - nejdříve výrazy v závorkách `\(\)`
 - potom operace `*`, `/`, `%` a nakonec operace `+` a `-`
- Operace se stejnou prioritou se vyhodnocují zleva doprava.

Příklady:

```
$ A=`expr 5 + 3 \* 2`
```

```
$ echo $A
```

```
11
```

```
$ A=`expr \( 5 + 3 \) \* 2`
```

```
$ echo $A
```

```
16
```

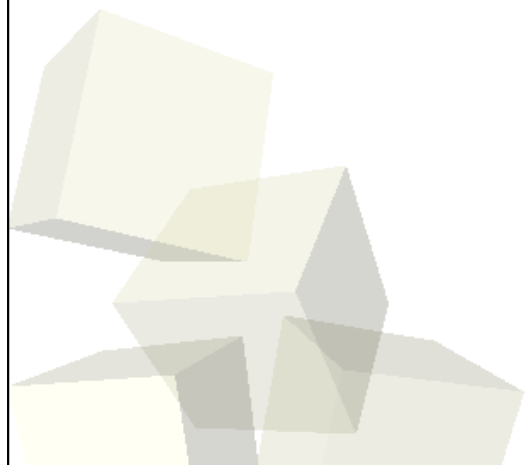
Celočíselná aritmetika III

- **Příkaz** `let výraz nebo ((výraz)) (neumí sh)`
 - Operandy a operátory ve výrazu se nemusí oddělovat mezerami.
 - Nehrozí kolize se speciálními znaky shellu.
 - Proměnné ve výrazu se automaticky nahrazují hodnotou (netřeba používat znak \$).

Operátor	Význam	Příklad
+	sčítání	<code>((N = N1 + 3))</code>
-	odčítání	<code>((N = N1 - N2))</code>
*	násobení	<code>((N = 10 * 21))</code>
/	celočíselné dělení	<code>((N = N1 / N2))</code>
%	zbytek po celočíselné dělení	<code>((N = N1 % 5))</code>
#	základ soustavy	<code>((N=2#1011))</code>
<<	bitový posun doleva	<code>((N= 2#1011 << 3))</code>
>>	bitový posun doprava	<code>((N= 2#1011 >> 3))</code>

Celočíselná aritmetika IV

Operátor	Význam	Příklad
&	bitový AND	$((N = 2\#1011 \ \& \ 2\#1101))$
	bitový OR	$((N = 2\#1011 \ \ 2\#1101))$
^	bitový XOR	$((N = 2\#1011 \ ^ \ 2\#1101))$



Návratový kód

- Každý proces při ukončení vrací návratový kód.
- **Návratový kód = přirozené číslo 0, 1, ... ,255**
 - 0 úspěšné ukončení procesu
 - 1,...,255 chyba při běhu procesu, např.
 - 1 program byl spuštěn, ale proces nebyl úspěšný
 - 2 program nepracoval
 - 127 program nebyl nalezen
- **Návratový kód posledního příkazu** spuštěného na popředí je **uložen v proměnné \$?**

- **Skript shellu lze ukončit s návratový kódem n** pomocí příkazu

```
exit [n]
```

- **Funkce shellu lze ukončit s návratový kódem n** pomocí příkazu

```
return [n]
```

Příklady

```
$ grep 'root' /etc/passwd
```

```
root:x:0:1:Super-User:/root:/sbin/sh
```

```
$ echo $?
```

```
0
```

```
$ grep 'XXX' /etc/passwd
```

```
$ echo $?
```

```
1
```

```
$ grep 'root' /XXX
```

```
grep: can't open /XXX
```

```
$ echo $?
```

```
2
```

```
$ XXXgrep 'root' /etc/passwd
```

```
-bash: XXXgrep: command not found
```

```
$ echo $?
```

```
127
```

Příkaz test

- **Vrací návratový kód na základě vyhodnocení výrazu** uvedeného jako jeho parametry.
- Parametry jsou analyzovány shellem, proto
 - musí být odděleny aspoň jednou mezerou
 - u speciálních znaků shellu musí předcházet znak \

- **Složené výrazy**

<code>\ (v1 \)</code>	přednostní vyhodnocení podvýrazu v1
<code>v1 -a v2</code>	logický součin (je true pokud v1 i v2 jsou true)
<code>v1 -o v2</code>	logický součet (je true pokud aspoň jeden podvýraz je true)
<code>! v1</code>	logická negace (je true, pokud je v1 false)

Synonyma příkazu test

[výraz]

- je synonymem příkazu `test výraz`

[[výraz]]

- jen u ksh a bash, zabudovaný příkaz, rozšířená varianta
 - a je nahrazeno `&&`
 - o je nahrazeno `||`

((číselný výraz))

- jen u ksh a bash,
 - true, pokud hodnota výrazu je různá od nuly
 - na místě výrazu lze použít i přiřazovací příkaz
 - numerickým proměnným se nepředřazuje znak `$`
 - operandy se nemusí oddělovat mezerami
-
- V ksh i bash je příkaz `test` zabudovaným příkazem (tzn. rychlejší).

