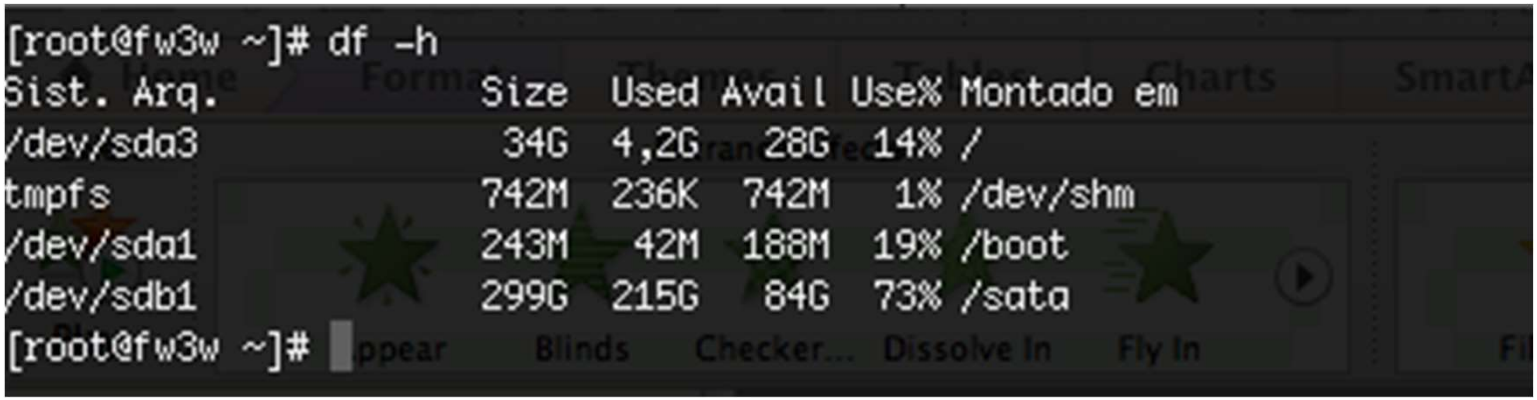
Redes de Computadores

Prof. André Nasserala

andre.nasserala@ufac.br

Comando DF

- Mostra o espaço livre/ocupado de cada partição.
- **df [opções]**
- Opções:
- -h, Mostra o espaço livre/ocupado em MB, KB, GB ao invés de blocos.
- Exemplo:
- **df –h**



Comando Ln

- Cria links para arquivos e diretórios no sistema. O link é um mecanismo que faz referência a outro arquivo ou diretório em outra localização.
- **ln [opções] [origem] [link]**
- Opções:
- -s: Cria um link simbólico. Cria ligações com o arq/dir de destino.
- Exemplo:
- **ln -s /dev/ttyS1 /dev/modem**
- Cria um link simbólico para o arquivo /dev/ttyS1 no arquivo /dev/modem.

Comando Du

- Mostra o espaço ocupado por arquivos e sub-diretórios do diretório atual.
- **du [opções]**
- Opções:
- -h, Mostra o espaço ocupado em formato legível por humanos (Kb, Mb) ao invés de usar blocos.
- -s, Faz uma totalização de todo conteúdo do diretório.
- **du –sh**

Comando Find

- Procura por arquivos/diretórios no disco.
- **find [diretório] [opções]**
- Opções:
- -name, especifica o nome do arq/diretório;
- -type [tipo]: Procura por arquivos do [tipo] especificado(f,d,l);
- Exemplo:
- **find / -name "*.doc"**
- Procura no diretório raíz e sub-diretórios um arquivo/diretório que terminem em .doc

Comando Free

- Mostra detalhes sobre a utilização da memória RAM do sistema.
- **free [opções]**
- opções
- -b: Mostra o resultado em bytes.
- -k: Mostra o resultado em Kbytes.
- -m: Mostra o resultado em Mbytes.
- -t: Mostra uma linha contendo o total.
- O free é uma interface ao arquivo /proc/meminfo.
- **free -m**

