Condições em Shell Script

Assim como qualquer outra *linguagem de programação*, o Shell também tem estruturas para se fazer condições.

O COMANDO TEST

O comando test é útil para fazer vários tipos de verificações em textos e arquivos, testa o conteúdo de uma string, pode ser um arquivo, uma variável, compara valores numéricos ou não. O test avalia uma determinada condição dada e se ela for verdadeira a variável $? é retornada com o valor 0 e se falsa o valor é 1.

```
test 1 = 1;
echo $?
# A saída desta condição é 0 por ser ela verdadeira.
```

Também pode ser escrita da seguinte forma

```
[ 1 = 1 ]; echo $?
```

Testes em arquivos	
`-b`	É um dispositivo de bloco
`-c`	É um dispositivo de caractere
`-d`	É um diretório
`-e`	O arquivo existe
`-f`	É um arquivo normal
`-g`	O bit SGID está ativado
`-G`	O grupo do arquivo é o do usuário atual
`-k`	O sticky-bit está ativado
`-L`	O arquivo é um link simbólico
`-O`	O dono do arquivo é o usuário atual
`-p`	O arquivo é um named pipe
`-r`	O arquivo tem permissão de leitura
`-s`	O tamanho do arquivo é maior que zero
`-S`	O arquivo é um socket
`-t`	O descritor de arquivos N é um terminal
`-u`	O bit SUID está ativado
`-w`	O arquivo tem permissão de escrita
`-x`	O arquivo tem permissão de execução
`-nt`	O arquivo é mais recente (NewerThan)
`-ot`	O arquivo é mais antigo (OlderThan)
`-ef`	O arquivo é o mesmo (EqualFile)

Comparação Numérica	
`-lt`	É menor que (LessThan)
`-gt`	É maior que (GreaterThan)
`-le`	É menor igual (LessEqual)
`-ge`	É maior igual (GreaterEqual)
`-eq`	É igual (EQual)
`-ne`	É diferente (NotEqual)
Comparação de Strings	
`=`	É igual
`!=`	É diferente
`-n`	É não nula
`-z`	É nula
Operadores Lógicos	
`!`	NÃO lógico (NOT)
`-a`	E lógico (AND)
`-o`	OU lógico (OR)

Estrutura de Condição (IF, THEN, ELSE, ELIF, FI)

Diferente de outras linguagens, o if testa um comando e não uma condição. Porém como já conhecemos qual o comando do shell que testa condições (test), é só usá-lo em conjunto com o if. Por exemplo, para saber se uma variável é maior ou menor do que 10 e mostrar uma mensagem na tela informando (ao final fecharemos a condição com fi, não esquecer, senão dá erro).

```
VARIAVEL=8;
if test "$VARIAVEL" -gt 10
    then
        echo "é maior que 10"
else
        echo "é menor que 10"
fi
```

Há um atalho para o test , que é o comando [. Ambos são exatamente o mesmo comando, porém usar o [deixa o if mais parecido com o formato tradicional de outras linguagens

```
VARIAVEL=8;
if [ "$VARIAVEL" -gt  ] 10
    then
        echo "é maior que 10"
else
        echo "é menor que 10"
fi
```

Se usar o [, também é preciso fechá-lo com o], e sempre devem ter espaços ao redor. É recomendado evitar esta sintaxe para diminuir suas chances de erro.

O elif constitui a terceira opção para uma condição, seria uma continuação do if, veja o exemplo de como usá-lo, nesse exemplo vou utilizá-lo com o test e com seu atalho o [ ] , pois o mesmo será adicionado ao nosso *meuscript.sh*, e no final do curso iremos ver como ele ficará

```
VARIAVEL=8;
if [ "$VARIAVEL" -gt 10 ]
then
        echo "é maior que 10"
elif test "$VARIAVEL" -lt 10
then
        echo "é menor que 10"
else
        echo "é 10"
fi
```

OBSERVAÇÃO: É recomendável usar dois colchetes [[ ]] do que 1 só [ ] , está referido ao POSIX e as novas implementações do Bash.

O case é para controle de fluxo, tal como é o if. Mas enquanto o if testa expressões não exatas, o case vai agir de acordo com os resultados exatos. Vamos ver um exemplo com case baseado no exemplo do if,elif e else anterior *(abre com case e fecha com esac, não esquecer de fechar senão dá erro)*

```
case $VARIAVEL in
    10) echo "é 10" ;;
    9) echo "é 9" ;;
    7|8) echo "é 7 ou 8" ;;
    *) echo "é menor que 6 ou maior que 10" ;;
esac
```