Shell Script do zero

Aula 1 – Básico do Básico

Apresentação

Este material é dedicado para aqueles que não sabem nada de lógica de programação e Shell Script, com as aulas a seguir você será capaz de criar scripts básicos a medianos e terá todas as condições de se aprofundar no tema sozinho, procuro sempre dar muitos exemplos para que você entenda de um jeito ou de outro.

Não se preocupem se os conceitos apresentados até na aula 3 ficarem vagos, a partir da aula 4 trabalharemos na pratica e tudo ficará mais fácil. O conteúdo das aulas são pequenos e objetivos, justamente para que você possa absolver melhor e te-los em mente no decorrer das outras aulas.

Esclarecendo que sei pouco, mas juntando este pouco com criatividade e persistência, eu consigo fazer muita coisa.

O que é Shell Script

Script é um arquivo com várias instruções, que são executadas pelo shell, que é o interpretador de comandos. Com ele podemos automatizar muitas tarefas no Linux criando grandes facilidades.

Onde escrever os scripts

Uma função interessante em alguns editores de texto no Linux, é que eles facilitam a visualização e escrita de scripts, colorindo os comandos e suas estruturas, recomendo o gedit ou pluma. Para que esta função seja ativada é necessário colocar no começo do script #!/bin/bash ou #!/bin/sh etc, e salvar, esta linha especifica o interpretador de comandos, caso não tenha, o shell executa qualquer um.

