Přednáška 8

Numerické výpočty, komprese a archivace, návratový kód.

Katedra číslicových systémů FIT, České vysoké učení technické v Praze

Příprava studijního programu Informatika je podporována projektem financovaným z Evropského sociálního fondu a rozpočtu hlavního města Prahy.
Praha & EU: Investujeme do vaší budoucnosti

Celočíselná aritmetika

- **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
`+`	sčítání	`N=`expr $N1 + 3``
`-`	odčítání	`N=`expr $N1 - $N2``
`*`	násobení	`N=`expr 10 * 21``
`/`	celočíselné dělení	`N=`expr $N1 / $N2``
`%`	zbytek po celočíselné dělení	`N=`expr $N1 % 5``

Celočíselná aritmetika

- **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

- **Interní příkaz shellu `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	`((N = N1 + 3))`
`-`	odčítání	`((N = N1 - N2))`
`*`	násobení	`((N = 10 * 21))`
`/`	celočíselné dělení	`((N = N1 / N2))`
`%`	zbytek po celočíselné dělení	`((N = N1 % 5))`
`#`	základ soustavy	`((N=2#1011))`
`<<`	bitový posun doleva	`((N= 2#1011 << 3))`
`>>`	bitový posun doprava	`((N= 2#1011 >> 3))`

Celočíselná aritmetika

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

Reálná aritmetika

- **Příkaz `bc [-c] [-l] soubor]`**
  - preprocesor k příkazu **`dc`** (**`-c`** zobrazí pouze příkazy pro příkaz **`dc`**)
  - **`-l`** zajistí volání funkcí z matematické knihovny a **`scale`**=20
  - příkazy načítá ze zadaného souboru, jinak ze standardního vstupu

Operátor	Význam	Příklad
`+`	sčítání	`N=`echo "$N1 + $N2" | bc``
`-`	odčítání	`N=`echo "$N1 - $N2" | bc``
`*`	násobení	`N=`echo "$N1 * $N2" | bc``
`/`	celočíselné dělení	`N=`echo "$N1 / $N2" | bc``
`%`	zbytek po celočíselné dělení	`N=`echo "$N1 % $N2" | bc``
`^`	mocnina	`N=`echo "$N1 ^ $N2" | bc``
`< <= > >=`	menší, menší nebo rovno, …	
`== !=`	rovno, různé od	

Reálná aritmetika

Řídící slova	Význam	Příklad
`ibase`	základ číselné soustavy na vstupu	`N=`echo "ibase=16; A + B" | bc``
`obase`	základ číselné soustav na výstupu	`N=`echo "obase=2; 5 + 2" | bc``
`scale`	počet desetinných míst na výstupu (default 0)	`N=`echo "scale 5; 10 / 3" | bc``

Identifikátor	Význam	Příklad
`x`	Proměnná (malé písmeno becedy)	`N=`echo "a=5;b=2; a + b" | bc``
`x[i]`	`i`-tý element pole `x`	`N=`echo "a[1]=3; a[1]+1" | bc``
`x(y,z)`	volání funkce `x` s parametry `y` a `z`	`N=`echo “length(3.1415)" | bc``

Reálná aritmetika

Funkce	Význam	Příklad
`sqrt(x)`	druhá odmocnina	`N=`echo "sqrt($A)" | bc``
`l(x)`	přirozený logaritmus	`N=`echo "l($A)" | bc``
`e(x)`	e^x	`N=`echo "e($A)" | bc``
`s(x)`	sin(x)	
`c(x)`	cos(x)	
`length(x)`	počet cifer čísla x	

- **Příkaz `awk/nawk`**

Archivace a komprese

- **Archivace adresářů/souborů**
  - zabalení adresářové struktury (jednotlivých souborů) do jednoho souboru (archivu)
- **Komprese (komprimace) souborů**
  - zmenšení objemu originálního souboru na základě odstranění redundantní informace z originálního souboru
  - jedná o bezeztrátovou kompresi
- **Použití**
  - přenos dat
  - zálohování (úplná záloha, přírůstková záloha!!!)
- **Problémy**
  - relativní/absolutní cesta
  - atributy souborů (vlastník, čas modifikace, přístupu,…)
  - hard linky
  - symbolické linky

