

OS

3 paskaita

# For ciklo sintaksė

```
#!/bin/bash for i in 1 2 3 4 5 do  
echo "Welcome $i times"  
done
```

**bash version 3.0+, tik šioje versijoje  
yra palaikomas**

```
#!/bin/bash for i in {1..5} do  
echo "Welcome $i times"  
done
```

# For ciklo sintaksė

**Bash v4.0+**, šioje versijoje yra galimybė nurodyti žingsnius, pvz: **{0..10..2}**, nuo nulio elemento iki 10 yra išvedamas kas 2 elementas.

```
#!/bin/bash echo "Bash version  
${BASH_VERSION}..." for i in {0..10..2} do  
echo "Welcome $i times" done
```

# C stiliaus for ciklo pavyzdys „bash’e“

- Struktūra yra sudaryta iš 3 išraiškų:
- (EXP1) – Inicializatorius.
- (EXP2) – ciklo testas arba sąlyga.
- (EXP3) – skaitliuko išraiška.

```
for (( EXP1; EXP2; EXP3 )) do  
command1 command2 command3  
done
```

```
#!/bin/bash for (( c=1; c<=5; c++ )) do  
echo "Welcome $c times"  
done
```

# Begalinis for ciklas naudojant išraiškas

Begalinis ciklas yra aprašomas taip, tiesiog yra nurodos tuščios išraiškos atskirtos kabliataškiais.

```
#!/bin/bash for (( ; ; )) do  
echo "infinite loops [ hit CTRL+C to stop]"  
done
```

# For ciklas išankstinis nutraukimas struktūra

Išankstinis išėjimas iš for ciklo naudojant break; kaip ir while until cikluose

```
for I in 1 2 3 4 5 do  
statements1 #Executed for all values of 'I', up to  
a disaster-condition if any.  
statements2  
if (disaster-condition)  
then  
break #Abandon the loop.  
fi  
statements3 #While good and, no disaster-condition.  
done
```

# For ciklo panaudojimas su išankstiniu nutraukimu

- Ciklas išveda visu failų esančių /etc/ direktorijoje ir veikia iki tol kol nėra surastas konkretus failas šiuo atveiu: .../etc/resolv.conf“

```
#!/bin/bash
for file in /etc/*
do
if [ "${file}" == "/etc/resolv.conf" ] then
countNameservers=$(grep -c nameserver /etc/resolv.conf)
echo "Total ${countNameservers} nameservers defined in ${file}"
break
fi
done
```

# Išankstinis tęsimas (angl. continue) for cikle

```
for I in 1 2 3 4 5 do
statements1 #Executed for all values of 'I', up to a
disaster-condition if any.
statements2
if (condition)
then
continue #Go to next iteration of I in the loop and
skip statements3
fi
statements3
done
```



# Išankstinis tęsimas (angl. continue) for cikle

Daro failų kopijas, jeigu kopija neegzistuoja nurodyto failo komandinėje eilutėje

```
#!/bin/bash
FILES="$@"
for f in $FILES
do
#if .bak backup file exists, read next file
if [ -f ${f}.bak ]
then
echo "Skipping $f file..."
continue # read next file and skip cp command
fi # we are hear means no backup file exists,
just use cp command to copy file /bin/cp $f $f.bak
done
```

- For ciklo pavyzdžiai nuroda:

<https://www.youtube.com/watch?v=ocXb3qeg7Es>

# Ciklas cikle For

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

```
#skriptas spausdina reikšmę 5 kartus
```

```
for (( i = 1; i <= 5; i++ )) ### Išorinis ciklas skirtas (i) eilutėms ###
```

```
do
```

```
for (( j = 1 ; j <= 5; j++ )) ### Vidinis ciklas skirtas (j) stulpeliams ###
```

```
do
```

**echo** -n "Ši " ###perbėga per visus stulpelius ir užpildo eilutės reikšmę, 1 eilutė yra užpildoma 1 reikšmėmis iki tol kol j <= 5 (kai j daugiau už 5 yra pereinama į sekančią eilutę) ir nauja i reikšmė i = 2, tada yra pereinama prie sekančios ir t.t.

```
done
```

```
echo "" ##### Atspausdina naują eilutę ###
```

```
done
```

**Užpildoma matrica 5 x 5 reikšmėmis i (šiuo atveju žr. apačioje ), kadangi for cikle yra nurodoma eilučių skaičius 5 ir stulpelių skaičius 5.**

Rezultatas

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5

Ciklas cikle for išklotinė. Pirmas matricos elementas  $i = 1$  (eilutė),  $j = 1$  (stulpelis), tai pora  $(i1, j1) = 1$ , kadangi nurodėme įrašyti  $i$  reikšmę...  $(i2, j2) = 1$  ir t.t. pvz. pasiekūs  $(i3, j1) = 3$ , kadangi eilutės  $i$  reikšmė yra lygi 3 ir t.t. žemiau yra pateikiama ciklas cikle for išklotinė.