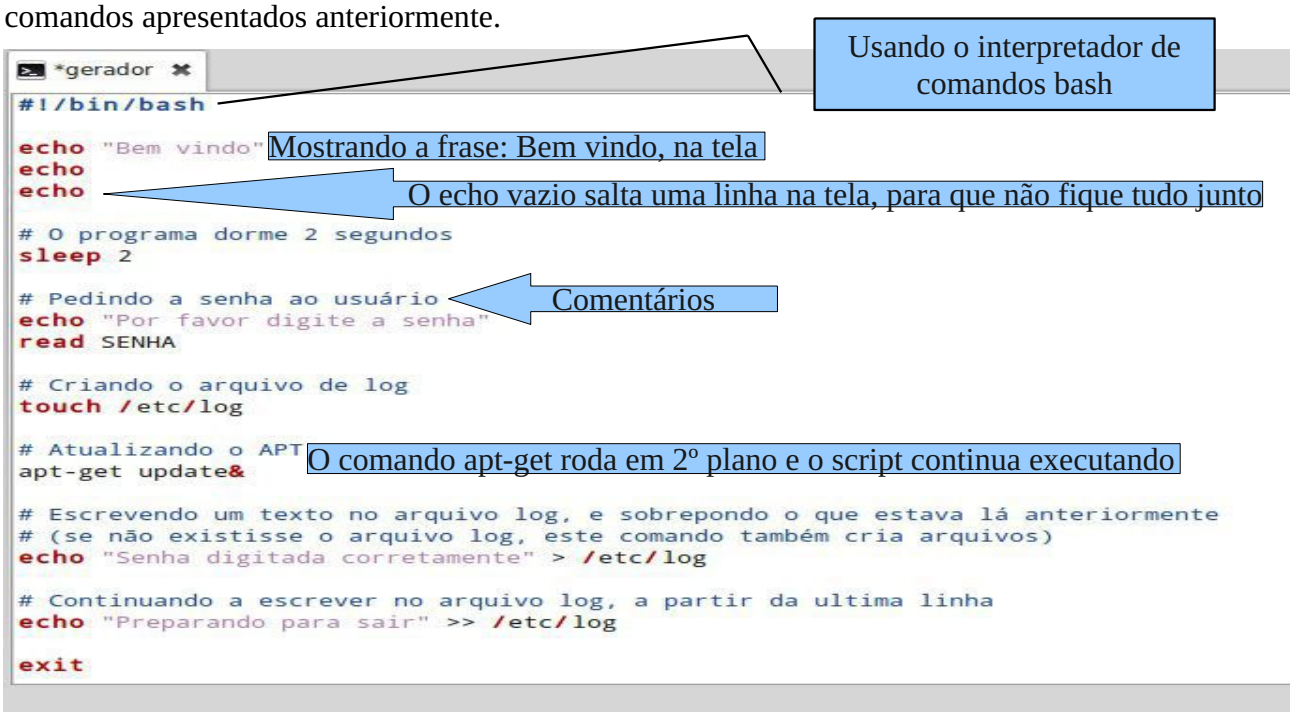
Primeiros Comandos

Qualquer comando do terminal podemos usar na escrita dos nossos scripts, desde comandos criados por você mesmo (colocando o script dentro de /bin), programas de terceiros e que tem seus comandos no terminal e principalmente alguns comandos do shell que são muito usados em scripts e pouco conhecidos no terminal. Vamos vê-los agora:

Comando	Descrição	Sintaxe
echo	Exibe o texto na tela	echo “texto a ser mostrado”
sleep	Dá um tempo antes de seguir	sleep segundos **exemplo:** Sleep 1
read	Recebe o valor de uma variável (próxima aula)	read variável **exemplo:** read dados
>	Escreve num arquivo texto *(apagando o que estava lá)*	echo “texto” > /home/luiz/arquivo
>>	Escreve num arquivo texto *(na ultima linha)*	echo “texto” >> /home/luiz/arquivo
&	Roda o comando em 2º plano e continua o script	Comando&
exit	Sai do script	exit
touch	Cria arquivos texto	touch arquivo
#	Comenta tudo depois deste simbolo	# Comentário

Exemplo destes comandos no script

É claro que o script a seguir não tem muito sentido, é só para visualizarmos a aplicação dos comandos apresentados anteriormente.



Podemos ter mais de um comando na mesma linha, basta usarmos o pipeline, que é este simbolo: "|" ele fica perto da letra z, mas teste, porque nem tudo vai funcionar assim.

Comandos mais conhecidos

São os comandos que estamos acostumados a usar no terminal e podemos usa-los também no script, se você não os conhece, vá aprendendo de acordo com a necessidade, pesquise na internet e consulte a tabela resumida abaixo sempre que preciso.

Diretórios		
Comando	**Sintaxe**	**Descrição**
rm -rf	rm -rf +diretório	Deleta arquivos/pastas e tudo que estiver dentro (cuidado)
pwd	pwd	Mostra em qual diretório estamos
chmod	chmod 777 arquivo_ou_pasta	Muda as permissões, 777 = permissão total
chown	chown user:grupo arq_ou_diret.	Muda o proprietário de arquivos e pastas
cd	cd diretório	Entra em diretórios
Usuários		
Comando	**Sintaxe**	**Descrição**
useradd	useradd luiz -g alunos (no grupo)	Adiciona um usuário
userdel	userdel usuário	Deleta usuário e seus arquivos
groupdel	groupdel grupo	Deleta um grupo
groups	groups nome_usuario	Mostra os grupos do usuário
addgroup	addgroup usuario grupo ou addgroup nomedogrupo	Cria um grupo ou adiciona um usuário ao grupo
sudo	sudo comando	Executa comandos como root
whoami	whoami	Identifica com qual usuário você esta logado

Rede		
Comando	**Sintaxe**	**Descrição**
ifconfig	ifconfig	Mostra as interfaces de rede
hostname	hostname	Mostra ou muda o nome de seu computador na rede
ping	Ping ip_desejado	Dispara pacotes para outro pc, para testar conexões etc
Sistema		
Comando	**Sintaxe**	**Descrição**
killall	Killall nome_do_programa	Mata um processo
whatis	Whatis +nome do programa	Descreve o que faz o comando
diff	diff arquivo1 arquivo2	Compara 2 arquivos
ps	ps -elf	Mostra os programas que estão rodando
cat	cat arquivo_texto	Mostra o conteúdo de um arquivo de texto
grep	Comando | grep palavra	Filtra a saída do comando, mostra a linha da palavra pedida
ln	ln -s arquivo_original atalho	Cria atalho
cp	cp arquivo destino	Copia um arquivo ou diretório (-R para diretórios)
apt-get	apt-get nome_programa	Instala aplicativos
find	Find +nome	Procura por arquivos e diretórios