Comandos More e Less

- More ou Less permite fazer a paginação de arquivos ou da entrada padrão. O comando more pode ser usado como comando para leitura de arquivos que ocupem mais de uma tela. Quando toda a tela é ocupada, o more/less efetua uma pausa e permite que você pressione Enter ou espaço para continuar avançando no arquivo sendo visualizado. Para sair do more pressione q;

- Exemplos:
- **cat /etc/passwd | more**
- **cat /var/log/messages | less**

Comando Sort

- Organiza as linhas de um arquivo texto ou da entrada padrão.
- **sort [opções] [arquivo]**
- Opções:
- -f: Ignora a diferença entre maiúsculas e minúsculas.
- -r: Inverte o resultado da comparação.
- -n: Caso estiver organizando um campo que contém números.
- -c: Verifica se o arquivo já esta organizado. Caso não estiver, retorna a mensagem "disorder on arquivo”.
- **sort texto.txt**
- Organiza o arquivo texto.txt em ordem crescente.

Uptime, Dmesg e Uname

- Uptime mostra o tempo de execução do sistema desde que o computador foi ligado.
- **uptime**
- Dmesg mostra as mensagens de inicialização do kernel. São mostradas as mensagens da última inicialização do sistema.
- **dmesg | less**
- Uname retorna o nome e versão do kernel atual.
- **uname –a**
- Retorna todas as informações disponíveis sobre o kernel (-a = all).

Comando Seq

- Imprime uma sequência de números começando em [primeiro] e terminando em [último], utilizando [incremento] para avançar.
- **seq [primeiro] [incremento] [último]**
- Exemplos
- **seq 10**
- Exibe de 1 a 10
- **seq 0 2 10**
- Exibe: 0,2,4,6,8 e 10;
- **seq 10 -1 0**
- Exibe: 10,9,8,7,6,5,4,3,2,1 e 0

Comandos Diversos

- cal: exibe um calendário;
  - Pode-se usar: ”cal ano” que ele exibirá o calendário do ano solicitado.
- cd diretório: abre um diretório.
  - Por exemplo, para abrir a pasta /mnt, basta digitar cd /mnt. Para ir ao diretório raiz a partir de qualquer outro, digite apenas cd;
- diff arquivo1 arquivo2: indica as diferenças entre dois arquivos.
  - Por exemplo: diff a.txt b.txt;
- file arquivo: mostra informações de um arquivo(codificação por exemplo);
- history: mostra os últimos comandos inseridos;
- top: exibe a lista dos processos, conforme os recursos de memória consumidos;

Editor VI

- O editor de textos VI pode ser usado:
- **vi arquivo.sh**
- Um pouco de vi:
- Ao abrir um arquivo, como anteriormente, devemos abrir o modo de inserção apertando a **tecla "i"**(aparecerá "INSERT" no rodapé);
- Agora podemos digitar normalmente, e quando terminarmos de digitar devemos sair do modo de inserção apertando a **tecla ESC**;
- Fora do modo de inserção podemos salvar o arquivo com os comandos:
- **ESC + :x**
- Esse comando sai salvando o arquivo, se quiser sair e ignorar o que foi feito use **ESC + :q!**

Editor VI

- **Comandos iniciais:**
- $ vi => Abre o vim vazio, sem nenhum arquivo e exibe a tela de apresentação.
- $ vi arquivo => Abre o arquivo de nome "arquivo".
- $ vi arquivo + => Abre o arquivo de nome "arquivo", com o cursor no final do mesmo.
- $ vi arquivo +10 => Abre o arquivo de nome "arquivo", com o cursor na linha 10.
- $ vi arquivo +/Vasco => Abre o arquivo de nome "arquivo", na primeira ocorrência da palavra ”Vasco".

Editor VI

- **<u>Comandos para salvar e/ou sair:</u>**
- :w => Salva o arquivo que está sendo editado no momento.
- :q => Sai.
- :wq => Salva e sai.
- :x => Idem.
- ZZ => Idem.
- :w! => Salva forçado.
- :q! => Sai forçado.
- :wq! => Salva e sai forçado.

