GUIA ESSENCIAL PARA CRIAÇÃO DE SCRIPT

LINUX ESSENCIAL

Lições preliminares em shell script mais exercícios

WILMAR BORGES LEAL JUNIOR
WILMARLEAL@GMAIL.COM

CIP-BRASIL. CATALOGAÇÃO-NA-FONTE

LEAL JUNIOR, WILMAR BORGES

Linux Essencial:: Guia essencial para criação de script / WILMAR BORGES LEAL JUNIOR. – 1. ed. – Porto Alegre : PLUS / Simplíssimo, 2019.

Recurso digital : il.

Formato: ePub2
Requisitos do sistema: Adobe Digital Editions
Modo de acesso: World Wide Web
ISBN 9788582456361

1. Linux. 2. Shell. 3. Script. 4. Bash. I. Título.

CDD: 0

Sumário

1 Apresentação 6
2 Lições preliminares 7
2.1 O shell 7
2.2 O Terminal 7
2.3 O prompt de comando 8
2.4 O Shell 9
2.5 Comandos no Shell 9
2.6 Consultando informações do comando 9
2.7 Traduzindo o man pages 10
2.8 Exercícios do Capitulo 11
3 Aprendendo Comandos Básicos 12
3.1 O comando – sudo / su 12
3.2 O comando – ls 13
3.3 O comando – mkdir 13
3.4 O comando – cd 14
3.5 O comando – touch 15
3.6 O comando – date 15
3.7 O comando – pwd 17
3.8 O comando – echo 17
3.9 O comando – cat 17
3.10 Os redirecionadores de entrada e saída `>` e `>>` 18
3.11 O comando – cp 18
3.12 O comando – mv 19
3.13 O comando – apt-get 19
3.14 O comando – chmod 20
3.15 O comando – shutdown 20
3.16 O comando – df 20
3.17 O comando – find 21
3.18 O comando – free 21
3.19 O comando – uptime 21
3.20 O comando – history 21
3.21 Caracteres especiais curinga 22
3.22 Uso das aspas 24
3.23 Exercícios do Capitulo 25

4 Arquivos e Diretórios no Linux 30
4.1 O Diretório Pessoal do usuário /home 32
4.2 Entendendo a listagem completa ls -l 32
4.3 Exercícios do Capitulo 35
5 Edição de arquivos 35
5.1 Exercícios do Capitulo 37
6 Empacotando e compactando arquivos 37
6.1 Empacotando os Arquivos 37
6.2 Compactando os Arquivos 39
6.3 Exercícios do Capitulo 40
7 Gerenciamento de Usuários no Linux 41
7.1 Exercícios do Capitulo 45
8 Variáveis de Ambiente 46
8.1 Exercícios do Capitulo 47
9 Expressão Regular 49
9.1 O comando – cut 50
9.2 O comando – grep/egrep 51
9.3 O comando – wc 52
9.4 O comando – sed 53
9.5 O comando – tail 54
9.6 O comando – expr 54
9.7 O comando – uniq 55
9.8 O comando – nl 55
9.9 O comando – more 55
9.10 O comando – less 55
9.11 O comando – find 56
9.12 Exercícios do Capitulo 57
10 Entendendo permissões de arquivos 60
10.1 Exercícios do Capitulo 63
11 Iniciando no script 63
11.1 Variável no script 66
11.2 O comando – curl 67
12 Operadores 71
12.1 Operadores de string 71
12.2 Operadores Numéricos 72
12.3 Operadores de Arquivos. 72

12.4 Operadores lógicos 72
13 Estrutura de Controle 73
13.1 Estruturas condicionais – if, else e elif 73
13.2 Estruturas condicionais – case 75
14 Estruturas de repetição 75
14.1 Estrutura de repetição – for 75
14.2 Estrutura de repetição – while 76
14.3 Exercícios Diversos 76
15 Referências 78

1 Apresentação

Sou professor da área de informática no Instituto Federal do Tocantins desde 2014, até o presente momento estou com uma disciplina no curso técnico em informática denominada sistemas operacionais e linguagem de script. O conteúdo dessa disciplina é voltado para resolução de problemas via script voltado para administração de sistemas. Hoje, boa parte dos dispositivos utilizam o sistema operacional Android que roda por baixo o kernell do linux, assim, fica fácil entender a demanda nas escolas técnicas por linux.

Esse livro é baseado em minha experiência na iniciativa privada e pública, aqui, reuni um pouco de conteúdo em Linux, voltado especialmente para a escola técnica, onde o aluno está no início de sua formação. A cada capitulo coloco alguns exercícios para serem feitos, dessa forma fica fácil compreender o uso do Linux capitulo a capitulo, de uma forma geral, esse livro não tem a pretensão de ser um curso completo e nem um básico, digamos que ele é essencial para quem precisa, de forma prática, utilizar o sistema com explicações teóricas e exercícios práticos, claro, que se você não for meu aluno, com esse guia de estudos você também, conseguira compreender com mais clareza o linux na sua essência. Não é um livro para se conhecer a história do linux, é um livro com pouco conteúdo teórico e muito conteúdo prático, mas sempre que precisar de algo teórico para que você consiga fazer os exercícios, vou busca-lo e cita-lo dando os devidos créditos, como é um material para curso, ele pode e deve ser divulgado nas escolas técnicas para uso e compartilhamento sobre a licença copyleft, Se você professor da área de informática se interessar pelo projeto entre em contato, vamos ampliar e divulgar nosso trabalho em parceria para nossos alunos. Espero que goste da leitura e dos exercícios, se tiver alguma dúvida, caro leitor, fique à vontade para entrar em contato por e-mail: wilmarleal@gmail.com e vamos em frente.

A parte que ignoramos é muito maior que tudo quanto sabemos. Platão

2 Lições preliminares

Em nosso livro trabalharemos com o terminal shell no Xubuntu, entretanto, você poderá utilizar qualquer distribuição derivada do Debian para o seu aprendizado. Não se preocupe, mesmo se você tiver pouca ou nenhuma experiência com o Linux, esse curso é o lugar de partida perfeito, abra sua mente para novas habilidades (skills), não se feche para essa nova experiência e vamos em frente.

2.1 O shell

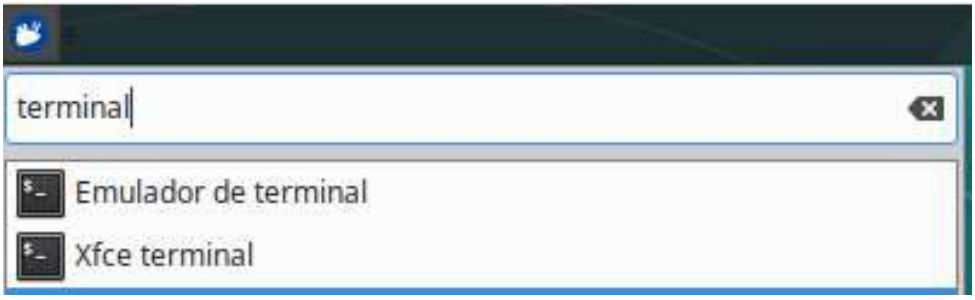
As ferramentas que permitem gerenciar o sistema possuem recursos de linguagens de computador, como loops, variáveis, estrutura de decisão dentre outros, e algumas linguagens de computador são amplamente utilizadas na automação de tarefas para a administração de sistemas. Apenas para contextualizar, você vai interagir com o sistema operacional por um shell, local ou remotamente. O trabalho do shell é aceitar comandos para manipular arquivos, iniciar ou parar processos, fazer instalações e configurações personalizadas, dentre outros, dessa forma, passando para o kernel do linux para sua execução.

O Shell do linux fornece uma linguagem rica para manipular arquivos e customização do ambiente, oferecendo ao administrador de sistemas uma possibilidade de automatizar processos que pode se tornar dispendioso e mecânico, como backup ou atualizações de sistema. Em um sistema operacional é inevitável ter que adicionar, remover e atualizar o software em determinado momento, como nosso curso é voltado para alunos em formação técnica, mas, nada impede ser utilizado por outros, mostraremos aqui como fazer, utilizaremos para tanto ferramentas como o `dpkg` ou `apt-get`, claro, existem outros para outras formas e em diferentes distribuições outras ferramentas, mas, basicamente esses dois conseguem suprir nossa necessidade no decorrer do curso.

2.2 O Terminal

A linha de comando é uma entrada de texto simples que permite inserir qualquer coisa, desde comandos, palavras até scripts (foco do nosso livro). Se você fizer login no modo texto, estará imediatamente no que chamamos de console. Se você fizer login graficamente,

será necessário iniciar um shell gráfico, para isso, utilize a tecla de atalho `CTRL + ALT + T` ou digite na busca `terminal`, que abrirá o console para começarmos nosso trabalho.



Se você é como a maioria das pessoas, provavelmente está mais familiarizado com o uso de uma interface gráfica, (Graphical User Interface, GUI), para controlar seu computador. Um sistema com interface gráfica fornece uma maneira fácil e intuitiva de gerenciar seu sistema.

A interface de linha de comando (Command Line Interface, CLI) é uma interface baseada em texto, onde o usuário digita um comando e o computador o executa. O ambiente da CLI é fornecido por um aplicativo no computador conhecido como terminal, que já vimos acima como abri-lo. O terminal aceita o que o usuário digita e passa para um shell. O shell interpreta o que o usuário digitou em instruções que podem ser executadas pelo sistema operacional. Se a saída for produzida pelo comando, este texto será exibido no terminal. Se problemas com o comando forem encontrados, uma mensagem de erro será exibida.

```
Toda reforma interior e toda mudança para melhor dependem exclusivamente da
aplicação do nosso próprio esforço. Immanuel Kant
```

2.3 O prompt de comando

Aqui é onde curso começa a ganhar vida, pois abriremos e conheceremos o ambiente a qual iremos trabalhar com um pouco mais detalhes. Essa parte merece algumas explicações introdutórias.

Quanto se abre o terminal pelo GUI aparecerá de modo semelhante à figura acima, nome do usuário @ nome do computador: seguido pelo `(~)` que indica que a sessão está aberta no diretório `/home` do usuário e por fim o símbolo `($)` que indica que esse usuário

é um usuário que não tem muitos privilégios no sistema, chamamos ele de usuário comum, agora seu terminal já está pronto para receber comandos.

2.4 O Shell

Um shell é o interpretador que traduz comandos inseridos por um usuário em ações a serem executadas pelo sistema operacional. O ambiente Linux oferece muitos tipos diferentes de shells. O shell mais usado para distribuições Linux é chamado de shell BASH. É um shell que fornece muitos recursos avançados, como o histórico de comandos por exemplo, `history` onde permite que você reexecute facilmente os comandos executados. No shell bash utilizaremos, `aliases`, variáveis e scripting, são muitos e diversos os recursos do bash não temos a intenção de esgotar nenhum comando e sim passar o essencial para que você possa começar a andar, o voo fica por sua conta.

2.5 Comandos no Shell

Existem vários comandos para o shell, e esses comandos podem ser executados sozinhos ou acompanhado de opções e argumentos, o formato básico de um comando é:

```
Comando [opção] [argumentos]
```

Vamos a um exemplo prático, voltemos ao terminal, digite o comando `ls,` esse comando é muito utilizado no terminal, seu análogo no windows é o comando `dir`, se já usou o `command` do ms-dos sabe do que estou falando. Ele sozinho mostra apenas o conteúdo do diretório atual, mas, com argumentos e a combinação deles você poderá operar milagres. Uma boa lista de comandos encontra-se em: https://www.devmedia.com.br/comandos-importantes-linux/23893, porém, veremos alguns, básicos para operar no shell.

2.6 Consultando informações do comando

Essa parte introdutória é importante para pegar algumas dicas uteis, para aqueles que não são familiarizados com o linux gastar um tempinho estudado a parte teórica adiantará e muito quando começarmos a desenvolver nossos scripts, portanto, não pule etapas, a não ser que já seja um usuário intermediário. Caso queira saber algo sobre determinado comando

basta olhar na vasta documentação oferecida pelo próprio terminal. As opções mais utilizadas no linux são `help, man` e `info` onde:

- `help` é um comando bash interno no shell (e somente no shell) que documenta alguns dos comandos e palavras-chave internos desse shell. Esse é um sistema interno de documentação desse shell
- `man` é um sistema de documentação para todo o sistema que fornece pequenos manuais de referência (páginas) para comandos individuais, funções API, conceitos, sintaxe de arquivo de configuração, formatos de arquivo organizados em seções (1 para comandos do usuário, 2 para chamadas do sistema ...). Esse é o sistema de documentação tradicional do Unix.
- `info` é outro sistema de documentação originado no projeto GNU. É hipertexto com links (antecede a web). Um manual de informações é como um livro digital com um conceito de índice e índice (pesquisável) que ajuda a localizar as informações. (StackExchange, 2011)

2.7 Traduzindo o man pages

Os manuais, originalmente vem escrito na língua inglesa, se ainda não tem familiaridade com o idioma o jeito é traduzi-lo, a LDP-BR (Linux Documentations Project – Brasil) está traduzindo essa documentação e já temos um grande número de comandos, pelo menos os mais usados já foram traduzidos, para isso, siga os passos abaixo para traduzir essa documentação para o português. Por hora não se preocupe com os comandos, irei explicar no seu devido momento, vamos deixar o sistema pronto para trabalharmos.

Se sua instalação é nova, ou seja, é a primeira vez que está utilizando, provavelmente ainda não fez uma atualização do sistema, se não, siga os passos a seguir.

```
sudo apt-get update
```

Quando finalizar o update, execute o comando abaixo

```
sudo apt-get upgrade
```

Os comandos acima deverão ser executados diretamente no terminal, nos comandos acima é mostrado como fazer a primeira atualização do sistema via `apt-get`, agora vamos passar o `man-pages` para português, execute os comandos na sequência.

```
sudo apt-get install manpages-pt
sudo apt-get install manpages-pt-dev
sudo apt-get update
```

Após finalizar, veja o se o comando `ls` encontra-se em português, como disse, nem toda documentação estará traduzida, mas já é um começo, caso precise ler sobre, pode acessar também a pagina https://pt.wikibooks.org/wiki/Guia_do_Linux/ e pesquisar o comando desejado, feito isso, já podemos começar de fato.

```
O homem não é nada além daquilo que a educação faz dele. Immanuel Kant
```

2.8 Exercícios do Capitulo

Observação, para facilitar, os exercícios deverão ser feitos separadamente em arquivos de texto com o número do exercício o seguido da extensão .txt, (ex1.txt). Caso o exercício peça para exportar para algum arquivo deverá seguir o padrão (log1.txt), assim, ex1 é para o exercício de número 1 e log1 a saída do exercício 1. Em todos os arquivos colocar na parte superior seu nome completo antecedido por um # cerquilha.