Compactação		
Tar.gz (A melhor em tempo vs compactação)		
Comando	**Função**	**Descrição**
tar -zcf novo.tar.gz pasta_ou_arquivo	Compactar	z=zip c=compact f=file
tar -zxf pasta_ou_arquivo	Descompactar	x=extrair z=zip f=file
Tar (Apenas junta)		
Comando	**Função**	**Descrição**
tar -cf arquivo_novo.tar arquivo	Apenas juntar os arquivos	-c comprimir -f file
tar -xf arquivo.tar	Extrair	-x extrair -f file
Rar		
Comando	**Função**	**Descrição**
rar a novo.rar arquivo	Compacta	a = Adiciona
unrar arquivo.rar	Descompacta	
Tar.xz (Compacta mais)		
Comando	**Função**	**Descrição**
tar -Jcf arquivo_novo.tar.xz arquivo	Compactar	-J = xz -c cria -f files
tar -Jxf arquivo.tar.xz	Descompactar	-x extrai -P preserva permissões

Shell Script do zero

Aula 2 – Variáveis

São muitos os valores que lidamos na programação e eles variam muito, por isto é que existem as variáveis, elas podem assumir qualquer valor, numéricos ou alfanuméricos a qualquer momento.

Portanto, variável é um nome com um valor dentro dela que fica armazenado na memória para ser usado quando preciso. (seu nome nunca começa com número).

Para confundir menos, é recomendado escrever as variáveis com letras maiúsculas e nem precisa falar que não se pode escreve-las com acento.

Valor:

Numérico → Números armazenados (para fazermos contas)
Alfanuméricos → Podem ser números, textos ou os dois juntos, o importante saber é que sempre será considerado como um texto

Exemplificando:

Posso criar um script que necessite colher o nome de alunos, mas a cada rodada o nome será diferente, então eu posso criar a variável ALUNO que armazenará este valor dentro dela.

Com o simbolo "$" antes da variável, prevalece o valor que esta dentro dela.
Exemplo: Dentro da variável "FAZER" tenho este valor → "mkdir programa"

No script digito uma linha assim: $FAZER
O que vai acontecer ???
O shell vai dar o comando "mkdir programa" criando o diretório pedido
Por causa do simbolo ele considera o conteúdo

Exemplos de como colher os dados para a variável:
Não se preocupe em decorar estes comandos, procure entender o raciocínio.

Você mesmo dá o valor dentro do script

```
ALUNO=$"Jonatan"
```

Aqui ao invés de chamar de variável, podemos chama-la de constante, já que o valor Jonatan não muda, a não ser que você crie outra linha modificando este valor

Recebendo o valor digitado pelo usuário

```
echo "Digite o nome do aluno"
read ALUNO
```

