

Introdução à Ciência da Computação

Shell Script – parte IV

Professor Iago Augusto de Carvalho
iago.carvalho@unifal-mg.edu.br

Teste de condições compostas

A declaração if-then permite usar lógica booleana para combinar testes.

Podemos usar dois operadores booleanos:

AND

[condição1] && [condição2]

OR

[condição1] || [condição2]

Testar de comparações compostas

```
#!/bin/bash
#Testar comparações compostas.
#Vamos verificar se o usuário logado atualmente é adriana
#e se ela tem permissão de escrito no seu arquivo .bashrc

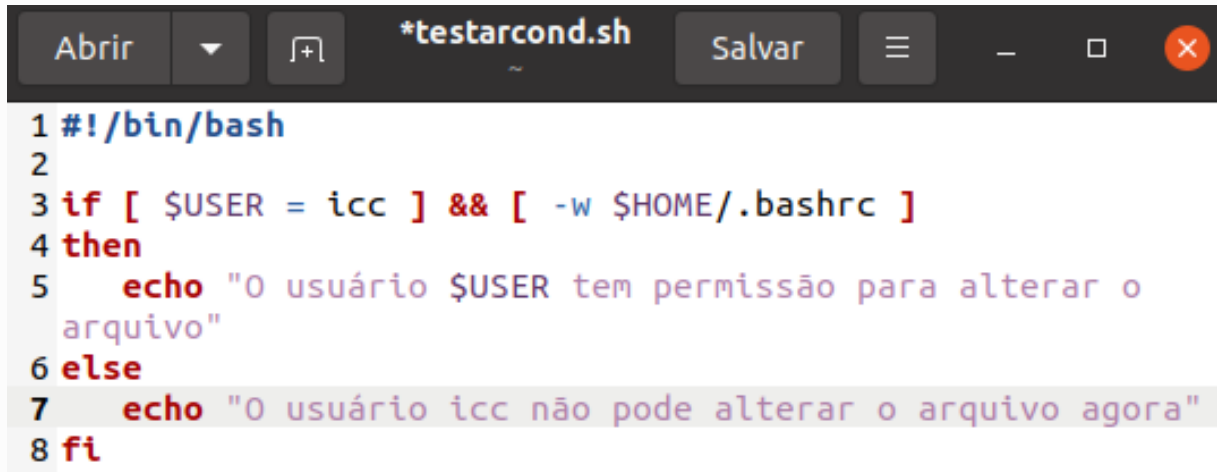
if[ $USER = adriana ] && [ -w $HOME/.bashrc ]
then
    echo "O usuários $USER tem permissão para alterar o arquivo"
else
    echo "O usuário adriana não pode alterar o arquivo agora"
fi
```

Testar de comparações compostas

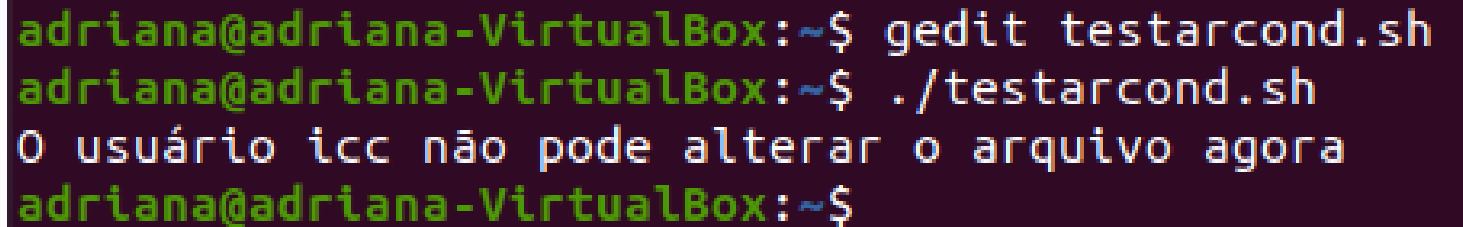
```
Abrir  *testarcond.sh  Salvar  ≡  
1 #!/bin/bash  
2  
3 if [ $USER = adriana ] && [ -w $HOME/.bashrc ]  
4 then  
5     echo "O usuário $USER tem permissão para alterar o arquivo"  
6 else  
7     echo "O usuário adriana não pode alterar o arquivo agora"  
8 fi
```

```
adriana@adriana-VirtualBox: ~  
adriana@adriana-VirtualBox:~$ gedit testarcond.sh  
adriana@adriana-VirtualBox:~$ chmod 755 testarcond.sh  
adriana@adriana-VirtualBox:~$ ./testarcond.sh  
O usuário adriana tem permissão para alterar o arquivo  
adriana@adriana-VirtualBox:~$
```

Testar de comparações compostas



```
1 #!/bin/bash
2
3 if [ $USER = icc ] && [ -w $HOME/.bashrc ]
4 then
5     echo "O usuário $USER tem permissão para alterar o
    arquivo"
6 else
7     echo "O usuário icc não pode alterar o arquivo agora"
8 fi
```



```
adriana@adriana-VirtualBox:~$ gedit testarcond.sh
adriana@adriana-VirtualBox:~$ ./testarcond.sh
O usuário icc não pode alterar o arquivo agora
adriana@adriana-VirtualBox:~$
```

Comando case

Esse comando verifica condições múltiplas que podem ocorrer em uma variável, em um formato parecido com uma lista.

A estrutura case substitui e simplifica o uso do if-then-elif com várias declarações.

O comando case compara o valor de uma variável ou expressão com os valores da lista criada.

