

Bash script alapok 4. rész

Ciklusok



Linux alapok

Varga Tibi 2020

Ciklusok és a Bash

A ciklusok arra jók, hogy egy parancsot vagy parancs-csomagot ismétlődően végrehajtsunk, amíg egy, általában a programozó által definiált feltételt el nem érünk.

A Bash háromféle ciklust ismer:

- for
- while
- until

A for ciklus - index alapú

```
for (( i=0; i<=$N; i++ )); do  
    parancs  
done
```

Az alábbi script kiírja 0-tól az első parancssori paraméter méretéig a számokat:

```
#!/bin/bash  
  
for (( i=0; i<$1; i++ )); do  
    echo $i  
done
```

```
#!/bin/bash  
START=5  
END=0  
echo „vissza számlálás”  
  
for (( c=$START; c>=$END; c-- )); do  
    echo -n "$c "  
    sleep 1  
done  
echo  
echo "Bumm!"
```

Példa: szöveg fájl tartalmának soronként kiíratása

```
#!/bin/bash
# szöveg fájl soronkénti kiíratása

echo "add meg a szöveg fájl nevét !"
"
read fnev
sorszam=`wc -l <$fnev`

for (( sor=1; sor!=$[sorszam+1]; sor++ )); do

    head -$sor $fnev | tail -1

done
```

```
tibi@tibi-server:~/test/ciklusok$ ./for2.sh
add meg a szöveg fájl nevét !

nevek.txt
Tibi
Tomi
Jani
Boci
Juci
```

A for ciklus - lista alapú

for változo **in** lista ; **do**

...

done

A változó rendre felveszi a lista elemeinek értékét és minden értékkel végrehajtódik az utasítás blokk minden utasítása.

A változó lehet:

- BASH változó neve, vagy
- Környezeti változó neve.

A lista lehet:

- Szavak szóközökkel elválasztott listája,
- Állománynév helyettesítő karakterekkel megadott állománylista,
- Változók listája,
- Végrehajtandó parancs a `$(...)` vagy ``...`` szerkezettel.

Példa for lista:

Az alábbi script végig megy a for-ban megadott listán (hétfő kedd szerda) és kiírja őket a képernyőre

```
#!/bin/bash  
for DAY in hetfo kedd szerda ; do  
    echo $DAY  
done
```

Az alábbi script végig megy a parancssori paramétereken és kiírhatja őket

```
#!/bin/bash  
for param in $* ; do  
    echo $param  
done
```

for lista számokból

- A ciklus annyiszor fut le ahány elem van a listában

```
#!/bin/bash
# Üdv a Bash világában 5 ször

for i in 1 2 3 4 5 ; do
    echo Szia üdv a Bash világában $i-szor
done
```

```
tibi@tibi-server:~/test/ciklusok$ ./for3.sh
Szia üdv a Bash világában 1-szor
Szia üdv a Bash világában 2-szor
Szia üdv a Bash világában 3-szor
Szia üdv a Bash világában 4-szor
Szia üdv a Bash világában 5-szor
```

for lista elemekkel

- A ciklus annyiszor fut le ahány elem van a listában és mind egy hogy mit tartalmaz értékben!

```
#!/bin/bash
# Üdv a Bash világában 5 ször

for i in M i n d 1 mi ; do
    echo Szia üdv a Bash világában $i-szor
done
```

```
tibi@tibi-server:~/test/ciklusok$ ./for33.sh
Szia üdv a Bash világában M-szor
Szia üdv a Bash világában i-szor
Szia üdv a Bash világában n-szor
Szia üdv a Bash világában d-szor
Szia üdv a Bash világában 1-szor
Szia üdv a Bash világában mi-szor
```


for - lista alapú tartománnyal

- A következő megoldásban tartomány beállítás látható **{1..5}** **{12..2}**

```
#!/bin/bash
# Üdv a Bash világában 5 ször

for i in {1..5} ; do
    echo Szia üdv a Bash világában $i-szor
done
```

```
tibi@tibi-server:~/test/ciklusok$ ./for4.sh
Szia üdv a Bash világában 1-szor
Szia üdv a Bash világában 2-szor
Szia üdv a Bash világában 3-szor
Szia üdv a Bash világában 4-szor
Szia üdv a Bash világában 5-szor
```

for - lista alapú tartomány lépésértékekkel

- A következő megoldásban tartomány beállítás lépésértékekkel, a beállításához a { START .. END .. INCREMENT } szintaxissal:

```
#!/bin/bash
# Üdv a Bas világában 8-szor 2-es lépésközzel

for i in {0..8..2} ; do
    echo Szia üdv a Bash világában $i-szor
done
```

```
tibi@tibi-server:~/test/ciklusok$ ./for5.sh
Szia üdv a Bash világában 0-szor
Szia üdv a Bash világában 2-szor
Szia üdv a Bash világában 4-szor
Szia üdv a Bash világában 6-szor
Szia üdv a Bash világában 8-szor
```

for - lista a **seq** paranccsal (elavult)