1. Exporte todos os comandos utilizados no shell para um arquivo de texto, siga as instruções caput. #questão deverá ser salvo como ex1.txt e seu log como log1.txt

2. Instale o programa tree, via terminal

3. Instale o programa vim, via terminal

4. Como abrir 3 terminais linux pela interface gráfica, utilizando teclas de atalho

5. Como abrir o terminal linux sem a GUI, qual o atalho no teclado para isso? TeleTypewriter tty

6. Descreva, como logar no terminal como root e como sair.

7. O shell é interpretador de comandos, como fazer para visualizar todos os comandos já digitados no shell

8. Como apagar todos os comandos que foram digitados no shell

9. Como executar novamente um comando, sem precisar digita-lo novamente

10. Exporte a página de manual de qualquer comando para um arquivo de texto

```
Não é livre quem não obteve domínio sobre si. Pitágoras
```

3 Aprendendo Comandos Básicos

Temos uma pequena lista de comandos do linux que devemos conhecer antes de começar a nos exercitar, não vou esgotar todos os argumentos do comando, já conhece o `man-pages, o info e o help`, quando estiver com dúvida acesse e de uma lida nas opções que o comando oferece.

3.1 O comando – sudo / su

Antes de começarmos vamos entender o que é e como funciona os comandos `su` e `sudo`. O comando `su` é o acrônimo para Super User, ou seja, você loga, entra no sistema, com o usuário root, que é padrão do linux. Já o comando `sudo` você executa como root e não sendo o root, é o acrônimo para Super User DO, ou, fazer como administrador. Todas as vezes que houver um `$` no terminal, significa que você está logado como usuário comum. Quando tiver um `#`, você está logado como root.

```
su:   Logar como Super User
sudo: Super User; DO = Fazer, executar
```

3.2 O comando – ls

Cada comando tem sua sintaxe própria, que pode ser acessada diretamente em seu `man`, o `ls`, esse simples comando lista o conteúdo do diretório atual. Todo o comando geralmente é precedida por letras ou números separado por um ou dois hifens, adicionados após o comando principal, vamos lá, vá em seu terminal e digite apenas o comando `ls`, repare que ele só retornou os diretórios e arquivos que estão no seu no seu diretório atual, veja a imagem abaixo.

```
wilmarborges@Linux:~$ ls
'Área de Trabalho'   Imagens   Público            wilmar_borges.jpg
 Documentos          Modelos   Vídeos             wilmar_borges.txt
 Downloads           Música    wilmar_borges.avi
```

O `(~)` como já sabe é o diretório padrão do usuário no linux, ou o home do usuário, é onde fica por padrão sua área de trabalho particular, seu diretório documentos, suas imagens, dentre outros. Vale ressaltar que o linux é case-sensitive, o sistema faz diferenciação entre maiúsculas e minúsculas ou seja, o sistema não reconhece o comando LS ou Ls, o mesmo só reconhece como é de fato, por isso muita atenção ao digitar comandos no terminal.

Bem, veja que o comando puro não me traz muitas informações, agora, faça um teste em seu terminal, digite `ls -l.` veja que o argumento `-l` modifica o comportamento do `ls` padrão listando os arquivo em outro formato, mais detalhado, agora digite `ls -la`, o acréscimo do `-a` ao comando principal faz com que o `ls` mostre os arquivos ocultos, aqui vale mais uma explicação, no linux arquivos que começam com `(.)` ponto, são arquivos ocultos do sistema e porque é necessário ocultar arquivos?, hora, arquivos de configuração são ocultados justamente para que usuários inexperientes não mexam, danificando o sistema, quer saber mais sobre o comando `ls`, acesse comandos ls de A a Z, (Silva, 2010).

3.3 O comando – mkdir

O comando `mkdir` é utilizado para manipular diretórios no sistema, podendo também ser combinado com outros argumentos. Vejamos, em seu diretório atual no `(~)`, digite, `mkdir fruta`, você estará criando um diretório chamado fruta no `/home` do seu usuário, simples assim, agora liste o conteúdo do diretório `/home,` com o comado `ls`, que o verá.

Para acessa-lo, ou melhor entrar nele digite `cd fruta`, será explicado mais sobre o comando `cd` logo abaixo. Veja que agora você está no diretório, deverá aparecer assim para você:

```
wilmarborges@Linux:~/fruta$
```

Note que temos o `~/frutas$` o que significa isso? simples, é que você está no diretório `/home` e dentro do diretório criado, frutas. Agora, vamos criar dentro desse mesmo diretório laranja. Depois acesse-o, crie o diretório doce, acesse-o.

```
wilmarborges@Linux:~/fruta/laranja/doce$
```

Note que os diretórios são separados pela `/` isso quer dizer que um diretório está dentro do outro. Agora, uma opção muito útil quando se usa o `mkdir` é o parâmetro `-p` que cria qualquer diretório pai faltante para cada diretório filho, ignora argumentos que correspondem a diretórios existentes. (Assim, se um diretório `/a` existe, então `mkdir /a` vai provocar um erro, mas `mkdir -p /a` não, assim é criado todos os diretórios de uma só vez, vamos testar, faremos o mesmo grupo acima, mas, agora de uma só vez e será um conjunto de verduras, veja como ficou:

```
wilmarborges@Linux:~$ mkdir -p arvores/goiabeira/doce
wilmarborges@Linux:~$ tree arvores/
arvores/
└── goiabeira
    └── doce
```

Tente fazer o mesmo sem o `-p` e veja se dará certo. Leia a mensagem que o shell retorna e tente entender o que aconteceu.

3.4 O comando – cd

Vamos aproveitar os diretórios acima criados para entender rapidamente o comando `cd`, no linux, que também é o mesmo no windows. O comando `cd` é acrônimo para change directory ou mudar de diretório e seu uso é simples, basta digitar `cd` e o nome/caminho do diretório que você quer acessar e você terá acesso a ele, vamos olhar o exemplo dos diretórios criados acima, se você quiser voltar um diretório, digita `cd ..` `[cd <espaço>..]` e assim poderá voltar até o diretório raiz o `/` (barra), vamos as praticidades, em qualquer

lugar que você estiver, se digitar `cd ~`, poderá ir diretamente par ao seu `/home/`. Há várias outras formas, mas, esse é a forma mais básica de se utilizar o comando `cd.`

3.5 O comando – touch

Comando simples para criação de arquivos pelo terminal. Em seu modo mais simples utiliza-se da seguinte forma:

```
touch /caminho/nome_do_arquivo
```

Observação, se estiver dentro do diretório que vai criar o arquivo o caminho é desnecessário, visto que já está no local, exemplo, suponha que está no home do usuário, não é necessário digitar `touch /home/wilmarborges/`arquivo para criar, basta digitar `touch arquivo`, pois já está dentro do diretório onde criará o arquivo.

Agora, se quiser criar vários arquivos de uma vez é só colocar um em frente ao outro, assim: touch pera uva maçã, se quiser criar vários arquivos em sequência utilize { } exemplos:

Vou criar arquivos de 1 a 10, apenas sequência numérica

```
touch {1..10}
```

Agora vamos criar de 1 a 10 com uma extensão .txt no final

```
touch {1..10}.txt
```

Pra finalizar. vamos criar nomes que se repetem com o final de 1 a 10

```
touch wilmar{1..10}.txt
```

Esse é o básico do touch, para saber mais, acesse o `man` do `touch`, ou o site seja livre, (Ludolf, 2013), link nas referências.

3.6 O comando – date

O comando date mostra a data e hora do sistema, isso é fato, entretanto, o que mais ele pode nos oferecer? Vamos ver.

```
date
```

O comando `date` puro apenas mostra o ano, mês, dia, hora minuto segundo, porém esse comando, combinado com outros pode nos oferecer uma excelente automação na criação de diretórios que necessitam ser criados diariamente, veja abaixo a lista de parâmetros para o comando `date`. Vamos ver uns exemplos:

Parâmetros que podem ser utilizados para formatação de datas:	
%A	dia da semana (domingo,..., sábado).
%B	nome do mês (janeiro,..., dezembro).
%H	hora do dia (0 a 23).
%M	minuto (0 a 59).
%S	segundos (0 a 61).
%T	hora no formato hh:mm:ss.
%Y	ano.
%a	dia da semana abreviado (dom,..., sab).
%b	nome do mês abreviado (jan,..., dez).
%c	dia da semana, data e hora.
%d	dia do mês (00-31).
%j	dia ano (1 a 366).
%m	mês (1 a 12).
%s	número de segundos desde das zero horas de 01/01/1970.
%w	dia da semana, onde 0 = domingo, 1 = segunda,..., 6 = sábado.
%x	representação da data local.
%y	os dois últimos dígitos do ano.
%r	formato de 12 horas completo (hh:mm:ss AM/PM)

Vamos a alguns exemplos práticos utilizando alguns parâmetros:

```
date +%d #mostra apenas o dia
date +%m #mostra apenas o mês
date +%y #mostra apenas o ano com 2 dígitos,
```

Para mostrar dia/mês/ano podemos concatenar todos eles, assim:

```
date +%d%m%y
```

Ou, caso queira mostrar apenas o dia da semana, o comando seria assim:

```
date +%A
```

É relativamente simples a utilização do `date` não é mesmo, consegue pensar na utilidade dele?

`Ajuda teus semelhantes a levantar a carga, mas não a carregues. Pitágoras`

3.7 O comando – pwd

O comando `pwd` é o acrônimo para `print working direcotry`, imprimir diretório de trabalho, serve para que você saiba está, ele mostra o caminho inteiro do diretório que estamos. Faça o teste digite `pwd` e veja o caminho completo.

```
wilmarborges@Linux:~/fruta$ pwd
/home/wilmarborges/fruta
```

Veja que ao digitar `pwd` no diretório fruta criado, ele mostra todo os diretórios anteriores até chegar no diretório raiz `/`, note também que o `~` esconde o `/home/wilmarborges` ou seja é um alias para o home do usuário.

3.8 O comando – echo

Este comando mostra os argumentos na saída padrão seguido por uma nova linha, ou seja, mostra na tela (saída padrão).

```
echo “meu é: wilmar borges leal junior”
```

Agora, coloque seu nome sem o echo no terminal e veja a mensagem que ele retorna

```
-n : não acrescenta uma nova linha
```

3.9 O comando – cat

O comando `cat`, do sistema operacional Unix é usado para unir, criar e exibir arquivos. O nome remete a "con`cat`enate" ("concatenar" em inglês). Em geral é usado desta maneira para exibir o conteúdo de um arquivo na saída padrão (Wikipédia, 2017). Muito utilizado para ver o conteúdo do arquivo sem precisar abri-lo. Vamos a um exemplo.

```
echo “esse conteúdo vai para dentro do arquivo” > arquivo.txt
```

Explicando o comando acima: ele está pegando o texto do comando `echo` e enviando e criando `( > )` para dentro do arquivo.txt, só isso. Agora vamos ver o que tem dentro, sem abri-lo.

```
wilmarborges@Linux:~$ cat arquivo.txt
esse conteúdo vai para dentro do arquivo
```

Viu, eu não preciso abrir o arquivo para visualizar seu conteúdo, para isso, utilizo o `cat`, só lembrando que ele tem várias outras utilidades.

3.10 Os redirecionadores de entrada e saída `>` e `>>`

Os redirecionadores de E/S definem para onde será a entrada/saída de um comando. A tabela abaixo mostra alguns exemplos de operadores de E/S. (Uniriotec, 2016).

Operador	Descrição
>	redireciona para saída; se tiver algum conteúdo ele substitui
>>	redireciona para fim de arquivo; não exclui o conteúdo do arquivo
<	redireciona para a entrada; se tiver algum conteúdo ele substitui
<<	redireciona para a entrada e mantém a entrada aberta até que seja digitado algum caractere de EOF (fim de arquivo) como, por exemplo, CTRL+D
|	redireciona a saída de um comando para a entrada de um outro comando, o comando se chama pipe | e tem uma enorme utilidade.

Exemplo prático, escreva os comandos abaixo em seu terminal e veja o que acontece, execute-os 3 vezes cada e veja o seu conteúdo com o `cat`

```
echo “esse conteúdo vai para dentro do arquivo” > arquivo1.txt
echo “esse conteúdo vai para dentro do arquivo” >> arquivo2.txt
```

3.11 O comando – cp

Comando análogo ao do windows, que faz apenas copias de arquivos, combinado com as opções do comando é uma ferramenta poderosa para realização de backup. Supomos que já foi criado o arquivo wilmar.txt com o comando `touch`, ou de outra forma. Aqui estou criando uma cópia do arquivo, com o mesmo nome e no /`tmp`.

```
cp ~/wilmar.txt /tmp/
```

Esse comando permite utilizar caracteres curingas para facilitar, lembra dos arquivos com final numérico é um bom exemplo, imagine que você queira copiar apenas os arquivos wilmar1.txt, wilmar2.txt e wilmar3.txt, simples, utilizando metacaracter, você pode especificar o range para realizar a copia.

```
cp wilmar[1-3].txt /tmp/
```

Supondo que os arquivos já existam, se não pode cria-los com o `touch wilmar{1..10}.txt`, o comando cp está copiando apenas o range wilmar de 1 a 3 para o diretório `/tmp/` do linux. Existe várias outras maneiras, vamos descobrindo com o passar das páginas.

```
A coisa mais indispensável a um homem é reconhecer o uso que deve fazer do seu próprio conhecimento. Platão.
- Sei que a leitura está agradável, mas pare um pouco relaxe, quando estiver pronto, volte a leitura.
```

3.12 O comando – mv

Esse comando move ou renomeia arquivos, ele possui opções de comando fazendo-o um utilitário de mão cheia. Se quiser apenas renomear o comando é simples. Aqui estou apenas renomeando o arquivo, visto que estou no mesmo local,

```
mv wilmar.txt borges.txt
```

Agora se quiser mover para outro local basta colocar o diretório de destino, como no `cp`, pode ser utilizado com caractere metacaracteres.

```
mv wilmar.txt /tmp/
```

3.13 O comando – apt-get

Permite a instalação, atualização e remoção de pacotes do sistema, no início do livro utilizamos ele para fazer algumas instalações, ao longo do livro faremos outras instalações, serve basicamente para instalar remover ou atualizar pacotes, para instalar basta digitar: `sudo apt-get install <o_pacote>`, exemplo prático, suponhamos que você queira baixar o utilitário `tree`, para ver seus diretórios e arquivos no terminal em forma de arvore.

```
sudo apt-get install tree
```

As opções abaixo podem ser utilizadas com o `apt`

```
-h - ajuda
-d - baixar arquivos apenas, não instalar
-f - conserta erros de instalações de pacotes
-s - não agir, apenas simular operação
-y - assume `sim' para todas as perguntas
-u - mostrar pacotes que serão atualizados também
```

3.14 O comando – chmod

Esse comando altera as permissões de arquivos, haverá uma sessão especial para ele, até porque é por meio desse comando é que conseguiremos executar nossos scripts, será um capitulo bem explicado, por hora, saiba que altera as permissões.

