

# Operacijski sistemi

vaje 6

# 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 ... elif ... else ... 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`.

# 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  $\text{expr}$
- primer uporabe:  $\text{expr } 1 + 2$
- 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 ; pogoji ; 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.

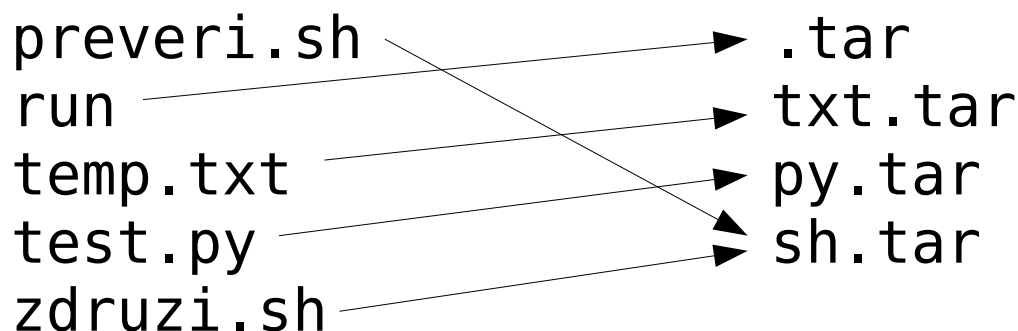
```
$ ./zdruzi.sh 2
```

|                         |   |                     |
|-------------------------|---|---------------------|
| <code>preveri.sh</code> | → | <code>pr.tar</code> |
| <code>temp.txt</code>   | → | <code>te.tar</code> |
| <code>test.py</code>    | → | <code>w.tar</code>  |
| <code>w</code>          | → | <code>zd.tar</code> |
| <code>zdruzi.sh</code>  | → |                     |

# 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` (pozor: ker so bo datoteka začela s piko, bo skrita).

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



The diagram illustrates the mapping of files to tar archives based on their extensions. Arrows point from the file names on the left to the resulting archive names on the right. Files with the same extension are grouped into a single archive.

| File       | Archive |
|------------|---------|
| preveri.sh | .tar    |
| run        |         |
| temp.txt   | txt.tar |
| test.py    | py.tar  |
| zdruzi.sh  | sh.tar  |
|            |         |