- A következő megoldásban tartomány beállítás **seq** paranccsal:

\$(seq tól> <lépés> > ig)

```
#!/bin/bash
# Üdv a Bash világában
seq parancsal.: $(seq 1 2 20)

for i in $(seq 1 2 20) ; do
    echo Szia üdv a Bash világában $i-szor
done
```

```
tibi@tibi-server:~/test/ciklusok$ ./for6.sh
Szia üdv a Bash világában 1-szor
Szia üdv a Bash világában 3-szor
Szia üdv a Bash világában 5-szor
Szia üdv a Bash világában 7-szor
Szia üdv a Bash világában 9-szor
Szia üdv a Bash világában 11-szor
Szia üdv a Bash világában 13-szor
Szia üdv a Bash világában 15-szor
Szia üdv a Bash világában 17-szor
Szia üdv a Bash világában 19-szor
```

While ciklus

```
while utasítás ; do  
    ...parancs...  
done
```

A ciklusfeltétel függhet előzőleg deklarált változóktól. A példában a számláló kiírja az egész számokat 0 és 10 közt.

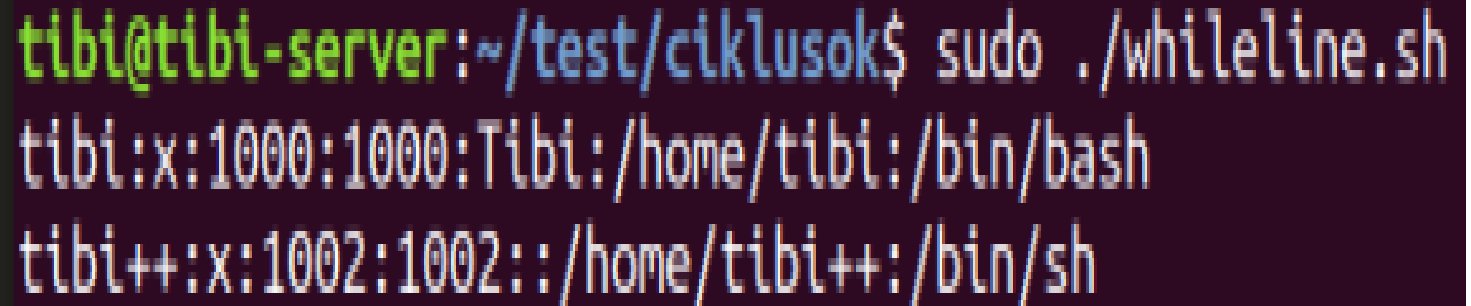
```
#!/bin/bash  
count=0  
while [ $count -le 10 ]; do  
    echo „sz é $count”  
    let ++count  
done
```

```
tibi@tibi-server:~/test/ciklusok$ ./while1.sh  
számláló értéke 0  
számláló értéke 1  
számláló értéke 2  
számláló értéke 3  
számláló értéke 4  
számláló értéke 5  
számláló értéke 6  
számláló értéke 7  
számláló értéke 8  
számláló értéke 9  
számláló értéke 10
```

Egy fájl soronként beolvasása

- A while ciklus egyik leggyakoribb használata egy fájl, adatfolyam vagy változó olvasása soronként.
- Íme egy példa, amely soronként beolvassa az / etc / passwd fájlt, és minden sort kiír amiben a „tibi szerepel”:

```
#!/bin/bash
file=/etc/passwd
while read -r line; do
    echo $line | egrep „tibi”
done < "$file"
```



```
tibi@tibi-server:~/test/ciklusok$ sudo ./whileline.sh
tibi:x:1000:1000:Tibi:/home/tibi:/bin/bash
tibi++:x:1002:1002::/home/tibi++:/bin/sh
```

- Ahelyett, hogy a while ciklus egy feltétellel irányítanánk, bemeneti átirányítást (<"\$ fájl") használunk, hogy egy fájlt továbbítsunk az read parancshoz, amely a ciklust vezérli. A ciklus addig fog futni, amíg az utolsó sort el nem olvassa.
- A fájl soronként olvasásakor mindig használjuk a **read -r** opciót, hogy megakadályozzuk a vissza-per (\)kilépési karakterként történő meghívását.

