Operacijski sistemi

vaje 5

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)

pisanje skript

- poganjanje
- zaključek skripte (`exit`)
- komentarji (#)
- spremenljivke
  - deklaracija:

    ```
    ime_spremenljivke=vrednost
    ```
  - uporaba:

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

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
```

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%% *}
```

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}`

preproste skripte

- primer programa: kopiraj.sh

```
#!/bin/bash
cp /etc/nekaj .
echo "Kopiranje končano."
```

Kaj pa, če se je zgodila napaka pri kopiranju?

if … then … else … elif … fi

```
#!/bin/bash
if cp /etc/nekaj .
then
  echo "Kopiranje končano."
else
  echo "Kopiranje neuspešno."
fi
```

Znamo napisati še kako drugače?

Naloga 1

```
#!/bin/bash
if cp /etc/nekaj .
then
  echo "Kopiranje končano."
else
  echo "Kopiranje neuspešno."
fi
```

Prepišimo program z uporabo pogojnega izvajanja in brez uporabe `if`.

Naloga 2

Napišite skripto `povezava.sh`, ki naredi trdo povezavo na datoteko, ki jo podamo kot prvi argument ob klicu skripte. Povezavo naredi v trenutnem delovnem imeniku in naj se imenuje enako kot izvorna datoteka. Če izvorna datoteka ne obstaja, pa naj jo ustvari in potem ustvari še trdo povezavo.

Primer uporabe:

```
/home/student$ /home/administrator/povezava.sh /etc/passwd
```

Po zaključku skripte se naredi nov datotečni zapis `/home/student/passwd`, ki kaže na isti inode kot `/etc/passwd`.

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

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`

Naloga 3

Napišite skripto `preveri.sh`, 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 (ugotavljate torej pravice na imeniku). Če uporabnik ne poda argumenta, ga opozorimo, da mora kot prvi argument podati uporabniško ime.

Primer klica kot uporabnik `administrator`:

```
$ ./preveri.sh administrator
uporabnik administrator obstaja
imenik /home/administrator obstaja
pravice za dostop do datotek v imeniku imamo

$ ./preveri.sh toor
uporabnik toor ne obstaja
```

test, [ ], [[ ]]

- stikala za preverjanje 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?

logični in, logični ali

- `pogoj1 && pogoj2`
- `pogoj1 || pogoj2`

```
#!/bin/bash
x=5
y=10
if [ "$x" -eq 5 ] && [ "$y" -eq 10 ]; then
  echo "Oba pogoja sta resnična."
else
  echo "Pogoja nista resnična."
fi
```

logični in, logični ali

- `pogoj1 && pogoj2`
- `pogoj1 || pogoj2`

```
#!/bin/bash
x=3
y=2
if [ "$x" -eq 5 ] || [ "$y" -eq 2 ]; then
  echo "En od pogojev je resnicen."
else
  echo "Noben pogoj ni resnicen."
fi
```

case … esac

```
#!/bin/bash
x=5

case $x in
  0) echo "Vrednost x je 0."
     ;;
  5) echo "Vrednost x je 5."
     ;;
  9) echo "Vrednost x je 9."
     ;;
  *) echo "Nepoznana vrednost."
esac
```

case ↔ if

```
#!/bin/bash
x=5

if [ "$x" -eq 0 ]; then
  echo "Vrednost x je 0."
elif [ "$x" -eq 5 ]; then
  echo "Vrednost x je 5."
elif [ "$x" -eq 9 ]; then
  echo "Vrednost x je 9."
else
  echo "Nepoznana vrednost."
fi
```

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"
```

while … do … done

```
#!/bin/bash
while true; do
  echo "Za izhod pritisni CTRL-C."
done
```

- `true` vgrajen?

while … do … done

```
#!/bin/bash
while :; do
  echo "Za izhod pritisni CTRL-C."
done
```

- `:` vgrajeno?

while … do … done

```
#!/bin/bash
x=0
while [ "$x" -le 10 ]; do
  echo "Trenutna vrednost spremenljivke x: $x"
  x=$x+1
  sleep 1
done
```

- Kaj dela ta koda?

while … do … done

```
#!/bin/bash
x=0
while [ "$x" -le 10 ]; do
   echo "Trenutna vrednost spremenljivke x: $x"
   x=$x+1
   sleep 1
done
```

- Kako bi popravili?

while … do … done

