Parâmetros de verificação

O if faz a verificação de acordo ao parâmetro usado para avaliar se a condição é verdadeira. Caso seja é executado depois de then os comandos caso não seja é executado os comandos depois de else.

-a Para saber se o arquivo existe

```
if [ -a file.txt ]
then
        echo Found archive Encontrado Aquivo!
fi
```

O if fará a verificação através do parâmetro -a se o file.txt existe. Exemplos posteriores segue a mesma lógica.

-b Para saber se o dispositivo é de bloco

Dispositivo de bloco: É um ponto de montagem de um dispositivo de armazenamento gerado pelo kernel. Por exemplo*:* /dev/sda é um arquivo que representa o ponto de montagem do HD gerado pelo kernel.

Obs: Terminais são normalmente acessados como dispositivo de caracteres. Quando usamos o comando tty vemos um ponto de montagem de dispositivo de caracteres.

Dispositivo de caractere: cada caractere não possui endereço e não pode ser acessado. A cada caractere é gerado uma interrupção podem ser: teclado, mouse, monitor, placas de áudio. Oque é escrito no terminal são caracteres portanto não pode montar caracteres já que não é um dispositivo de armazenamento.

```
If  [ -b /dev/sda ]
then
   echo Found device block Encontrado dispositivo de bloco
fi
```

-c Para saber se o arquivo é um caractere especial

Arquivo de caractere especial: São arquivos gerados pelo kernel feitos para armazenar caracteres. Arquivos como: */dev/stdin , /dev/null , /dev/stdeer , /dev/stdout.*

O arquivo de caractere especial são feitos para os dispositivos de caractere que por exemplo pode ser o teclado. O terminal mesmo usa um arquivo especial usando o comando tty mostra a localização do arquivo de caractere especial que está usando.

```
if [ -c /dev/null ]
then
   echo Arquivo especial
fi
```

-d Para saber se o diretório existe

```
if [ -d /usr ]
then
  echo Dir existe
fi
```

-e Para saber se o arquivo existe

```
if [ -e file.txt ]
then
  echo Arquivo existe
fi
```

-f Saber se o arquivo é um arquivo normal.

```
if [ -f file.txt ]
then
    echo Arquivo normal
fi
```

A definição de normal seria os arquivos que a maioria das pessoas usam.

Para não precisar usar if para fazer a verificação a todo momento se o arquivo é normal pode ser utilizado o comando stat.

-g Saber se o arquivo tem permissão de SGID

SGID é conceder permissão para um arquivo ser executado atravéz do usuário.

Usa-se por exemplo chmod g+s filename.txt

O parâmetro g tem o significado de grupo do arquivo
O parâmetro +s define o SGID
g+s = conceder permissão ao grupo do usuário

```
if [ -g filename.txt ]
then
   echo Tem permissão do usuário
fi
```

Para saber se o arquivo tem permissão de ser executado através do usuário deve ser usado ls -l para ver a permissão do arquivo. Olha a comparação antes de ter suporte SGID e depois.
ls -l file.txt
-rw-rw-r-- 1 user user 0 jan 2 15:44 file.txt

Agora com chmod g+s file.txt

ls -l file.txt

-rw-rwSr-- 1 user user 0 jan 2 15:44 file.txt

Como pode ser visto depois do -rw-rw foi acrescentado um S

-h Para saber se é um link simbólico

```
if [ -h /dev/stderr  ]
then
  echo É link simbólico
fi
```

-k Para saber se o diretório tem o sticky bit ativado

Sticky bit: Essa permissão geralmente é aplicada a diretórios. Quando isso ocorre, os arquivos criados dentro do mesmo apenas podem ser renomeados ou apagados pelo dono do arquivo, do diretório ou pelo superusuário.

Embora exista uma concordância sobre a funcionalidade dessa permissão quando aplicada a diretórios, quando ela é aplicada a arquivos sua função varia de acordo com o sistema operacional utilizado.

Pode ser útil caso uma distro Linux tenha muitos usuários usando-a ao mesmo tempo. Com isso um usuário não pode apagar um diretório de outro usuário por exemplo.

O sticky bit pode ser definido usando o comando chmod, usando seu modo octal 1000 ou por meio de seu símbolo **t**

```
if [ -k bits ]
then
 echo Diretório tem o bit sticky.
fi
```

Usando chmod -t poderá remover o Sticky Bit.

