Operacijski sistemi

vaje 5

spremenljivke

- deklaracija:

```
ime_spremenljivke=vrednost
```

- uporaba:

```
${ime_spremenljivke}
$ime_spremenljivke
```

primeri:

```
a = os
a=os
a=operacijski sistemi
a="operacijski sistemi"
a='operacijski sistemi'
echo a
echo $a
echo "$a"
echo '$a'
```

spremenljivke

```
${ime_spremenljivke}
$ime_spremenljivke
${ime_spremenljivke:-vrednost}
${ime_spremenljivke:=vrednost}
${#ime_spremenljivke}     ... dolžina niza
${niz:poz}                ... podniz z leve
${niz:poz:dolžina}        ... podniz z leve
${niz:(-poz)}             ... podniz z desne
${niz:(-poz):dolžina}     ... podniz z desne
```

test, [ ], [[ ]]

- nekaj stikal za preverjanje lastnosti datotek:

`-e datoteka` … Ali datoteka obstaja?
`-d datoteka` … Ali je datoteka imenik?
`-f datoteka` … Ali je datoteka navadna datoteka?
`-s datoteka` … Ali velikost datoteke ni enaka 0?
`-b datoteka` … Ali je datoteka bločno-orientirana?
`-c datoteka` … Ali je datoteka znakovno-orientirana?
`-h datoteka` … Ali je datoteka simbolična povezava?
`-L datoteka` … Ali je datoteka simbolična povezava?
`-r datoteka` … Ali lahko datoteko (kot trenutni uporabnik) beremo?
`-w datoteka` … Ali lahko v datoteko (kot trenutni uporabnik) zapisujemo?
`-x datoteka` … Ali lahko datoteko (kot trenutni uporabnik) poganjamo?

spremenljivke (odstranitev podniza)

- najkrajše ujemanje
  `${niz#podniz}`
- najdaljše ujemanje
  `${niz##podniz}`
- najkrajše ujemanje z zadnje strani
  `${niz%podniz}`
- najdaljše ujemanje z zadnje strani
  `${niz%%podniz}`

Kako delujejo naslednje operacije?

```
x="To je test, 123, 123."
echo ${x#* }
echo ${x##* }
echo ${x% *}
echo ${x%% *}
```

test, [ ], [[ ]]

- preverjanje enakosti med števili

`x -eq y` … preverimo, ali je x enak y

`x -ne y` … preverimo, ali je x različen od y

`x -gt y` … preverimo, ali je x večji kot y

`x -ge y` … preverimo, ali je x večji ali enak kot y

`x -lt y` … preverimo, ali je x manjši od y

`x -le y` … preverimo, ali je x manjši ali enak od y

spremenljivke (zamenjava podniza)

- zamenja prvo ujemanje
  `${niz/podniz/zamenjava}`
- zamenja vsa ujemanja
  `${niz//podniz/zamenjava}`
- zamenja začetek niza
  `${niz/#podniz/zamenjava}`
- zamenja konec niza
  `${niz/%podniz/zamenjava}`

test, [ ], [[ ]]

- preverjanje enakosti med nizi

`x = y` … preverimo, če je niz x enak nizu y
`x != y` … preverimo, če x ni enak y
`-n x` … preverimo, če x ni ničeln (null)
`-z x` … preverimo, če je x ničeln (null)

testiranje

- `if test -f "/etc/blabla"; then`
- `if [ -f "/etc/blabla" ]; then`
- `if [[ -f /etc/blabla ]]; then`

- `if test "$ime" -eq 5; then`
- `if [ "$ime" -eq 5 ]; then`
- `if [[ $ime -eq 5 ]]; then`

programiranje v BASH-u

- BASH - **B**ourne **A**gain **Sh**ell
- lupina za ukaze
- skriptni jezik
  - omogoča v celoti uporabiti zmožnosti lupine
  - avtomatizacija opravil (npr. zagonske skripte v Linuxu z `init` → `/etc/init.d/`)

- prevedeni programi skripte
  - hitrost, prenosljivost, prevajanje, …

pisanje skript

- pozivna vrstica
  - interaktivni način
- skripta
  - neinteraktivni način (+ možnost interaktivnega načina)
  - kako poženemo?
  - zaključek skripte (`exit`)
  - komentarji (#)

posebne spremenljivke

`$_` … zadnji argument predhodno izvedenega ukaza
`$#` … število podanih argumentov
`$0` … ime skripte
`$1 $2 … ${n}`… zaporedni argumenti skripte
`$?` … izhodni status zadnjega (v ospredju) izvedenega ukaza
`$$` … PID lupine
`$!` … PID procesa, ki je bil zadnji zagnan v ozadju
`$*, $@` … vsi argumenti skripte skupaj
`$-` … opcije podane lupini, ki poganja skripto

naloge - skripte

1. Napišite skripto, ki preveri, ali smo skripto pognali kot skrbnik sistema "superuporabnik" (angl. superuser)! Če nismo, izpiše opozorilo in se zaključi. Če smo, požene ukaz, ki izpiše zadnjih 5 uspešnih in 5 neuspešnih prijav v sistem.

2. V spremenljivko okoljsko spremenljivko `PATH` dodajte imenik, kjer se trenutno nahajate (ne trenutni imenik v splošnem) in podimenik vaje v vašem domačem imeniku!

3. Napišite skripto, ki preveri, ali uporabnik, ki ga podamo kot argument skripti, obstaja (na našem sistemu)? Če uporabnik obstaja, preverite, ali obstaja njegov domači imenik! Če domači imenik obstaja, izpišite, če imate pravico za dostop do posameznih datotek v tem imeniku! Če uporabnik ne poda argumenta, ga opozorimo, da mora kot prvi argument podati uporabnika.

aritmetika

- ukaz `expr`
- primer uporabe: `expr 1 + 2`
- katere operacije?

- vgrajenost?
- `$((...))`

aritmetika

```
#!/bin/bash
x=8
y=4
z=$(expr $x + $y)
echo "Vsota števil $x + $y je $z"
```

aritmetika

```
#!/bin/bash
x=8
y=4
z=$(($x + $y))
echo "Vsota števil $x + $y je $z"
```

aritmetika

```
#!/bin/bash
x=5
y=3
add=$(($x + $y))
sub=$(($x - $y))
mul=$(($x * $y))
div=$(($x / $y))
mod=$(($x % $y))
# izpišemo rezultate:
echo "vsota: $add"
echo "razlika: $sub"
echo "zmnožek: $mul"
echo "količnik: $div"
echo "ostanek: $mod"
```