3.15 O comando – shutdown

O comando serve para desligar ou reiniciar o computador pelo terminal, `shutdown [opção] [mensagem]`, a mensagem não é necessária, mas, caso queria alertar ao usuário que seu computador será desligado ou reiniciado é só colocar entre aspas. Abaixo algumas opções uteis para o comando.

```
-c - cancela um shutdown sendo executado.
-h - desliga o sistema.
-k - envia mensagens de alerta aos usuários logados
+m, hh:mm - executa o comando após m minutos ou na hora hh:mm.
now - executa o comando imediatamente.
-r - reinicializa o sistema após a parada.
```

Exemplo de como utilizar as opções no comando, caso queira reiniciar o computador imediatamente

```
sudo shutdown -r now
```

Para enviar uma mensagem para os usuários e desligar o sistema em 10 minutos, dando tempo para o usuário salvar os arquivos abertos, basta digitar:

```
sudo shutdown -h +10 "O sistema será encerrado em 10 minutos"
```

Temos também o comando `reboot` que reinicializa o sistema, e o `init` que pode ser usado no lugar do shutdown, `init 0` desliga, `init 6` reinicia, para saber mais acesse o site da bóson treinamentos, (Reis, 2013), link na referência.

3.16 O comando – df

Basicamente o comando `df`, exibe informações sobre espaço livre e espaço usado nas partições do sistema. Estou utilizando um disco virtual dinamicamente alocado, veja como

está sendo utilizado pelo sistema linux a opção `-h` serve para ver de forma mais legível o tamanho das partições.

```
wilmarborges@Linux:/tmp$ df -h
Sist. Arq.      Tam. Usado Disp. Uso% Montado em
udev            1,4G     0  1,4G   0% /dev
tmpfs           290M 1012K  289M   1% /run
/dev/sda1       9,8G  4,8G  4,5G  52% /
tmpfs           1,5G     0  1,5G   0% /dev/shm
tmpfs           5,0M  4,0K  5,0M   1% /run/lock
tmpfs           1,5G     0  1,5G   0% /sys/fs/cgroup
tmpfs           290M   16K  290M   1% /run/user/1000
/dev/sr0         56M   56M     0 100% /media/wilmarborges/VBox_GAs_5.2.20
```

3.17 O comando – find

Iremos estudar mais sobre ele, por hora é um utilitário que serve para pesquisar arquivos em uma hierarquia de diretórios, existe várias opções para ele, que veremos ao longo do livro, entretanto, se não estiver se aguentado de curiosidade, veja aqui várias opções de uso (Oliveira, 2015) está bem detalhado, link nas referências.

3.18 O comando – free

Usado para monitorar o uso da memória do sistema, utiliza como base as informações contidas no arquivo `/proc/meminfo/` útil para administradores de sistema.

```
free
```

3.19 O comando – uptime

Comando que verifica o tempo em que o computador está ativo, sua sintaxe é simples `uptime [opções]` `uptime -p` mostra de uma forma mais legível. Comando simples e de fácil compreensão.

```
uptime
```

3.20 O comando – history

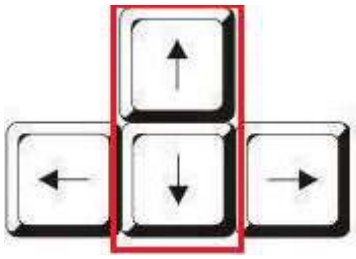
Quando você executa um comando em um terminal, o comando é armazenado em uma "lista de histórico". Isso foi projetado para facilitar a execução do mesmo comando mais

tarde, pois você não precisará digitar novamente o comando inteiro. Para ver a lista de comandos digitados no terminal basta digitar `history`.

Para saber mais, (Haeder et al., 2012, p. 72). Abaixo, alguns argumentos adicionais para o comando:

Exemplo	Significado
history 5	Mostrar os últimos cinco comandos da lista do histórico
!!	Execute o último comando novamente
!-5	Execute o quinto comando na parte inferior da lista do histórico
!ls	Execute o ls comando mais recente

Se quiser navegar nos comandos já digitados é só utilizar as teclas para navegar.



3.21 Caracteres especiais curinga

Os caracteres glob são geralmente chamados de "curingas", estes são símbolos que possuem um significado especial para o shell. Ao contrário dos comandos que o shell executa, ou das opções e argumentos que o shell passa para os comandos, os caracteres curinga são interpretados pelo próprio shell antes de tentar executar qualquer comando. Isso significa que os caracteres curinga podem ser usados com qualquer comando.

Assim, os curingas são muito uteis pois permitem especificar padrões que correspondem a nomes de arquivos em diretórios, de modo que, ao invés de manipular um único arquivo por vez, você poderá facilmente executar comandos que afetarão diversos arquivos, já vimos um pouco do caractere curinga quando falamos sobre o comando `cp`, vamos ver mais exemplos, essa é uma parte muito importante para resolução de diversos problemas cotidianos, sugiro que dê uma olhada no site do Aurelio Jargas que possui um

excelente material, onde também, sempre que preciso me recorro a ele aurelio.net, link nas referências, (Jargas, 2013).

O Asterisco `(*)` é usado para representar zero ou uma cadeia de caracteres. Abaixo um exemplo simples com o uso do asterisco, vamos exibir todos os arquivos que começam com a letra h dentro do diretório `/etc.`

```
Ls -l /etc/h*
```

Assim conseguimos listar todos os arquivos que começam com h, pois o * substitui todos os outros caracteres possíveis depois do `h`.

```
wilmarborges@Linux:/etc$ ls -l /etc/h*
-rw-r--r-- 1 root root 6748 jan 21  2018 /etc/hddtemp.db
-rw-r--r-- 1 root root 4861 fev 22  2018 /etc/hdparm.conf
-rw-r--r-- 1 root root   92 abr  9  2018 /etc/host.conf
-rw-r--r-- 1 root root    6 jan 24 15:26 /etc/hostname
-rw-r--r-- 1 root root  220 jan 24 15:26 /etc/hosts
-rw-r--r-- 1 root root  411 jul 25  2018 /etc/hosts.allow
-rw-r--r-- 1 root root  711 jul 25  2018 /etc/hosts.deny
```

O Ponto de interrogação `(?)` representa qualquer caractere. Cada caractere de ponto de interrogação corresponde exatamente a um caractere, vamos utilizar o mesmo diretório, `/etc` como exemplo prático. Suponha que você queira buscar tanto `Hosts*` quanto `hosts*,` maiúsculo e/ou minúsculo, dentro do arquivo `/etc.` Usando o caractere poderia listar da seguinte forma.

```
ls -l ?ost*
```

Assim, sabendo que o linux é case sensitive, dessa forma eu consigo listar qualquer palavra que seja maiúscula e minúscula no diretório pesquisado. O uso dos Colchetes `[],` já falado anteriormente, são usados para corresponder a um único caractere, representando um intervalo de caracteres que são possíveis características de correspondência, exemplo, vou usar o mesmo do anterior, imagine que quero pesquisar palavras dentro do `/etc` que se iniciam com H ou h, somados ao ost e com qualquer outro caracter posterior, ficaria assim:

```
ls -l [hH]ost*
```

E assim temos dezenas de operadores, para não estender muito, veja Operadores no site do aurelio.net, lá você vai encontrar operadores aritméticos, de atribuição, logico dentre outros, explore bem esse site.

3.22 Uso das aspas

Existem três tipos de aspas que têm significado especial para o shell Bash: aspas duplas ", aspas simples ' e apóstrofo invertido ` (no teclado é a crase). Cada conjunto de aspas indica ao shell que ele deve tratar o texto entre aspas de maneira diferente do que normalmente seria tratado.

As aspas duplas "" impedirão que o shell interprete alguns metacaracteres, incluindo caracteres curinga. Dentro de aspas duplas, um asterisco * é apenas um asterisco, um ponto de interrogação é apenas um ponto de interrogação e assim por diante.

As aspas simples '' impedem que o shell faça interpretações de caracteres especiais. Isso inclui os caracteres curingas, variáveis, substituição de comandos e outros metacaracteres que ainda serão discutidos.

O apóstrofo invertido ` (crase), são usados para especificar comando dentro de comando, exemplo para ficar claro. Digite o comando `date` em seu terminal como mostrado abaixo.

```
wilmarborges@Linux:~$ date
sex jan 25 12:49:15 -03 2019
wilmarborges@Linux:~$ echo "hoje é..:date"
hoje é..:date
wilmarborges@Linux:~$ echo hoje é date
hoje é date
wilmarborges@Linux:~$ echo hoje é..: `date`
hoje é..: sex jan 25 12:50:01 -03 2019
wilmarborges@Linux:~$
```

Quando colocamos entre aspas duplas o `date` é apenas um nome, `date`, mas, quando coloco entre `` `date` `` apostrofo invertido, o shell consegue retornar o comando dado. Daqui a pouco faremos alguns exercícios de fixação para que você possa compreender melhor, em frente.

Uso do `ponto-e-vírgula`, pode ser usado para executar vários comandos, um após o outro. Cada comando é executado de forma independente e consecutiva; não importa o resultado do primeiro comando, o segundo será executado assim que o primeiro for concluído, o terceiro e assim por diante. Teste você mesmo e veja o resultado.

Double `E comercial (&&).` O duplo comercial `&&` funciona como um "e" lógico, se o primeiro comando for bem-sucedido, em seguida, o segundo comando (à direita do `&&)` também será executado. Se o primeiro comando falhar, o segundo comando não será executado`,` diferente do `; (ponto e vírgula)`

Antes te terminar o parágrafo anterior eu sei que você deve ter se perguntado, se tem o "e" lógico, também deve ter o "ou" lógico, certamente que sim, é só utilizar o pipe duplo ||, ele funcionará de maneira semelhante ao &&; assim, dependendo do resultado do primeiro comando, o segundo comando será executado ou será ignorado, olha que legal; com o pipe duplo, se o primeiro comando for executado com êxito, o segundo comando será ignorado; se o primeiro comando falhar, o segundo comando será executado. Em outras palavras, você está essencialmente dizendo ao shell: "Execute este primeiro comando ou o segundo". Legal né! Imagina quantas possibilidades de questões bem legais podemos fazer só até aqui, vamos fazer alguns?

```
O livro é um mestre que fala, mas que não responde. Platão
```

3.23 Exercícios do Capitulo

Sem sombra de dúvidas esse será o capitulo com mais exercícios, visto que, os comandos básicos estão todos nesse capitulo, não se esqueça de seguir o padrão.

11. Liste todos os arquivos e diretórios existentes no home do usuário logado

12. Você está no GUI e deseja acessar o primeiro TTY do sistema, quais conjunto de teclas devem ser utilizados? (Essa você já fez, é só para não esquecer)

13. Liste todos os subdiretórios do ~ recursivamente

14. Liste os subdiretórios do ~ recursivamente e envie para um arquivo, siga o padrão

15. Liste os subdiretórios recursivamente do diretório /etc, envie para um arquivo com as linhas numeradas:

16. Abra o terminal e crie um diretório no /tmp com seu nome

17. Após criado um diretório com seu nome acesse-o e crie 1000 arquivos de texto em um só comando, com o nome linux de 0 a 1000

18. Após realizar a tarefa anterior volte para o seu home, crie um novo diretório chamado copiar e copie apenas os arquivos de 200 a 299 para esse arquivo

19. Crie um diretório no seu /home com seu primeiro nome, dentro dele crie um arquivo de texto, que deverá ter seu sobrenome, faça tudo em uma só linha de comando, dica: echo + > + mkdir.

20. Crie um novo diretório, agora com o nome InfoApp em /home, dentro, crie um arquivo chamado aprendendo_linux.txt em uma só linha de comando

21. Agora, pegue o aprendendo_linux.txt e mova para o diretório para o /tmp

22. Crie um diretório chamado diversos em seu home, depois, crie 20 arquivos com o nome foto.jpg, músicas.mp3, arquivos.txt e sistema.py

23. Agora liste tudo que existe no diretório ~/diversos e envie um relatório para um arquivo da forma padrão, log<numero_questão> em /tmp/ (padrão).

24. Mova apenas os arquivos com extensão .jpg para /tmp

25. Mova apenas os arquivos de música de 1 a 12 para o /tmp

26. Mova todos os arquivos com extensão .txt para /tmp

27. Utilizando o comando date envie para um arquivo a data e hora do sistema, quando o comando for rodado outra vez ele substituirá o conteúdo do anterior

28. Utilizando o comando date envie para um arquivo a data e hora do sistema, quando o comando for rodado outra vez ele NÃO substituirá o conteúdo do anterior

29. Escreva um comando em que crie um diretório com o dia da semana correspondente, sendo o dia da semana variável, mudando-se o calendário o comando criará com um novo dia da semana, utilize o date para isso.

30. Ainda, utilizando o date e seus argumentos, crie um diretório ano, e dentro o mês correspondente e dentro o dia, em seu diretório /home.

31. Crie um arquivo em que, o nome do arquivo seja o ano, mês, dia, hora, min, segundo.txt e toda vez que rodar esse comando o mesmo irá gerar um novo arquivo, o nome do seu arquivo ficará assim: 20190310221232.txt

32. Faça o mesmo da questão anterior, porém ao invés de ciar um arquivo.txt irá colocar dentro de um log apenas a numeração gerada e todas as vezes que rodar o comando ele irá colocar um novo número abaixo, útil quando se quer gerar número de protocolo irrepetível.

33. Qual o comando para visualizar esse log gerado na questão anterior sem precisar abri-lo?

34. Está gostando? Na questão 32 no arquivo gerado, coloque uma coluna com o número da linha. Pode gerar outro arquivo com o número de linha mais a numeração gerada, vamos chamar de número do protocolo.

35. Comando para ver os últimos 5 comandos digitados no terminal

36. Em um arquivo no `/tmp`, envie seu nome completo, a data completa, e em baixo os últimos 20 comandos digitados no seu terminal, um abaixo do outro e gere um log padrão no caso `log36.txt`, como já vimos.

37. Mova o arquivo criado da questão anterior para o `/home` do usuário e renomeie para o novo número da questão `log37.txt`

38. Escreva o comando para que o computador desligue em 10 minutos e aparece uma mensagem na tela dizendo que o computador será desligado em 10 minutos

39. Qual o comando para cancelar o desligamento do sistema?

40. Qual o comando para desligar imediatamente?

41. Qual o comando para reiniciar?

42. Crie um arquivo em seu ~, dê permissão apenas de leitura pra todos (faça em octal e literal)

43. No mesmo arquivo dê apenas permissão de escrita (faça em octal e literal)

44. No mesmo arquivo dê, leitura e escrita para o dono, leitura para o grupo e nada para os outros (faça em octal e literal)

45. De uma só vez, deixe a permissão apenas de execução do arquivo (faça em octal e literal)

46. Deixe agora a apenas permissão de execução para o dono e para outros, mas, pra o grupo nada. (faça em octal e literal)

47. Crie um diretório chamado teste em /home, agora crie 10000 arquivos lá dentro, verifique qual o tamanho do diretório /home/teste

48. Coloque um pendrive em seu computador com linux, agora, qual o caminho do pendrive? Onde ele é montado, qual o seu diretório padrão de montagem?