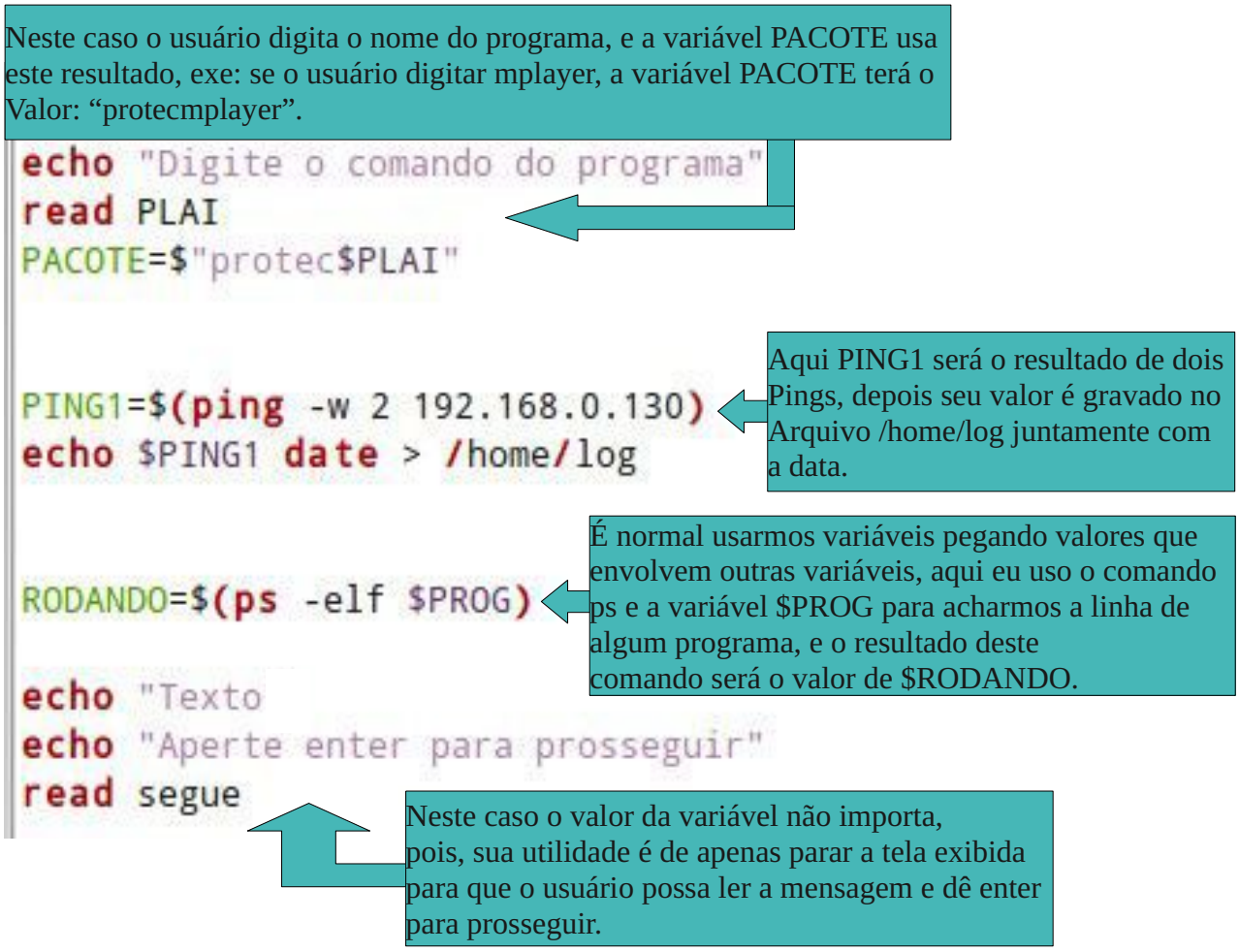
Onde "read" é o comando para que o usuário digite o valor da variável em questão

Pegando o valor de um arquivo texto

```
ALUNO=$(cat /etc/matricula)
```

O valor da variável será o resultado do comando dentro do parênteses, neste caso mostra o conteúdo do arquivo matricula.

O Linux é case sensitive, ou seja, se escreveu determinada variável em maiúscula, você não pode mudar para minuscula no meio do Script, porque ele reconhece como outra palavra.



As variações são muitas e a medida que for necessário ou pedido, podemos abordar o assunto mais minuciosamente.

Shell Script do zero

Aula 3 - Operadores lógicos de comparação

Nós vamos ver estes operadores dentro de um determinado contexto, pois é o que necessitamos por enquanto, qualquer função que seja um pouco diferente do descrito aqui, vamos abordar depois da aula 7.

Os operadores lógicos de comparação mostram ao shell uma condição a ser testada, o resultado do teste pode ser **verdadeiro** ou **falso**, este resultado é usado por vários comandos (vamos entender estes comandos na medida do possível), um deles e um dos mais importantes é o IF, que veremos na próxima aula.

Estes operadores são fundamentais em scripts/programação.