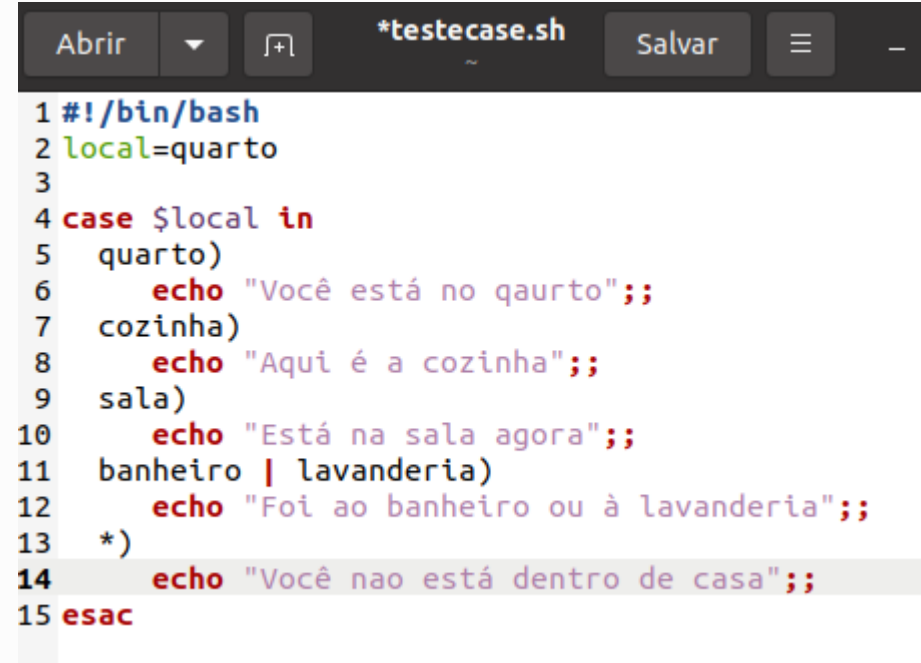
Se o valores forem iguais, o shell executará os comandos especificados para o valor.

Comando case – sintaxe

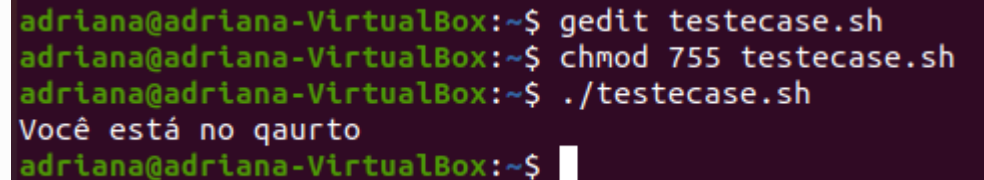
```
case <variável> in
    valor1)
        comandos 1 ;;
    valor2)
        comandos 2 ;;
    valor3 | valor4)
        comandos 3 e 4 ;;
    *)
        comandos-padrão ;;
esac
```

Comando case

```
#!/bin/bash
#Usando o comando case
local=quarto
case $local in
    quarto)
        echo "Você está no quarto;;"
    cozinha)
        echo "Aqui é a cozinha;;"
    sala)
        echo "Está na sala agora;;"
    banheiro | lavanderia)
        echo "Foi ao banheiro ou à lavanderia;;"
    *)
        echo "Você não está dentro de casa;;"
esac
```



```
1 #!/bin/bash
2 local=quarto
3
4 case $local in
5     quarto)
6         echo "Você está no qaurto;;"
7     cozinha)
8         echo "Aqui é a cozinha;;"
9     sala)
10        echo "Está na sala agora;;"
11    banheiro | lavanderia)
12        echo "Foi ao banheiro ou à lavanderia;;"
13    *)
14        echo "Você nao está dentro de casa;;"
15 esac
```



```
adriana@adriana-VirtualBox:~$ gedit testecase.sh
adriana@adriana-VirtualBox:~$ chmod 755 testecase.sh
adriana@adriana-VirtualBox:~$ ./testecase.sh
Você está no qaurto
adriana@adriana-VirtualBox:~$
```

Comando case

```
Abrir  ▾  +  *testecase.sh  Salvar  ≡

1 #!/bin/bash
2 local=sala
3
4 case $local in
5   quarto)
6     echo "Você está no qaurto";;
7   cozinha)
8     echo "Aqui é a cozinha";;
9   sala)
10    echo "Está na sala agora";;
11   banheiro | lavanderia)
12    echo "Foi ao banheiro ou à lavanderia";;
13   *)
14    echo "Você nao está dentro de casa";;
15 esac

adriana@adriana-VirtualBox:~$ gedit testecase.sh
adriana@adriana-VirtualBox:~$ ./testecase.sh
Está na sala agora
adriana@adriana-VirtualBox:~$
```

```
Abrir  ▾  +  testecase.sh  Salvar  ≡

1 #!/bin/bash
2 local=banheiro
3
4 case $local in
5   quarto)
6     echo "Você está no qaurto";;
7   cozinha)
8     echo "Aqui é a cozinha";;
9   sala)
10    echo "Está na sala agora";;
11   banheiro | lavanderia)
12    echo "Foi ao banheiro ou à lavanderia";;
13   *)
14    echo "Você nao está dentro de casa";;
15 esac

adriana@adriana-VirtualBox:~$ gedit testecase.sh
adriana@adriana-VirtualBox:~$ ./testecase.sh
Foi ao banheiro ou à lavanderia
adriana@adriana-VirtualBox:~$
```

```
Abrir  ▾  +  *testecase.sh  Salvar  ≡

1 #!/bin/bash
2 local=quintal
3
4 case $local in
5   quarto)
6     echo "Você está no qaurto";;
7   cozinha)
8     echo "Aqui é a cozinha";;
9   sala)
10    echo "Está na sala agora";;
11   banheiro | lavanderia)
12    echo "Foi ao banheiro ou à lavanderia";;
13   *)
14    echo "Você nao está dentro de casa";;
15 esac

adriana@adriana-VirtualBox:~$ gedit testecase.sh
adriana@adriana-VirtualBox:~$ ./testecase.sh
Você nao está dentro de casa
adriana@adriana-VirtualBox:~$
```


Estruturas de Repetição

O shell bash oferece três estruturas de repetição para criação de loops estruturados. São:

for

while

until

Comando for

O comando for permite criar um loop que itera através de uma série de valores.