```
#!/bin/bash
x=0
while [ "$x" -le 10 ]; do
  echo "Trenutna vrednost spremenljivke x: $x"
  x=$(expr $x + 1)
  sleep 1
done
```

until … do … done

```
#!/bin/bash
x=0
until [ "$x" -gt 10 ]; do
  echo "Trenutna vrednost spremenljivke x: $x"
  x=$(expr $x + 1)
  sleep 1
done
```

- Ali dela isto kot različica na prejšnji prosojnici?

for … in … do … done

```
#!/bin/bash
for x in papir svincnik pero; do
  echo "Vrednost spremenljivke x je: $x"
  sleep 1
done
```

for … in … do … done

```
#!/bin/bash
echo -n "Kontrola sistema za napake"
for dots in 1 2 3 4 5 6 7 8 9 10; do
  echo -n "."
  sleep 1
done
echo "Sistem je pregledan."
```

Naloga 4

S pomočjo **(a)** zanke `for` in **(b)** zanke `while` napišite program `stevec.sh`, ki bo štel od 1 do 500 v enosekundnih intervalih.

Primer delovanja:

```
$ ./stevec.sh
1
2
3
⋮
499
500
$
```

for … in … do … done

```
#!/bin/bash
for datoteka in *; do
  echo "Dodaj koncnico .html datoteki $datoteka..."
  mv $datoteka $datoteka.html
  sleep 1
done
```

zanka for

```
for var in spisek ; do
  ukazi
done
```

- od BASH 2.04 naprej:

```
for (( inicializacija ; pogoj ; inkrementiranje )); do
  ukazi
done
```

zanka for

```
for i in `seq 24 42`; do
  echo -n "$i "
done

for i in {24..42}; do
  echo -n "$i "
done

for ((i=24; i<=42; i++)); do
  echo -n "$i "
done
```

Uporaba tabel

```
tabela=(rdeča oranžna rumena zelena modra vijolična)
dolzina=${#tabela[*]}
echo "Mavrica ima $dolzina barv:"
i=0
while [ $i -lt $dolzina ]; do
   echo "$((i+1)): ${tabela[$i]}"
   let i++
done
```

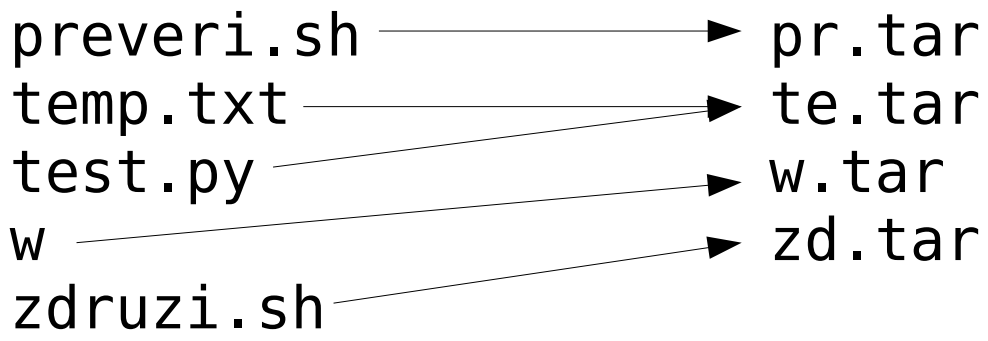
- prazno tabelo definiramo z
  `declare -a ime_tabele`

- prazno asociativno tabelo (slovar) definiramo z
  `declare -A ime_tabele`

Naloga 5

Napišite skripto `zdruzi.sh`, ki bo združila vse navadne datoteke iz trenutnega imenika v pakete **`tar`**. V posameznem paketu naj bodo vse datoteke, ki imajo prvih **n** črk imena istih (da bo naloga lažja, vam ni treba odstranjevati končnic). Ime paketa naj bo sestavljeno iz prvih skupnih **n** črk in končnice **`tar`**. Število črk n podamo kot argument skripte. Če je število črk manjše od **n**, potem vzemite toliko črk, kolikor jih je na voljo.

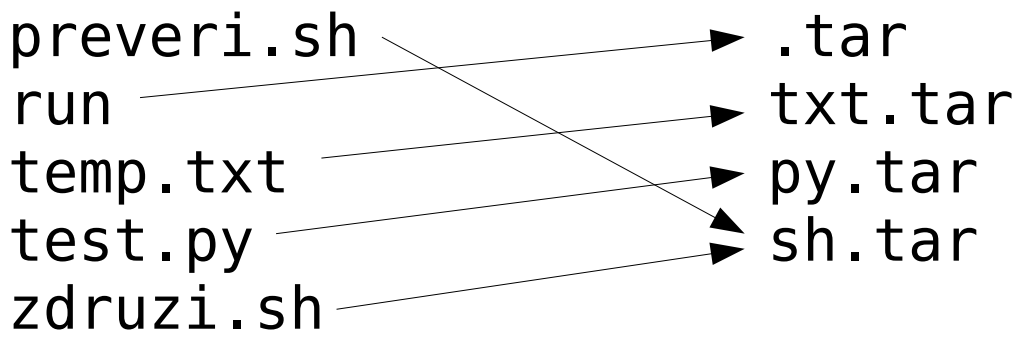
```
$ ./zdruzi.sh 2
```



Naloga 6

Napišite skripto `koncnice.sh`, ki bo združila vse navadne datoteke iz trenutnega imenika v pakete **`tar`**. V posameznem paketu naj bodo vse datoteke, ki imajo isto končnico (torej imajo isti niz za zadnjo piko). Ime paketa naj bo ime končnice. Če datoteka končnice nima, potem je ime paketa prazen niz s končnico `.tar`.

