Programiranje v Bash

Ukazni skript – Komentarji – Vzdevki – Vhodna substitucija – Izhodna substitucija – Spremenljivke – Klicni parametri in spremenljivke – Statusna spremenljivka ukazov – Pogoji in testiranja – Krmiljenje izvajanja – Funkcije – Vhod in izhod

Ukazni skript

Ukaze, ki bi jih tipkali neposredno lupini bash v ukazno vrstico, lahko zapišemo z urejevalnikom v ukazno datoteko, skript, in tega poženemo iz ukazne vrstice. Možni so trije načini zagona:

1) Tekoča lupina naj izvede skript SKRIPT. Spremenljivke, definirane v skriptu (glej naprej), bodisi izvozne ali ne, naj po končanju ostanejo definirane, to je, dostopne iz tekoče lupine:

```
$ . SCRIPT
```

2) Tekoča lupina naj požene lupino bash (ki je lahko enaka ali drugačna od tekoče lupine), ta pa naj izvede skript. Skriptne spremenljivke naj po končanju izginejo:

```
$ bash SCRIPT
```

3) Tekoča lupina naj pogleda prvo vrstico v skriptu in naj izvede lupino, ki je v njej zapisana, ta pa naj potem izvaja skript. Skript mora imeti dovoljenje za izvajanje. Spremenljivke po končanju izginejo:

```
$ SCRIPT
```

Za zagon lupine bash mora imeti prva vrstica skripta obliko:

```
#!/bin/bash
```

Komentarji

Vse, kar sledi znaku "#" do konca vrstice, je komentar:

```
# Tole je komentar
```

Vzdevki

Poljubnemu nizu znakov lahko priredimo njegov vzdevek. Kadarkoli potem uporabimo vzdevek, se nadomesti z originalnim nizom znakov. Primer:

```
alias ll="ls -l"
```

Vhodna substitucija

Če hoče kakšen ukaz COMMAND sprejemati znake s stdin, mu jih lahko namesto tega posredujemo neposredno iz skripta:

```
COMMAND <<"
tekst
tekst
"
```

Izhodna substitucija

Vse znake, ki jih kakšen ukaz piše na stdout, lahko namesto tega zapiše kar v skript sam. Konstrukcija

```
$(COMMAND)
```

se v skriptu nadomesti z izhodom ukaza COMMAND. Pri tem se znaki za konec vrstic, LF, ne zapišejo.

Spremenljivke

V skriptu lahko uporabljamo lupinske spremenljivke. Imena imajo sestavljena iz črk, številk in podčrtaja. Začnejo se s črko. Ni jih treba vnaprej deklarirati. Njihov tip je "niz", to je, vsebujejo nize (zaporedja znakov). Preden jih napolnimo, so prazne – vsebujejo niz "".

Spremenljivko X napolnimo z nizom "abc" takole:

```
X=abc
```

Okrog enačaja ne sme biti presledkov, saj se ti za enačajem interpretirajo kot del niza.

Vsebino spremenljivke X dosežemo s konstrukcijo $X, na primer:

```
echo $X
```

Ime spremenljivke X lahko pišemo tudi kot {X}, če je to potrebno, na primer:

```
Y=${X}def
```

Če ne bi pisali oklepajev, bi se v Y zapisala vsebina spremenljivke Xdef, ki morebiti sploh ni definirana in je torej prazna.

Spremenljivko X lahko naredimo izvozno:

```
export X
```

To pomeni, da je poznana vsem ukazom, ki jih obdelovalna lupina poganja.

Klicni parametri in spremenljivke

Skript lahko poženemo s klicnimi parametri, na primer:

```
$ SCRIPT abc def
```

Znotraj skripta so na razpolago posebne spremenljivke, ki vsebujejo informacijo o klicnih parametrih:

- `0` Vsebuje ime skripta, torej "SCRIPT"
- `1` Vsebuje prvi parameter, torej "abc"
- `2` Vsebuje drugi parameter, torej "def"
- `#` Vsebuje število parametrov, torej 2
- `*` Vsebuje vse parametre kot en sam niz z vmesnimi presledki, torej "abc def"

Statusna spremenljivka ukazov

Vsak ukaz, ki se izvede, zapiše v posebno spremenljivko ? status, kako je bil ukaz izvršen:

- `?` Vsebuje "0" ali "1"

Status skripta ob izhodu določimo z ukazom

```
exit 0
```

Pogoji in testiranja

Ukazi true, false in test vračajo vrednosti true ali false:

Ali obstaja imenik?

```
test -d DIRNAME
```

Ali obstaja datoteka?

```
test -f FILENAME
```

Ali sta niza X in Y enaka / različna?

```
test X = Y
```

```
test X != Y
```

Ali sta niza N in M, interpretirana kot celi števili,
enaka / različna itd.:

```
test N -eq M
```

```
test N -ne M
```

```
test N -lt M
```

```
test N -gt M
```

Krmiljenje izvajanja

Pogojna razvejitev:

```
if test izraz
then
    ukazi
fi
```

```
        if test izraz
        then
            ukazi
        else
            ukazi
        fi

        if test izraz
        then
            ukazi
        elif test izraz
        then
            ukazi
        fi
```

Pogojna iteracija:

```
        while test izraz
        do
            ukazi
        done
```

Iteracija:

```
        for N in 1 2 3 4 5
        do
            ukazi
        done
```

Namesto eksplicitno navedenih nizov "1" … "5" lahko uporabimo ukazno substitucijo. Praktični primeri so naslednji:

```
        $*      Vsi klicni parametri

        *.txt   Vse datoteke *.txt v tekočem imeniku
```

Funkcije

V skriptu definiram funkcijo FUNCT:

```
        function FUNCT {
            ukazi
        }
```

V telesu funkcije označujejo spremenljivke 1, 2, … #, *

parametre funkcije na identičen način, kot v skriptu označujejo parametre skripta.

Klic funkcije z dvema parametroma "abc" in "def":

```
FUNCT abc def
```

Vhod in izhod

Čitaj s tipkovnice in zapiši v spremenljivke X, Y, Z:

```
read X Y Z
```

Prva beseda vrstice se zapiše v X, druga v Y, preostanek v Z.

Izpiši na zaslon vsebino spremenljivk X, Y, Z:

```
echo $X $Y $Z
```

Izpiši na zaslon znak z razširjeno ASCII kodo AB:

```
echo -e "\xAB"
```