Editor VI

- **Subcomandos para localização de texto**
- /palavra => Procura pela palavra ou caractere acima ou abaixo do texto.
- ?palavra => Move para a ocorrência anterior da palavra (para repetir a busca use "n").
- n => Repete o ultimo comando utilizando / ou ?.
- N => Repete o ultimo comando / ou ? ao contrario (baixo para cima).
- Ctrl+g => Mostra o nome do arquivo, o número da linha corrente e o total de linhas.

Editor VI

- **Opções para remoção de caracteres**
- x => Apaga o caractere onde o cursor estiver.
- dd => Apaga a linha inteira onde o cursor estive
- D => Apaga a linha a partir da posição do cursor até o fim.
- J => Une a linha corrente à próxima.
- 5dd => Remove as próximas 5 linhas a partir da posição do atual do cursor (qualquer número).
- **Copiar e colar**
- yy => Copia a linha onde o cursor se encontra.
- 5yy => Copia as próximas 5 linhas a partir da posição atual do cursor.
- p => Cola o que foi copiado na linha abaixo do cursor atual.

Editor VI

- Você pode executar comandos do shell sem precisar sair do editor. Digite:
- :! comando
- O comando desejado será executado no shell e você terá volta à edição do arquivo pressionando ENTER.
- É possível ainda inserir a saída do comando no corpo do arquivo, por exemplo:
- :r! ls
- Lista os arquivos do diretório atual e insere o resultado no texto, a partir da posição do cursor, em vez de abrir um shell e retornar para o texto posteriormente.

Compactadores e Empacotadores

- **Tar**
- É um programa de arquivamento desenvolvido para armazenar e extrair arquivos de um arquivo tar (que contém os demais) conhecido como tarball.
- Quando ocorre o trabalho conjunto entre TAR e GZIP, o arquivo formado tem a extensão tar.gz.
- Ele é um aplicativo capaz de armazenar vários arquivos em um só. Porém, não é capaz de compactar os arquivos armazenados. Como é possível notar, o TAR serve de complemento para o GZIP e vice-versa.
- A síntaxe do TAR é :
- **tar [parâmetros] [-f arquivo] [-C diretório] [arquivos...].**

Compactadores e Empacotadores

- **Parâmetros**:
- -c - cria um novo arquivo tar;
- -M - cria, lista ou extrai um arquivo multivolume;
- -p - mantém as permissões originais do(s) arquivo(s);
- -r - acrescenta arquivos a um arquivo tar;
- -t - exibe o conteúdo de um arquivo tar;
- -v - exibe detalhes da operação;
- -w - pede confirmação antes de cada ação;
- -x - extrai arquivos de um arquivo tar;
- -z - comprime ou extrai arquivos tar resultante com o gzip;
- -j - comprime ou extrai arquivos tar resultante com o bz2;
- -f - especifica o arquivo tar a ser usado;
- -C - especifica o diretório dos arquivos a serem armazenados.

Compactadores e Empacotadores

- Para compactar arquivos no formato TAR.GZ usando use:
- **tar -zcvf arquivos.tar.gz arquivos/**
- Para compactar arquivos no formato TAR.BZ2 usando use:
- **tar -jcvf arquivos.tar.bz2 arquivos/**
- Para descompactar no formato TAR.GZ, no diretório corrente:
- **tar -zxvf arquivos.tar.gz**
- Para descompactar arquivos TAR.BZ2, no diretório corrente:
- **tar -jxvf arquivos.tar.bz2**

Compactadores e Empacotadores

- **Outros exemplos**
- **tar -c pasta > arq.tar**
- **tar -cvf arq.tar arq1 arq2**
- **tar -cvf /dev/fd1 /dir1/***
- **tar -cvMf /dev/fd1 /dir1 /dir2/subdir /dir3 /dir4**
- **tar -c -v -f arq.tar *.ext**
- **tar -cwf arq.tar pasta**
- **tar -czvf /pasta/arq.tgz ***
- **tar -czwf arq.tar.gz -C /dir1 arq1 -C /dir2 arq2 arq3**
- **tar -T /root/lista.txt -cf backup.tar**

Introdução ao Shell Script

- O que é o shell
- O shell é o "prompt" da linha de comando do Unix e Linux, é o servo que recebe os comandos digitados pelo usuário e os executa.
- O shell é aquele que aparece logo após digitar-se a senha do usuário e entrar na tela preta. Ou na interface gráfica, ao clicar no ícone do Xterm, rxvt, Terminal ou Console.
- Um script é um arquivo que guarda vários comandos e pode ser executado sempre que preciso.
- Os comandos de um script são exatamente os mesmos que se digita no prompt, é tudo shell.