Archivace

- **Příkaz `tar`** (Tape ARchive)
- Vytvoření archivu (**c**reate), používá se přípona .tar

```
tar cvf archiv.tar adresáře/soubory
find . > list.txt ; tar cvf archiv.tar –I list.txt
```

- Přidání adresářů/souborů do archivu (**u**pdate)

```
tar uvf archiv.tar adresáře/soubory
```

- Prohlížení archivu (**t**est)

```
tar tvf archiv.tar
```

- Rozbalení archivu (**e**xtract)

```
tar xv[op]f archiv.tar
```

- Obnova jednotlivých souborů z archivu (**r**ecover)

```
tar rv[op]f archiv.tar soubory
```

Komprese

- **Příkazy `compress, uncompress, zcat`**
  - Komprese dat pomocí LZW (Lampel-Ziv-Welch kódování)
  - Komprimování (používá standardně příponu .Z)

```
compress soubor
cat soubor | compress > soubor.Z
tar cv soubory | compress > archiv.tar.Z
```

  - Dekomprimování

```
uncompress [-f] soubor.Z
zcat soubor.tar.Z | tar xvf -
```

Komprese

- **Příkazy `gzip, gunzip, gzcat`**
  - Koprese pomocí LZ77 (Lempel-Ziv kódování)
  - Komprimování (používá standardně příponu .gz)

```
gzip [-9] soubor
cat soubor | gzip > soubor.gz
tar cv soubory | gzip > archiv.tar.gz
```

  - Dekomprimování

```
gunzip soubor.gz
gzcat soubor.tar.gz | tar xvf -
```

Komprese

- **Příkazy `bzip2, bunzip2, bzcat`**
  - Koprese pomocí kombinace BWT (Burrows-Wheelerova transformace), MTF (Move-to-Front) transformace a Huffmanova kódování
  - Komprimování (používá standardně příponu .bz2)

```
bzip2 [-9] soubor
cat soubor | bzip2 > soubor.bz2
tar cv soubory | bzip2 > archiv.tar.bz2
```

  - Dekomprimování

```
bunzip soubor.bz2
bzcat soubor.tar.bz2 | tar xvf -
```

Archivace a komprese

- **Příkaz `zip, unzip`**
  - Kombinuje kompresi a archivaci do jednoho příkazu.
  - Používá formát vytvořený Philem Katzem (program PKZIP).

  - Vytvoření koprimovaného archivu (používá se standardně přípona .zip)

```
zip archiv.zip soubory
zip –r[9] archiv.zip adresáře
```

  - Zobrazení obsahu archivu

```
unzip –l archiv.zip
```

  - Rozbalení archivu

```
unzip archiv.zip [adresáře/soubory]
```

Archivace a komprese

- **Příkaz `jar`** (Java ARchive tool)
  - Založen na ZIP formátu a ZLIB.
  - Vznikl jako nástroj pro archivaci JAVA balíčků.
  - Syntaxe podobná příkazu tar. Používá se přípona .jar.

- Vytvoření archivu (**c**reate)

```
jar cvf archiv.jar adresáře/soubory
```

- Prohlížení archivu (**t**est)

```
jar tvf archiv.jar
```

- Rozbalení archivu (**e**xtract)

```
jar xv[op]f archiv.jar
```

Archivace a komprese

- **Příkaz `gtar`** (GNU tar)
- GNU implementace příkazu tar, který umí např. při archivaci zavolat příslušný komprimační program.

- Vytvoření komprimovaného archivu

```
gtar cvZf archiv soubory      (pomocí compress)
gtar cvzf archiv soubory      (pomocí gzip)
gtar cvjf archiv soubory      (pomocí bzip2)
```

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 a jeho synonyma

`test výraz`

- **Vrací návratový kód na základě vyhodnocení výrazu** uvedeného jako jeho parametry.

`[ 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).

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 neni citelny"
soubor /etc/shadow neni citelny

$ ! [ -r $P ] && echo "soubor $P neni citelny"
soubor /etc/shadow neni citelny
```

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 neni rovno $B"
10 neni rovno 7
$ [ $A -gt $B ] &&  echo "$A > $B"
10 > 7
```

Příklady – využití návratového kódu

```
#!/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 – využití návratového kódu

- **Skript while1.sh:**

```
#!/bin/sh

MAX=5
I=1

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