49. Agora que já sabe que o diretório padrão de montagem de um pendrive é no /media, de um comando que mostre o tamanho do seu pendrive

50. Liste todo o conteúdo do pendrive para um arquivo, não se esqueça do padrão log49.txt

51. Converta a permissão -r–r–r– para o formato octal.

52. Converta a permissão -rwx—r-x para a o formato octal.

53. Existe várias maneiras de se ver qual o tamanho da memória livre, qual a forma mais simples, pelo terminal

54. Procure todos os arquivos criados com a extensão .txt, liste e envie o relatório par um log54.txt

55. Procure e delete todos os arquivos com a extensão .mp3 criadas no diretório `/home` do usuário logado, apenas do usuário logado

56. No diretório `/tmp` crie 100 arquivos musica.mp3, foto.jpg e programas.sh, agora, aplique de uma só vez aos arquivos programas.sh a permissão de execução para todos, transforme-os todos em um arquivo executável de uma só vez. Apenas os com extensão .sh

57. Copie o conteúdo do `passwd` que está em `/etc` para `/tmp` com o mesmo nome,

58. Copie o `passwd` para /`tmp` com um novo nome

`É fazendo que se aprende a fazer aquilo que se deve aprender a fazer. Aristóteles`

4 Arquivos e Diretórios no Linux

Nesse capitulo vamos aprender um pouco mais sobre diretórios no linux, não muito profundo, o foco aqui é desenvolver script e não entender a fundo o sistema operacional. para saber mais leia o livro do (Boanan, 2010), rico em detalhes.

Os arquivos são usados para armazenar dados, como texto, gráficos e programas. Diretórios são usados para fornecer uma estrutura organizacional hierárquica e também para organizar os arquivos. No linux, essa estrutura é um pouco diferente do que você pode estar acostumado, se já trabalhou anteriormente em sistemas Microsoft Windows lembra como é a hierarquia. Em um sistema Windows, o nível superior da estrutura de diretório é chamado `Este Computador`. Cada dispositivo físico (disco rígido, drive de DVD, pen drive USB) aparece em `Este Computador`, a cada um é atribuída uma letra de unidade, como `C: D: E:`. Veja de forma visual a estrutura no Windows 10.

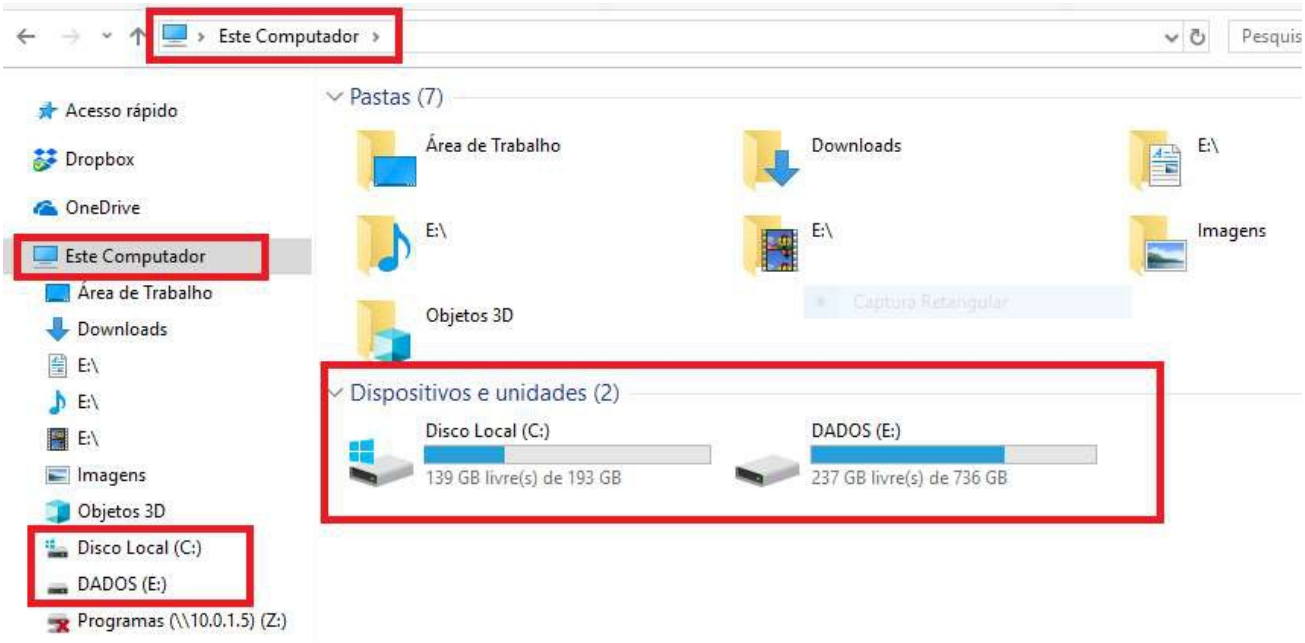


Figure 1: Windows 10

No Linux, o diretório raiz e é simbolizado pelo caractere `/` (barra). Também não há drives no Linux; cada dispositivo físico está acessível em um diretório, não em uma letra de unidade. Vejamos uma representação visual de uma estrutura de diretório típica do Linux:

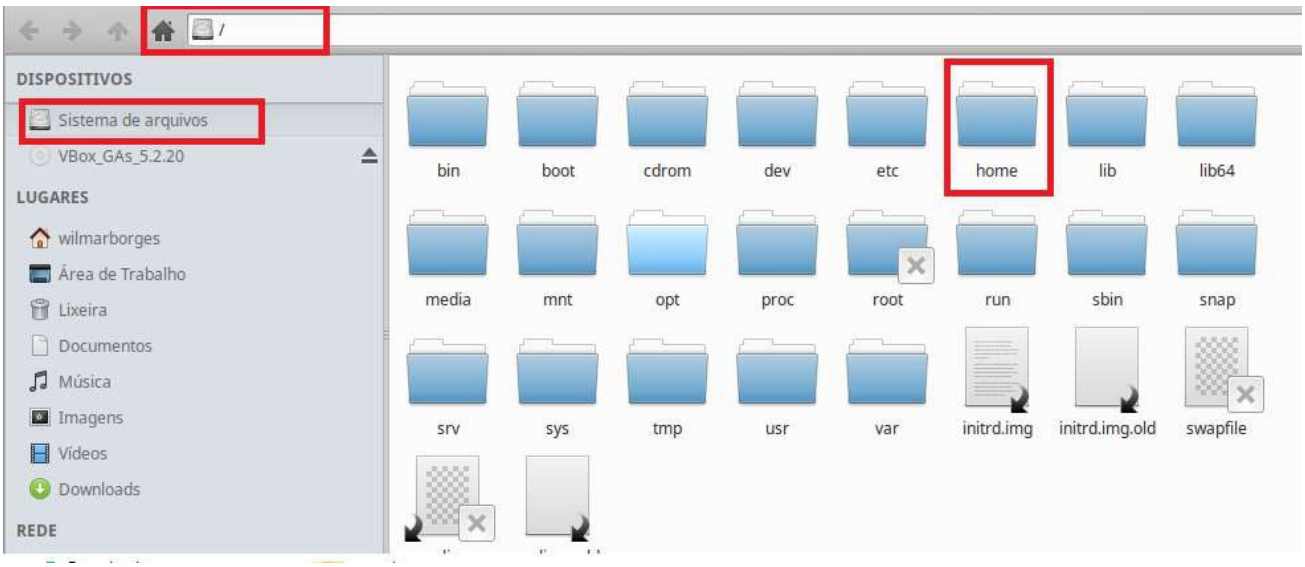


Figure 2: Xubuntu 18

Veja que todos os diretórios que estão abaixo da raiz `/` aparecem dentro do sistema de arquivos, a exemplo o diretório `/home` do usuário está dentro do `/` "raiz". No linux é possível visualizar tudo que está dentro `/` utilizando o comando `ls /`. Agora, liste todos os diretórios que existem dentro do `/boot`. Como disse, não vamos explicar sobre cada

diretório, estamos aqui para ser o mais prático possível para que possamos entender o sistema linux de uma forma rápida e simples.

4.1 O Diretório Pessoal do usuário /home

O diretório pessoal geralmente causa confusão aos iniciantes no sistema linux. Para começar, na maioria das distribuições linux há um diretório chamado `home` sob o diretório raiz, sempre haverá um diretório para cada usuário no sistema, o nome desse diretório geralmente é nomeado com o nome do usuário criado no meu caso, meu usuário chama `wilmarborges`, e meu `/home` fica em `/home/wilmarborges`.

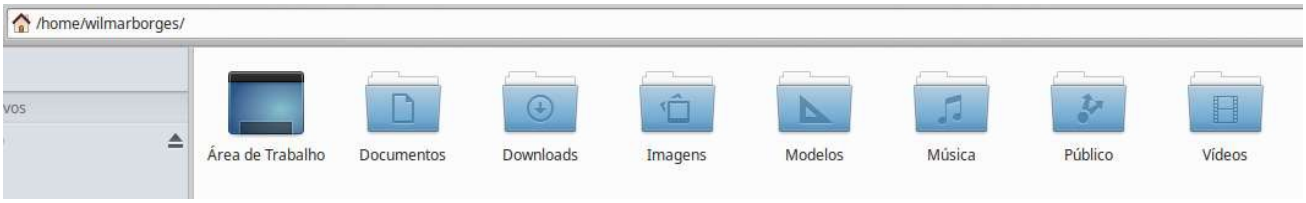


Figure 3: home do usuário logado

Seu diretório `home` é um diretório muito importante. Para começar, quando você abre um shell, você deve ser colocado automaticamente no seu diretório `home`, pois é onde você fará a maior parte do seu trabalho. O comando `pwd` mostra onde está.

```
wilmarborges@Linux:~$ pwd
/home/wilmarborges
```

O seu diretório pessoal é um dos poucos diretórios onde você tem o controle total para criar e excluir arquivos. A maioria dos outros diretórios em um sistema de arquivos do linux é protegida com permissões em que se faz necessário ser ou executar como `root`, o outro diretório é `/tmp,` onde também passaremos boa tarde trabalhando nele, já que todos os usuários tem privilégios de leitura, escrita e execução nesse diretório, além do `/home`, claro.

4.2 Entendendo a listagem completa ls -l

O comando `ls` lista os diretórios e arquivos de forma simples, com a opção `-l` é exibida informações sobre cada arquivo, chamados metadados, que às vezes são úteis para

serem exibidos podendo incluir informações uteis como, quem possui um arquivo, o tamanho de um arquivo e a última vez que o conteúdo de um arquivo foi modificado. Você pode exibir essas informações usando a `-l` opção do `ls comando`.

```
wilmarborges@Linux:~$ ls -l
total 32
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52 'Área de Trabalho'
-rw-rw-r-- 1 wilmarborges wilmarborges    0 jan 26 11:10  arquivo.txt
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Documentos
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Downloads
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Imagens
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Modelos
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Música
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Público
-rw-rw-r-- 1 wilmarborges wilmarborges    0 jan 26 11:10  video.avi
-rw-rw-r-- 1 wilmarborges wilmarborges    0 jan 26 11:10  video.jpg
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Vídeos
```

Na saída acima, cada linha descreve metadados sobre um único arquivo. Vamos descrever cada um dos campos de dados que você verá na saída do `ls -l`. O primeiro caractere de cada linha indica o tipo de arquivo. Os tipos de arquivos são:

Símbolo	Tipo de arquivo	Descrição
d	diretório	Um arquivo usado para armazenar outros arquivos.
-	arquivo regular	Inclui arquivos legíveis, arquivos de imagens, binários, compactados, vídeo, mp3, etc.
l	link simbólico	Aponta para outro arquivo.
s	soquete	Permite a comunicação entre processos.
p	tubo	Permite a comunicação entre processos.
b	arquivo de bloco	Usado para se comunicar com o hardware.
c	arquivo de caracteres	Usado para se comunicar com o hardware.

O primeiro caractere de cada linha mostra o tipo de arquivo, veja a tabela acima.

```
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Documentos
-rw-rw-r-- 1 wilmarborges wilmarborges    0 jan 26 11:10  video.jpg
```

Essa parte é a permissão do arquivo, veremos em detalhes, sobre permissões.

```
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Documentos
-rw-rw-r-- 1 wilmarborges wilmarborges    0 jan 26 11:10  video.jpg
```

Nessa parte mostra a quantidade de links físicos que apontam para esse arquivo

```
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Documentos
-rw-rw-r-- 1 wilmarborges wilmarborges    0 jan 26 11:10  video.jpg
```

Aqui é bem simples, todo arquivo foi criado por alguém com uma conta, isso é importante pois o dono do arquivo tem o direito de definir permissões em um arquivo. Veremos mais detalhes sobre isso quando tratarmos de criação de usuário e permissões

```
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Documentos
-rw-rw-r-- 1 wilmarborges wilmarborges    0 jan 26 11:10  video.jpg
```

Aqui é indica qual grupo possui este arquivo, também é importante compreender porque qualquer membro desse grupo tem um conjunto de permissões no arquivo. Sobre permissões haverá um capitulo próprio.

```
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Documentos
-rw-rw-r-- 1 wilmarborges wilmarborges    0 jan 26 11:10  video.jpg
drwxr-xr-x   2 root root 4096 jan 25 08:17 /bin #veja que o diretório bin é do root
```

Indica o tamanho do arquivo em bytes.

```
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Documentos
-rw-rw-r-- 1 wilmarborges wilmarborges    0 jan 26 11:10  video.jpg
```

Timestamp, indica a hora em que o conteúdo foi modifica pela última vez. Para diretórios, esse registro de data e hora indica a última vez em que um arquivo foi incluído ou excluído do diretório.

```
drwxr-xr-x 2 wilmarborges wilmarborges 4096 jan 24 15:52  Documentos
-rw-rw-r-- 1 wilmarborges wilmarborges    0 jan 26 11:10  video.jpg
```

Um minuto para reflexão, você entendeu a necessidade de todas essas “marcações” para o sistema operacional e para a operacionalidade de um técnico, e até mesmo para

computação forense[1], um arquivo traz muitos detalhes que para o dia-a-dia de um usuário comum, pode ser ignorado, mas, para outras profissões são essenciais, pense nisso.

```
A partir de certo ponto não há retorno. Esse é o ponto que é preciso alcançar.
Franz Kafka
```

4.3 Exercícios do Capitulo

59. Crie um atalho do seu usuário no diretório /tmp

```
.
└── DiretorioRaiz
    ├── contas
    │   ├── agua
    │   └── energia
    ├── escola
    │   ├── informatica
    │   └── matematica
    └── fruta
        ├── laranja
        └── pera
```

60. Em um só comando crie quatro diretórios um dentro do outro, o primeiro será chamado de DiretorioRaiz seguidos pelo diretório fruta que deverá conter laranja e pera um diretório gastos dentro do DiretorioRaiz que deverá conter água e energia e um diretório Escola dentro do DiretorioRaiz que deverá conter informática e matemática. Não se esqueça tudo em uma só linha de comando e utilizando metacaracteres. Exemplo de como ficará os diretórios criados, ao lado.