Introdução ao Shell Script

- Por exemplo, se de tempos em tempos você quer saber informações do sistema como horário, ocupação do disco e os usuários que estão logados, é preciso digitar três comandos:
- **date, df e w**
- É melhor fazer um script chamado "sistema.sh" e colocar estes comandos nele. O conteúdo do arquivo "sistema.sh" seria o seguinte:
- **#!/bin/bash**
- **date**
- **df**
- **w**

Introdução ao Shell Script

- O editor de textos VI pode ser usado para isso:
- **vi sistema.sh**
- Um pouco de vi:
- Ao abrir um arquivo, como anteriormente, devemos abrir o modo de inserção apertando a **tecla "i"**(aparecerá "INSERT" no rodapé);
- Agora podemos digitar normalmente, e quando terminarmos de digitar devemos sair do modo de inserção apertando a **tecla ESC**;
- Fora do modo de inserção podemos salvar o arquivo com os comandos:
- **ESC + :x**
- Esse comando sai salvando o arquivo, se quiser sair e ignorar o que foi feito use **ESC + :q!**

Introdução ao Shell Script

- Voltando ao exemplo:
- E para chamar este script, basta agora executar apenas uns comandos:
- **chmod +x sistema.sh**
- **./sistema.sh**
- Ao executar o ultimo comando o shell executará os comandos listados no arquivo, ou seja, **date, df e w;**
- Isso é um shell script, um arquivo de texto que contém comandos do sistema e pode ser executado pelo usuário.

Introdução ao Shell Script

- Passos para criar um shell script:

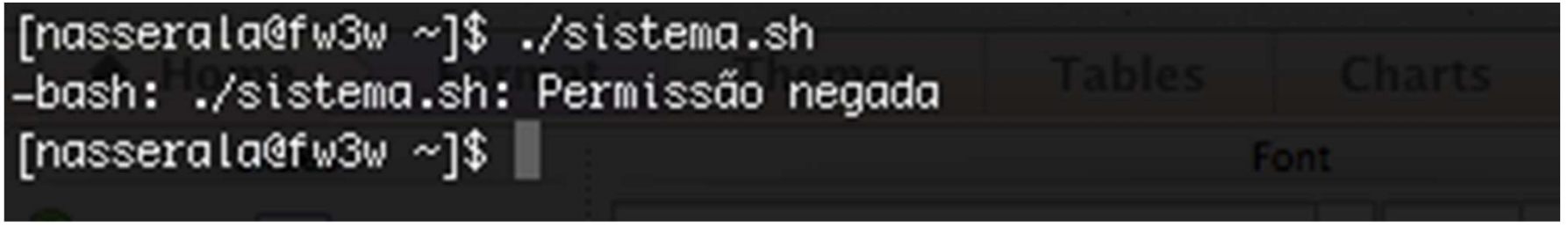
1. Escolher um nome para o script(sistema.sh);
2. Escolher o diretório onde colocar o script(/root);
3. Criar o arquivo e colocar nele os comandos(vi sistema.sh);
4. Colocar a chamada do shell na primeira linha(#!/bin/bash);
5. Tornar o script um arquivo executável(chmod +x sistema.sh).

- Assim é aparentemente simples a criação de shell script's, mas devido a vasta gama de comandos e funções existente em Linux podemos criar scripts poderosos para administração e gerencia de um servidor/desktop.

