Operacijski sistemi

Skriptiranje

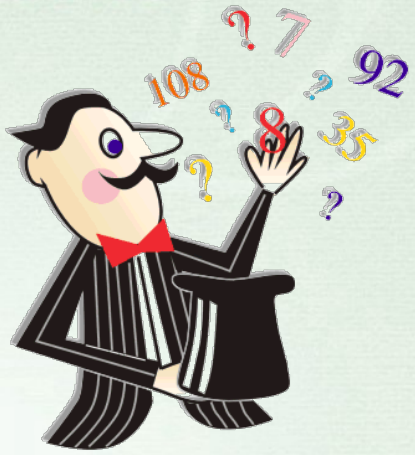
Vaje

Jurij Mihelič, FRI, Uni LJ

Vsebina

- Skripta
- Spremenljivke
- Pogojni izrazi
- Programski stavki
- Ujemanje vzorcev

Skripta



- Zgradba skripte
  - Shebang #!
    - čarobno število
    - `/bin/bash`
    - `/usr/bin/env bash`
  - Komentar
  - Zaporedje ukazov
    - konec vrstice ali podpičje
  - Zaključek skripte
    - impliciten zaključek ali
    - ukaz `exit` STATUS

```
#!/bin/bash

# komentar

echo 1
echo 2; echo 3;
echo 4
echo 5;
echo 6; echo 7;

exit 0
```

Skripta

- Izhodni status
  - 8 bitna vrednost od 0 do 255
  - spremenljivka `$?`
  - vrednost 0
    - uspešen zaključek skripte
    - resnično (true) v pogojnih izrazih
    - ukaz `true`
  - vrednost >0
    - neuspešen zaključek skripte
    - vrednost predstavlja kodo napake
    - neresnično (false) v pogojnih izrazih
    - ukaz `false`

true.sh

```
#!/bin/bash
exit 0
```

true.c

```
int main() {
    exit(0);
}
```

false.sh

```
#!/bin/bash
exit 1
```

false.c

```
int main() {
    exit(1);
}
```

Skripta

- Zagon skripte
  - Neposreden zagon
    - `./`*skripta.sh*
    - Zakaj moramo dodati predpono `./`?
    - Kako dodamo dovoljenje za izvajanje?
  - Preko ukaza bash
    - `bash` *skripta.sh*
  - Uvoz v trenutno lupino
    - uporaba za knjižnice
    - `source` *skripta.sh*

Spremenljivke

- Argumenti skripte
  - Vgrajene spremenljivke

`$#`	… število vseh argumentov
`$0`	… ime skriptne datoteke
`$1, $2, …`	… posamezni argument skripte
`$*, $@`	… vsi argumenti skripte

```
#!/bin/bash

echo ”Ime skripte: $0”

echo $1
echo $1$2

# 12. argument
echo $12      # previdno
echo ${12}    # pravilno
```

Spremenljivke

- Argumenti skripte
  - Ukaz `shift`

  - Kako bi zaporedoma enega za drugim obdelal vse argumente skripte?

```
while [[ -n $1 ]]; do
    echo $1
    shift
done
```

  - * Želiš napredno obdelovati stikala?
    - Glej ukaza `getopt` in `getopts`

Spremenljivke

- Definicija spremenljivk
  - **globalne** spremenljivke
    - *ime=vrednost*
  - **lokalne** spremenljivke
    - lahko le znotraj funkcij
    - `local` *ime=vrednost*

Napaka: presledki okrog =

```
a = 123
a =123
a= 123
```

Pravilno: brez presledkov

```
a=123
a=Triglav
a=1+2+3
```

Kaj hranita a in b?

```
a=b
b=a
```

- Katerega tipa so spremenljivke v `bash`?
- Kaj pa če vrednost vsebuje presledke?
- Čemu služi ukaz `unset`?

