Tópicos Especiais em Fundamentos da Computação
Shell Scripting

Estruturas de Repetição

Andrei Rimsa Álvares
andrei@cefetmg.br

Sumário

- Introdução
- Comando `for`
- Comando `while`
- Comando `until`
- Sequenciadores `break` e `continue`
- Comando `select`

INTRODUÇÃO

Controle de Fluxo

- A *shell* usa as mesmas construções para controle de fluxo que linguagens de programação de alto nível
  - Controle de fluxo condicional
    - `if` e `case`
  - Controle de fluxo de repetição
    - `for`, `while`, `until` e `select`
  - Sequenciadores
    - `break` e `continue`

Exemplo

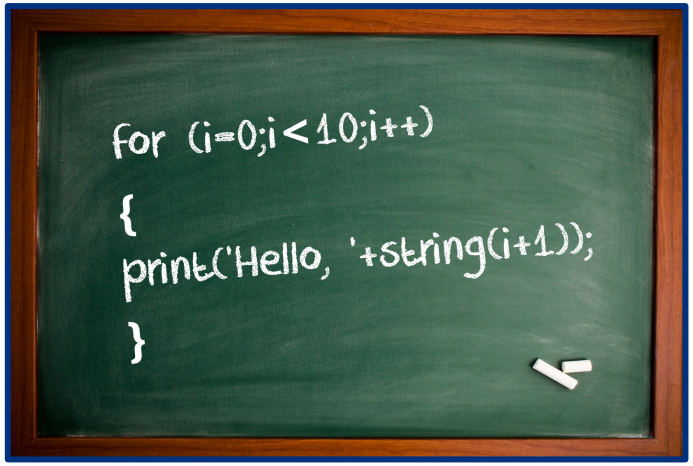
- Exemplo de fluxo de repetição, em que são exibidos número sequencialmente de 1 a 5

count_5

```
#!/bin/bash

count=1
while [ $count -le 5 ]; do
  echo $count;
  count=$((count + 1))
done
```

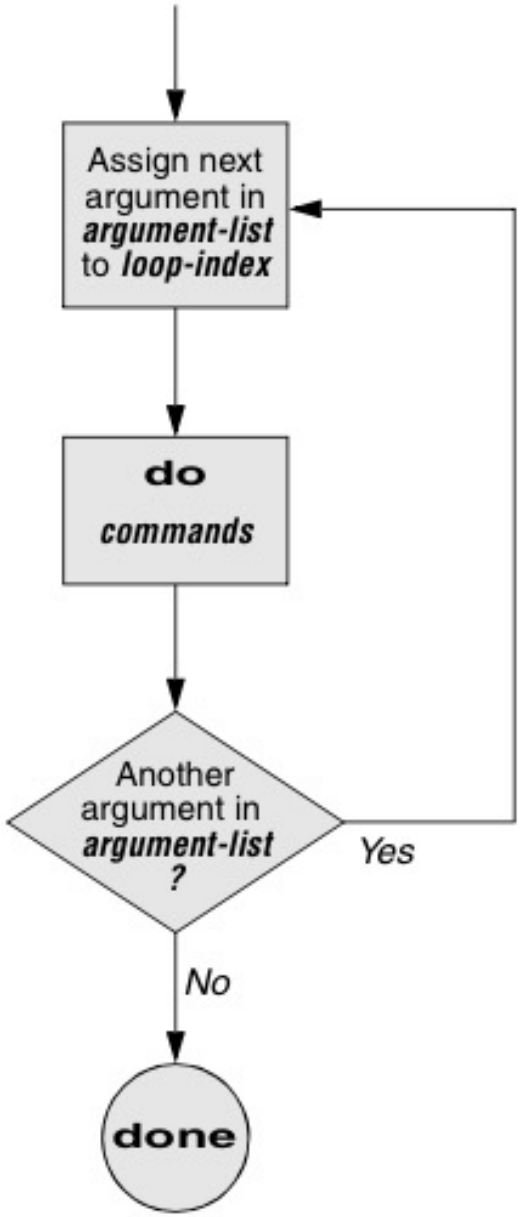
```
$ ./count_5
1
2
3
4
5
```

COMANDO `for`



`for...in`

- A estrutura de controle de repetição `for...in` possui a seguinte sintaxe




```
for loop-index in argument-list; do
    commands
done
```

