# Condicionais

João Marcelo Uchôa de Alencar

Universidade Federal do Ceará - Quixadá

8 de outubro de 2025

# Estruturas de Controle

Enquanto outras linguagens testam **condições**, o *Shell* testa o código de retorno do comando que o segue:

- O código de retorno está associado à variável $?
- Sucesso (verdadeiro) equivale à 0.
- Erro (falso) equivale à 1.

No caso de um |, é retornado uma vetor PIPESTATUS, cada posição equivale ao código de retorno do comando na posição equivalente do *pipe*.

- ${PIPESTATUS[0]} é o código de retorno do primeiro comando.
- ${PIPESTATUS[*]} exibe todo o vetor.

# A estrutura if/then/else

```
if <comando>
then
    <comando1>
    <comando2>
    <...>
else
    <comando3>
    <comando4>
    <...>
fi
```

# Exemplo

Como seria um *script* chamado **numeroOuNao.sh** que recebe um parâmetro e afirma se o mesmo é um número ou não?

# O comando *test*

- Como fazer para analisar condições booleanas no estilo de comparações $a > b$, $a == b$, etc?

```
$ test <expressao>
```

- Sendo **expressao** a condição a ser testada.
- Ao usar o **test** é bom colocar as variáveis entre aspas. Há diferença entre comparar a *string* vazia e uma variável não existente.

# Exemplo

- Como seria um *script* chamado *param.sh* que verifica se o usuário passou parâmetros ou não?
- Caso não tenha passado parâmetros, deve imprimir apenas o nome do *script*.
- Caso tenha passado parâmetros, deve imprimir todos.

# Testando Arquivos

| | |
|---|---|
| -r arquivo | Permissão de Leitura |
| -w arquivo | Permissão de Escrita |
| -x arquivo | Permissão de Gravação |
| -f arquivo | Arquivo Regular |
| -d arquivo | Diretório |
| -s arquivo | Tamanho maior que zero |

# Testando Cadeia de Caracteres

| | |
|---|---|
| -z cad1 | Se o tamanho é zero |
| -n cad1 | Se o tamanho é diferente de zero |
| cad1 = cad2 | Se são idênticas |
| cad1 | Se é não nula |

# Testando Números

| | |
|---|---|
| int1 -eq int2 | Igualdade |
| int1 -ne int2 | Desigualdade |
| int1 -gt int2 | Maior |
| int1 -ge int2 | Maior ou igual |
| int1 -lt int2 | Menor que |
| int1 -le int2 | Menor ou igual |

# Simplificando a escrita do *test*

- No lugar de:

  ```
  $ test <expressao>
  ```

- Podemos escrever:

  ```
  $ [ <expressao> ]
  ```

- Importante notar que há um espaço entre os colchetes e a expressão.
- Se houver um sinal de ! antes da expressão, ela está negada. Mas lembre, tudo separado por **espaços**!!!

# Simplificando a escrita do *test*

- O **e** lógico pode ser criado usando -**a** para separar as expressões.
- O **ou** lógico pode ser criado usando -**o** para separar as expressões.
- Você pode usar parênteses, mas eles devem ser precedidos da barra invertida para o shell não interpretá-los:

```
\( $a = $b -o $c = $d \) -a \( $a -gt $d \)
```

# Usando o && e o ||

- Executar o segundo comando apenas se o primeiro for sucesso:

  ```
  comando1 && comando2
  ```

- Executar o primeiro comando, e caso este falhe, executar o segundo:

  ```
  comando1 || comando2
  ```

# Exemplo

Vamos definir dois comandos:

- Caso um diretório chamado **novoDir** não exista, deve ser criado.
- Mover todos os arquivos *.txt* para um diretório chamados **textos** e em seguida compacte o diretório.

# Testando Padrões

Você pode testar se uma cadeia casa com um padrão usando:

```
[[ expressão ]]
```

Sendo que expressão obecede:

| | |
|---|---|
| cadeia == padrão | Casamento |
| cadeia != padrão | Incompatibilidade |
| cadeia1 < cadeia2 | Ordem alfabética |
| cadeia1 > cadeia2 | Ordem alfabética |
| expr1 && expr2 | Lógica |
| expr1 \|\| expr2 | Lógica |

# Expressões Regulares

Além de padrões simples, podemos casar expressões regulares:

```
[[ $cadeia =~ expressaoregular ]]
```

Se a expressão regular usar parêntenses, o vetor BASH_REMATCH armazena as subcadeias que casaram com cada subexpressão.

- ${BASH_REMATCH[0]} é cadeira inteira
- ${BASH_REMATCH[1]} é a subcadeia que casou com a subexpressão no primeiro parênteses.

No caso das expressões, os parênteses não precisam ser escritos precedidos por barras.

```
$ num=100
$ [[ $num =~ ([0-9])[0-9][0-9] ]]
```

# Exemplo

Escrever um comando que preencha o vetor com os octetos de um endereço IP.
Por exemplo, para o IP 200.19.190.1:
${BASH_REMATCH[1]} = 200
${BASH_REMATCH[2]} = 19
${BASH_REMATCH[3]} = 190
${BASH_REMATCH[4]} = 1

# O comando *case*

```
case <valor> in
   padr1)
      <comando1>
      <comando2>
      <...>
      ;;
   padr2)
      <comando1>
      <comando2>
      <...>
      ;;
   ...
   padrN)
      <comando1>
      <comando2>
      <...>
      ;;
   *)
      <comando1>
      <comando2>
      <...>
esac
```

Os padrões só usam um conjunto básico de meta-caracteres, não são expressões regulares completas.