Spremenljivke

- Enojne in dvojne navednice
  - **Enojne** navednice
    - dobesedna predstavitev niza
  - **Dvojne** navednice
    - izvedejo se morebitne notranje substitucije
  - Uporaba navednic
    - nujna kadar niz vsebuje presledke
    - skoraj vedno priporočljiva

```
a=Triglav
echo $a
echo '$a'
echo "$a"

b='Martin        Krpan'
echo $b
echo '$b'
echo "$b"
```

Spremenljivke

- Priklic vrednosti spremenljivke

`$ime`	… krajši način
`${ime}`	… osnovni način
`${ime:-privzeto}`	… shranjena ali privzeta vrednost

$ kot substitucija

```
a=1; b=2

echo $a $a $b $b
echo $a  $a  $b  $b
echo $a$a$b$b

a=$a$a; a=$a$a; echo $a
```

```
cmd=ls
arg=-alp

echo Ukaz: $cmd z argumenti $arg

$cmd $arg
```

Spremenljivke

- Substitucija vrednosti spremenljivk
  - veliko možnosti, RTFM ☺, glej `man bash`

```
# Nizi in podnizi
${#ime}              … dolžina niza, shranjenega v ime
${ime:poz}           … podniz od indeksa poz do konca
${ime:poz:dol}       … podniz od indeksa poz dolžine dol

# Odstranjevanje podnizov
${ime#vzorec}        … najkrajša predpona
${ime##vzorec}       … najdaljša predpona
${ime%vzorec}        … najkrajša pripona
${ime%%vzorec}       … najdaljša pripona

# Zamenjava podniza
${ime/vzorec/vrednost}      … prva pojavitev
${ime//vzorec/vrednost}     … vse pojavitve
```

Pogojni izrazi

- Ovrednotenje pogoja
  - več možnosti, tako vgrajeni kot zunanji ukazi

```
test pogoj
[ pogoj ]
[[ pogoj ]]
```

bash specific, but easier to use

  - rezultat dobimo preko izhodnega statusa
    - preko spremenljivke `$?`

```
test OS == os; echo $?

[ -f /etc/passwd ]; echo $?

[[ 3 -lt 42 ]]; echo $?
```

Pogojni izrazi

- Primerjava nizov

`[[ -z niz ]]`	… Je niz prazen (dolžina 0, zero)?
`[[ -n niz ]]`	… Je niz neprazen (dolžina >0, nozero)?
`[[ niz1 = niz2 ]]`	… Sta niza enaka?
`[[ niz1 != niz2 ]]`	… Sta niza različna?
`[[ niz1 < niz2 ]]`	… Je niz1 leksikografsko pred niz2?
`[[ niz1 > niz2 ]]`	… Je niz1 leksikografsko za niz2?

 # Težave brez navednic

```
unset x

test -n $x
[ -n $x ]

test $x = $x
test $x != $x
```

 # Uporaba "" in [[…]]

```
test -n "$x"
[ -n "$x" ]

[[ -n $x ]]
[[ -n "$x" ]]
[[ $x = $x ]]
[[ "$x" = "$x" ]]
```

Pogojni izrazi

- Preverjanje datotek

```
[[ -e pot ]]
[[ -f pot ]]
[[ -d pot ]]
[[ -L pot ]]
[[ -b pot ]]
[[ -c pot ]]
...
```

… Ali datoteka **obstaja**?
… Je datoteka **navadna** (regular file)?
… Je datoteka **imenik**?
… Je datoteka **simbolična povezava**?
… Je datoteka **bločna** naprava?
… Je datoteka **znakovna** naprava?

```
test -e /etc/passwd
test -f /etc/passwd
test -f /etc
test -L /etc/init.d
test -b /dev/sda1
test -c /dev/tty0
```

```
file="${1:-vhod.txt}"

path="/tmp/output/$file"

test -f "$path"
```

Pogojni izrazi

- Kaj pa logični izrazi?

```
# Logični IN
[[ -n "$x" && -f /etc/passwd ]]

# Logični ALI
[[ -f "$f" || test -L "$f" ]]

# Negacija
[[ ! -f /etc/passwd ]]
```

bash specific, but easier to use

```
# Logični IN
test -n "$x" -a -f /etc/passwd
# Logični ALI
[ -f "$f" -o test -L "$f" ]
# Negacija
test ! -f /etc/passwd
```

Pogojni izrazi

- Logična *alternativa*
  - ukaz **&&** ukaz … logični in
  - ukaz **||** ukaz … logični ali
  - kratkostično ovrednotenje izraza
  - oba && in || sta enake prioritete

```
true || echo "To se ne izpiše"

false || echo "To pa se"

[[ -f "$file" ]] && cat "$file" || echo "No file"

mkdir test && cd test || echo "Error"
```