-r Para saber se o arquivo existe e se é um arquivo que tenha permissão de ler

```
if [ -r file.txt ]
then
  echo Tem permissão de ler arquivo              Resp=Tem permissão de ler arquivo
fi
```

chmod 600 file.txt
600=Só consegue ler o a arquivo como root

```
if [ -r file.txt ]
then
   echo Tem permissão de ler arquivo                Resp=
fi
```

cat file.txt
Permission danied   Permissão Negada

-s Para saber se o tamanho do arquivo é maior que zero

  du -h /dev/null = 0

```
if [ -s /dev/null ]
then
   echo O tamanho é maior que zero         R=
fi
```

  du -h file.txt 4,0K

```
if [ -s file.txt ]
then
 echo O tamanho é maior que zero              R=O tamanho é maior que zero
fi
```

-t Saber se é um descritor de arquivos

Descritor de arquivo pode ser o stdin=0 , stdout=1 , stdeer=2. Stdin representa a entrada que seria o teclado, stdout é oque é imprimido na tela, stdeer é o erro imprimido na tela. Descritores de arquivos são muito importante pode por exemplo silenciar uma mensagem de comando, usar um arquivo pré-escrito como entrada para um comando.

Usa-se > para dizer saída. Mandar o stdout para /dev/null um “arquivo” que não acumula cache.

```
if [ -t >/dev/null ]
then
  echo É um descritor de arquivo          R=É um descritor de arquivo
fi
```

```
if [ -t < xxx ]
then
  echo É um descritor de arquivo          R=É um descritor de arquivos
fi
```

< Tem o significado como um descritor de arquivo já que simboliza a entrada que é o stdin.

-u Para saber se o arquivo está definido com bit SUID

Se este bit estiver ligado em um arquivo executável, isso indica que que ele vai rodar com as permissões do seu dono (o proprietário do arquivo) e não com as permissões do usuário que o executou. O SUID é definido por uma permissão por quem criou o executável e em todos computadores terá esta permissão pré-definida, é um pouco perigoso já pode executar algo mesmo sem precisar ter acesso root.

Na maioria das versões do UNIX, você pode criar scripts de shell que são SUID ou SGID. Ou seja, você pode criar um script de shell e, ao definir o proprietário do script de shell como root e definir seu bit SUID, pode forçar a execução do script de shell com privilégios de superusuário.

OBS: Só tem efeito em arquivos executáveis.

Por causa de uma falha fundamental com a implementação UNIX de scripts de shell e SUID, você não pode executar scripts de shell SUID de uma maneira completamente segura.

SUID = Acesso root
SGID = Acesso elevado para o usuário

ls -l file.txt   -rw-rw-r--

```
if [ -u file.txt ]
then
  echo Tem bit SUID            R=
fi
```

chmod 4555 file.txt ou chmod u+s

Chmod 4555 (chmod a + rwx, u-w, g-w, o-w, ug + s, + t, g-s, -t) define permissões para que o proprietário possa ler, escrever e executar. Outros usuários podem ler, não podem escrever e podem executar.

ls -l file.txt -r-sr-xr-x

```
if [ -u file.txt ]
then
  echo Tem bit SUID            R=Tem bit SUID
fi
```

SUID só funciona com executáveis
SGID funciona com executáveis, arquivos , diretórios etc

-w Para saber se o arquivo é editável pela permissão do arquivo

ls -l file.txt -rw-rw-r--

```
if [ -w file.txt ]
then
   echo O arquivo é editável                    R=O arquivo é editável
fi
```

chmod 4 file.txt
4=permissão de leitura
ls -l file.txt -------r--

```
if [ -w file.txt ]
then
   echo É um arquivo editável                    R=
fi
```

-x Para saber se o arquivo é executável

```
if [ -x file.txt ]
then
   echo É um arquivo executável          R=
fi

chmod +x  file.txt

if [ -x file.txt ]
then
  echo É um arquivo executável          R=É um arquivo executável
fi
```

-O Para saber se o arquivo ou diretório pertence ao USER ID, ou seja saber se o usuário tem permissão no momento de modificar