Introdução ao Shell Script

- Problemas na execução do script

1. Comando não encontrado: Verifique se o comando que você está chamando tem exatamente o mesmo nome do seu script;
2. Permissão Negada: O shell encontrou seu script, mas ele não é executável;
3. Erro de Sintaxe: O shell encontrou e executou seu script, porém ele tem erros;



Introdução ao Shell Script

- Nesse ponto, já sabemos o básico necessário para fazer um script em shell do zero e executá-lo.
- Mas apenas colocar os comandos em um arquivo não torna este script útil. Vamos fazer algumas melhorias nele para que fique mais compreensível.
- **Melhorar a saída na tela**
- Executar os três comandos seguidos resulta em um bolo de texto na tela, misturando as informações e dificultando o entendimento. É preciso trabalhar um pouco a saída do script, tornando-a mais legível.
- O comando "echo" serve para mostrar mensagens na tela. Que tal anunciar cada comando antes de executá-lo?

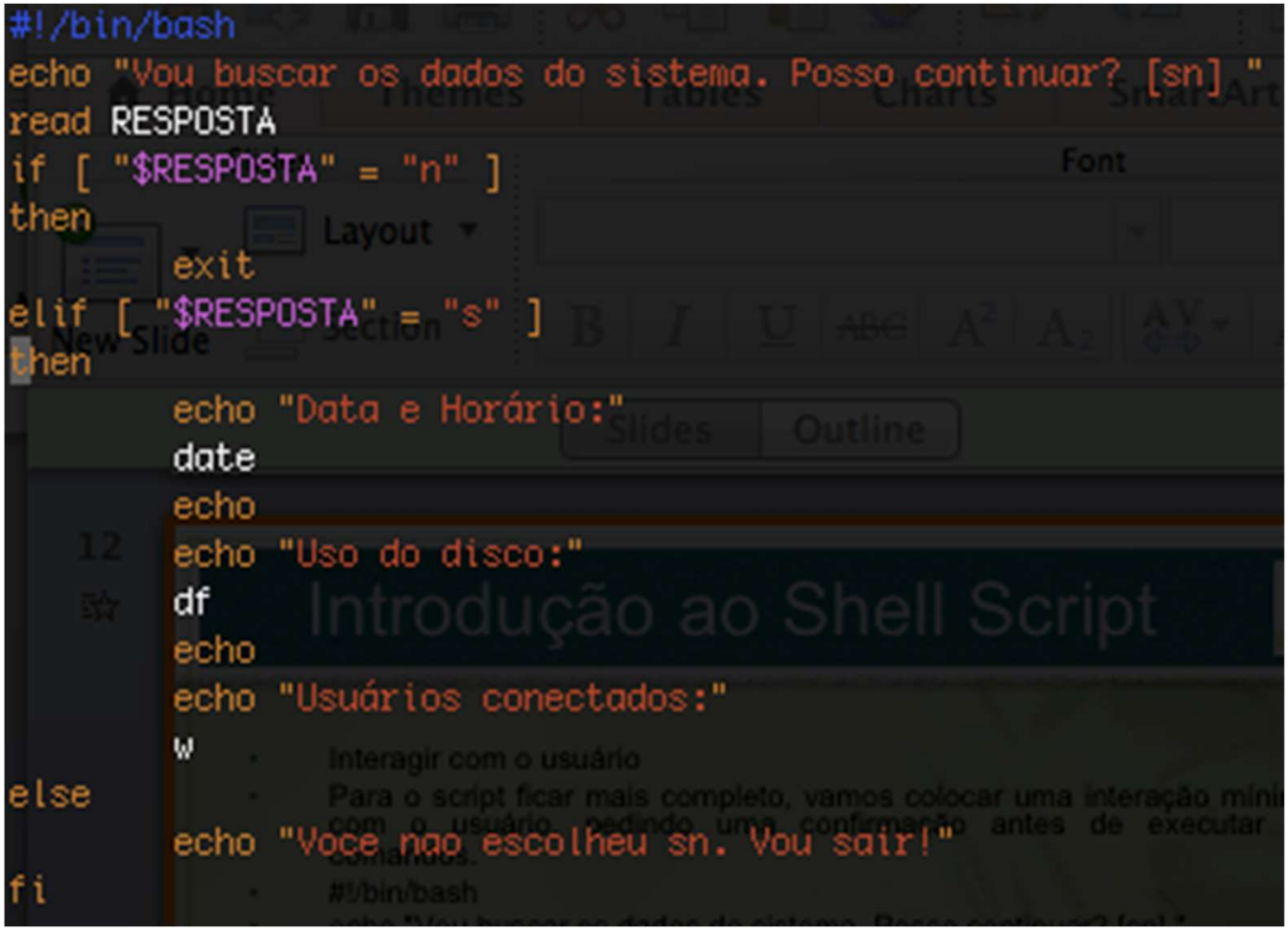
Introdução ao Shell Script

```
#!/bin/bash
echo "Data e Horário:"
date
echo
echo "Uso do disco:"
df
echo
echo "Usuários conectados:"
w
```