Určování atributů souborů

Přepínač	Příklad	Význam
-f	[-f soubor]	soubor existuje a je obyčejným souborem?
-d	[-d soubor]	soubor r existuje a je adresářem?
-s	[-s soubor]	soubor existuje a není prázdný?
-e	[-e soubor]	soubor existuje? (neumí sh)
-L	[-L soubor]	soubor existuje a je symbolickým linkem? (neumí sh)
-r	[-r soubor]	soubor existuje a má nastaveno právo r?
-w	[-w soubor]	soubor existuje a má nastaveno právo w?
-x	[-x soubor]	soubor existuje a má nastaveno právo x?

- Další podrobnosti lze najít pomocí: **man sh**, **man ksh**, **man bash**.



Příklady

```
$ test -f /etc/passwd ; echo $?
```

```
0
```

```
$ [ -f /etc/passwd ] ; echo $?
```

```
0
```

```
$ test -d /etc/passwd ; echo $?
```

```
1
```

```
$ [ -d /etc/passwd ] ; echo $?
```

```
1
```

```
$ test -f /etc/passwd -a -r /etc/passwd ; echo $?
```

```
0
```

```
$ [ -f /etc/passwd -a -r /etc/passwd ] ; echo $?
```

```
0
```



Příklady

```
$ P="/etc/group"
```

```
$ [ -r $P ] ; echo $?
```

```
0
```

```
$ [ -r $P ] && echo "soubor $P je citelny"
```

```
soubor /etc/group je citelny
```

```
$ P="/etc/shadow"
```

```
$ [ -r $P ] ; echo $?
```

```
1
```

```
$ [ -r $P ] || echo "soubor $P není citelny"
```

```
soubor /etc/shadow není citelny
```

```
$ ! [ -r $P ] && echo "soubor $P není citelny"
```

```
soubor /etc/shadow není citelny
```

Porovnávání řetězců

Test	Význam
[<code>r1 = r2</code>]	Řetězce <code>r1</code> a <code>r2</code> jsou stejné?
[<code>r1 != r2</code>]	Řetězce <code>r1</code> a <code>r2</code> jsou různé?
[<code>r1 < r2</code>]	Je řetězec <code>r1</code> v abecedě před řetězcem <code>r2</code> ?
[<code>r1 > r2</code>]	Je řetězec <code>r1</code> v abecedě za řetězcem <code>r2</code> ?
[<code>-z r1</code>]	Je řetězec <code>r1</code> prázdný?
[<code>-n r1</code>]	Není řetězec <code>r1</code> prázdný?

- Pozor!
 - Použije-li se jako operand relace nebo unární operace proměnná, je vhodné ji uvést v uvozovkách ("). Jinak dojde, v případě že hodnota proměnné je prázdný řetězec, k syntaktické chybě.



Příklady

```
$ test "Jana" \< "Petr" ; echo $?
```

0

```
$ [ "Jana" \< "Petr" ] ; echo $?
```

0

```
$ A=Ales ; B=Jiri ; C="Dobry den"
```

```
$ test $A \< $B ; echo $?
```

0

```
$ test $A = $B ; echo $?
```

1

```
$ test $B \< $C ; echo $?
```

-bash: test: too many arguments

2

```
$ test "$B" \< "$C" ; echo $?
```

1

Porovnávání čísel

Test	Význam
[n1 -eq n2]	Číslo n1 je rovno číslu n2 ?
[n1 -ne n2]	Číslo n1 není rovno číslu n2 ?
[n1 -lt n2]	Číslo n1 je menší než číslo n2 ?
[n1 -gt n2]	Číslo n1 je větší než číslo n2 ?
[n1 -le n2]	Číslo n1 je menší nebo rovno číslu n2 ?
[n1 -ge n2]	Číslo n1 je větší nebo rovno číslu n2 ?



Příklady

```
$ test 2 -lt 7 ; echo $?
```

```
0
```

```
$ [ 2 -lt 7 ] ; echo $?
```

```
0
```

```
$ test 2 -gt 7 ; echo $?
```

```
1
```

```
$ [ 2 -gt 7 ] ; echo $?
```

```
1
```

```
$ A=10 ; B=7
```

```
$ test $A -eq $B || echo "$A není rovno $B"
```

```
10 není rovno 7
```

```
$ [ $A -gt $B ] && echo "$A > $B"
```

```
10 > 7
```


Příkazy

- **Jednoduché příkazy**

- **vnitřní (vestavěné) příkazy shellu:**

- např. `kill`, `echo`, `.`, `:`, `true`, `false`, `exit`, `return`, `continue`,....

- **vnější příkazy (binární soubory nebo skripty):**

- např. `/bin/l`s, `/etc/rc3.d/S50apache`

- **Složené příkazy**

- Pokud není uvedeno jinak, návratový kód složeného výrazu je návratový kód posledního provedeného jednoduchého příkazu.