61. Agora, utilizando o radical [ama] no futuro do presente do indicativo, faça sua conjugação verbal e envie para um arquivo chamado verbo.txt, utilize-se os metacaracteres para isso. ficará assim: (amarei amarás amará amaremos amareis amarão); faça da forma mais sintética possível, utilize os metacarateres.

62. Agora, utilizando o radical [ama] no futuro do presente do indicativo, faça sua conjugação verbal e crie arquivos de texto com os nomes gerados, utilize-se os metacaracteres para isso.

5 Edição de arquivos

Já disse e repito, esse livro não é voltado para o profissional linux, é para o iniciante, por essa razão, vamos facilitar um pouco na edição dos nossos scripts, não utilizaremos o vi,

[1] Computação forense (também conhecida como ciência forense computacional) é um ramo da ciência forense digital pertencente às evidências encontradas em computadores e em mídias de armazenamento digital.

para isso utilizaremos dois programas o `gedit` que já é padrão em algumas distribuições, e por isso deve ser instalado, caso não tenha e o `geany` que é um editor de programa para o linux, claro, muitos podem pensar: que absurdo usuário raiz do linux tem que usar o vi, bem, quando nosso aluno iniciante se transformar em um usuário avançado, ai sim, tenho certeza que ele sentira a necessidade de utilizar o vi, mas, para o nosso aprendizado, vamos facilitar, vamos usar o `geany` para criar nossos scripts e `gedit` para abrir arquivos de modo visual ou até mesmo para editar os scripts.

Instalando o Geany

```
sudo apt-get install geany
```

Instalando o Gedit

```
sudo apt-get install gedit
```

Feito isso, já explicado porque não utilizaremos o vi, vamos em frente. Como abriremos os arquivos agora, simples, com o `gedit` ou o `geany` instalados, depois de um arquivo criado digite o programa a qual será aberto e o caminho/nome do arquivo. Exemplo no terminal.

```
wilmarborges@Linux:~$ date > ArquivoCriado.txt
wilmarborges@Linux:~$ gedit ArquivoCriado.txt
wilmarborges@Linux:~$ geany ArquivoCriado.txt
```

Veja que na primeira linha acima, a saída do comando `date` foi enviado para o arquivo chamado `ArquivoCriado.txt`, para abrir o arquivo criado usei o programa `gedit ArquivoCriado.txt` e na última linha abri como `geany` da mesma forma, vale lembrar que esse arquivo foi criado pelo usuário e aberto pelo usuário, se fosse um arquivo de outro usuário e seu usuário não tivesse permissão para abri-lo, necessitaria utilizar antes o comando: `sudo gedit/geany ArquivoCriado.txt.`

```
Não vês que somos viajantes?
E tu me perguntas:
Que é viajar?
Eu respondo com uma palavra: é avançar!
Experimentais isto em ti
Que nunca te satisfaças com aquilo que és
Para que sejas um dia aquilo que ainda não és.
Avança sempre! Não fiques parado no caminho. Santo Agostinho
```

5.1 Exercícios do Capitulo

63. Crie um arquivo chamado logData.txt, coloque dentro, via comando, envie apenas a data do sistema de forma que, todas as vezes que a data mudar muda-se também a data no logData.txt, envie também os últimos 5 comandos que você deu, gravados no history e seu nome completo, um abaixo do outro separado por um traço (----)

64. Crie um comando que limpe o histórico digitado no shell, mas, antes envie o histórico criado para um arquivo de texto em `/tmp` o arquivo deverá estar oculto para o sistema

65. Abra o gedit, via terminal, apague o logData.txt, criado anteriormente e salve novamente

66. Abra o geany, via terminal, escreva seu nome no arquivo logData.txt e salve em um novo arquivo em `/tmp/`

6 Empacotando e compactando arquivos

Primeiramente vamos definir empacotar e compactar. Empacotar é colocar tudo em um pacote, ou seja, o empacotamento não diminui seu tamanho, vai o tamanho original, ele apenas faz a junção de um ou mais arquivos em um pacote.

6.1 Empacotando os Arquivos

Vamos ver uns exemplos para facilitar o entendimento, faça em seu terminal para melhor compreensão.

```
wilmarborges@Linux:~$ mkdir empacotamento
wilmarborges@Linux:~$ cd empacotamento/
wilmarborges@Linux:~/empacotamento$ touch foto{1..1000}.jpg
wilmarborges@Linux:~/empacotamento$ ls
```

Descrevendo o passo-a-passo, criei um diretório empacotamento, depois acessei esse diretório através do comando `cd`, criei vários arquivos com o comando `touch foto{1..1000}.jpg` , depois listei para ver se foi criado com o `ls`. Bem, com o `ls`.

Agora imagine que você precisa enviar todas essas imagens para alguém por e-mail, ou pelo whatsapp, você tem duas opções, apenas empacotar, para que os arquivos fiquem em um só arquivo para facilitar anexar no e-mail/whatsapp (upload), ou compactar para diminuir seu tamanho, isso mesmo, quando se `compacta`, você também, está de uma certa maneira, empacotando, mas, se está apenas empacotando não está compactando os arquivos.

Usando o utilitário `tar`, que é a forma abrevia para **TA**pe a**R**chive. (O nome é derivado de uma época em que os arquivos eram normalmente copiados para backup e ocasionalmente recuperados da fita magnética como um dispositivo de armazenamento permanente, por isso o nome arquivo de fita). O `tar` carrega vários arquivos em um só lugar, como um pacote mesmo.

Vamos ver 3 modos utilizados com o comando `tar.`

```
Criar:   crie um novo arquivo a partir de uma série de arquivos
Extrair: extrair um ou mais arquivos de um arquivo
Lista:   mostra o conteúdo do arquivo sem extrair
```

Algumas opções do comando

```
-c : cria um novo arquivo tar.
-j ou --bzip2 : compacta/descompacta os arquivos usando bzip2.
-J ou --xz : descompacta os arquivos .xz e .lzma.
-t : lista o conteúdo do arquivo tar.
-x : extrai o conteúdo do arquivo tar.
-v : mostra mensagens.
-f arquivo - define o nome do arquivo tar.
-z ou --gzip ou --gunzip - compacta/descompacta os arquivos usando gzip/gunzip.
-Z ou --compress ou --uncompress - compacta/descompacta os arquivos usando compress.
```

Tabela retirada de site: http://www.uniriotec.br/~morganna/guia/tar.html

A combinação `cf` é necessária pois o `-c` cria e `-f` define o nome do pacote `tar` criado, após a criação do pacote ele estará no local onde foi empacotado.

```
tar -cf pacote.tar arq*
```

Aqui vejo o conteúdo do pacote `tar`, sempre precisar abri-lo.

```
tar tvf pacote
```

Aqui estamos extraindo os arquivos.

```
tar xvf pacote
```

```
Conhece-te. Aceita-te. Supera-te. Santo Agostinho
```

6.2 Compactando os Arquivos

Depois que entendemos como empacotar, agora vamos compactar, faça em seu terminal o exemplo abaixo para entender melhor, aqui estou compactando em `gzip`.

```
tar czvf pacote_compactado arq*
```

Pacote Gzip	14,5 kB	pacote_compactado
Pacote Tar	1,0 MB	pacote_empacotado

Figure 4: Diferença entre tar e gzip

Aqui, de modo visual a diferença entre empacotar e compactar, os arquivos empacotados ficaram no pacote com 1,0MB, compactado, os arquivos ficaram com 14,5kB, para descompactar, basta utilizar o `-x` (extrair)

```
tar -xf pacote_empacotado
```

Uma observação importante: há várias formas de compactar arquivos no linux, não se prenda apenas ao que foi passado aqui, faça uma busca na internet por compactadores de arquivos no linux, entretanto, o `gzip` está bom para nosso aprendizado inicial, quando quiser se aprofundar, aprofunde.

6.3 Exercícios do Capitulo

67. Crie um diretório chamado pacote, crie 1000 arquivos `.txt; .jpg; .py; .avi; .sh`; utilize os metacaracteres para realizar a tarefa de criação dos arquivos, pensa na menor forma possível para criar os arquivos.

68. Agora empacote todos os arquivos do diretório em .tar

69. Desempacote todos os arquivos empacotados da questão anterior em `/tmp/`

70. Desempacote todos os arquivos no diretório extração dentro do /tmp, porém o comando deverá ser dado na mesma linha desempacotar direto no diretório que deverá ser criado junto

71. Agora empacote apenas os arquivos com extensão .avi

72. Desempacote todos os arquivos empacotados da questão anterior em /tmp em um diretório chamado vídeos em /home/vídeos, lembre-se, tudo em uma só linha de comando

73. Agora empacote os arquivos como extensão .sh e envie-os, empacotado para `/home/seu_usuário_/scriptsh`, uma só linha de comando

74. Desempacote todos os arquivos empacotados da questão anterior e de permissão de execução para todos se uma vez em uma só linha

75. Liste todos os scripts do diretório colocando-os em `/tm/ex75.log`

76. Faça a compactação dos arquivo `gzip` dos arquivos `.sh`

77. Faça a descompactação dos arquivo `gzip` dos arquivos `.sh`

78. Essa vale a pena: Gere um arquivo de backup de tudo em `/home/<seu_usuário>/` com o nome do arquivo sendo a data (`Ano/Mês/Dia/Hora/Minuto/Segundo)tar.gz` < ficando assim o arquivo para facilitar a data e hora do backup

79. Agora compacte os arquivos como extensão `.sh` e envie-os, empacotado para `/tmp`

80. Desempacote todos os arquivos empacotados da questão anterior em /tmp em um diretório chamado `script_gzip`, lembre-se, tudo em uma só linha de comando

81. Gere um backup do `/home/<usuário_logado>` com `(Ano/Mês/Dia/Hora/Minuto/Segundo)tar.gz` porém deverá gerar um log externo do conteúdo desse backup, o log conterá nome do usuário que gerou o backup, a data, a lista de arquivos dentro do backup gerado.

82. Depois de criados os arquivos com extensão, do início do exercício, ex67, faça: Empacote apenas os 10 primeiros arquivos de cada e envie para 10deCada em `/tmp`

```
A força fez os primeiros escravos, a sua covardia perpetuou-os. Jean-Jacques Rousseau
```

7 Gerenciamento de Usuários no Linux

Já sabemos que o linux é um sistema multiusuário, e o acesso simultâneo de várias pessoas é algo rotineiro em qualquer servidor de pequeno, médio ou grande porte. O técnico que deve gerenciar todos esses usuários, criar, excluir, alterar, dentre outros, por isso, vamos ver um pouquinho de gerenciamento de usuários.

As tarefas básicas no gerenciamento de usuário são:

- Configurar um ambiente;
- Criar conta e definir senha
- Alterar senhas

- Remover a conta
- Definir políticas de uso da conta, privilégios

É salientar que há inúmeras ferramentas para isso, como o `webmin`, isso facilita e muita a vida de um administrador de sistemas, pois de modo visual fica mais fácil o entendimento, mas, como esse livro é sobre criação de `scripts`, então, vamos trabalhar apenas no modo texto, temos que conhecer a base. A tabela abaixo mostra os comandos para gerencias usuários.

Comando	Descrição
useradd	Cria usuário
userdel	Remove usuário
usermod	Edita dados do usuário
passwd	Define a senha do usuário
groupadd	Cria grupo
groupdel	Remove um grupo
groupmod	Edita dados de um grupo

Basicamente criar um usuário consiste em definir qual vai ser o seu nome de `login` e `senha`, assim, usamos os dois comandos `useradd` e `passwd` para isso. Acompanhe, fazendo em seu terminal.

```
sudo useradd aluno
sudo passwd aluno
Digite a nova senha UNIX:
Redigite a nova senha UNIX:
passwd: senha atualizada com sucesso
```

Utilize os comandos `useradd` e `passwd`. Para criar usuário e definir a senha. Importante ressaltar que, nos comandos acima, só criei um usuário e defini a senha para esse usuário, no caso o `/home` desse usuário <u>não foi criado</u>, para isso né necessário compreender alguns parâmetros para cria-lo, vamos lá, se tentar fazer o login com esse novo usuário, o mesmo não vai logar, vamos ver mais opções para resolver isso.

```
sudo useradd -d /home/aluno aluno
```

Comando	Argumento	Descrição
useradd usermod	-d	Especifica o caminho do diretório HOME
	-g	Especifica o grupo do usuário
	-c	Especifica o nome completo do usuário
	-s	Especifica a shell do usuário
	-m	Cria o diretório pessoal do usuário
userdel	-r	Remove também o diretório HOME
passwd	-d	Deixa o usuário logar sem precisar de senha
	-l	Trava a conta do usuário, impedindo de fazer login
	-u	Destrava a conta
	-S	Mostra o status da senha do usuário
groupadd	-g	Especifica o GID (número de identificação) do grupo
groupmod	-g	Especifica o GID (número de identificação) do grupo
	-n	Especifica um nome novo para o grupo

Para saber mais sobre o comando digite-o sem nenhuma opção apenas `useradd`

```
sudo useradd -m aluno
sudo passwd aluno
Digite a nova senha UNIX:
Redigite a nova senha UNIX:
passwd: senha atualizada com sucesso
```

Veja como é simples, a combinação dos comandos e opções gera várias possibilidades. Todos esses comandos, criação de usuário, senhas etc, vai para algum lugar, algum arquivo, uma base de dados, onde o sistema possa recuperar quando necessário, existem vários, mas, não é a intenção aqui de sermos completos, vejamos alguns apenas para entender.

Comando	Descrição
/etc/passwd	Arquivo com informações de usuários, login, nome, UID, grupo etc
/etc/shadow	Arquivo onde ficam armazenadas as senhas, criptografadas" (hashs) dos usuários e outras informações como validade
/etc/gshadow	Semelhante ao /etc/shadown, porém, para grupos do sistema
/etc/group	Arquivo com texto que é exibido após o login de cada usuário

Se você revelar os seus segredos ao vento, não pode culpar o vento por revelá-los às arvores. Khalil Gibran

Vamos criar dois usuários, Finn e Jake

```
sudo useradd -m finn
sudo passwd finn
Digite a nova senha UNIX:
Redigite a nova senha UNIX:
passwd: senha atualizada com sucesso
```

```
sudo useradd -m jake
sudo passwd jake
Digite a nova senha UNIX:
Redigite a nova senha UNIX:
passwd: senha atualizada com sucesso
```

Pronto, vamos ver como está o arquivo `/etc/passwd`

```
cat /etc/passwd
```

No final do arquivo você verá

```
finn:x:1002:1002::/home/finn:/bin/sh
jake:x:1003:1003::/home/jake:/bin/sh
```

Agora vamos criar um grupo chamado jurídico

```
sudo groupadd juridico
```

Veja como ficou o arquivo `group` em `/etc/group`

```
cat /etc/group
```

Veja o resultado, faça em seu terminal para melhor compreensão.

```
juridico:x:1004:
```

Pois bem, qual grupo, `finn` e `jake`, que foram recém-criados pertence? hora, cada um está em seu próprio grupo, grupo finn e grupo jake, veja o conteúdo do arquivo `/etc/group` que os encontrará como grupo, assim, quando não se define o grupo à qual pertencerá o usuário é criado com seu próprio nome de usuário. Agora vamos colocar finn e jake no grupo jurídico.

```
sudo addgroup finn juridico
sudo addgroup jake juridico
```

Se acessar ou apenas visualizar com o `cat` o arquivo de grupo `/etc/group` verá algo do tipo: `juridico:x:1004:finn,jake`. Aqui a intenção não é esgotar as possibilidades de criação de grupo, claro, se você fizer uma busca no google de como adicionar usuários ao grupo verá outros modos, mas, o que importa é saber o conceito por trás disso tudo.