- Para usar o echo, basta colocar o texto entre "aspas". Se nenhum texto for colocado, uma linha em branco é mostrada.

Introdução ao Shell Script

- Interagir com o usuário
- Para o script ficar mais completo, vamos colocar uma interação mínima com o usuário, pedindo uma confirmação antes de executar os comandos.



Introdução ao Shell Script

- **Melhorar o código do script**
- Com o tempo, o script vai crescer, mais comandos vão ser adicionados e quanto maior, mais difícil encontrar o ponto certo onde fazer a alteração ou corrigir algum erro.
- Para poupar horas de estresse, e facilitar as manutenções futuras, é preciso deixar o código visualmente mais agradável e espaçado, e colocar comentários esclarecedores.
- Basta iniciar a linha com um "#" e escrever o texto do comentário em seguida. Estas linhas são ignoradas pelo shell durante a execução. O cabeçalho com informações sobre o script e seu autor também é importante para ter-se uma visão geral do que o script faz, sem precisar decifrar seu código.

```
#!/bin/bash
# Script para buscar informacoes do sistema
# Autor: Andre Luiz Nasserala Pires
# Versao: 1.0b

# Pede uma confirmação do usuário antes de executar
echo "Vou buscar os dados do sistema. Posso continuar? [sn] "
read RESPOSTA
# Se ele digitou 'n', vamos interromper o script
if [ "$RESPOSTA" = "n" ]
then
        exit
# se e;e dogitou 's', continuamos
elif [ "$RESPOSTA" = "s" ]
then
        echo "Data e Horário:"
        date # Mostra a data
        echo
        echo "Uso do disco:"
        df # Mostra a ocupacao dos discos
        echo
        echo "Usuários conectados:"
        w # Mostra os usuarios logados
else
        echo "Voce nao escolheu sn. Vou sair!"
fi
```

Introdução ao Shell Script

- É possível armazenar a saída de um comando dentro de uma variável. Ao invés de aspas, o comando deve ser colocado entre "$(...)", veja:
- **HOJE=$(date)**
- **echo "Hoje é: $HOJE"**
- **Hoje é: Sáb Abr 24 18:40:00 BRT 2004**
- **unset HOJE**
- **echo $HOJE**
- E finalmente, o comando "unset" apaga uma variável. Para ver quais as variáveis que o shell já define por padrão, use o comando "env“.

Introdução ao Shell Script

Comando	Função	Opções úteis
cat	*Mostra arquivo*	-n, -s
cut	*Extrai campo*	-d -f, -c
date	*Mostra data*	-d, +'...'
find	*Encontra arquivos*	-name, -iname, -type f, -exec
grep	*Encontra texto*	-i, -v, -r, -qs, -w -x
head	*Mostra Início*	-n, -c
printf	*Mostra texto*	*nenhuma*
rev	*Inverte texto*	*nenhuma*
sed	*Edita texto*	-n, s/isso/aquilo/, d
seq	*Conta Números*	-s, -f
sort	*Ordena texto*	-n, -f, -r, -k -t, -o
tail	*Mostra Final*	-n, -c, -f
tr	*Transforma texto*	-d, -s, A-Z a-z
uniq	*Remove duplicatas*	-i, -d, -u
wc	*Conta Letras*	-c, -w, -l, -L

Introdução ao Shell Script

	Testes em variáveis
-lt	Núm. é menor que (LessThan)
-gt	Núm. é maior que (GreaterThan)
-le	Núm. é menor igual (LessEqual)
-ge	Núm. é maior igual (GreaterEqual)
-eq	Núm. é igual (EQual)
-ne	Núm. é diferente (NotEqual)
=	String é igual
!=	String é diferente
-n	String é não nula
-z	String é nula