Exemplo

- Exemplo em que cada fruta (*apples*, *oranges*, *pears* e *bananas*) é associada a variável `fruit`, que posteriormente é impressa na tela, até esgotar a lista por completo

fruit

```
#!/bin/bash

for fruit in apples oranges pears bananas; do
    echo "$fruit"
done
echo "Task complete"
```

```
$ ./fruit
apples
oranges
pears
bananas
Task complete
```

Expansões na Lista de Argumentos

- A lista de argumentos do comando `for` aceita expansões de arquivo

- Exemplo: listar somente os diretórios da pasta atual de trabalho

dirfiles

```
#!/bin/bash

for  i in *; do
        if [ -d "$i" ]; then
                echo "$i"
        fi
done
```

```
$ ./dirfiles
Applications
Desktop
Documents
Downloads
Library
Movies
Music
Pictures
Public
```

Outro Exemplo

- Usando comandos como expansão na lista de parâmetros para listar os nomes dos identificadores dos grupos do usuário

groups

```
#!/bin/bash

echo "Groups: "

for gid in $(id -G); do
    group=$(cat /etc/group |
            grep ":$gid:" |
            cut -f 1 -d":")
    echo "  $group";
done
```

```
$ ./groups
Groups:
   rimsa
   adm
   cdrom
   sudo
   dip
   plugdev
   lpadmin
   sambashare
```

for

- Uma versão alternativa do comando `for` permite navegar na lista de argumentos da linha de comando

```
for loop-index; do
    commands
done
```

- O mesmo comportamento pode ser simulado pelo comando a seguir, tal que `"$@"` expande em argumentos em linha de comando entre aspas (`"$1"`, `"$2"`, ...)

```
for loop-index in "$@"; do
    commands
done
```

Exemplo

- Listando os parâmetros em linha de comando

candies

```
#!/bin/bash

for arg; do
    echo "$arg"
done
```

```
$ ./candies candy gum chocolate
candy
gum
chocolate
```

for ((expr ; expr ; expr))

- Uma outra versão do `for` com sintaxe similar àquelas de linguagens de programação como C pode ser usada

```
for (( expr1 ; expr2 ; expr3 )); do
    commands
done
```

  - A expressão `expr1` é avaliada antes da primeira iteração;
  - Os comandos (`commands`) são executados repetidamente enquanto o valor de `expr2` avaliar em verdadeiro
  - Após cada iteração, a expressão `expr3` é avaliada

Múltiplas expressões podem ser avaliadas se separadas por vírgula

Dica: um loop infinito pode ser simulado usando a sintaxe: `(( ; ; ))`

Exemplos

a) Listar três números aleatórios

rand3
```
#!/bin/bash

for (( i=1; i <= 3; i++ )); do
    echo "Random $i: $RANDOM"
done
```

```
$ ./rand3
Random 1: 26851
Random 2: 26233
Random 3: 26699
```

b) Mostrar três números de 5 em 5 a partir de 10

by3
```
#!/bin/bash

for ((i=1, j=10; i <= 3 ; i++, j=j+5)); do
    echo "Number $i: $j"
done
```

```
$ ./by3
Number 1: 10
Number 2: 15
Number 3: 20
```

c) Mostrar números de 1 até o infinito com espera linear

inf
```
#!/bin/bash

for (( i=1 ; ; )); do
   sleep $i
   echo "Number: $((i++))"
done
```

