Bash script alapok 2. rész
Elágazások

Linux alapok

Varga Tibi 2020

Feltételes elágazás

if szintaxis

- **Figyeljünk arra, hogy a '[' és ']' zárójel karakterek előtt is után is egy-egy szóköz kell, hogy legyen !!!**

```
if [ feltétel1 ] || [ feltétel2 ]  ; then
        . . . elif [ feltétel ] ; then
        . . .
else
        . . .
fi
```

Feltételes kifejezések

A feltételeket csak speciális kapcsolókkal tudjuk megadni. Ellenkező esetekben más művelet történne (pl. > esetén beleirányítás.)

- **Numerikus és logikai operátorok:**

Kif1 **-eq** Kif2	Egyenlő # EQual
Kif1 **-ne** Kif2	Nem egyenlő #Not Equal
Kif1 **-lt** Kif2	Kisebb # Less Then
Kif1 **-le** Kif2	Kisebb vagy egyenlő # Less or Equal
Kif1 **-gt** Kif2	Nagyobb # Greater Then
Kif1 -**ge** Kif2	Nagyobb vagy egyenlő # Greather or Equal

Kif1 **-a** Kif2	Logikai ÉS # And
Kif1 **-o** Kif2	Logikai VAGY # Or
!Kif	Logikai tagadás
Kif1 **&&** Kif2	Logikai ÉS ez is
Kif1 **||** Kif2	Logikai VAGY
-z String	0 hosszúságú # Zero
String	nem 0 hosszúságú
String **!=** String	nem egyenlők
String **=** String	egyenlők

Példák

```
#!/bin/bash
if [ 8 -lt 9 ] ; then
    echo "Nyolc kisebb, mint kilenc."
fi
```

```
#!/bin/bash
if [ $# == 1 ] && [ $1 -gt 1 ] && [ $1 -lt 10 ] ; then
        echo "A parancssori parameter erteke 1 es 10 kozott van."
fi
```

```
#!/bin/bash
VAR1="Bela"
VAR2="Jeno"
if [ $VAR1 != $VAR2 ] ; then
   echo "A ket String nem egyezik meg."
fi
```

Példa:

```
 1  #!/bin/bash
 2  echo   "   Ez a bash script elágazásokat mutat be"
 3  sleep 1
 4  echo "Kérem az első számot"
 5  read szam1
 6  echo "Kérem az masodik számot"
 7  read szam2
 8  if [ $szam1 != $szam2 ] ; then
 9       echo " A kétszám nem egyenlő"
10  elif [ $szam1 == $szam2 ] ; then
11       echo " A kétszám egyenlő"
12
13  fi
```

```
tibi@tibi-server:~/test$  ./elagazas1.sh
   Ez a bash script elágazásokat mutat be
Kérem az első számot
22
Kérem az masodik számot
22
 A kétszám egyenlő
tibi@tibi-server:~/test$  ./elagazas1.sh
   Ez a bash script elágazásokat mutat be
Kérem az első számot
22
Kérem az masodik számot
23
 A kétszám nem egyenlő
```

Példa:Számok egyenlőség vizsgálata

```
1  #!/bin/bash
2  echo   "   Ez a bash script számok
3   egyenlöségét mutat be"
4  sleep 1
5  echo "Kérem az első szamot"
6  read szam1
7  echo "Kérem az masodik szamot"
8  read szam2
9
10 if    [ $szam1 -eq $szam2 ] ; then
11      echo " a kétszám egyenő"
12 elif [ $szam1 -gt $szam2 ] ; then
13      echo " Az első szám nagyobb"
14 elif [ $szam1 -lt $szam2 ] ; then
15      echo " Az első szám a kissebb"
16 else
17      echo " Isten tudja valami geba van"
18 fi
```

Kimenet

```
tibi@tibi-server:~/test$ bash elagazas3.sh
   Ez a bash script számok
 egyenlőségeket mutat be
Kérem az első szamot
10
Kérem az masodik szamot
20
 Az első szám a kissebb
tibi@tibi-server:~/test$ bash elagazas3.sh
   Ez a bash script számok
 egyenlőségeket mutat be
Kérem az első szamot
21
Kérem az masodik szamot
11
 Az első szám nagyobb
tibi@tibi-server:~/test$ bash elagazas3.sh
   Ez a bash script számok
 egyenlőségeket mutat be
Kérem az első szamot
22
Kérem az masodik szamot
22
 a kétszám egyenő
```

Példa : sztringek vizsgálata

```
#!/bin/bash
echo  "  Ez a bash script elágazásokat mutat be"
sleep 1
echo "Kérem az első nevet"
read nev1
echo "Kérem az masodik nevet"
read nev2

if [ -z $nev1 ]||[ -z $nev2 ] ; then
    echo " nem adtál meg nevet"
elif [ $nev1 ]||[ $nev2 ] ; then
    echo " A nevek  léteznek"


    if [ $nev1 != $nev2 ] ; then
    echo " A két név  nem azonos"
    else [ $nev1 == $nev2 ]
    echo " A nevek  azonosak"
    fi

fi
```

Kimenetek:

```
tibi@tibi-server:~/test$ ./elagazas2.sh
   Ez a bash script elágazásokat mutat be
Kérem az első nevet

Kérem az masodik nevet

 nem adtál meg nevet
tibi@tibi-server:~/test$ ./elagazas2.sh
   Ez a bash script elágazásokat mutat be
Kérem az első nevet
béla
Kérem az masodik nevet
laci
 A nevek  léteznek
 A két név  nem azonos
tibi@tibi-server:~/test$ ./elagazas2.sh
   Ez a bash script elágazásokat mutat be
Kérem az első nevet
laci
Kérem az masodik nevet
laci
 A nevek  léteznek
 A nevek  azonosak
```

Fájlvizsgálat 1.