- **Větvení výpočtu**

- podmíněný příkaz `if`
- příkaz výběru `case`

- **Cykly**

- `while`, `until`, `for`

- **Funkce shellu**

Příkaz if

Syntaxe

```
if seznam_příkazů_1
then seznam_příkazů_2
[   elif seznam_příkazů_3
      then seznam_příkazů_4   ]
...
[ else seznam_příkazů_5   ]
fi
```

Popis

- Pokud `seznam_příkazů_1` vrátí nulový návratový kód, je prováděn `seznam_příkazů_2` jinak, pokud `seznam_příkazů_3` vrátí nulový návratový kód, je prováděn `seznam_příkazů_4` jinak

...

Příklady

- **Skript if.sh:**

```
#!/bin/sh

if [ $# -ne 1 ] ; then
    echo "Volani: $0 cislo_navratoveho_kodu"
    exit 2
fi

exit $1
```

- **Spuštění:**

```
$ chmod 755 if.sh
```

```
$ ./if.sh
```

```
Volani: ./if.sh cislo_navratoveho_kodu
```

```
$ ./if.sh 65
```

```
$ echo $?
```

```
65
```



Příklady

```
$ grep root /etc/passwd  
root:x:0:1:Super-User:/root:/sbin/sh
```

```
$ echo $?
```

```
0
```

```
$ grep root /etc/passwd > /dev/null
```

```
$ echo $?
```

```
0
```

```
$ if grep root /etc/passwd > /dev/null
```

```
> then
```

```
> echo "ucet root existuje"
```

```
> fi
```

```
ucet root existuje
```

Varianty k příkazu if

```
seznam_příkazů_1 && seznam_příkazů_2
```

je ekvivalentní:

```
if seznam_příkazů_1 ; then
    seznam_příkazů_2
fi
```

```
seznam_příkazů_1 || seznam_příkazů_2
```

je ekvivalentní:

```
if seznam_příkazů_1 ; then :
else
    seznam_příkazů_2
fi
```

Příklady

```
$ grep XYZ soubor && lp soubor
```

```
$ grep XYZ soubor || echo "XYZ nenalezeno"
```

Příkaz case

Syntaxe

```
case hodnota in
    vzor_1) seznam_příkazů_1 ;;
    ...
    vzor_n) seznam_příkazů_n ;;
esac
```

Popis

- **hodnota** je postupně porovnávána se vzory (podle pravidel náhrad jmen souborů, nikoliv regulárních výrazů).
- Pokud dojde ke shodě, je proveden seznam příkazů uvedených za vzorem a tím příkaz **case** končí.
- Vzor může být tvořen více variantami oddělenými znakem |.
- Bývá zvykem uvádět jako poslední vzor *) a za něj příkazy provedené v případě selhání ostatních vzorů.

Příklady

```
#!/sbin/sh

case "$1" in
'start')
    [ -x /usr/lib/lpsched ] && /usr/lib/lpsched
    ;;

'stop')
    [ -x /usr/lib/lpshut ] && /usr/lib/lpshut
    ;;

*)
    echo "Usage: $0 { start | stop }"
    exit 1
    ;;
esac
```

Příklady

- **Skript case.sh:**

```
#!/bin/sh

case "$1" in
    -[tT])
        echo "Parametr -t nebo -T"
        ;;
    yes|no)
        echo "Parametr yes nebo no"
        ;;
    *)
        echo "Neznany parametr."
        exit 1
        ;;
esac
```


Cyklus while

Syntaxe

```
while seznam_příkazů_1
do
    seznam_příkazů_2
done
```

Popis

- Dokud `seznam_příkazů_1` vrací nulový návratový kód, je prováděn `seznam_příkazů_2`.
- Cyklus je možno předčasně ukončit příkazem **break**.
- Předčasný návrat na začátek příkazu lze příkazem **continue**.

Příklady

- **Skript while1.sh:**

```
#!/bin/sh

MAX=5
I=1

while [ $I -le 10 ]
do
    echo "Hodnota I je $I"
    I=`expr $I + 1`
done
```

Příklady

- **Skript while2.sh:**

```
#!/bin/sh

while :
do
  /bin/echo "Zadej cele cislo [0,...99][k=konec]: \c"
  read C
  case $C in
    k)
      break
      ;;
    [0-9]|[0-9][0-9] )
      echo "Druha mocnina cisla $C je `expr $C \* $C`."
      ;;
    *) echo "Spatny parametr."
  esac
done
```

Cyklus until

Syntaxe

```
until seznam_příkazů_1
do
    seznam_příkazů_2
done
```

Popis

- Obdoba příkazu **while** s obrácenou podmínkou.
- Dokud **seznam_příkazů_1** vrací nenulový návratový kód, je prováděn **seznam_příkazů_2**.
- Příkazy **break** a **continue** lze použít jako u cyklu **while**.

Cyklus for

Syntaxe

```
for proměnná in seznam_argumentů
do
    seznam_příkazů
done
```

Popis

- Cyklus **for** proběhne tolikrát, kolik má **seznam_argumentů** elementů.
- V každém cyklu nabývá **proměnná** hodnotu jednoho elementu ze **seznam_argumentů**.
- Příkazy **break** a **continue** lze použít jako u cyklu **while**.

Příklad

- **Skript for.sh:**

```
#!/bin/sh

I=1

for E in Petr Jana Jiri Karel
do
    echo "Element $I je $E."
    I=`expr $I + 1`
done
```