Através da variável $UID dar para saber se está no usuário normal ou no modo root. Caso esteja no usuário normal apresenta o número 1000 no modo root o número 0.

```
echo $UID  R=1000

if [ -O file.txt ]
then
  echo O arquivo pode ser modificado pelo USER ID no momento    R=O arquivo pode ...
fi
```

Agora está no modo root o terminal portanto vai apresentar como verdadeiro a condição

```
echo $UID  R=0

if [ -O /usr ]
then
  echo O User ID tem permissão de modificar o diretório          R=O User ID tem ...
fi
```

-G Para saber se pertence ao grupo do usuário

```
 cat /etc/group | grep ^$USER     # Mostrá o grupo do usuário
```

ls -l file.txt R= -rw-rw-r-- 1 ale ale 0 jan 4 12:20 file.txt
Como pode ser visto o usuário é ale=1 e o ale=2 é o grupo

```
if [ -G file.txt ]
then
    echo Arquivo pertence ao grupo do usuário  R=Arquivo pertence ….
fi
```

Para alterar o grupo do arquivo usa-se o comando chgrp

```
chgrp root file.txt
```

sudo chgrp root file.txt # Tem que usar sudo para ter permissão para alterar o arquivo já que vai pertencer ao grupo root.

ls -l file.txt -rw-rw-r-- 1 ale root 0 jan 4 12:29 file.txt

```
if [ -G file.txt ]
then
  echo Arquivo pertence ao grupo do usuário     R=
fi
```

-L Para saber se é um link simbólico

```
if [ -L file.txt ]
then
    echo É um link simbólico  R=
fi
```

```
ln -s file.txt arq.txt
```

```
if [ -G arq.txt ]
then
    echo É um link simbólico    R=É um link simbólico
fi
```

-N Para saber se o arquivo foi modificado desde sua ultima leitura

```
touch file.txt
cat file.txt
```

```
if [ -N file.txt ]
then
  echo O arquivo foi modificado desde sua ultima leitura  R=
fi
```

if não executou comando nenhum já que eu fiz ler o arquivo usando cat, mas não modifiquei-o

```
cat  file.txt
echo mod  >> file.txt
```

```
if [ -N file.txt ]
then
  echo O arquivo foi modificado desde sua ultima leitura     R=O arquivo foi modificado….
fi
```

-nt Para saber se FILE1 foi modificado primeiro que FILE2

```
FILE1 -nt FILE2         #Sintaxe

touch file1.txt
touch file2.txt
echo  somar >> file2.txt

if [   file1.txt -nt file2.txt ]
then
    echo O arquivo file1.txt foi modificado primeiro que file2.txt  R=
fi

echo mod >> file1.txt   # Perceba que quando modifica o arquivo usando o comando ls -l o horário fica da atual modificação.

if [  file1.txt   -nt file2.txt ]
then
  echo O arquivo file1.txt foi modificado primeiro que file2.txt  R=O arquivo file1.txt ….
fi
```

* Usando o comando touch pode-se além de criar um arquivo, modificar seu horário de criação para o atual. Pode enganar a verificação usando touch ….

-ot Para saber se o arquivo FILE1 é mais antigo que FILE2

```
touch A.txt
touch B.txt

if [  A.txt -ot B.txt ]
then
   echo O arquivo A.txt é mais antigo que B.txt    R=O arquvio A.txt ….
fi
```

Mesmo o tempo de criação dos arquivos serem iguais no horário a verificação do if verificou corretamente dizendo que A.txt era mais antigo, mas poderá enganar a verificação após colocar touch A.txt para simular que criação é atual

```
touch A.txt     #FILE1 mais antigo = falso

if [  A.txt -ot B.txt ]
then
  echo  O arquivo A.txt é mais antigo que B.txt        R=
fi
```

-z Para saber se a string está vazia

```
if [ -z  ooo ]
then
  echo A string está vazia dentro do colchete          R=
fi

if [ -z    ]
then
  echo A string está vazia dentro do colchete          R=A string está vazia…
fi
```

* Uma ideia boa para usar este parâmetro por exemplo é verificar se uma variável está vazia ou inverter a função do operador -z com um sinal de ! antes para verificar se a variável possui uma string.