Grupo, são conjuntos de usuários onde sua criação facilita a administração, ao invés de aplicar uma ação usuário a usuário, aplica-se ao grupo e as alterações servem para todos. Exemplo prático - imagina uma ação para vários usuários, do tipo atualização de sistema, quando se cria grupos bem definidos posso aplicar aquela ação apenas a um determinado grupo que servirá para todos os usuários pertencentes aquele grupo.

`Uma vida sem desafios não vale a pena ser vivida. Sócrates`

7.1 Exercícios do Capitulo

83. Crie os usuários: Gato, Cachorro, Galinha, Pintado, Tilápia, Carpa, Tubarão, Andorinha, Arara, canário

84. Crie uma senha para todos os usuários cadastrados com o padrão 12345

85. Crie os grupos: Animais, Peixes, Pássaros .

86. Coloque os usuários em seus grupos: `Animais{Gato, Cachorro, Galinha}; Peixes{Tilápia, Carpa, Tubarão}; Pássaros{Andorinha, Arara, Canário}`

87. Qual o arquivo está localizado os usuários criados

88. Qual o arquivo está localizado os grupos criados

89. Faça com que os usuários do grupo Animal sejam forçados a redefinir a senha no próximo logon no sistema

90. Crie um usuário com seu primeiro nome para logon e defina o diretório `/home` com o nome tecnologia e coloque seu nome completo para o sistema.

91. Remova o seu usuário logado e deixe o seu diretório `/home/tecnologia`

92. Remova o seu usuário logado e apague também o seu diretório `/home`

93. Como travar a conta do usuário criado, teste em seu usuário

94. Como destravar a conta do usuário travado anteriormente

95. Apague os usuários criados

96. Apague os grupos criados

8 Variáveis de Ambiente

Trabalharemos nesse capitulo com variáveis de ambiente no linux, a linha de comando é um ambiente que hora o outro o administrador estará lá, trabalhando no seu dia a dia, é importante a personalização para um uso mais confortável. O shell bash é muito flexível quando se trata de configurações, aqui vamos trabalhar com as variáveis de ambiente.

Uma variável é um nome especial que está associado a um valor. Este valor pode ser um número, uma palavra, um texto ou um comando.

Vejamos um exemplo:

```
MEUNOME="Wilmar Borges Leal Junior"
echo $MEUNOME
Wilmar Borges Leal Junior
```

Veja que criei uma variável chamada `MEUNOME` e logo abaixo mostrei o valor dessa variável com o comando `echo`, com `$MEUNOME` para chamar as variáveis criadas basta colocar o `$`. Aproveitando, para remove-la use o comando `unset MEUNOME, unset <nome_da_variavel>.`

O comando `env` mostra todas as variáveis de ambiente no linux, gora, para que a variável criada pelo usuário também seja acessada ou válida para os programas, faz-se necessário exporta-las com o comando `export`.

```
MEUNOME="Wilmar Borges Leal Junior"
export MEUNOME #nome da variável
```

Assim, a variável criada fará parte do ambiente de todos os processos que forem executados desde ponto em diante, porém elas funcionarão apenas na sessão aberta, fazendo `logout`, elas serão apagadas. Para torna-las permanentes devemos editar os arquivos de configuração do bash, existem vários locais onde você pode salvar suas variáveis de ambiente para torná-las permanentes, cada um afeta algo diferente. Por exemplo, se você pretende que todos os usuários do sistema também tenham acesso à variável de ambiente que vai criar, edite o seguinte arquivo: /etc/bash.bashrc

```
sudo gedit /etc/bash.bashrc
```

No final do arquivo, coloque suas variáveis, (`export MEUNOME=”Wilmar Borges Leal Junior”`), assim, ela ficará permanente no sistema, isso para todos, caso queira apenas para o seu perfil, basta fazer o mesmo, mas, dentro do `.bashrc` que encontra-se dentro do seu `/home.`

```
"Os lábios da sabedoria estão fechados, exceto aos ouvidos do Entendimento." - O CAIBALION
```

8.1 Exercícios do Capitulo

97. Liste todas as variáveis de ambiente existentes e envie para um arquivo chamado variáveis.txt

98. Crie uma variável de ambiente chamado $USUARIO que receba seu nome completo, lembre-se, por convenção, variáveis de ambiente são criadas em letras maiúsculas.

99. Faça com que a variável anteriormente criada $USUARIO, com seu nome, não se perca, seja uma variável global

100. Crie uma variável de ambiente que limpe

101. Crie uma variável de ambiente que liste todos os usuários cadastrados e envie para um arquivo chamado usu.txt no `/tmp,` com o nome $LISTUSE.

102. Crie uma variável de ambiente que resuma a criação o comando: Em um só comando execute, crie quatro diretórios um dentro do outro, o primeiro será chamado de DiretorioRaiz seguidos pelo diretório fruta que deverá conter laranja e pera um diretório gastos dentro do DiretorioRaiz que deverá conter água e energia e um diretório Escola dentro do DiretorioRaiz que deverá conter informática e matemática. Não se esqueça tudo em uma só linha de comando e utilizando metacaracteres. Exemplo de como ficará ao lado. Chamando a variável $RAIZ, ele criará todos os diretórios anteriores.

103. Crie duas variáveis de ambiente, uma chama: $MEUNOME e a outra $MINHAIDADE, e, usando-as crie uma mensagem e envie-a pra /tmp/saída.txt com a frase, meu nome é .<sua variável> e minha idade é <sua idade>

Diz o Caibalion: "Em qualquer lugar que estejam os vestígios do Mestre, os ouvidos daquele que estiver preparado para receber o seu Ensinamento se abrirão completamente"

9 Expressão Regular

Até aqui estávamos apenas fazendo um apanhado geral, agora iniciaremos a proposta desse guia, os scripts, para tanto, a um conteúdo a ser explicado para entrarmos no script propriamente dito, expressão regular, desse ponto em diante não é necessário mais explicar o comando em si, pois tenho certeza que você não pulou etapas, se você fez todos os exercícios, tenho certeza que já entendeu bem o conteúdo que é só uma questão de praticar, a cada comando um manual que você pode e deve consultar.

Agora sim, começaremos a separar os técnicos dos usuários, saber usar expressão regular não é para qualquer um é para todos, é essencial para administrar um sistema, tarefas e rotinas das simples as mais complexas, repare que falei em rotinas, sim, scripts servem para automatizar essas rotinas, as vezes um administrador de sistema repete infinitamente algo que poderia resolver com um simples script, sabendo utiliza-las, você poderá economizar horas de trabalho que seriam gastas em tarefas manuais e repetitivas, pense nesse capitulo como mais uma habilidade que você adquirira na sua formação técnica.

Expressão Regular é um método formal de se especificar um padrão de texto. Podendo ser usado em vários outros contextos, como em um banco de dados ou, na limpeza de dados em uma planilha eletrônica, esse método pode abranger ou até prever falhas de digitação, as expressões regulares são formadas por caracteres curingas, (viu porque não se pode pular etapas), `*, +, ^, e outros,` cada um possui uma função especial, funcionando como uma ferramenta para auxilia-los na busca de textos, acesso o aurelio.net lá você encontrará um material de excelente qualidade é uma fonte de agua limpa para aqueles que buscam conhecer.

Vamos a um exemplo prático, você precisa fazer uma pesquisa genérica de palavras que começam com vogal e ao mesmo tempo, linhas que contenham 4 números. Que seria algo do tipo: `[0-9]{4}` e `^[AaEeIiOoUu]`, respectivamente, complicado? calma, vamos por parte. Vários são os aplicativos que entendem expressões. Leitores de e-mail, Navegadores, Editor de Texto, Suíte Office e grande maioria das linguagens de programação. Para entender um pouco mais sobre expressão regular, veremos algumas ferramentas de linha de comando `(grep, find, sed, awk).`

Para o administrador de sistemas, as expressões regulares são úteis para pesquisar e extrair informações de arquivos de log. Basicamente serve para você dizer algo `abrangente`

`de forma especifica,` definindo seu padrão de busca, você tem uma lista, finita ou não, de possibilidades. Imagine a seguinte situação para exemplificar melhor, você tem um arquivo que contém `n` palavras, entretanto, você quer buscar as palavras, `gato, rato, prato`, há duas formas, fazer a busca um a um, ou, usar o `regex`, ficando assim: `^[grp]ato`. Vamos ver algumas ferramentas uteis para trabalhar com arquivos, lembrando que não é minha intenção esgotar seu conteúdo e sim dar os primeiros passos

Caracteres especiais utilizados nas Expressões Regulares

Meta Caractere	Sua utilização
^	Representa o começo da linha
$	Representa o fim da linha
[abc]	Casa as lebras "a" ou "b" ou "c"
[a-d]	Casa as letras "a" ou "b" ou "c" ou "d" (range)
[^abc]	Qualquer caractere, exceto "a","b", e "c"
(esse|aquele)	Casa as palavras esse ou aquele
a{2}	casa a letra a duas vezes
a{2,4}	casa a letra a duas a quaro vezes
a{2,}	casa a letra a no mínimo duas vezes
a?	Casa a letra a zero ou uma vez
a*	casa a letra a zero ou mais vezes
a+	Casa a letra a zero ou mais vezes
.	casa um caractere qualquer
.*	casa qualquer coisa

```
O importante não é aquilo que fazem de nós, mas o que nós mesmos fazemos do que os
outros fizeram de nós. Jean Paul Sartre
```

9.1 O comando – cut

O comando `cut` pode ser usado para mostrar apenas seções específicas de um arquivo de texto ou da saída de outros comandos, assim é ótimo para trabalhar com delimitadores entre os dados como `“:”` ou o `“;”`.

```
-b, -bytes [bytes] - Mostra a lista de bytes do arquivo.
-c, -characters [número] - Mostra o número de caracteres do arquivo
 (o Tab e o espaço são contados como caracteres).
-f, -field [campos] - Mostra a lista de campos
-d, delimite [caractere] - Opção para uso com "-f", determina o caractere delimitador e
na ausência o padrão é o Tab.
```

O melhor arquivo para se trabalhar com o `cut` como exemplo, continua sendo o `/etc/passwd` que você já conhece de capítulos anteriores. Execute os comandos abaixo em seu terminal e veja o que acontece.

```
cat /etc/passwd
```

Note que a separação das colunas é dada pelo `":"` (dois-pontos), veja abaixo um infográfico

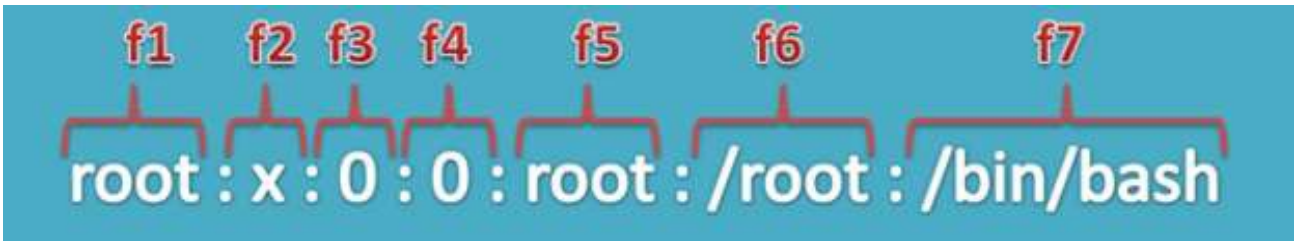


Figure 5: retirado do site: https://tecadmin.net/linux-cut-command/

Perceba que a coluna 1 `(f1)` é onde encontra-se o `nome` do usuário cadastrado, logo, para listar apenas os usuários, sem as outras informações, digitamos:

```
cut -d : -f1 /etc/passwd
# resumindo: o cut está mostrando apenas a primeira coluna f1 que é delimitada pelo : -d
```

Execute em seu terminal para ver o que ele mostrará, percebeu as possibilidades desse simples comando, veja o manual de utilização do comando `cut` para mais informações, por hora esse exemplo já está ótimo.

9.2 O comando – grep/egrep

Os comandos `grep` e `egrep` servem para realizar buscas em arquivos, pastas e linhas dentro de um documento, podemos usar expressões regulares para realizar as buscas de forma que o texto ou palavra digitada case com a expressão. Exemplo, durante uma pesquisa, quero que você me retorne somente os arquivos com extensão `.doc` isso é possível com o `grep`, de outra forma também, mas, aqui é para exemplificar o comando estudado.

O `egrep` é um atalho para o `grep` mais o parâmetro `-E.` A opção `-E` permite o uso de expressões regulares na filtragem de texto: Exemplo: `+, ^, [],` e outros já visto.
Vejamos um pequeno exemplo simples do uso do `grep`.

```
grep jake /etc/passwd
```

Aqui, estou pesquisando se existe o usuário `jake` dentro do arquivo `/passwd`

```
grep fin /etc/passwd
```

Aqui estou pesquisando se existe o usuário `finn` dentro do arquivo `/passwd`, e utilizando a opção `-E` estou pesquisando os dois juntos, o parâmetro `-E` serve para o `grep` utilizar expressão regular.

```
grep -E "jake|root" /etc/passwd
```

Agora, vamos passar o resultado do `ls` para o `egrep` realizar o filtro, utilizando o `pipe` `|`, já estudado.

```
ls -lR ~ | egrep .txt
```

Você já deve ter entendido, mesmo assim, vou explicar: estou listando tudo que há no diretório `~` recursivamente `-R` e o resultado, é enviado para o comando `egrep,` através do pipe `|`, assim, estou dizendo para o `egrep` mostrar apenas os arquivos que tenham .txt como extensão. Poderoso não é, há outras formas, mas, vamos ficar por aqui, por enquanto, pra não confundir muito.

```
Quanto aos homens, não é o que eles são que me interessa, mas o que eles podem se
tornar. Jean Paul Sartre
```

9.3 O comando – wc

Basicamente o `wc` é usado para contar, isso mesmo, contar, seja caracteres ou bytes em um arquivo, vejamos uns exemplos com seus parâmetros para facilitar, ele é bem simples mesmo.

```
wc /etc/passwd
wc -l /etc/passwd #conta o número de linhas
wc -w /etc/passwd # conta as palavras
wc -c /etc/passwd # conta os caracteres
```

Simples assim, mas, lembre-se que sempre podemos combinar comandos no linux para obter um resultado.