Cada iteração executa um conjunto definido de comandos usando um dos valores da lista.

Sintaxe básica:

```
for valor in lista  
do  
    comandos  
done
```

A cada iteração, a variável *valor* contém o valor atual da lista

Comando for

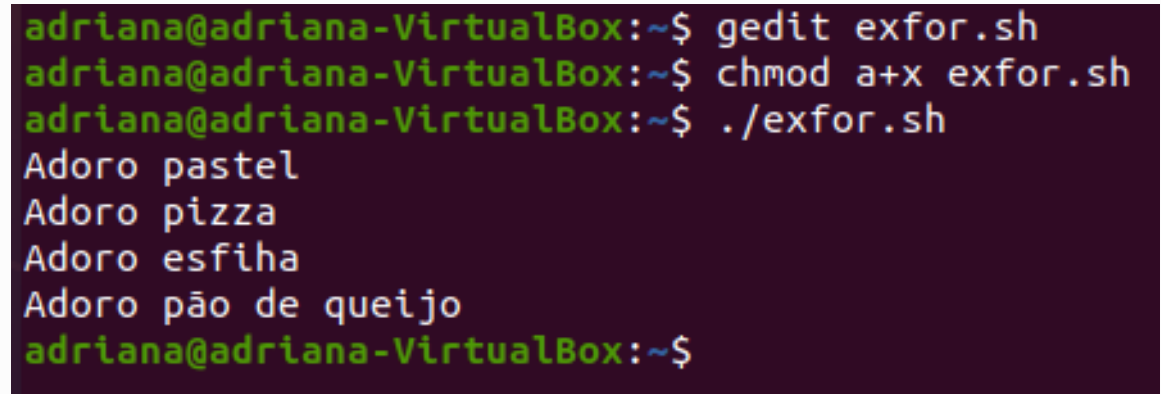
`#!/bin/bash`

`#Ler valores de uma lista, declarada no próprio comando for`

```
for valor in pastel pizza esfiha 'pão de queijo'
do
    echo Adoro $valor
done
```

A screenshot of a gedit editor window. The title bar shows 'Abrir', a dropdown arrow, a '+l' icon, '*exfor.sh', 'Salvar', a hamburger menu icon, and a '-' icon. The editor content shows a bash script with line numbers 1 through 6. Line 1 is '#!/bin/bash' in blue. Line 2 is empty. Line 3 is 'for valor in pastel pizza esfiha 'pão de queijo'' in green. Line 4 is 'do' in red. Line 5 is ' echo Adoro \$valor' in red. Line 6 is 'done' in red.

```
1 #!/bin/bash
2
3 for valor in pastel pizza esfiha 'pão de queijo'
4 do
5     echo Adoro $valor
6 done
```

A screenshot of a terminal window with a dark background. It shows the execution of the script 'exfor.sh'. The prompt is 'adriana@adriana-VirtualBox:~\$'. The commands entered are 'gedit exfor.sh', 'chmod a+x exfor.sh', and './exfor.sh'. The output of the script is 'Adoro pastel', 'Adoro pizza', 'Adoro esfiha', and 'Adoro pão de queijo'. The prompt returns to 'adriana@adriana-VirtualBox:~\$'.

```
adriana@adriana-VirtualBox:~$ gedit exfor.sh
adriana@adriana-VirtualBox:~$ chmod a+x exfor.sh
adriana@adriana-VirtualBox:~$ ./exfor.sh
Adoro pastel
Adoro pizza
Adoro esfiha
Adoro pão de queijo
adriana@adriana-VirtualBox:~$
```

Separador de campos: IFS

Caso os itens iterados sejam compostos por palavras separadas por espaço (como Pão integral), o comando `for` irá considerar cada palavra como um item separado.

Isso é devido á variável de ambiente `IFS` (Internal Field Separator), a qual define uma lista de caracteres que o shell `bash` usa como separadores de campos.

São eles, por padrão: espaço, tabulação e newline.

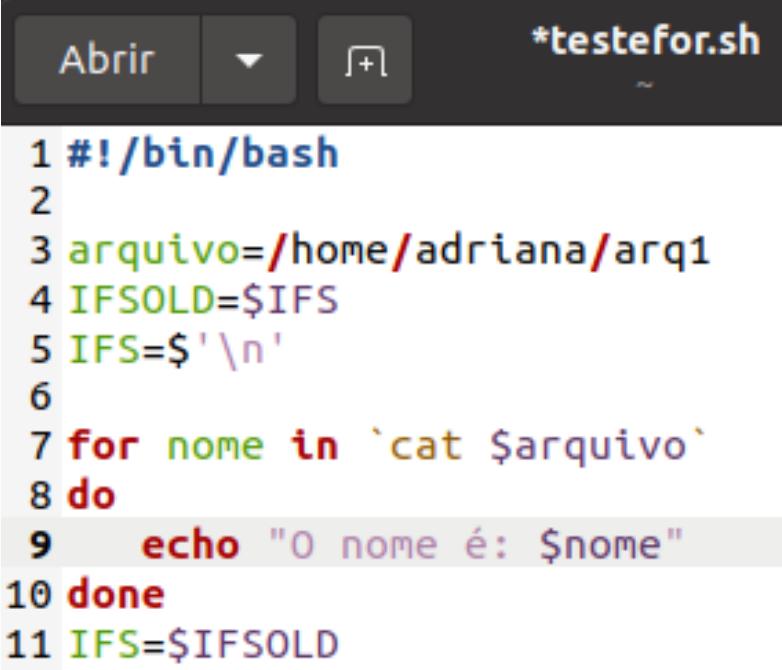
É possível alterar essa lista de separadores.

