Rozšíření ksh vůči sh při práci s proměnnými

(X36UNX, Jan Skalický, 2006)

Pole

- **homogenní lineární struktura**
- **implicitně s číselným rozsahem indexů 0...1023 (někde 4095)**
- **implicitně řetězcově orientovaná hodnota (jako obyčejné proměnné)**

definice:

```
pole[index]=hodnota
```

hromadná definice:

```
set -A pole hodnota[0] hodnota[1] ...   // bash, ksh (dle verze)
pole=(hodnota[0] hodnota[1] ... )       // bash
```

čtení položky:

```
$pole[index]            // ŠPATNĚ (sémanticky)
${pole[index]}          // SPRÁVNĚ (viz echo nebo print v ksh)
$pole                   <=> ${pole[0]}
```

hromadné čtení:

```
${pole[*]}          // substituce najednou
${pole[@]}          // ...po částech (bez úvozovek eqivalentní)
“${pole[*]}”        // substituce do jednoho řetězce
“${pole[@]}”        // ...do více řetězců (pro parametrizaci)
${#parameter[*]}    // počet obsazených (definovaných) buněk
${#parameter}       // délka řetězce (netýká se přímo polí)
```

```
pole[0]=nula
pole[1]=jednicka
pole[2]=”dvojka a trojka”
echo ${pole[*]}         <=> nula jednicka dvojka a trojka
echo “${pole[*]}”       <=> “nula jednicka dvojka a trojka”
echo “${pole[@]}”       <=> “nula” “jednicka” “dvojka a trojka”
echo ${#pole[*]}        <=> 3
```

pozn.: je zvykem (nejen v ksh) psát globalní (exportované) proměnné velkými a lokální malými písmeny

Aritmetické výrazy

- **expandované podobně jako sekce v uvozovkách**
- **celočíselná aritmetika**
- **formát čísel nejen v dekadické soustavě**

syntaxe:

```
   $[vyraz]                  // bash
   $(( vyraz ))              // bash, ksh
   (( prirazeni ))           /* bash, ksh; syntaxe C, není třeba $
                                před identifikátory, nehrozí kolize s
                                metaznaky, může obsahovat mezery */
   let promenna=vyraz        // ksh; není třeba $, nesmí být mezera

   x=$(( x - 1 ))            <=>dekrementace x (bash, ksh)
   let x=x-1                 <=>dekrementace x (ksh)
```

Typy

- **proměnným v ksh lze při definici explicitně přiřadit typ**
- **typy mají vliv na interpretaci proměnné**

syntaxe:

```
typeset [ -HLRZfilrtux[n] ]  [ name[=value ]  ]          // ksh
```

L zarovnání (+ev. oříznutí) zleva (šířka n nebo 1. přiřazení)
R zarovnání zprava
Z doplnění nulami zleva
t name se vztahuje k definovaným funkcím
i integer (rychlejší manipulace), n udává použitou soustavu
l lower-case (vstupní konverze)
r readonly => konstanta
t tag (generické použití), u funkcí má význam trasování
u upper-case
x automaticky exportované symboly

pozn. + místo – za typeset má inverzní účinek

pozn. vynecháním name se zobrazí definované proměnné vyhovující přepínačům

pozn. Integer se dá definovat tokenem integer (je to alias na typeset -i)

pozn. Konstanta se dá definovat tokenem readonly (není to alias na typeset -r),
ev. S přepínačem -p vypíše jejich seznam

Specialitky

odřezávání prefixů a sufixů podle vzorů:
(vzor je vzor ve stylu shellu, nikoliv regexp)

```
${parameter#pattern}   // odříznutí minimálního výskytu zleva
${parameter##pattern}  // odříznutí maximálního výskytu zleva
${parameter%pattern}   // odříznutí minimálního výskytu zprava
${parameter%%pattern}  // odříznutí maximálního výskytu zprava
```

```
cesta=usr/local/bin/dosbox
${cesta##*/}           <=> dosbox
${cesta%/*}            <=> /usr/local/bin
```

další, novější rozšíření závislá na konkrétním shellu:
(u ksh na verzi, např. KornShell93+)

- **asociativní pole proměnných (možná syntaxe pole=[index]=hodnota)**
- **složené proměnné (struktura v C)(možná syntaxe osoba=jmeno=Bohous)**
- **concat řetězcových proměnných v syntaxi otazka+=" $odpoved"**
- **nepřímé referencování proměnných v syntaxi nameref ref=prom**
  => $ref manipuluje s $prom (např. pro výstupní parametry funkcí)

Bonusy

- **ksh/bash obsahují oproti sh navíc některé shellové proměnné:**

  $SECONDS – počet sekund od spuštění shellu (časování skriptů)
  $ERRNO – číslo poslední chyby
  $LINENO (readonly) – aktuální řádek právě vykonávaného skriptu
  (užitečné např. pro debugovací účely)
  $RANDOM (readonly) – pseudonáhodné číslo z ntervalu <0, 32767>
  (přiřazení řetězce "seed" inicializuje generátor)
  $PPID (readonly) – processID rodiče

- **přehled speciálních proměnných shellu (i sh):**
  $0 - $9 **parametry příkazové řádky** (ev. $argv[n])
  $# **počet parametrů příkazové řádky**
  $? **návratový kód posledního procesu** (decimální string)
  $$ **PID shellu**
  $! **PID posledního procesu spuštěného na pozadí**
  $- **přepínače shellu**
  $* **parametry příkazové řádky** (za sebou od indexu 1)
  $@ (rozdíl od $* v expanzi při použití uvozovek)
  $PWD, $IFS, $SHELL ...