Until ciklus

A while mellett az **until** ciklust is használhatjuk, ami nagyon hasonlít a while-ra. A két ciklus szintaxisa megegyezik, bár a ciklusfeltétel pont a while ellentéte. **Ebben az esetben a ciklusmag akkor hajtódik végre, ha a feltétel hamis.**

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

```
count=0
```

```
until [ $count -gt 7 ] ; do
```

```
    echo „a számláló:$count”
```

```
    let count++
```

```
done
```

```
tibi@tibi-server:~/test/ciklusok$ ./until1.sh
a számláló:0
a számláló:1
a számláló:2
a számláló:3
a számláló:4
a számláló:5
a számláló:6
a számláló:7
```

A break utasítás

- A **break** utasítás megszünteti az aktuális ciklust, és átadja a programvezérlést a megszüntetett ciklust követő parancsoknak. Ezt általában a ciklus lezárására használják, amikor egy bizonyos feltétel teljesül.
- A következő példában a ciklus végrehajtása megszakad, ha az aktuális iterált elem 6-al osztható lesz.

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

```
i=1
```

```
while [[ $i -lt 15 ]]; do
```

```
echo A szám: $i
```

```
if [[ $i%6 -eq 0 ]]; then
```

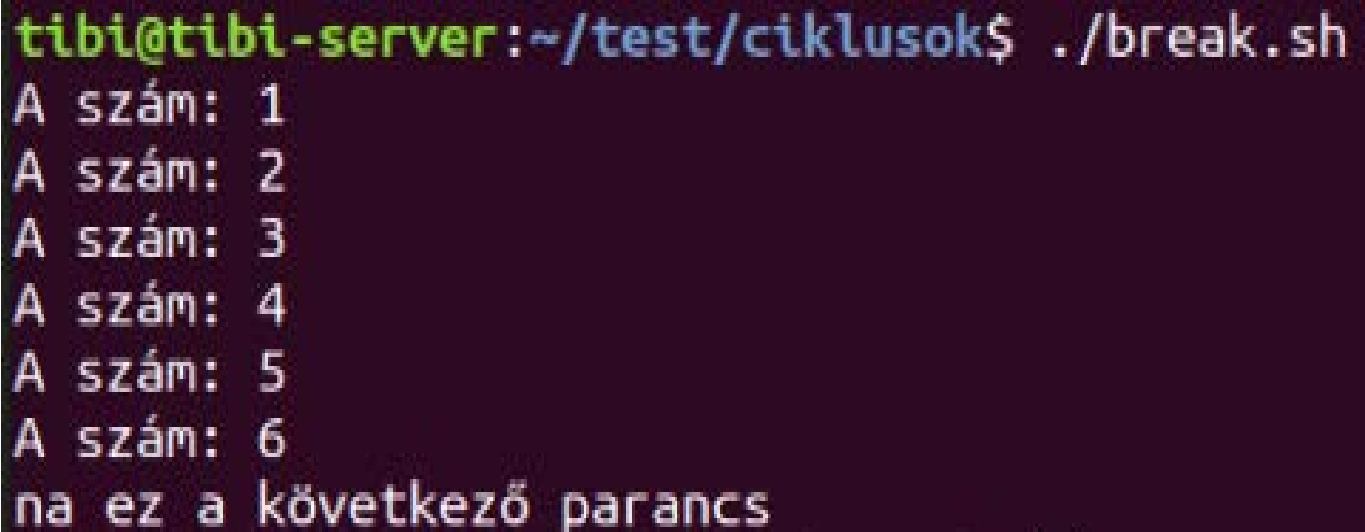
```
    break
```

```
    let i++
```

```
fi
```

```
done
```

```
echo na ez a következő parancs
```



```
tibi@tibi-server:~/test/ciklusok$ ./break.sh
A szám: 1
A szám: 2
A szám: 3
A szám: 4
A szám: 5
A szám: 6
na ez a következő parancs
```

A continue utasítás

- A **continue** utasítás kilép a ciklus aktuális iterációjáról, és a programvezérlést a ciklus következő iterációjára továbbítja.
- Az alábbiakban, ha az aktuális iterált elem páros, a folytatás utasítás visszaadja a végrehajtást a ciklus elejére nem ír ki semmit és folytatja a következő iterációval.

```
#!/bin/bash
i=0
while [ $i -lt 15 ]; do
    let i++
    if [[ $i%2 -eq 0 ]]; then
        continue
    fi
    echo „A számok: $i”
done
echo ”na ez a vége”
```

```
tibi@tibi-server:~/test/ciklusok$ ./continue.sh
A számok: 1
A számok: 3
A számok: 5
A számok: 7
A számok: 9
A számok: 11
A számok: 13
A számok: 15
A számok: 17
na ez a vége
```