Comando for

```
#!/bin/bash
#Arquivo arq1 contém uma lista de nomes,
# um por linha,
#incluindo nomes compostos
arquivo=/home/adriana/arq1
IFSOLD=$IFS
IFS=$'\n'

for nome in `cat $arquivo`
do
    echo "O nome é: $nome"
done
IFS=$IFSOLD
```

```
adriana@adriana-VirtualBox:~$ cat < arq1
Adriana
Maria
Joaquim José
Catarina
Maria José
adriana@adriana-VirtualBox:~$
```



```
Abrir ▼ [+]
```

```
*testefor.sh
1 #!/bin/bash
2
3 arquivo=/home/adriana/arq1
4 IFSOLD=$IFS
5 IFS=$'\n'
6
7 for nome in `cat $arquivo`
8 do
9     echo "O nome é: $nome"
10 done
11 IFS=$IFSOLD
```

Comando for

```
adriana@adriana-VirtualBox: ~  
adriana@adriana-VirtualBox:~$ gedit testefor.sh  
adriana@adriana-VirtualBox:~$ ls  
'Área de Trabalho'  Documentos  Imagens  Música  snap  Vídeos  
arq1                Downloads  Modelos  Público  testefor.sh  
adriana@adriana-VirtualBox:~$ chmod 755 testefor.sh  
adriana@adriana-VirtualBox:~$ ./testefor.sh  
0 nome é: Adriana  
0 nome é: Maria  
0 nome é: Joaquim José  
0 nome é: Catarina  
0 nome é: Maria José  
adriana@adriana-VirtualBox:~$
```

Comando for

```
Abrir ▼ [+]
```

```
#testeFor.sh  
1 #!/bin/bash  
2  
3 arquivo=/home/adriana/arq1  
4  
5 for nome in `cat $arquivo`  
6 do  
7     echo "O nome é: $nome"  
8 done
```

```
adriana@adriana-VirtualBox:~$ cat < arq1  
Adriana  
Maria  
Joaquim José  
Catarina  
Maria José  
adriana@adriana-VirtualBox:~$
```

```
[+] adriana@adriana-VirtualBox: ~
```

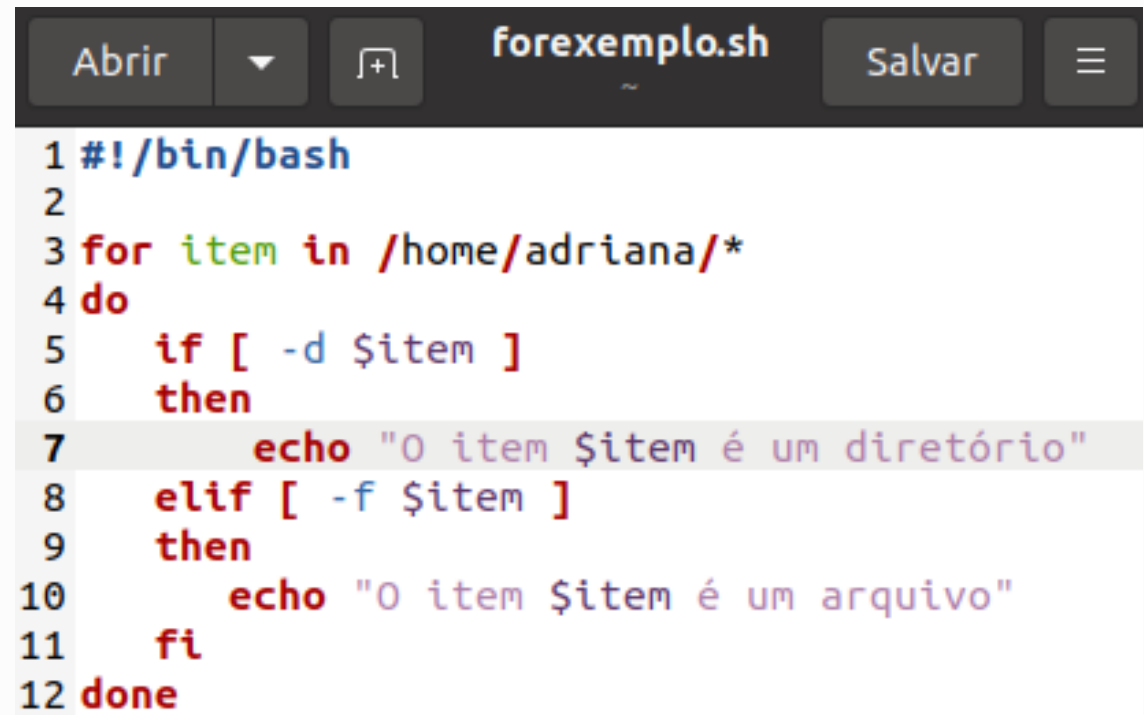
```
adriana@adriana-VirtualBox:~$ gedit testeFor.sh  
adriana@adriana-VirtualBox:~$ ./testeFor.sh  
O nome é: Adriana  
O nome é: Maria  
O nome é: Joaquim  
O nome é: José  
O nome é: Catarina  
O nome é: Maria  
O nome é: José  
adriana@adriana-VirtualBox:~$
```