9.4 O comando – sed

E por fim o `sed`, (Strem Editor) que também não aprofundaremos, pelo menos nesse momento é juntamente com a `awk`, uma linguagem de manipulação de arquivos e streams do Unix/Linux. Ambas possuem vasta abrangência e o que uma não pode fazer a outra provavelmente o fará, com o `sed` é possível substituir e casar padrões, sempre por meio de regex, é uma grande ferramenta no mundo computacional, o `sed`, assim como o `awk`, lê um arquivo, linha por linha e aplica a expressão do parâmetro a cada uma delas. `$man sed`

Um excelente material com inúmeros parâmetros do sed, encontra-se em um artigo publicado na revista eletrônica (CCM, 2011), link nas referências.

Parâmetros do sed

-i altera o arquivo
-e imprime na tela sem alterar o arquivo
-n faz a supressão, mostra só o resultado do comando
-s substitui um trecho de texto por outro
-! inverte a lógica do comando
-; separador de comandos *separador de strings*
-d no final deleta (CUIDADO)
-p no final imprime (SEMPRE SUBSTITUA PELO d para TESTAR)
-g no final (como se usa o d e p) altera todas as ocorrências
-q sai do sed , não continua o comando

Nada melhor do que exemplos para se ter uma melhor compreensão, vamos lá. Imagina que tenho um arquivo com uma lista, nessa lista era para ter apenas meu nome Wilmar, porém quem a fez acabou escrevendo meu nome em um arquivo chamado `nomes.txt` de várias formas em várias linhas diferentes, ficando mais ou menos assim:

```
Wilmar
wilmar
vilmar
Vilmar
```

```
Uilmar
uilmar
```

Preciso fazer uma correção nessa lista, substituindo todos esses nomes por WILMAR, tudo maiúsculo, "o `sed` pode ser usado para muito mais que isso", esse exemplo é para introdução apenas.

```
sed 's/[WwVvUu]ilmar/WILMAR/' nomes.txt
```

O processo é autoexplicativo, mas, vamos lá. Veja que `/s` é parâmetro do `sed` para substituir, ao invés de escrever um por um, use expressão regular `[WwVvUu]ilmar`, para procurar linha a linha e mudar para /WILMAR, tudo maiúsculo em `nomes.txt`, veja a utilidade do `sed` e da expressão regular. Resumindo, o `sed`, permite encontrar e substituir expressões em arquivos (Eduardo Morimoto, 2004, p. 30)

9.5 O comando – tail

É geralmente usado para examinar as últimas linhas de um arquivo, muito útil para examinar arquivos de log, `tail /etc/passwd` mostra as 10 ultimas linhas do arquivo `passwd`, entretanto, utilizando o parâmetro `-n` pode-se alterar esse valor `tail -n 5 /etc/passwd.` Para ver as ultimas 5 linhas, basicamente é isso, basicamente.

9.6 O comando – expr

O comando `expr` trabalha com expressões matemáticas simples e somente números inteiros, exemplo:

```
expr 2 + 14
```

O comando – bc

O comando `bc` é utilizado para expressões numéricas complexas, ou com casas decimais. Pode-se também ser utilizada para conversão de base numérica. Exemplo:

```
echo "2 ^ 4 + 3" | bc
```

O comando – sort

O `sort`, por padrão, ordena os elementos de um arquivo ou saída de comando. Veja a saída do comando abaixo no arquivo `passwd`, faça em seu terminal.

```
sort /etc/passwd
```

9.7 O comando – uniq

O comando `uniq` remove as linhas duplicadas de um arquivo ou de uma saída de comando redirecionada, lembra do pipe |, é isso a saída redirecionada, sendo necessário que o arquivo esteja ordenado para que as linhas duplicadas sejam consecutivas dentro do arquivo.

9.8 O comando – nl

O comando `nl` numera todas ou algumas linhas de um arquivo, há muito mais, claro, mas, a princípio é isso, vamos a um exemplo, faça em seu terminal e veja o resultado

```
nl /etc/passwd
```

9.9 O comando – more

Mostra o arquivo na saída padrão, caso o tamanho do arquivo for maior do que o número de linhas de tela, faz uma pausa e aguarda a tecla, enter, espaço ou `ctrl + esc` para continuar a exibição.

```
ls -la / | more
```

9.10 O comando – less

O less é muito parecido com o comando more, mas permite a navegação dentro do arquivo ou saída do comando, utilizando as setas, com possiblidade até de pesquisa dentro do documento ou saída do comando

```
ls -la / | less
```

/; pesquisar

q; sair

espaço; avançar uma tela

9.11 O comando – find

Um dos desafios que os usuários enfrentam ao trabalhar com o sistema de arquivos é tentar recuperar o local onde os arquivos estão armazenados. Existem milhares de arquivos e centenas de diretórios em um sistema de operacional, portanto, lembrar onde esses arquivos estão localizados pode representar um grande desafio, por essa razão, utilizamos o localizador `find`, embora o Linux possibilite trabalhar com a interface gráfica — assim como no Windows — e isso possa parecer mais prático, a utilização das linhas de comandos promove mais agilidade e eficácia para realizar tarefas.(Pedrodelfino, 2016).

O layout do comando é `find <Diretório_Inicial> <opções> <termo pesquisado>.` Vamos a uns exemplos, procurando por nome: `find . -name "arquivo que você quer buscar",` só para relembrar o `"."` (ponto) é o diretório atual, diretório onde você está. O argumento `-name` é `case-sentive`, ou seja, diferencia maiúscula de minúscula.

Você pode procurar entre múltiplos arquivos com um formato comum como por exemplo `.txt`, que pode ser útil em algumas buscas: `find . -name "*.txt"`, veja que estou usando metacaracteres. Caso queria buscar todos os arquivos e exclui-los, utilize o `-delete` no final: `find . -name "arquivo_procurado" -delete`, o `find` é tão útil que você pode procurar o arquivo por `tipo`, ou seja, aqui, o argumento `-type` com as seguintes opções quando procuramos arquivos pelo tipo.

- **f** – Arquivo normal
- **d** – Diretório ou pasta
- **l** – Link simbólico
- **c** – Dispositivos de caracteres
- **b** – Dispositivos bloqueados

Um exemplo simples de utilização. Preciso buscar apenas os diretórios, como saber se é diretório? Retorne a aula entendendo permissões de arquivos, `find / -type d` . Existe vários outros modos de utilização do `find`, utilize o manual do `find` para saber mais, ou leia o artigo em (Rafael H, 2017). Um exemplo bem interessante, fazendo a combinação de dois comandos utilizando o pipe, canalizando a saída do `find` e enviando para o `grep` filtrar apenas o `libxt` para mostrar na tela.

```
find /usr -name '*.so' | grep libxt
```

```
É preciso ter a coragem de fazer como todo mundo para não ser como ninguém. Jean
Paul Sartre
```

9.12 Exercícios do Capitulo

Vale lembrar que, a partir daqui os exercícios serão um pouco mais extensos e um pouquinho mais complexos, pois podemos misturar vários comandos até aqui já estudados.

104. Liste apenas a coluna onde estão os grupos existentes no arquivo onde são cadastrado os grupos de usuários e envie para `/tmp/<`padrão>

105. Mude, dentro do arquivo gerado o nome do usuário root para ROOT.

106. Liste apenas a coluna com os nomes do usuário do arquivo onde ficam os usuários cadastrados no linux

107. Veja o arquivo `/etc/passwd`, sabe-se que dentro desse arquivo fica todos os usuários cadastrados no sistema, entretanto nem todos usuários (apenas usuário) que contenham um /home, então, liste apenas os usuários que tenham um `/home`

108. Lembra nos exercícios iniciais onde você pegava a data e dividia em `+%d%m%y, (dia, mês e ano).` Agora, utilizando o `cut`, recorte apenas a semana e gere um diretório no seu /home

109. Lembra nos exercícios iniciais onde você pegava a data e dividia em `+%d%m%y, (dia, mês e ano).` Agora, utilizando o `cut`, recorte apenas o mês e gere um diretório no seu /home

110. Lembra nos exercícios iniciais onde você pegava a data e dividia `em +%d%m%y, (dia, mês e ano)`. Agora, utilizando o `cut`, recorte apenas o dia da semana e gere um diretório no seu /home

111. Pegue as variáveis criadas anteriormente e crie uma mensagem e envie para um arquivo de texto com os seguintes dizeres, tudo com variável, e utilizando o `cut`: mensagem: Hoje é $SEMANA, dia $DIA, do mês de $MÊS

112. Vamos trabalhar com nosso clássico arquivo `/etc/passwd`, onde fica o cadastro dos usuários do linux. Filtre dentro desse arquivo todas as linhas que começam com as vogais

113. Pegando a questão anterior coloque em ordem alfabética [a-u]

114. Pegando a questão anterior já em ordem alfabética, envie para um arquivo de texto apenas a primeira letra achada [a-u]

115. Depois que gerar um arquivo com a saída do comando anterior, coloque-as todas em lentas maiúsculas

116. Agora, retire as letras duplicadas deixe apenas 1 de cada letra

117. Filtre apenas as linhas que começam com root e aluno

118. Filtre apenas as linhas que contenham no mínimo 4 dígitos na terceira coluna

119. Numere cada linha da questão anterior e gere um log.txt com a linha e o [a-u]

120. Faça uma expressão que extraia do arquivo `/var/log/syslog` somente as mensagens geradas hoje

121. Conte quantas linhas há no arquivo que contém os grupos dos usuários

122. Gere um log que contenha, quantas linhas há no arquivo `/var/log/syslog`, qual o tamanho do arquivo do arquivo, seu nome completo, data que foi criado esse logo e a data completa que foi gerado esse log, todas as vezes que rodar o comando o mesmo colocará o log abaixo, sem apagar anterior

123. Gere um log com as últimas 10 linhas do syslog em /var/log/syslog

124. Gere um log para análise do syslog dos 30 últimos eventos gerados hoje, coloque o arquivo em `/tmp/<`padrão>

125. Calcule os valores abaixo no terminal, utilize o expr e bc
5 + 3 / 2
(5 +3)/ 2^3
(2^2) / 2 * 4

126. Liste somente as permissões dos arquivos e diretórios e subdiretórios do /home

127. Para treinar com o comando find, liste todos os arquivos ocultos do diretório /home do usuário e gere um log em `/tmp` (você pode utilizar o ls também, lembra? Porém, faça com o find para treinar o comando)

128. Localizar arquivos pertencentes ao usuário root iniciando no / .

129. Localizar, dentro do diretório raiz /, todos os arquivos pertencentes ao usuário root e que tenham a extensão `.html` apenas

```
A existência precede a essência. Jean Paul Sartre
```

10 Entendendo permissões de arquivos

Antes de começarmos com o script propriamente dito, precisamos saber um pouco sobre permissões, visto que será por meio dela que o script pode ser ou não executado no terminal. As vezes um usuário apenas executa um determinado comando sem saber o que está fazendo ou até mesmo o que está acontecendo, assim, quando um tutorial pede para dar a permissão `777` em um arquivo o usuário simplesmente cópia, cola e executa. Como estamos formando técnicos é obrigação do técnico saber pelo menos o que está fazendo quando executa o comando `chmod 777 <arquivo>,` vamos lá.

Lembra dos atributos, veja a saída do comando `ls -l ~.`

```
ls -l ~
drwxr-xr-x 2 wilmarborges wilmarborges  4096 jan 24 15:52 'Área de Trabalho'
```

```
d rwx r-x r-x
```

`d:` tipo de arquivo (diretório) relembre no capitulo que trato de arquivos e diretórios no linux, onde tratamos da saída do comando `ls -l`

```
rwx: permissões do proprietário e/ou usuário;
rwx : permissões para usuários do mesmo grupo;
rwx: permissões para todos usuários.
```

O que significa o `rwx`

```
r - read (permissão de leitura);
w - write (permissão de escrita);
x - execute (permissão de execução.
```

Assim, temos 3 casas com `rwx` na forma literal, vejamos na figura abaixo o que significa:

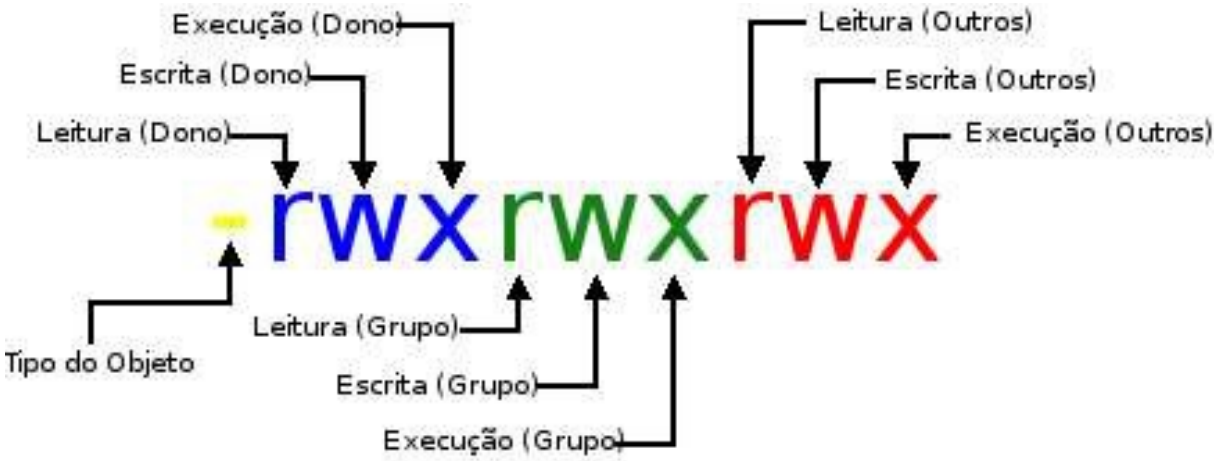


Figure 6: Fonte: http://robertors.blogspot.com/2006/09/linux-permisses-de-acesso-especiais.html

```
O Tipo do Objeto significa:
d - Diretório
b - Arquivo de bloco, armazenamento
c - Arquivo especial de caractere
p – Canal ou pipe de comunicação entre processos
s – Socket de comunicação entre processos
-   Arquivo comum
l – Link simbólico
```

- Vamos chamar a primeira casa de **U**SUÁRIO/DONO
- Vamos chamar a segunda casa de **G**RUPO
- Vamos chamar a terceira casa de **O**UTROS

Formamos a palavra **UGO**, as propriedades são para o `usuário | grupo | outros`.

Podemos ter o método `numérico` para permissões o comando `chmod` aceita também valores numéricos, é uma forma bastante prática e ao meu ver, mais fácil de se entender. Exemplificando, sabendo que são 3 casas para aplicar as permissões no UGO, essas permissões trabalham em `octal`, cada casa pode trabalhar com decimal de 0 a 7, para ficar mais fácil o entendimento vamos ver uns exemplos que montei.