[ ] Se usarmos um colchete sem parâmetro o if verificará a existência de um caractere ou uma string para emitir verdadeiro

```
if [    ]
then
   echo Contém conteúdo dentro do colchete     R=
fi


if [ hello ]
then
  echo Contém contéudo dentro do colchete       R=Contém contéudo ...
fi
```

== Para saber se a STRING1 é igual a STRING2

```
if  [ A  == B ]
then
  echo A == A          R=
fi

if [ A == A ]
then
  echo A == A string1=string2      R = A == A …
fi
```

!= Para saber se existe diferença entre a STRING1 e a STRING2

```
if  [  AA  != AA ]
then
   echo A string1 é diferente da string2          R=
fi
```

```
if [  AA  != BB  ]
then
   echo A string1 é diferente da string2       R= A string1 é diferente ...
fi
```

< Saber se a string1 é menor que a string2 lexicográficamente

lexicográfica é transformar o alfabeto em sequência: a=1, b=2 ..

Comparar a string chocolate e chocotone qual é maior lexicográficamente

chocolate
chocotone

choco se repete nas duas strings portanto não se compara, mas logo após tem t e l separados

choco late
choco tone   #Na ordem do alfabeto t vem bem depois de l

l < t portanto chocotone é maior.

--No alfabeto l=12 e t=20

```
A=chocotone
B=chocolate

if [[ $A < $B ]]
then
   echo A variável $A é menor lexicográficamente que a variável $B  R=
fi

A=chocolate
B=chocotone

if  [[  $A < $B ]]
then
   echo A variável $A é menor lexicográficamente que a variável $B  R=A variável $A …
fi
```

[na verdade é o comando test (mas requer um argumento final de]). Embora todos os shells modernos tenham implementações embutidas de [, geralmente ainda há um executável externo com esse nome, por exemplo /bin/[

[[também é o comando test só que é mais moderno com funções atualizadas por exemplo o < que só é interpretado corretamente quando se usa dois colchetes.

[= Vem do shell POSIX do unix

[[= Vem do shell bash

-eq Para saber se o DIGITO1 é igual ao DIGITO2

Este parâmetro em inglês tem o significado de equal que em português é igual

```
if [  1 -eq  4 ]
then
  echo O DIGITO1 é igual o DIGITO2     R=
fi


if [  5 -eq 5 ]
then
   echo O DIGITO1 é igual o DIGITO2        R=O DIGITO1 é igual….
fi
```

-ne Para saber se o DIGITO1 é diferente do DIGITO2

```
if [  5  -ne  5 ]
then
  echo O DIGITO1 é diferente do DIGITO2     R=
fi

if [  1 -ne  2 ]
then
  echo O DIGITO1 é diferente do DIGITO2    R=O DIGITO1 é …...
fi
```

-lt Para se o DIGITO1 é menor que o DIGITO2

```
if [  4 -lt  9 ]
then
  echo O DIGITO1 é menor que o DIGITO2     R=O DIGITO1 é …..
fi


if [ 4 -le 4 ]
then
   echo O DIGITO1 é menor que o DIGITO2    R=
fi
```

* Como o DIGITO1 é igual o DIGITO2 e não menor não executa nada, mas no -le se o DIGITO1 for menor que o DIGITO2 ou igual emite verdadeiro.

-le Para saber se o DIGITO1 é menor que o DIGITO2 ou se o DIGITO1 é igual o DIGITO2

```
 if [  1 -le  7 ]
then
   echo O DIGITO1 é menor que o DIGITO2 ou igual   R=O DIGITO1….
fi
```

```
if [ 1 -le 1 ]
then
   echo  O DIGITO1 é menor que o DIGITO2 ou igual    R=O DIGITO1 …..
fi
```

-gt Para saber se o DIGITO1 é maior que o DIGITO2

```
 if [  8 -gt 1 ]
then
   echo O digito1 é maior que o digito2          R=O digito1 é...
fi
```

-ge Para saber se o DIGITO1 é maior que o DIGITO2 ou igual

```
if [  9 -ge  8 ]
then
  echo O DIGITO1 é maior que o DIGITO2 ou igual  R=O DIGITO1 é…
fi

if [  1 -ge 1 ]
then
   echo O DIGITO1 é maior que o DIGITO2 ou é igual   R=O DIGITO1….
fi
```