	Testes em arquivos
-d	É um diretório
-f	É um arquivo normal
-r	O arquivo tem permissão de leitura
-s	O tamanho do arquivo é maior que zero
-w	O arquivo tem permissão de escrita
-nt	O arquivo é mais recente (NewerThan)
-ot	O arquivo é mais antigo (OlderThan)
-ef	O arquivo é o mesmo (EqualFile)
-a	E lógico (AND)
-o	OU lógico (OR)

Introdução ao Shell Script

- Até agora vimos o básico, o necessário para se fazer um script de funcionalidade mínima.
- Recebimento de opções e parâmetros: Assim como os comandos do sistema que possuem e opções e parâmetros, os scripts também podem ser preparados para receber dados via linha de comando.
- Dentro do script, algumas variáveis especiais são definidas automaticamente, em especial, "$1" contém o primeiro argumento recebido na linha de comando, "$2" o segundo, e assim por diante.
- Veja o script "argumentos":

Introdução ao Shell Script

- **#!/bin/bash**
- **# argumentos - mostra o valor das variáveis especiais**
- **echo "O nome deste script é: $0"**
- **echo "Recebidos $# argumentos: $*"**
- **echo "O primeiro argumento recebido foi: $1"**
- **echo "O segundo argumento recebido foi: $2”**

- Ele serve para demonstrar o conteúdo de algumas variáveis especiais, acompanhe:

Introdução ao Shell Script

- **./argumentos um dois três**
- O nome deste script é: ./argumentos
- Recebidos 3 argumentos: um dois três
- O primeiro argumento recebido foi: um
- O segundo argumento recebido foi: dois

- O acesso é direto, basta referenciar a variável que o valor já estará definido.
- Assim é possível criar scripts que tenham opções como --help, --version e outras

Introdução ao Shell Script

- Expressões aritméticas: O shell também sabe fazer contas. A construção usada para indicar uma expressão aritmética é "$((...))", com dois parênteses.

- **prompt$ echo $((2*3)) : 6**
- **prompt$ echo $((2*3-2/2+3)) : 8**
- **prompt$ NUM=44**
- **prompt$ echo $((NUM*2)) : 88**
- **prompt$ NUM=$((NUM+1))**
- **prompt$ echo $NUM : 45**

Introdução ao Shell Script

- **If, for e while**
- Assim como qualquer outra linguagem de programação, o shell também tem estruturas para se fazer condicionais e loop. As mais usadas são if, for e while.

```
if COMANDO
then
  comandos
else
  comandos
fi
```

```
for VAR in LISTA
do
  comandos
done
```

```
while COMANDO
do
  comandos
done
```

Introdução ao Shell Script

- 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, 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:

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

Introdução ao Shell Script

- 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:
- **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.

Introdução ao Shell Script

- Já o while é um laço que é executado enquanto um comando retorna OK.
- Novamente o test é bom de ser usado. Por exemplo, para segurar o processamento do script enquanto um arquivo de lock não é removido:
- **while test -f /tmp/lock**
- **do**
- **echo "Script travado..."**
- **sleep 1**
- **done**

Introdução ao Shell Script

- E por fim, o for percorre uma lista de palavras, pegando uma por vez:
- **for numero in um dois três quatro cinco**
- **do**
- **echo "Contando: $numero"**
- **done**
- Uma ferramenta muito útil para usar com o for é o seq, que gera uma seqüência numérica.
- Para fazer o loop andar 10 passos, pode-se fazer:
- **for passo in $(seq 10)**

Introdução ao Shell Script

- O mesmo pode ser feito com o while, usando um contador:

```
i=0
while test “$i” -le 10
do
    i=$((i+1))
    echo "Contando: $i"
done
```

Introdução ao Shell Script

- E temos ainda o loop infinito, com condicional de saída usando o "break¨:
- **while :**
- **do**
- **if test -f /tmp/lock**
- **then**
- **echo "Aguardando liberação do lock..."**
- **sleep 1**
- **else**
- **break**
- **fi**
- **done**