O que podemos entender como **verdadeiro** e **falso ?**

A frase abaixo, por mais obvia que seja, ela é verdadeira:
4 é menor que 5

Esta também é verdadeira:
O nome camila é diferente do nome Julia

Por fim, esta é falsa:
50 é diferente de 50

O conteúdo destes comparadores podem ser **numéricos** ou **Alfanuméricos** (vai fazer comparação que necessita considerar o número matematicamente, então é numérico o resto é alfanumérico), por exemplo, se eu vou lidar com números de cadastros de funcionários vou considerá-los Alfanuméricos porque além de não ter conta matemática, eu só vou ler como se fosse um texto, já o numérico pode ser quantidade, comparação (1 menor que 2), cálculos etc. Nunca poderemos confundir estas duas categorias.

Comparadores Numéricos

-lt	Número é menor que (Less Than)
-gt	Número é maior que (Greater Than)
-le	Número é menor ou igual (Less Equal)
-ge	Número é maior ou igual (Greater Equal)
-eq	Número é igual (EQual)
-ne	Número é diferente (Not Equal)

Exemplificando:

Supondo que a variável **A** tem o valor de **30** e **B** tem o valor de **20**, então poderíamos exemplificar as seguintes situações:

Ele não executa o comando, porque a condição não é satisfeita, pois, A não é menor que B.
se [$A **-lt** $B]; então faça (se A é menor que B então faça o comando) Condição Falsa
comando

Neste caso, ele executaria o comando, porque a condição é satisfeita, já que A é diferente de B
se [$A **-ne** $B]; então faça Condição Verdadeira
comando

Comparadores Alfanuméricos

=	Texto é igual
!=	Texto é diferente
-n	Texto não nulo
-z	Texto é nulo

Suponhamos agora que **A** contém a palavra "**noite**" e **B** contém a palavra "**dia**", então:

Aqui ele não executa o comando, porque A não é igual a B.
se [$A = $B]; então faça
comando

Aqui ele executa o comando, já que A é diferente de B.
se [$A **!=** $B]; então faça
comando

Com o "-n" ele executará o comando caso a variável tenha algum valor, já com o "-z", só executará se a variável estiver vazia. Embora esteja na tabela de alfanuméricos eles também funcionam se a variável conter apenas números. Vamos ver como escrevê-los na próxima aula.

Shell Script do zero

Aula 3 - Operadores lógicos de comparação

Nós vamos ver estes operadores dentro de um determinado contexto, pois é o que necessitamos por enquanto, qualquer função que seja um pouco diferente do descrito aqui, vamos abordar depois da aula 7.

Os operadores lógicos de comparação mostram ao shell uma condição a ser testada, o resultado do teste pode ser **verdadeiro** ou **falso**, este resultado é usado por vários comandos (vamos entender estes comandos na medida do possível), um deles e um dos mais importantes é o IF, que veremos na próxima aula.

Estes operadores são fundamentais em scripts/programação.

O que podemos entender como **verdadeiro** e **falso ?**

A frase abaixo, por mais obvia que seja, ela é verdadeira:
4 é menor que 5

Esta também é verdadeira:
O nome camila é diferente do nome Julia

Por fim, esta é falsa:
50 é diferente de 50

O conteúdo destes comparadores podem ser **numéricos** ou **Alfanuméricos** (vai fazer comparação que necessita considerar o número matematicamente, então é numérico o resto é alfanumérico), por exemplo, se eu vou lidar com números de cadastros de funcionários vou considerá-los Alfanuméricos porque além de não ter conta matemática, eu só vou ler como se fosse um texto, já o numérico pode ser quantidade, comparação (1 menor que 2), cálculos etc. Nunca poderemos confundir estas duas categorias.

Comparadores Numéricos

-lt	Número é menor que (Less Than)
-gt	Número é maior que (Greater Than)
-le	Número é menor ou igual (Less Equal)
-ge	Número é maior ou igual (Greater Equal)
-eq	Número é igual (EQual)
-ne	Número é diferente (Not Equal)

Exemplificando:

Supondo que a variável **A** tem o valor de **30** e **B** tem o valor de **20**, então poderíamos exemplificar as seguintes situações:

Ele não executa o comando, porque a condição não é satisfeita, pois, A não é menor que B.
se [$A **-lt** $B]; então faça (se A é menor que B então faça o comando) Condição Falsa
comando