Comando for

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

```
#Iterando por todos os itens de um diretório
```

```
for item in /home/adriana/*  
do  
    if [ -d "$item" ]  
    then  
        echo "O item $item é um diretório"  
    elif [ -f "$item" ]  
    then  
        echo "O item $item é um arquivo"  
    fi  
done
```



```
1 #!/bin/bash  
2  
3 for item in /home/adriana/*  
4 do  
5     if [ -d $item ]  
6     then  
7         echo "O item $item é um diretório"  
8     elif [ -f $item ]  
9     then  
10        echo "O item $item é um arquivo"  
11    fi  
12 done
```


Comando for

```
adriana@adriana-VirtualBox: ~  
adriana@adriana-VirtualBox:~$ gedit forexemplo.sh  
adriana@adriana-VirtualBox:~$ chmod 755 forexemplo.sh  
adriana@adriana-VirtualBox:~$ ./forexemplo.sh  
./forexemplo.sh: linha 5: [: número excessivo de argumentos  
./forexemplo.sh: linha 8: [: número excessivo de argumentos  
O item /home/adriana/arq1 é um arquivo  
O item /home/adriana/Documentos é um diretório  
O item /home/adriana/Downloads é um diretório  
O item /home/adriana/forexemplo.sh é um arquivo  
O item /home/adriana/Imagens é um diretório  
O item /home/adriana/Modelos é um diretório  
O item /home/adriana/Música é um diretório  
O item /home/adriana/Público é um diretório  
O item /home/adriana/snap é um diretório  
O item /home/adriana/testefor.sh é um arquivo  
O item /home/adriana/Vídeos é um diretório  
adriana@adriana-VirtualBox:~$
```

Comando while

O comando while permite definir um comando a testar e então iterar por um conjunto de comandos enquanto o comando definido de teste retornar status de saída zero.

Quando o comando de teste retornar status de saída diferente de zero, o while para de executar seu bloco de comandos e o loop é encerrado.

Sintaxe:

```
while comando_de_teste  
do  
    bloco de comandos  
done
```

O comando_de_teste usa o mesmo formato da estrutura if-then, e podemos usar o comando test para testar condições.

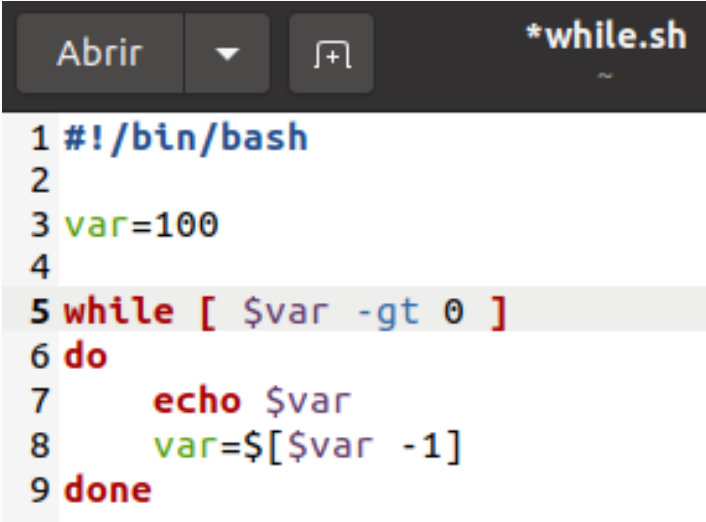
O status de saída do comando de teste deve mudar em algum momento durante as iterações, ou teremos um loop infinito.

Comando while

```
#!/bin/bash
#Testando o comando while
```

```
var=100
while [ $var -gt 0 ]
do
    echo $var
    var=$(( $var - 1 ))
done
```

```
adriana@adriana-VirtualBox:~$ gedit while.sh
adriana@adriana-VirtualBox:~$ chmod a+x while.sh
adriana@adriana-VirtualBox:~$ ./while.sh
100
99
98
97
```

A screenshot of a code editor window titled '*while.sh'. The editor shows a bash script with line numbers 1 through 9. The script starts with a shebang, sets a variable, and enters a while loop that echoes the variable and decrements it until it reaches 0. The code is color-coded: blue for comments, green for variable assignments, red for control flow, and purple for arithmetic operations.

```
Abrir ▼ [+]
```

```
1 #!/bin/bash
2
3 var=100
4
5 while [ $var -gt 0 ]
6 do
7     echo $var
8     var=$(( $var - 1 ))
9 done
```

A screenshot of a terminal window showing the output of the script. It displays the numbers 100, 99, 98, and 97 on separate lines, followed by the prompt 'adriana@adriana-VirtualBox:~\$' with a cursor.

```
100
99
98
97
adriana@adriana-VirtualBox:~$
```

Comando until

O comando until opera de forma oposta ao comando while.

É necessário especificar um comando de teste que retorne um status de saída diferente de zero para que o bloco de comandos listado no loop seja executado.