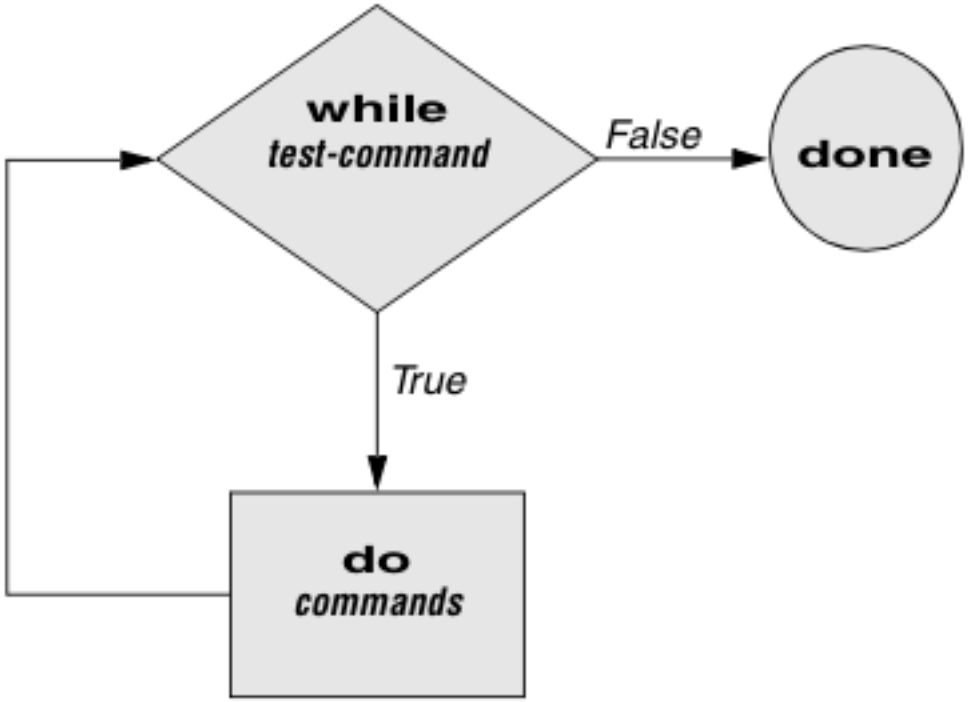
```
$ ./inf
Number: 1
Number: 2
^C
```

COMANDO `while`

while

- A estrutura de controle de repetição `while` possui a seguinte sintaxe

```
while test-command; do
    commands
done
```



Exemplo

- Imprimir dígitos de `0` a `9` sem pular linha usando o comando `while`

count

```
#!/bin/bash

number=0
while [ "$number" -lt 10 ]; do
    echo -n "$number"
    ((number +=1))
done
echo
```

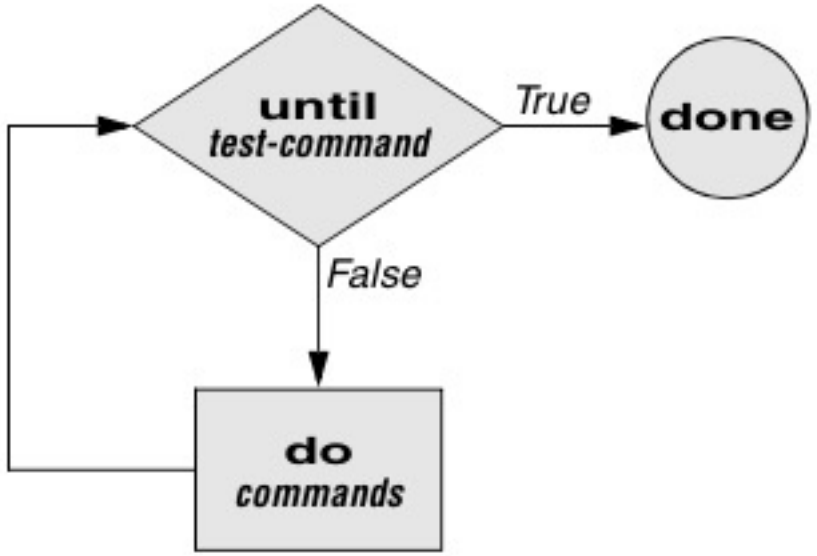
```
$ ./count
0123456789
```

COMANDO `until`

until

- O comando `until` tem estrutura muito similar ao comando `while`, a diferença está apenas na forma que é feita o teste condicional

```
until test-command; do
    commands
done
```



Exemplo

- Usar o comando `until` para adivinhar um nome secreto

guess

```
#!/bin/bash

secretname=zach
name=

echo "Try to guess the secret name!"
echo

until [ "$name" = "$secretname" ]; do
  echo -n "Your guess: "
  read name
done

echo "Very good"
```

```
$ ./guess
Try to guess the secret name!

Your guess: helen
Your guess: barbara
Your guess: rachel
Your guess: zach
Very good
```

Um Exemplo Real

- Travar a tela até que a senha correta, escolhida previamente pelo usuário, seja especificada

locktty

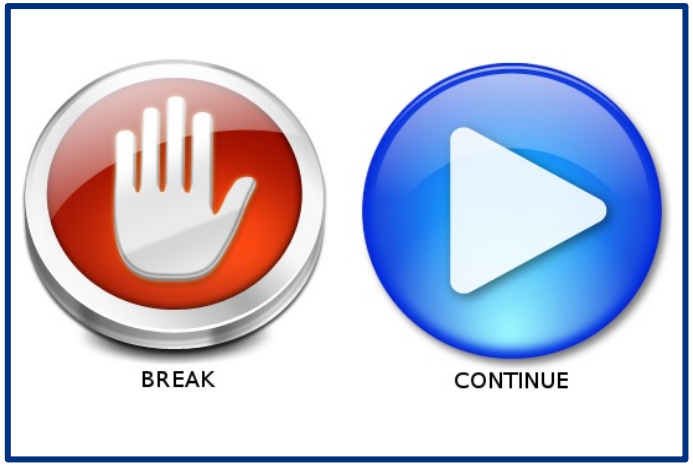
```
#!/bin/bash

trap '' 1 2 3 18
stty -echo

echo -n "Key: "
read key_1
echo
echo -n "Again: "
read key_2
echo

key_3=
if [ "$key_1" = "$key_2" ]; then
    tput clear

    until [ "$key_3" = "$key_2" ]; do
        read key_3
    done
else
    echo "locktty: keys do not match" 1>&2
fi

stty echo
```