1 (  $i=1$   $j=1$ ) 1 (  $i=1$   $j=2$ ) 1 (  $i=1$   $j=3$ ) 1 (  $i=1$   $j=4$ ) 1 (  $i=1$   $j=5$ )

2 (  $i=2$   $j=1$ ) 2 (  $i=2$   $j=2$ ) 2 (  $i=2$   $j=3$ ) 2 (  $i=2$   $j=4$ ) 2 (  $i=2$   $j=5$ )

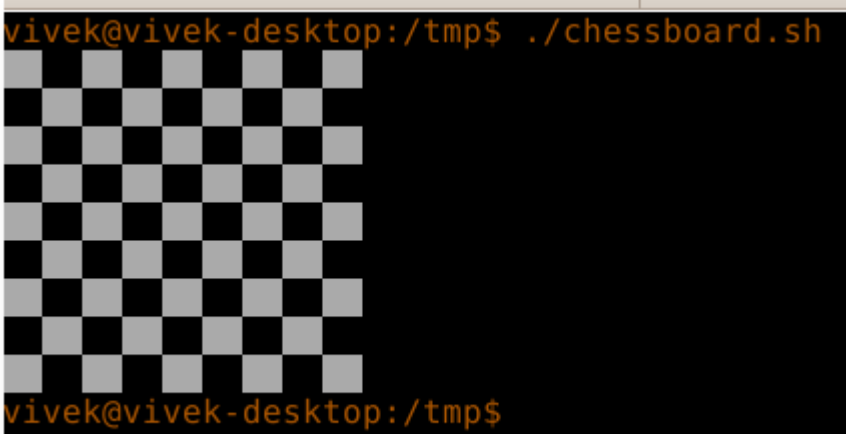
3 (  $i=3$   $j=1$ ) 3 (  $i=3$   $j=2$ ) 3 (  $i=3$   $j=3$ ) 3 (  $i=3$   $j=4$ ) 3 (  $i=3$   $j=5$ )

4 (  $i=4$   $j=1$ ) 4 (  $i=4$   $j=2$ ) 4 (  $i=4$   $j=3$ ) 4 (  $i=4$   $j=4$ ) 4 (  $i=4$   $j=5$ )

5 (  $i=5$   $j=1$ ) 5 (  $i=5$   $j=2$ ) 5 (  $i=5$   $j=3$ ) 5 (  $i=5$   $j=4$ ) 5 (  $i=5$   $j=5$ )

# Šachmatų lentos pavyzdys

```
#!/bin/bash for (( i = 1; i <= 8; i++ )) ### Outer for  
loop ###  
do  
for (( j = 1 ; j <= 8; j++ )) ### Inner for loop ###  
do  
total=$(( $i + $j )) # total  
tmp=$(( $total % 2 )) # modulus # Find out odd and even  
number and change the color # alternating colors using  
odd and even number logic  
if [ $tmp -eq 0 ];  
then  
echo -e -n "\033[47m "  
else  
echo -e -n "\033[40m "  
fi  
done  
echo "" #### print the new line ###  
done
```



A terminal window with a black background and orange text. The prompt is 'vivek@vivek-desktop:/tmp\$'. The command './chessboard.sh' has been executed, resulting in an 8x8 chessboard pattern of alternating black and white squares. The prompt is now 'vivek@vivek-desktop:/tmp\$' again.

# While ciklo sintaksė

```
while [ condition ]  
do  
command1  
command2 .. ....  
commandN  
done
```

# While ciklo pavyzdys

```
#!/bin/bash # set n to 1
n=1
# continue until $n equals 5
while [ $n -le 5 ] do
echo "Welcome $n times."
n=$(( n+1 )) # increments $n
done
```

# While ciklas naudojant išraiškas pagerinti skaitomumui

```
#!/bin/bash  
n=1  
while (( $n <= 5 ))  
do  
  echo "Welcome $n times."  
  n=$(( n+1 ))  
done
```

# While ciklas failo nuskaitymas

```
#!/bin/bash file=/etc/resolv.conf
while IFS= read -r line
do # echo line is stored in $line
echo $line
done < "$file"
```

## REZULTATAS

```
nameserver 127.0.0.1
nameserver 192.168.1.254
nameserver 4.2.2.1
```



# While ciklas failo nuskaitymas

```
#!/bin/bash
file=/etc/resolv.conf # set field
separator to a single white space while
IFS=' ' read -r f1 f2
do
echo "field # 1 : $f1 ==> field #2 : $f2"
done < "$file"
```

## REZULTATAS

```
field # 1 : nameserver ==> field #2 : 127.0.0.1
field # 1 : nameserver ==> field #2 : 192.168.1.254
field # 1 : nameserver ==> field #2 : 4.2.2.1
```

# Masyvų pavyzdžiai bash

```
array=( one two three )  
files=( "/etc/passwd" "/etc/group" "/etc/hosts" )  
limits=( 10, 20, 26, 39, 48)
```

```
printf "%s\n" "${array[@]}"  
printf "%s\n" "${files[@]}"  
printf "%s\n" "${limits[@]}"
```

# \$i masyvo visų elementų saugojimui (laikymui)

```
for i in "${arrayName[@]}"  
do : # do whatever on $i  
done
```

\$i will hold each item in an array. Here is a sample working script:

```
#!/bin/bash  
# declare an array called array and define 3  
vales  
array=( one two three )  
for i in "${array[@]}"  
do  
echo  
$i done
```

# Begalinio while ciklo pavyzdys

```
#!/bin/bash # Recommend syntax for  
setting an infinite while loop  
while :  
do  
echo "Do something; hit [CTRL+C]  
to stop!"  
done
```

# Begalinis while ciklas skirtas meniu

```
#!/bin/bash
# set an infinite loop
while :
do
clear
# display menu
echo "Server Name - $(hostname)" echo "-----"
echo " M A I N - M E N U" echo "-----"
echo "1. Display date and time." echo "2. Display
what users are doing." echo "3. Display network connections."
echo "4. Exit" # get input from the user read -p "Enter your
choice [ 1 -4 ] " choice
# make decision using case..in..esac
case $choice in 1) echo "Today is $(date)" read -p "Press
[Enter] key to continue..." readEnterKey ;; 2) w read -p
"Press [Enter] key to continue..." readEnterKey ;; 3) netstat
-nat read -p "Press [Enter] key to continue..." readEnterKey
;; 4) echo "Bye!" exit 0 ;; *) echo "Error: Invalid option..."
read -p "Press [Enter] key to continue..." readEnterKey ;;
esac
done
```