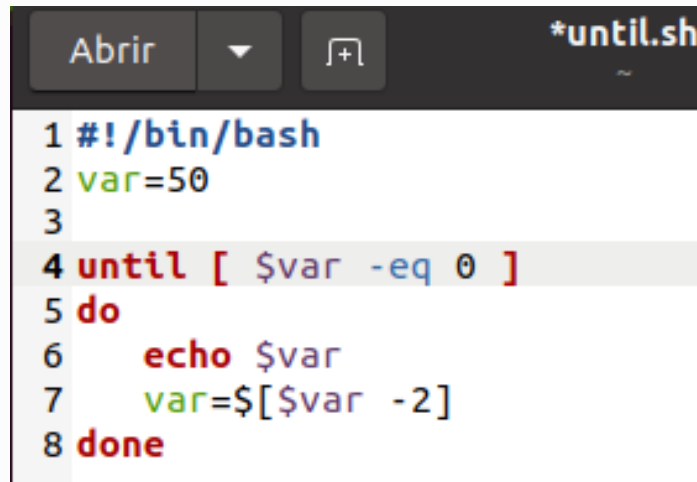
Quando o comando de teste retornar status de saída zero, o loop termina.

Sintaxe:

```
until comando_de_teste  
do  
    bloco de comandos a executar  
done
```

Comando until

```
#!/bin/bash
#Teste da estrutura de repetição until
var=50
until [ $var -eq 0 ]
do
    echo $var
    var=$(( $var - 2 ))
done
```



```
Abrir ▼ [+]
```

```
1 #!/bin/bash
2 var=50
3
4 until [ $var -eq 0 ]
5 do
6     echo $var
7     var=$(( $var - 2 ))
8 done
```

```
adriana@adriana-VirtualBox:~$ gedit until.sh
adriana@adriana-VirtualBox:~$ chmod 755 until.sh
adriana@adriana-VirtualBox:~$ ./until.sh
50
48
46
44
42
40
38
36
34
32
30
28
26
24
22
20
18
16
14
12
10
8
6
4
2
adriana@adriana-VirtualBox:~$
```

Comando for no estilo Linguagem C

É possível usar uma estrutura de repetição for no estilo da linguagem C em um script do shell.

Neste caso, teremos uma variável contadora que irá controlar o número de iterações do loop.

Sintaxe:

```
for (( atrib_variável; condição; processo_iteração ))
```

Exemplo:

```
#!/bin/bash  
#Exemplo de comando for no estilo linguagem C  
for (( i = 1; i <= 15; i++ ))  
do  
    echo "Número: $i"  
done
```

Comando for no estilo Linguagem C

```
Abrir ▼ [+]
```

***cfor.sh**

```
1 #!/bin/bash
2
3 for (( i = 1; i <= 15; i++ ))
4 do
5     echo "Número: $i"
6 done
```

```
adriana@adriana-VirtualBox:~$ gedit cfor.sh
adriana@adriana-VirtualBox:~$ chmod 755 cfor.sh
adriana@adriana-VirtualBox:~$ ./cfor.sh
Número: 1
Número: 2
Número: 3
Número: 4
Número: 5
Número: 6
Número: 7
Número: 8
Número: 9
Número: 10
Número: 11
Número: 12
Número: 13
Número: 14
Número: 15
adriana@adriana-VirtualBox:~$
```

Manipulação de entradas de usuários

Frequentemente, precisamos escrever scripts que interajam com o usuário.

Há algumas formas de se obter dados dos usuários no shell, como parâmetros de linhas de comando, opções e leitura de dados diretamente do teclado.

Parâmetros de linha de comando

É o método para passar dados ao script do shell. Os parâmetros de linha de comando permitem adicionar valores de dados à linha de comandos ao executar o script.

Para passar parâmetros a um script, digite-os após o nome do script, ao executá-lo:

`./scriptnome param1 param2 ... paramN`

Leitura dos parâmetros

O shell bash atribui variáveis especiais, denominadas parâmetros posicionais, a todos os parâmetros digitados na linha de comandos.

Os parâmetros posicionais são números, sendo \$0 o nome do programa, \$1 o primeiro parâmetro, \$2 o segundo, e assim por diante até o nono, que é \$9.

Para adicionar mais parâmetros, englobe o número do parâmetro entre parênteses: \$(10)

Exemplo:

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

```
#Teste de parâmetros na linha de comandos
```

```
echo "Programa que calcula o quadrado de um número"
```

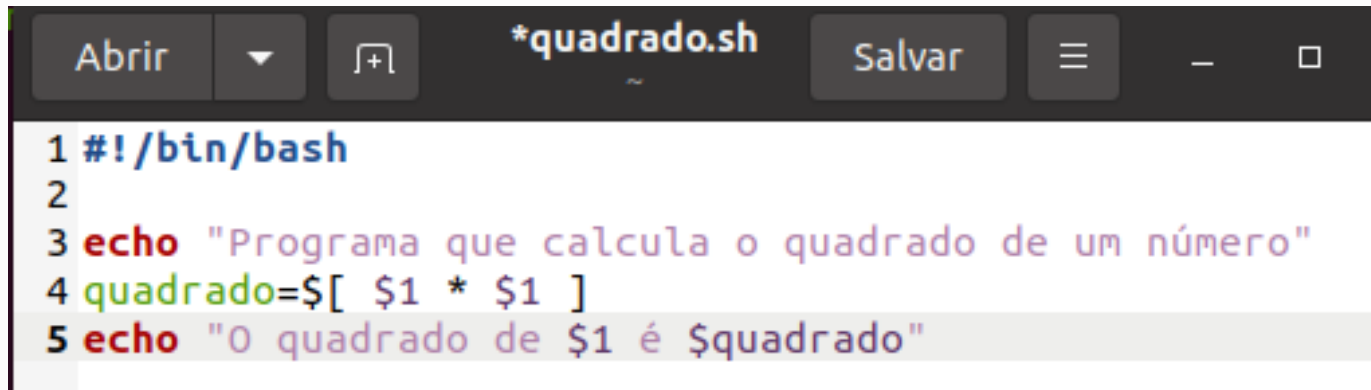
```
quadrado=$(( $1 * $1 ))
```

```
echo "O quadrado de $1 é $quadrado"
```