Neste caso, ele executaria o comando, porque a condição é satisfeita, já que A é diferente de B
se [$A **-ne** $B]; então faça Condição Verdadeira
comando

Comparadores Alfanuméricos

=	Texto é igual
!=	Texto é diferente
-n	Texto não nulo
-z	Texto é nulo

Suponhamos agora que **A** contém a palavra "**noite**" e **B** contém a palavra "**dia**", então:

Aqui ele não executa o comando, porque A não é igual a B.
se [$A = $B]; então faça
comando

Aqui ele executa o comando, já que A é diferente de B.
se [$A **!=** $B]; então faça
comando

Com o "-n" ele executará o comando caso a variável tenha algum valor, já com o "-z", só executará se a variável estiver vazia. Embora esteja na tabela de alfanuméricos eles também funcionam se a variável conter apenas números. Vamos ver como escrevê-los na próxima aula.

Shell Script do zero

Aula 4 – Condição IF e escrevendo o primeiro script

Condição IF

No inglês “if” significa “se”, (SE a condição for satisfeita eu executo o comando), condição que será testada usando o conceito da última aula. O teste de comparação acusou verdadeiro, então ele executa o comando que está dentro do if, o teste acusou falso então ele pula este if e segue com o script.

Pense nos operadores de comparação como se fossem chaves, e o if sendo a porta, se a chave for “verdadeira” ela abre a porta e executa o que tem lá dentro, se a chave for falsa o shell não consegue abrir a porta e consequentemente não executa o que esta lá dentro. Nesta aula aprenderemos como escrevê-lo na linguagem do shell e consequentemente o primeiro script.

Estrutura do if

```
if [ condição ];then
   comando
fi
```

- if → Abre o comando
- Condição → Condição comparativa para execução dos comandos dentro do if
- Comando → Qualquer comando do shell e quantos você quiser
- fi → Fecha o if (escrito ao contrário)

Exemplo na linguagem do shell:

␣ → Espaço obrigatório

A palavra julia esta entre aspas porque é um texto (numéricos ficam sem aspas), a mesma coisa vale para a váriavel.

```
if [␣"$USUARIO"␣=␣"julia"␣];then
   mkdir $USUARIO
fi
```

```
#!/bin/bash

if [ "$USUARIO" = "julia" ];then
mkdir $USUARIO
fi
```

No Shell

Tradução ↑
Se o conteúdo da variável USUARIO é igual a palavra julia então execute o comando, que no caso cria um diretório com o nome de julia.

Comando auxiliar Else

Uma função complementar e muito útil no comando if é o else (senão), caso a condição do if não seja verdadeira, ele automaticamente executa o que está no else, seu uso é opcional.

Sentido do comando	Sintaxe no shell
se [condição é verdadeira]; **então execute o comando** mkdir $USUARIO **senão execute este outro comando** mkdir $USUARIO2	if [condição];then comando else comando fi

↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓ **Nosso primeiro script**

Vamos fazer nosso primeiro script/exercício, no link do final da página temos o script pronto, mas é claro que você vai resolvê-lo antes de conferir, não precisa ficar igual, basta funcionar.

Passos a passo de como fazer o script, antes do exercício:

- Abra o editor de texto escolhido
- Escreva na primeira linha #!/bin/bash
- Agora vamos salvar para que as linhas fiquem coloridas
- Tem pessoas que colocam a extensão .sh no nome do arquivo (tanto faz)
- Para não termos problemas futuros, já vamos dar permissão de execução (chmod +x script)
- Agora já podemos escrever o script
- Quando o script estiver pronto entre no diretório em questão e dê o comando ./nome_do_script para executá-lo

Lembretes:

- Preste muita atenção nos espaços, porque se der espaço a mais ou a menos não vai rodar.
- Lembre-se de sempre fechar o if → "fi"
- Os comandos sempre são executados de cima para baixo sequencialmente, ou seja, só executa a próxima linha quando terminar a atual, a não ser que usemos o "&"
- Quando houver erros, o terminal mostra em qual linha o mesmo está, nem sempre é aquela linha indicada, pode ser algumas depois
- Verticalmente pode saltar quantas linhas quiser

Exercício

Função do script → Temos 3 médicos, cada um atende num turno diferente, o usuário informa qual turno deseja se consultar e o programa mostra o nome do médico do turno escolhido, só isso!

