Računalniški praktikum 1

Uvod v bash

Vida Groznik

Source: Ryan‘s tutorials.

Stavek IF

```
1. #!/bin/bash
2. # Osnovni stavek IF
3.
4. if [ $1 -gt 100 ]
5. then
6.       echo Hey that\'s a large number.
7.       pwd
8. fi
9.
10. date
```

```
user@bash: ./if_example.sh 15
Mon 6 Nov 17:20:40 2017
user@bash: ./if_example.sh 150
Hey that's a large number.
/home/vida/bin
Mon 6 Nov 17:20:40 2017
user@bash:
```

Osnovni stavek `if` pravi, če je vrednost določenega testa prava (true), potem izvedi podan set ukazov. Če vrednost ni prava (false), potem teh ukazov ne izvedi.

```
if [ <test> ]
then
        <ukazi>
fi
```

Vse med besedama `then` in `fi` (if brano nazaj) bo izvedeno le v primeru, če test (med oglatimi oklepaji) vrne vrednost true.

- **Vrstica 4** - Če je vrednost prvega elementa ukazne vrstice večja od 100.
- **Vrstici 6 in 7** - Se bosta izvedli le v primeru, če test v vrstici 4 vrne vrednost true. Tu je lahko poljubno število ukazov.
- **Vrstica 6** - Obratna poševnica (\ ) pred enojnimi narekovaji (') je potrebna, ker imajo enojni narekovaji poseben pomen, ki ga v tem primeru ne želimo.
- **Vrstica 8** - `fi` označuje konec stavka if. Vsi nadaljnji ukazi bodo izvedeni kot običajno.
- **Vrstica 10** - Ta ukaz bo izveden ne glede na rezultat testa v vrstici 4.

Vgnezdeni stavki IF

```
1.  #!/bin/bash
2.  # Vgnezdeni stavki IF
3.
4.  if [ $1 -gt 100 ]
5.  then
6.      echo Hey that\'s a large number.
7.
8.      if (( $1 % 2 == 0 ))
9.      then
10.         echo And is also an even number.
11.     fi
12. fi
```

V skripti imate lahko toliko stavkov **`if`** kolikor je potrebno. Prav tako je mogoče, da imate stavek `if` znotraj nekega drugega stavka `if`.

- **Vrstica 4** – Izvedi sledeče ukaze, če je vrednost prvega argumenta ukazni vrstici večja od 100.
- **Vrstica 8** – To je variacija stavka `if`. Če želimo preveriti aritmetični izraz, lahko uporabimo dvojne oklepaje.
- **Vrstica 10** – Se izvede le v primeru, da sta vrednosti obeh stavkov `if` pravi (true).

Stavek IF - ELSE

```
1.  #!/bin/bash
2.  # if-else primer
3.  # beri iz dokumenta, če je podan kot
4.  # argument ukazne vrstice, sicer beri
5.  # iz STDIN.
6.
7.  if [ $# -eq 1 ]
8.  then
9.      nl $1
10. else
11.     nl /dev/stdin
12. fi
```

Včasih si želimo izvesti niz ukazov, ko je določena trditev prava (true) in nek drug niz ukazov, če je trditev napačna (false). To lahko naredimo z uporabo mehanizma `else`.

```
if [ <test> ]
then
        <ukazi>
else
        <drugi ukazi>
fi
```

Vaja

Napiši skripto, ki ugotovi ali določen dokument obstaja ali ne. Ime dokumenta je podano kot argument ukazne vrstice. Prav tako preveri, če je podanih dovolj argumentov ukazne vrstice.