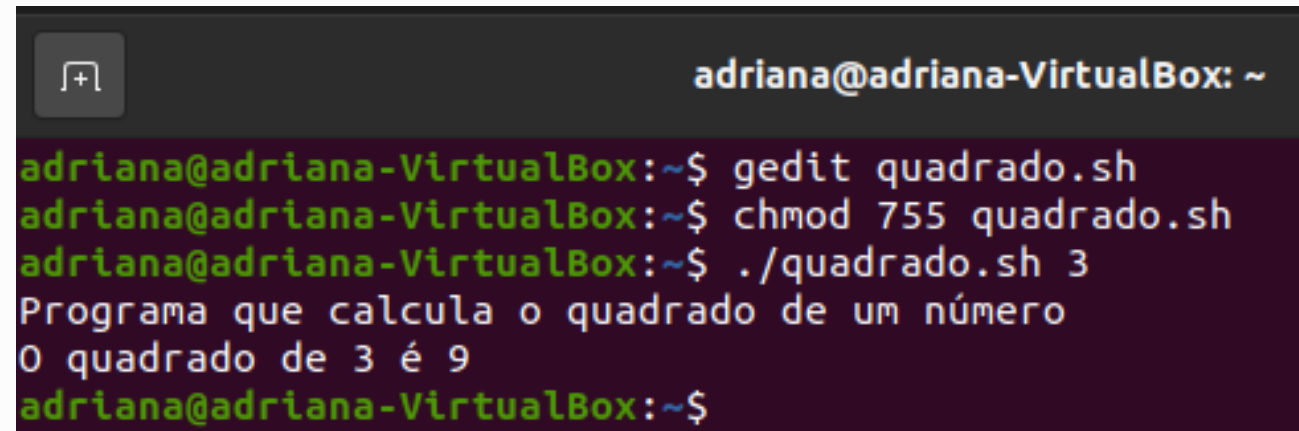
Para executá-lo e calcular o quadrado de 3:

```
./quadrado.sh 3
```

Leitura dos parâmetros



```
1 #!/bin/bash
2
3 echo "Programa que calcula o quadrado de um número"
4 quadrado=$(( $1 * $1 ))
5 echo "O quadrado de $1 é $quadrado"
```



```
adriana@adriana-VirtualBox: ~$ gedit quadrado.sh
adriana@adriana-VirtualBox: ~$ chmod 755 quadrado.sh
adriana@adriana-VirtualBox: ~$ ./quadrado.sh 3
Programa que calcula o quadrado de um número
O quadrado de 3 é 9
adriana@adriana-VirtualBox: ~$
```

Variáveis de parâmetros especiais

Há algumas variáveis especiais disponíveis no shell bash.

A variável especial `$#` contém o número de parâmetros de linhas de comando fornecidos ao rodar o script. Podemos usá-la para verificar se o usuário digitou o número de parâmetros necessários para rodar o programa corretamente.

Exemplo:

```
#!/bin/bash
#Verificação de número de parâmetros
if [ $# -ne 1 ]
then
    echo "Digite ao menos um valor!"
else
    resultado=$(( $1 * 3 ))
    echo "O triplo de $1 é $resultado"
fi
```

Variáveis de parâmetros especiais

```
Abrir ▼ [+]
```

***triplo.sh**

```
Salvar
```

```
1 #!/bin/bash
2
3 if [ $# -ne 1 ]
4 then
5     echo "Digite ao menos um valor!"
6 else
7     resultado=$(( $1 * 3 ))
8     echo "O triplo de $1 é $resultado"
9 fi
```

```
[+] adriana@adriana-VirtualBox: ~
```

```
adriana@adriana-VirtualBox:~$ gedit triplo.sh
adriana@adriana-VirtualBox:~$ chmod 755 triplo.sh
adriana@adriana-VirtualBox:~$ ./triplo.sh 4
O triplo de 4 é 12
adriana@adriana-VirtualBox:~$
```

Comando shift (deslocamento)

O comando shift auxilia na manipulação de parâmetros de linha de comando.

O comando shift desloca os parâmetros em suas posições relativas.

Ao ser usado, o comando shift diminui cada parâmetro em uma posição, de modo que o valor da variável \$3 é movido para \$2, o valor de \$2 é movido para \$1 e o valor de \$1 é descartado.

Exemplo:

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

```
#Teste de comando shift
```

```
i=1
```

```
while [ -n "$1" ]
```

```
do
```

```
    echo "O parâmetro $i tem o valor: $i"
```

```
    i=$(( i + 1 ))
```

```
    shift
```

```
done
```

Comando shift (deslocamento)

```
Abrir  *shift.sh  Salvar  ≡  
1 #!/bin/bash  
2  
3 i=1  
4 while [ -n "$1" ]  
5 do  
6   echo "O parâmetro $i tem o valor: $1"  
7   i=$(( i + 1 ))  
8   shift  
9 done
```

```
adriana@adriana-VirtualBox: ~  
adriana@adriana-VirtualBox:~$ gedit shift.sh  
adriana@adriana-VirtualBox:~$ chmod a+x shift.sh  
adriana@adriana-VirtualBox:~$ ./shift.sh banana  
O parâmetro 1 tem o valor: banana  
adriana@adriana-VirtualBox:~$ ./shift.sh banana maçã  
O parâmetro 1 tem o valor: banana  
O parâmetro 2 tem o valor: maçã  
adriana@adriana-VirtualBox:~$ ./shift.sh banana maçã uva melancia  
O parâmetro 1 tem o valor: banana  
O parâmetro 2 tem o valor: maçã  
O parâmetro 3 tem o valor: uva  
O parâmetro 4 tem o valor: melancia  
adriana@adriana-VirtualBox:~$
```