Etapas do script:

1- Mostra um texto de boas vindas ao usuário
2- Pede que o usuário escolha qual turno em que deseja se consultar
3- Mostra o texto nestes moldes: Médico TAL é o único a atender no turno TAL
4- "Dorme" por 2 segundos
5- Informa que a consulta está marcada
6- O programa "dorme" por mais 4 segundos e sai

Shell Script do zero

Aula 5 – Operadores Lógicos de Conexão

Com estes operadores podemos conectar duas ou mais condições criadas com os operadores de comparação, criando assim mais de um evento a ser testado pelo shell e a "porta" só será aberta se o resultado desta conexão for verdadeira. Abaixo aprenderemos sobre os conectores **E** e **OU**.

Vamos visualizar como ficam os conectores para ficar mais claro:

* No exemplo acima temos dois conectores, mas é só exemplo, já que o correto ali seria apenas um conector.

Operador lógico (E)

Entenda **conjunção** sendo a união das condições comparativas feitas pelos conectores.
rasgando o verbo: É tudo que esta dentro dos colchetes do if [ ]

A **conjunção** é verdadeira se todas as condições de comparação forem verdadeiras, então mostro a tabela para entendermos a lógica.

Nº	Condição 1	Conector	Condição 2	Resultado do teste	Explicação
1º	V	**e**	V	Verdadeiro	Porque as duas condições são verdadeiras
2º	F	**e**	V	Falso	É falso porque apenas uma condição atende
3º	V	**e**	F	Falso	É falso porque apenas uma condição atende
4º	F	**e**	F	Falso	Nenhuma condição atende

Exemplificando:

Vamos considerar esta lógica como se fosse um porteiro e ele libera o acesso de acordo com a situação descrita abaixo:

Só entra na festa casais que o homem se chama César **E** a mulher Juliana:

1º	César	**e**	Juliana	Entrada permitida (as duas condições são verdadeiras)
2º	Paulo	**e**	Juliana	O nome Juliana bate mas o nome Paulo não atende, barrados !
3º	César	**e**	Mônica	O nome César está na lista mas o nome Mônica não, barrados !
4º	Júlio	**e**	Carolina	Nenhum dos dois nomes estão na lista, barrados !

Muitas vezes vamos esbarrar com a necessidade de usar os conectores, vamos supor que eu preciso de um if rodando apenas em duas situações:

Eu fiz um script que diminuía a velocidade dos meus downloads para 100k quando a minha irmã conectava o notebook na internet, mas quando o pc da minha mãe estava ligado simultaneamente, como eu poderia diminuir a velocidade para 50k compensando duas máquinas ligadas? Então eu criei um if para esta situação.

Esta é a parte que identifica as duas máquinas ligadas:

```
if [ “$NOTE” = “ligado” E “$PCMAE” = “ligado” ];then
  Comando (wondershaper eth0 400 que é igual a → 50k )
fi
```

Para o conector **E**, basta lembrar:
se é tudo verdadeiro então executa.

Eu usei vários comandos para chegar no valor ligado antes de ser comparado e para outras velocidades, mas aqui vamos nos prender apenas ao sentido deste comando citado acima.

Operador lógico (OU)

Aqui a conjunção é verdadeira se uma ou outra condição for verdadeira (sendo as duas verdadeiras também é válido):

Nº	Condição 1	Conector	Condição 2	Resultado do teste	Explicação
1º	V	**ou**	V	Verdadeiro	Porque pelo menos uma condição é verdadeira
2º	F	**ou**	V	Verdadeiro	Temos uma condição verdadeira, e é suficiente
3º	V	**ou**	F	Verdadeiro	Temos uma condição verdadeira, e é suficiente
4º	F	**ou**	F	Falso	Nenhuma condição verdadeira para validarmos

Exemplificando:

Vamos usar o mesmo exemplo da conexão anterior, só que desta vez eu preciso que apenas uma condição seja verdadeira para que ele execute o comando, trocaremos o **e** pelo **ou**.