Dicas:

1) use o comando `trap` para inibir sinais (control+c, ...)
2) consulte o manual do comando `signal` para obter o código de cada sinal

Cuidado: se esquecer a senha desse programa será preciso matar o processo (`kill`) usando um outro terminal

SEQUENCIADORES `break` e `continue`



`break` e `continue`

- Os comandos `break` e `continue` podem interromper o fluxo das estruturas de repetição `for`, `while` e `until`

`break`: transfere o controle para o comando imediatamente após o comando `done` do laço

`continue`: transfere o controle para o comando imediatamente após o comando `do` do laço

Exemplo

- Um laço de `1` a `10` que imprime apenas os números entre `4` e `8`

flow

```
#!/bin/bash

for index in {1..10}; do
    if [ $index -le 3 ]; then
        echo "continue"
        continue
    fi

    echo $index

    if [ $index -ge 8 ] ; then
        echo "break"
        break
    fi
done
```

```
$ ./flow
continue
continue
continue
4
5
6
7
8
break
```

COMANDO `select`

select

- O comando `select` (baseado no encontrado em *Korn Shell*) exibe um menu, atribui o conteúdo do item escolhido à variável definida pelo usuário e executa uma série de comandos; a sintaxe é

```
select varname [in arg...]; do
    commands
done
```

Assim como no `for`, se in não for especificado, será utilizada a expansão `"$@"`

select

- Por exemplo, um comando `select` que começa com

```
select fruit in apple banana blueberry kiwi
               orange watermelon STOP
```

exibe um menu como

```
1) apple 3) blueberry   5) orange          7) STOP
2) banana 4) kiwi       6) watermelon
```

- O comando `select` usa as variáveis `LINES` (padrão: `24`) e `COLUMNS` (padrão: `80`) que especificam o tamanho do display; se `COLUMNS` for definido com `40` o menu é exibido como

```
1) apple        5) orange
2) banana       6) watermelon
3) blueberry    7) STOP
4) kiwi
```

`select`

- Após exibir o menu o comando `select` exibe o valor da variável `PS3` (*select prompt*), em que o valor padrão é `#?`, mas que normalmente é modificado para um valor mais apropriado

- Ao entrar com um número válido (na faixa do menu), o comando `select` atribui a variável `varname` o argumento correspondente ao número do item escolhido; caso contrário, o comando `select` define `varname` como `null` (vazio)
  - De qualquer forma, o comando `select` coloca o número do item escolhido na variável `REPLY`

Apertar *enter* sem definir uma opção causa a reexibição do menu com o *select prompt*

`select`

- Depois, executa-se os comandos especificados no corpo do comando `select`

- A estrutura de fluxo `select` continua exibindo um *select prompt* (`PS3`) e executa os comandos definidos no seu corpo até que o fluxo seja interrompido explicitamente, tipicamente por um comando como `break` (para sair do laço) ou `exit` (para sair do script)

Exemplo

- Um laço para escolha de frutas

favorite

```
#!/bin/bash

PS3="Choose your favorite fruit: "

select FRUIT in apple banana blueberry kiwi \
                orange watermelon STOP; do
    if [ "$FRUIT" == "" ]; then
        echo "Invalid entry"
        echo
        continue
    elif [ $FRUIT = STOP ]; then
        echo "Thanks for playing!"
        break
    fi

    echo "You chose $FRUIT as your favorite"
    echo "That is choice number $REPLY"
    echo
done
```

```
$ export COLUMNS=40
$ ./favorite
1) apple        5) orange
2) banana       6) watermelon
3) blueberry    7) STOP
4) kiwi
Choose your favorite fruit: 3
You chose blueberry as your favorite
That is choice number 3

Choose your favorite fruit: 99
Invalid entry

Choose your favorite fruit: 7
Thanks for playing!
$
```

ISSO É TUDO, PESSOAL!