Stavek IF – ELIF – ELSE

Primer: Če ste stari 18 let ali več, greste lahko na zabavo.
Če niste dovolj stari, ampak imate pisno soglasje staršev, greste lahko a morate biti doma pred polnočjo.
V nasprotnem primeru ne smete na zabavo.

```
1.  #!/bin/bash
2.  # ukazi elif
3.
4.  if [ $1 -ge 18 ]
5.  then
6.      echo Lahko greste na zabavo.
7.  elif [ $2 == 'yes' ]
8.  then
9.     echo Lahko greste na zabavo a se
       morate vrniti domov pred polnočjo.
10. else Ne smete na zabavo.
12. fi
```

Včasih imamo lahko vrsto pogojev, ki lahko vodijo do različnih ciljev.

```
if [ <test> ]
then
        <ukazi>
elif [ <test> ]
then
        <drugi ukazi>
else
        <drugi ukazi>
fi
```

Lahko imate kolikor si želite ukazov `elif`. Tudi končni ukaz `else` je neobvezen.

Booleanove operacije

```
1. #!/bin/bash
2. # primer za „in“
3.
4. if [ -r $1 ] && [ -s $1 ]
5. then
6.     echo This file is useful.
7. fi
```

```
1. #!/bin/bash
2. # primer za „ali“
3.
4. if [ $USER == 'bob' ] || [ $USER == 'andy' ]
5. then
6.     ls -alh
7. else
8.     ls
9. fi
```

Včasih si želimo izvesti nekaj, ko je veljavnih več pogojev.

Spet drugič si želimo izvesti nekaj, ko je veljaven vsaj en izmed pogojev.

Pri tem si lahko pomagamo z booleanovi operatorji.

in (and) - &&

ali (or) - ||

Vaja

Napišite skripto, ki bo poiskala najvišje število izmed podanih treh števil. Števila so podana kot argumenti ukazne vrstice.

Izpišite napako, če ni podanih zadostno število argumentov v ukazni vrstici.

Case stavek

```
1.  #!/bin/bash
2.  # primer za „case“ stavek
3.
4.  case $1 in
5.      start)
6.          echo starting
7.          ;;
8.      stop)
9.          echo stopping
10.         ;;
11.     restart)
12.         echo restarting
13.         ;;
14.     *)
15.         echo don\'t know
16.         ;;
17. esac
```

Če si želimo, da program poteka po različnih poteh glede na ujemanje spremenljivke nekemu vzorcu, sicer lahko uporabimo zaporedje večih `if` in `elif` stavkov, ampak bi koda hitro postala nejasna.

Za dosego tega cilja lahko uporabimo `case` stavek.

```
case <spremenljivka> in
    <vzorec 1>)
        <ukazi>
        ;;
    <vzorec 2>)
        <drugi ukazi>
        ;;
esac
```

Case stavek (2)

```
1.  #!/bin/bash
2.  # primer za „case“ stavek
3.
4.  case $1 in
5.      start)
6.          echo starting
7.          ;;
8.      stop)
9.          echo stoping
10.         ;;
11.     restart)
12.         echo restarting
13.         ;;
14.     *)
15.         echo don\'t know
16.         ;;
17. esac
```

- **Vrstica 4** – Začetek `case` stavka.
- **Vrstica 5** – če je `$1` 'start' potem izvedi naslednje ukaze. Oklepaj `)` predstavlja konec vzorca.
- **Vrstica 7** – Konec tega skupka ukazov označimo z dvema podpičjema (`;;`). Temu sledi nov vzorec.
- **Vrstica 14** – Test za vsak primer je nek vzorec. Zvezdica `*` predstavlja kateri koli znak (lahko tudi več ponovitev znakov). Ta del se izvede, če se vzorec ne sklada s katerim koli drugim vzorcem pred tem. Je neobvezen.
- **Vrstica 17** - `esac` predstavlja konec case stavka. Karkoli za tem bo izvedeno kot običajno.

Vaja

Napiši skripto z uporabo **case stavka** za izvajanje osnovnih matematičnih operacij kot sledi:

+ seštevanje
- odštevanje
x množenje
/ deljenje

Ime skripte naj bo 'q4, ki jo pokličemo tako:
$ **./q4 20 / 3**, preveri, da bo imel vnos zadostno število argumentov ukazne vrstice.s