Só entra na festa casais que o homem se chama César **OU** a mulher Juliana:

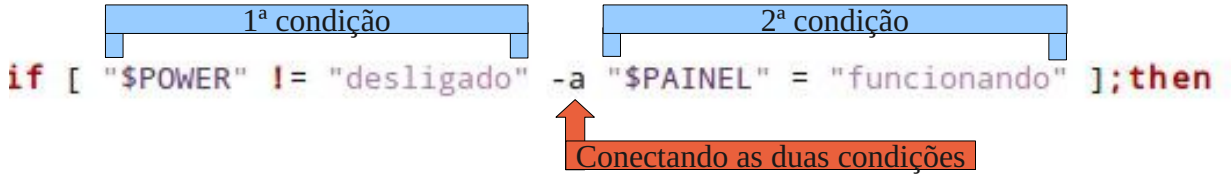
1º	César	**ou**	Juliana	Entrada permitida, as duas condições são verdadeiras
2º	Paulo	**ou**	Juliana	Entrada permitida, pelo menos uma condição verdadeira (juliana)
3º	César	**ou**	Mônica	Entrada permitida, a condição (César) valida a entrada
4º	Júlio	**ou**	Carolina	Nenhum dos dois nomes estão na lista, barrados !

Pegando o exemplo anterior do script que diminui a velocidade da internet, podemos pensar na seguinte situação: E se eu quisesse diminuir a velocidade da internet para 100k independentemente da quantidade de pcs ligados, ou seja, se o pc da minha irmã **OU** o pc da minha mãe estiverem ligados, ou se os dois estiverem simultaneamente, diminui para 100k e pronto.

```
if [ “$NOTE” = “ligado” OU “$PCMAE” = “ligado” ];then
  Comando (wondershaper eth0 800 que é igual a → 100k)
fi
```

Na linguagem do shell o operador “**e**” é representado como “**-a**”
e o operador “**ou**” é representado por “**-o**” (não confundir com -zero)

Como mostrado abaixo:



Do exemplo acima podemos entender: Se a variável POWER é diferente de desligado **E** a Variável PAINEL é igual a funcionando então faça o comando.

Shell Script do zero

Aula 6 – Usando os Conectores

Vamos abordar mais alguns exemplos dos conectores e partiremos para a prática a seguir.

Podemos usar quantos conectores quisermos:

```
if [ "$P1" = "of" -a "$P2" = "of" -a "$P3" = "of" ];then
```

Só todas as condições sendo verdadeiras é que ele executa

```
if [ "$P1" != "of" -o "$P2" != "of" -o "$P3" != "of" ];then
```

Se pelo menos uma for verdadeira ele executa o comando

Exemplo de um script completo, que usa o recurso dos conectores:

Note que os comandos if não estão usando aspas nas variáveis e nem nos valores, justamente por se tratarem de valores numéricos/matemáticos.

saudacao

```
#!/bin/bash

# Este Script dá boa tarde bom dia e boa noite de acordo com o horario

# Escreve as horas na variável DATA
DATA=$(date +%H)

# Agora que identificamos a hora, o programa executará os audios nos horarios corretos

# Se a variável DATA é igual ou maior que 13 e menor que 18 entao execute o arquivo boatarde.mp3
if [ $DATA -ge 13 -a $DATA -lt 18 ] ; then
   mpg123 /usr/share/boatarde.mp3 # Chama o programa para rodar o boatarde.mp3
fi

# se a hora é menor que 13 entao execute o arquivo bomdia.mp3
if [ $DATA -lt 13 ] ; then
   mpg123 /usr/share/bomdia.mp3
fi

# se a hora é maior que 18 entao execute o arquivo boanoite.mp3
if [ $DATA -ge 18 ] ; then
   mpg123 /usr/share/boanoite.mp3
fi
exit
```

Usando o conector

Neste script fui obrigado a usar o conector -a para definir o horário da tarde, já que “estar de tarde” significa ser mais que 13 horas **E** menos que 18 horas.