```
$ ./koncnice.sh
```



branje vhoda: ukaz read

- interakcija z uporabnikom

```
#!/bin/bash
# prebere ime uporabnika in izpiše dobrodošlico
echo -n "Vpisi svoje ime: "
read user_name
echo "Pozdravljen $user_name!"
```

branje vhoda: ukaz read

```
#!/bin/bash
# prebere ime uporabnika in izpiše dobrodošlico
echo -n "Vpisi svoje ime: "
read user_name
# če uporabnik ne vnese nič:
if [ -z "$user_name" ]; then
  echo "Nisi mi povedal svojega imena!"
  exit 10
fi
echo "Pozdravljen $user_name"
exit 0
```

branje vhoda: ukaz read

```
#!/bin/bash
# prebere ime uporabnika in izpiše dobrodošlico
echo -n "Vpisi svoje ime: "
read user_name
# če uporabnik ne vnese nič:
if [ -z "$user_name" ]; then
  echo "Nisi mi povedal svojega imena!"
  exit 10
fi
echo "Pozdravljen $user_name"
exit 0
```

- Kako bi spremenili program, da bi spraševal uporabnika po imenu dokler ga le-ta ne vpiše?

branje vhoda: ukaz read

```
#!/bin/bash
while [ -z $user_name ]; do
  echo -n "Vpisi svoje ime: "
  read user_name
  if [ -n "$user_name" ]; then
    echo "Pozdravljen $user_name"
    exit 0
  fi
  echo "Nisi mi povedal svojega imena!"
done
exit 1
```

Naloga 7

Napišite skripto `veriga.sh`, ki bo v podanem imeniku izpisala vse verige mehkih povezav. Za vsako mehko/simbolično povezavo naj skripta pogleda kam kaže, če pa je ciljna datoteka spet povezava, naj pogleda, kam kaže in tako naprej do datoteke, ki ni mehka povezava – tako sestavljeno verigo povezav nato izpišemo.

Imenik, kjer skripta poišče mehke povezave, prejme tako, da uporabniku izpiše poziv in čaka, da uporabnik vpiše imenik.

Ko skripta obdela podani imenik in izpiše verige povezav, ponovno izpiše uporabniku poziv za vpis novega imenika itd.

Če uporabnik ne vpiše imenika v petih sekundah, se skripta ustavi.

Primer uporabe:
predpostavimo, da imamo v imeniku `.` tri mehke povezave: `konfiguracije`, ki kaže na `/etc`; `podatki`, ki kaže na `data.txt` in `novo`, ki kaže na `podatki`.

```
$ ./veriga.sh
Vpišite imenik: .
konfiguracije -> /etc
novo -> podatki -> data.txt
podatki -> data.txt
Vpišite imenik:
$
```