Az alábbi kapcsolókkal a rendszerben megtalálható fájlokra kérdezhetünk rá.

-b fájlnév	blokkeszköz-meghajtó	# Block
-c fájlnév	karaktereszköz-meghajtó	# Character
-d fájlnév	könyvtár	# Directory
-f fájlnév	szabályos állomány	# File
-l fájlnév	közvetett hivatkozás	# Link
-p fájlnév	csővezeték	# Pipe line
-e fájlnév	létezik	# Exists
-G fájlnév	saját csoportba tartozik	# Group
-O fájlnév	saját tulajdon	# Own
-s fájlnév	fájlnév üres	# String

Fájlvizsgálat 2.

-r fájlnév olvasható # Read

-w fájlnév írható # Write

-x fájlnév futtatható # eXecute

-**h** fájlnév igaz, ha a fájlnév létező szimbolikus lánc neve

fájl1 **–nt** fájl2 a fájl1 újabb, mint a fájl2 # Newer

fájl1 **–ot** fájl2 a fájl1 régebbi, mint a fájl2 # Older

fájl1 -**ef** fájl2 a fájl1 és fájl2 azonos állományt jelöl # Equal

Példák:

Az alábbi script megnézi, hogy létezik-e a valami.txt fájl

```
#!/bin/bash

if [ -e valami.txt ] ; then
    echo "A fájl létezik."
Fi
```

Az alábbi script megnézi, hogy a VAR üres-e

```
#!/bin/bash

VAR="Lala"
if [ ! -s $VAR ] ; then
    echo "A változó tartalma nem üres."
fi
```

A case elágazás

A case elágazás egy változó értékétől függően hajtja végre valamelyik parancsot. Formája:

```
case szo in
        szo1a | szo1b )
                utasitasblokk1 ;;
        szo2 )
                utasitasblokk2 ;;
        * )
                utasitasblokk0  ;;      # default ág (ha egyik sem)
esac                                    # fun fact: ez a "case" fordítva
```

Példa:

Az alábbi script az egy darab parancssori paraméter alapján választ az alábbi opciókból, majd kiírja a képernyőre az illeszkedő esetet

```
#!/bin/bash
case   $1    in
          1 )
                    echo "Hinnye, de nagyon hetfo"  ;;
          2 )
                    echo "Szinte hetfo„                ;;
          3 | 4 )
                    echo "Na kozeledunk„               ;;
          5 | 6 )
                    echo "Azert ez mar valami„         ;;
          7 )
                    echo "Jajj, anyam..„               ;;
          * )       echo "rossz input„                 ;;
esac
```

Az exit utasítás

- Úgy lehet elképzelni, mint C-ben a return. Ahol kiadjuk ezt a parancsot ott terminál a program. Közvetlen az exit utasítás után meg lehet adni a kilépési állapotot (egy szám).

- Ahogy C-ben, itt is a 0 jelzi, ha minden rendben volt, 1 pedig ha vmi hiba történt. 2-255-ig mi magunk is adhatunk meg valamilyen kilépési státuszt.

- Rövid emlékeztető: a **$?** az előző parancs exit állapotára hivatkozó változó. Segítségével könnyedén leellenőrizhető, hogy az előző parancs sikeresen lefutott-e. Ha igen, akkor 0 értéket vesz fel, ellenkező esetben 1 lesz az értéke.

Példa:

Az alábbi script első sorában megpróbálunk létrehozni egy "mappa" nevű könyvtárat.

- Ha sikerül, akkor a $? érteké 0 lesz es az if ág teljesül, ellenkező esetben az else ág fut le.
- Például akkor lehet sikertelen, ha az adott mappa mar létezik. (Próbáljuk ki mindkét esetben.)

```
#!/bin/bash
mkdir mappa
if [ $? -eq 0 ] ; then
        echo "Sikeres volt az elozo muvelet."
else
        echo "Sikertelen volt az elozo muvelet."
        exit
fi
echo "Ha sikertelen volt, akkor ide mar nem jut el a program."
```

A break és continue utasítások

- A break-et és a continue-t már jól ismerhetjük C-ből. A szerepük itt is ugyanaz.
- A **break** -et kiadva azonnal kilépünk az adott ciklusból,
- A **continue** -val pedig a ciklus következő iterációjára lépünk

Példa:

- Az következő script egy végtelen ciklust tartalmaz, mely 1-től felfelé összeadja a számokat.
- Mielőtt hozzá adnánk az összeghez az aktuális számot, megnézzük, hogy az összeg meghaladta-e a 100-as érteket. Amennyiben igen, a ciklusból kilépünk.
- Amennyiben az adott szám osztható 5-tel, akkor nem adjuk hozzá az összeghez, hanem a következő iterációra lépünk.

```
#!/bin/bash
osszeg=0
val=0
while true ; do
        let val=$val+1
        echo val: $val
        if [ $osszeg -ge 100 ] ; then
                break
        fi
                let maradek=$val%5
        if [ $maradek -eq 0 ] ; then
                continue
        fi
        let osszeg=$osszeg+$val
        echo osszeg: $osszeg
done
echo $val
echo $osszeg
```