Pogojni izrazi

- In še aritmetični pogoji

`[ a -eq b ]`	… Je *a* enako *b*?
`[ a -ne b ]`	… Je *a* različno od *b*?
`[ a -gt b ]`	… Je *a* večje od *b*?
`[ a -ge b ]`	… Je *a* večje ali enako od *b*?
`[ a -lt b ]`	… Je *a* manjše od *b*?
`[ a -le b ]`	… Je *a* manjše ali enako od *b*?

```
test 7 -eq 007
test 42 -ne 42
test 3 -gt 2
test 2 -le 6
```

Pogojni izrazi

- Aritmetika in dvojni oklepaji
  - ovrednotenje izraza: ((*izraz*))
  - ovrednotenje in substitucija: $ ((*izraz*))
  - Cjevsko / javansko “izrazoslovje” ☺

```
(( sedem = 1 + 2 * 3 ))
sedem=$(( 1 + 2 * 3 ))

(( a = 1, a++, a += 2, b = a + a, c = a**b ))

(( a <= 42 )) && echo Malo || echo Veliko

echo $a $(( 2 * a + b ))
```

Programski stavki

- Odločitveni stavek if

`if` *pogoj*; `then` *ukazi*; `fi`
`if` *pogoj*; `then` *ukazi*; `else` *ukazi*; `fi`
`if` *pogoj*; `then` *ukazi*; `elif` *pogoj*; `then` *ukazi*; `fi`
`if` *pogoj*; `then` *ukazi*; `elif` *pogoj*; `then` *ukazi*; `else` *ukazi*; `fi`

```
if ping -c 1 s.ki; then
    echo Strežnik je dosegljiv.
else
    echo Alarm: strežnik je nedosegljiv.
    exit 42
fi
```

Programski stavki

- Zanke
  - `break`
  - `continue`

`while` ***pogoj***; `do` ***ukazi***; `done`

`until` ***pogoj***; `do` ***ukazi***; `done`

```
while true; do false; done
i=42; while (( i > 0 )); do echo $i; (( i-- )); done
i=42; until (( i == 0 )); do echo $i; (( i-- )); done

while read line; do echo $line; done
while ping -c 1 s.ki; do
    echo Juhuhu smučanje.
    sleep 2
done

while [[ -e "$file" ]] && grep syscall "$file"; do
    . . .
Done
```

Programski stavki

- Zanke

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

`for` ((***init***; ***pogoj***; ***inkrement***)); `do` ***ukazi***; `done`

```
seznam='a b c d'
for x in $seznam; do echo $x; done
for x in "$seznam"; do echo $x; done
for x in '$seznam'; do echo $x; done
for x in $seznam $seznam $seznam; do echo $x; done

for (( i = 0; i <= 42; i++ )); do echo $i; done
```

Programski stavki

- Stavek case

```
case niz in
    (vzorec1) ukazi ;;
    (vzorec2) ukazi ;;
    (*) ukazi ;;
esac
```

```
akcija=$1
shift
case $akcija in
    (help)          do_help ;;
    (list_all)      ls -R "$1" ;;
    (update)        shift; do_update $* ;;
    (*)             echo Napačna akcija.
esac
```

Ujemanje vzorcev

- Ujemanje vzorcev z nizi
  - Pozor: To niso regularni izrazi!

*	… poljuben niz, vključno s praznim nizom
?	… poljuben znak
[…]	… poljuben znak znotraj oglatih oklepajev
ostalo	… znak predstavlja sam sebe

```
case $niz in
    bingo)       echo 1 ;;
    bin*)        echo 2 ;;
    [abc]ingo)   echo 3 ;;
    ???go)       echo 4 ;;
    ??)          echo 5 ;;
    *)           echo 6 ;;
esac
```

Ujemanje vzorcev

- Ujemanje vzorcev z imeni datotek

```
touch foo.txt bar.txt b1 a3.png c2.txt file.jpg

ls *
ls *.txt
echo b*.txt
echo b?
ls ?2.txt f*
echo ???
ls ?
echo ?
echo c[[:digit:]].txt
echo [ac][13].*
echo [a-c][13]
echo [a-c]?[02].*xt
```