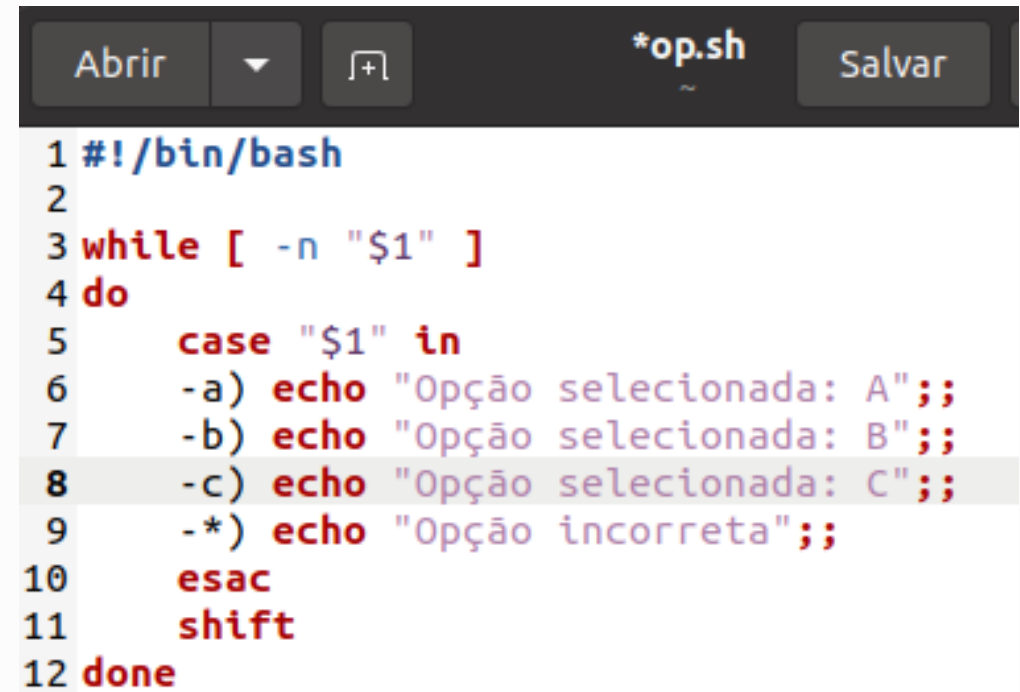
Trabalhos com opções

Opções são letras precedidas por um hífen que alteram o comportamento de um comando.

Para processar opções passadas ao script, vamos usar uma combinação dos comandos case e shift.

Exemplo:

```
#!/bin/bash
#Usando opções em um script
while [ -n "$1" ]
do
    case "$1" in
        -a) echo "Opção selecionada: A";;
        -b) echo "Opção selecionada: B";;
        -c) echo "Opção selecionada: C";;
        -*) echo "Opção incorreta";;
    esac
    shift
done
```

A screenshot of a code editor window with a dark theme. The window has a title bar with buttons for 'Abrir', a dropdown arrow, a file icon, and a file name '*op.sh'. There is also a 'Salvar' button. The code is a bash script with line numbers 1 through 12 on the left. The script uses a while loop and a case statement to process command-line options. The code is color-coded: blue for shell keywords, red for control flow, and purple for strings. The line numbers are in a light gray font.

```
1 #!/bin/bash
2
3 while [ -n "$1" ]
4 do
5     case "$1" in
6         -a) echo "Opção selecionada: A";;
7         -b) echo "Opção selecionada: B";;
8         -c) echo "Opção selecionada: C";;
9         -*) echo "Opção incorreta";;
10    esac
11    shift
12 done
```

Trabalhos com opções

```
adriana@adriana-VirtualBox: ~  
adriana@adriana-VirtualBox:~$ gedit op.sh  
adriana@adriana-VirtualBox:~$ chmod a+x op.sh  
adriana@adriana-VirtualBox:~$ ./op.sh  
adriana@adriana-VirtualBox:~$ ./op.sh -a  
Opção selecionada: A  
adriana@adriana-VirtualBox:~$ ./op.sh -b  
Opção selecionada: B  
adriana@adriana-VirtualBox:~$ ./op.sh -c  
Opção selecionada: C  
adriana@adriana-VirtualBox:~$ ./op.sh -f  
Opção incorreta  
adriana@adriana-VirtualBox:~$
```

```
adriana@adriana-VirtualBox: ~  
adriana@adriana-VirtualBox:~$ ./op.sh -a -c  
Opção selecionada: A  
Opção selecionada: C  
adriana@adriana-VirtualBox:~$ ./op.sh -a -b  
Opção selecionada: A  
Opção selecionada: B  
adriana@adriana-VirtualBox:~$ ./op.sh -c -t  
Opção selecionada: C  
Opção incorreta  
adriana@adriana-VirtualBox:~$ ./op.sh -c -t -a  
Opção selecionada: C  
Opção incorreta  
Opção selecionada: A  
adriana@adriana-VirtualBox:~$
```


Referências

PRITCHARD, S.; PESSANHA, B. G.; LANGFELDT, N.; STANGER, J.; DEAN, J. 2007. **Certificação Linux LPI Rápido e Prático. Guia de Referência nível 1: Exames 101 e 102.** 2ª Ed. Rio de Janeiro: Editora Alta Books.

Curso de Shell Scripting – Bóson Treinamentos

<http://www.bosontreinamentos.com.br/curso-de-shell-scripting/>