1	Decimal	4	2	1	4	2	1	4	2	1
2	Literal	r	w	x	r	w	x	r	w	x
3	Casas	U	G	O	U	G	O	U	G	O
4	Binário	1	1	1	1	0	1	1	0	0
5	Binário x Decimal		**7**			**5**			**1**	

Na linha 1, temos os valores UGO em decimal, na linha 2, os valores do UGO literal, ou seja `read, write  e execution`, na linha 3 apenas para delimitar as casas UGO que são três, primeira casa permissão para os usuários, segunda casa permissão para o grupo e terceira casa permissão para outros, ou seja aqueles que não está no grupo e também não é o dono do arquivo e na linha 5 a permissão em formato numérico que nada mais é que o valor binário (0 ou 1) x os valores decimais (1,2,4), formando assim a permissão em formato numérico.

Dessa forma uma permissão `777` é o mesmo que `rwx,` e uma permissão `755` é o mesmo que `r-x`, sendo o `(-)` zero em seu valor binário. O que isso tem a ver com script?. Simples, o linux só executa um script se ele tiver, primeiro, permissão para executar, segundo conteúdo de script válido, sem esses dois fatores, não há como executar um script.

Execute os comandos abaixo em seu terminal, depois do # é apenas comentário, não precisa copiar.

```
touch arquivo.sh #Cria o arquivo vazio
ls -l arquivo.sh #lista as permissões UGO
```

Vamos agora modificar as permissões, ao criar um arquive esse é o padrão: `-rw-rw-r--`
Ou seja, o dono pode ler e escrever, o grupo também e outros podem apenas ler o conteúdo do arquivo. Bem, ele não é um arquivo executável, para torna-lo, vamos dar a permissão de executável para todos, para isso utilizamos o comando `chmod`. Ficando assim:

```
chmod 755 arquivo.sh
ou
chmod +x arquivo.sh
```

Veja o resultado utilizando o comando `ls -l arquivo.sh,` viu como é simples, o que precisamos para executar nossos scripts é isso. Criar um arquivo com conteúdo que

possa ser executado e dar permissão para ser executado, e como executar esse script no terminal? basta digitar `./arquivo.sh`

```
O homem nada mais é do que aquilo que ele faz de si mesmo: é esse o primeiro princípio do existencialismo. Jean Paul Sartre
```

10.1 Exercícios do Capitulo

130. Crie uma variável de ambiente chamada EX, essa variável de ambiente deverá ser a variável que dá a permissão de execução para os arquivos, ao invés escrever o comando. Variável + arquivo = EXecução.

131. Crie 10 arquivos .sh em um diretório, em um só comando, coloque todos de uma vez como arquivos de execução.

132. Que opção para chmod atribuiria a permissão de leitura/escrita para o proprietário do arquivo, permissão de somente-leitura para o grupo proprietário e permissão somente leitura para todos os outros? (mostre em octal e em literal)

133. Com ver as permissões de um arquivo?

134. Um arquivo de texto com o nome script.txt, pode se tornar um script executável sem mudar sua extensão? Como e porquê?

11 Iniciando no script

Depois da parte essencial, que nada mais é do que uma visão geral de comandos isolados com o básico de alguns comandos do terminal, vamos iniciar o script propriamente dito, que, nada mais é do que colocar os comandos em pacotinhos para serem executados.

Um script shell é um arquivo que contém comandos executáveis que ficam em um arquivo. Um script pode, portanto, testar a presença de um arquivo ou procurar uma saída

específica e alterar seu comportamento. Você pode criar scripts para automatizar partes repetitivas do seu trabalho, o que libera seu tempo e garante a consistência, toda vez que você usa o script.

Por exemplo, se você executar os mesmos 3 comandos todos os dias, poderá transformá-los em um script que reduzira seu trabalho a um comando, por isso precisávamos ver alguns comandos e seus comportamentos, não foram todos, claro, porque é impossível ver todos os comandos e seus argumentos em um só livro, o importante é você saber onde procurar, quando precisar, saberá buscar para resolver o seu problema, ou, pelo menos não ficará na mão dos outros.

Para criar um script, vamos inicialmente criar um arquivo com a extensão `.sh`, o `sh` é extensão para script shell, mas lembrando que para que um arquivo seja executável ele deve ser transformado em script com `chmod`, veja as aulas anteriores, outra coisa, não é qualquer texto dentro de um `arquivo.sh` que será executado como script, uma dica que sempre dou aos meus alunos é, execute o comando no terminal, se funcionar vai montando o script, se já é um profissional, claro que você já vai fazer diretamente no arquivo, mas, se tiver iniciando, faça devagar, teste antes, depois coloque dentro do pacote e teste tudo, só mais uma dica, sempre utilize maquinas virtuais para seus scripts e sempre coloque um ponto de restauração, snapshot, na máquina virtual, caso aconteça de algo dar errado.

Lembra do comecinho do livro, onde pedi para instalar dois programas, o `gedit` e o `geany`, bem, vamos utiliza-los agora para fazer os scripts, os dois tem uma interface mais amigável para quem está começando, mas, se você quiser fazer em outro programa, fique à vontade.

Vou criar um arquivo para o script e colocar permissão para executar apenas uma vez, quando estiver na dúvida volte aqui e veja como foi feito.

```
touch script.sh    # criando o arquivo que será nosso script
chmod +x script.sh # dando permissão de execução para todos UGO
```

Há várias maneiras de criar um arquivo, vou mostrar apenas uma e fique à vontade para criar sua própria rotina, dentro do `/home` do usuário.

O programa `geany` é de fácil compreensão para sua utilização, depois que arquivo está com permissão de execução `chmod +x <arquivo.sh>`, abra-o no `geany` com o comando:

```
geany script.sh
```

Só mais uma coisinha antes de começarmos. Existem muitos shells, cada um com variações de linguagem de programação, os shells mais comuns são: Bourne Shell (sh) é um shell simples encontrado em qualquer sistema Unix ou Linux. Temos o Bourne-Again shell (bash) é o shell padrão da maioria das distribuições linux. Temos o Korn shell (ksh) que é um shell com mais recursos que o Bourne Shell,o Z shell (zsh) é siminar ao ksh, com mais recursos adicionais e o C shell (csh) proveniente do BSD é uma versão com melhorias do (csh).

Esse pedacinho de história foi só para dizer que quando utilizamos a sintaxe do `Bash` devemos colocar em todos os scripts, na primeira linha o `#!/bin/bash`. Do contrário, scripts que usam a sintaxe estendida do Bash podem não funcionar corretamente.

Veja a figura a coma como ficará o cabeçalho dos seus scripts. Se quiser executa-lo no terminal coloque `./script.sh`, você verá a saída do comando `echo`. Se quiser ver a saída do comando no `geany`, basta clicar em executar ou visualizar o arquivo atual, marcado em vermelho na figura acima.

Vamos fazer mais alguns exemplos para fixação do conteúdo e assim entender com um pouco mais de profundidade. No script podemos colocar sequencias de comandos para serem executados, como por exemplo:

```
#!/bin/bash
date; cal; ifconfig; ls -l ~ #vários comandos em sequência
```

Vale ressaltar que o caractere "#" cerquilha, indica que o restante da linha é um comentário e não será interpretado. Veja que, a saída dos comandos acima é mostrada independente da anterior funcionar, pois, não é uma passagem de parâmetro apenas uma execução de comandos sequenciais, só pra entendermos o poder do script.

```
A felicidade do homem depende de si mesmo. Marco Aurélio
```

11.1 Variável no script

Para atribuir variável utiliza-se o operador `=` (igual), sem espaço em branco antes ou depois. O símbolo `"$"` (cifrão), precedendo o nome da variável, é utilizada para chamar seu conteúdo. Dessa forma temos:

```
#!/bin/bash
NOME="Wilmar Borges Leal Junior"
echo "Seu nome é: $NOME"
```

Utiliza-se o comando read (Lê uma linha da entrada padrão e separa em campos) como input, para script shell, ou seja, entrada de dados. Veja que o script anterior a variável nome já recebe um valor e abaixo é impresso o valor da variável estática, isso é muito útil quando é preciso a todo momento chamar um valor especifico, dessa forma, muda-se apenas em um lugar seu valor ao invés de sair mudando em todo o código. Para saber mais utilize o `help read`

```
#!/bin/bash
Echo -n "Escreva seu nome"
read nome
echo "Seu nome é: $nome"
```

No script acima o valor da variável é aguardado externamente, a diferença é que não é necessário editar o script para mudar o valor da variável, ela esperará o valor para continuar. Com esses dois exemplos fica fácil entender as variáveis já estudadas em outros exemplos. Vamos fazer alguns exercícios de fixação, agora não é mais só comando no terminal, coloque tudo dentro de um pacote de script, não se esqueça do padrão. Relembrando, número do exercício + extensão .sh (Exemplo: ex10.sh), na primeira linha seu nome completo como comentário, ficando assim o cabeçalho:

```
#!/bin/bash
#Wilmar Borges Leal Junior
<inicio do script>
...
<fim do script>
```

`read leitor`.

Finalmente chegamos ao nosso propósito, que é desenvolver scripts, só para deixar claro, caro `$leitor`, não teria como fazermos script sem entender o shell-bash, alguns comandos e como eles se comportam no terminal. Desse ponto em diante fica mais fácil, pois você já sabe que todo o comando que vai no terminal pode ir pro pacotinho chamado `script.sh`, vamos exercitar nosso cérebro fazendo exercícios e a medida em que vamos precisando de mais algumas explicações vou colocando-as aqui e mais, não deixe de usar e abusar do `help, man, info e claro, o bom e velho google`, outra coisa, seu script deverá funcionar em qualquer computador, com qualquer usuário logado, ou seja, abuse também das `variáveis de ambiente.`

11.2 O comando – curl

Para esse exercício, precisamos da ferramenta `curl`, se não lembra com instalar, faça: `sudo apt-get install curl.`

Explicando essa poderosa ferramenta. Segundo artigo publicado no site Linux descomplicado, escrito por: (Costa, 2016). O `curl` é uma ferramenta de linha de comando para manipulação de URLs e transferência de dados (download). O principal benefício do curl é que você pode usá-lo em arquivos shell scripts para automatizar a manipulação de URL's. Ele suporta protocolos, como: FTP, HTTP, FTPS, TELNET, IMAP e outros.

Em termos simplificados, o cURL executa várias solicitações de um cliente para um servidor estabelecendo uma conexão entre eles por meio de um protocolo específico e seus métodos associados. Por exemplo, através de um cliente HTTP, o cURL pode enviar um pedido para ler ou fazer download de conteúdo (GET método de solicitação), ou postar conteúdo através de um formulário em um site (método de solicitação POST). Muitas aplicações e serviços web permitem cURL para interagir com suas APIs (application programming interface). Bem explicado esse artigo não é mesmo? Vamos a alguns exemplos, ainda retirados do mesmo artigo para explicar melhor.

Acesse o site http://wttr.in e veja que ele traz a previsão do tempo em texto puro "sem imagens", agora vamos ver como fica no terminal, vai no seu terminal e digite `curl http://wttr.in/LOCALIDADE`, veja que o curl mostra na tela os dados meteorológicos carregados do site.

Agora um exemplo prático.

```
#Utilize o comando curl para ver todo o conteúdo da página abaixo:
curl https://ww2.uft.edu.br/index.php/ensino/76-ensino/10908-corpo-docente
#Viu como é simples?
```

Como baixar o conteúdo do link para um arquivo?

```
curl https://ww2.uft.edu.br/index.php/ensino/76-ensino/10908-corpo-docente --output
docentes.txt #aqui a saída dos dados output é transferida para um arquivo docentes.txt
```

Veja o exemplo abaixo, estou realizando um simples filtro online, diretamente de uma `url`, compreende o poder do shell? Acesse a página abaixo e veja a relação de docentes, estou extraindo através de comandos no terminal uma lista dos Docente, Curso, Campus e Titularidade, conforme é mostrado no site da UFT (observação, só fiz com vários comandos apenas para demonstrar como é possível a saída de um ser a entrada do outro comando, dá para diminuir o código, utilizando apenas o `sed,` por exemplo) assim, a saída o comando `curl` é passado para o `grep` fazer o filtro onde tiver a `tag <tr>` depois envia para o comando `cut` cortar até `>` na coluna 2 e por fim, o `sed` busca todas as marcações `>*` e as apaga para mostrar o texto. Veja como os comandos trabalham encadeados sinta o poder em suas mãos, você pode fazer online, direto na `url`, ou, caso queira testar localmente é só salvar o arquivo. Vamos ao exemplo.

```
curl https://ww2.uft.edu.br/index.php/ensino/76-ensino/10908-corpo-docente | grep "<tr"*
| cut -d ">" -f2 | sed 's/<[^>]*//g'
```

Aqui, salvei a saída do `curl` em docentes.txt, assim fica mais rápido e fácil trabalhar.

```
cat docentes.txt | grep "<tr"* | cut -d ">" -f2 | sed 's/<[^>]*//g'
```

O exemplo a cima foi aplicado em uma página qualquer, mas, você pode aplicar em uma página ou em um arquivo, tabela, banco de dados, arquivos de log, em uma ou ∞ linhas e colunas, a limpeza de dados é essencial para uma boa mineração.

Não desanime de você, ainda que a colheita de hoje não seja muito feliz. Santo Agostinho.

135. Desenvolva um script que: ao abrir peça ao usuário um $nome e crie um diretório em sua área de trabalho

136. Desenvolva um script que: ao abrir peça ao usuário que digite um nome para um arquivo e envie para o diretório `/tmp`

137. Desenvolva um script que: mostre o nome do usuário logado com a seguinte mensagem: Seu Usuário logado é: _____ e, com o nome do usuário logado crie um diretório em /tmp

138. Desenvolva um script que: Mostre da tela a mensagem, "Olá $NomeDoUusuárioLogado", hoje é "DiaDaSemana" e já são "$HoraDoSistema", O seu computador é: "NomeDoCompudor". <tudo variável>

139. Desenvolva um script que: Crie um diretório no diretório documentos do usuário logado com o nome do próprio usuário

140. Desenvolva um script que: Mostre sua idade calculando sua data de aniversário e sua data de nascimento e mostre na tela, $SeuNome você tem $Idade anos.

141. Desenvolva um script que: Crie dentro do `/tmp/` a quantidade de arquivos que você definir na variável inicial e final, igual o touch {1..100}, só que, ao invés de pré definido como no exemplo será variável, entre como valor inicial 3 entre com o valor final 9 cirando assim arquivos do 3 ao 7 e a extensão também é variável, você escolho.

142. Desenvolva um script que: Desenvolva um script que busque qualquer usuário dentro do arquivo de usuários `passwd`, com uma mensagem, "qual usuário deseja pesquisar" e retorne apenas esse usuário cadastrado.

143. Desenvolva um script que: Desenvolva um script que busque qualquer usuário dentro do arquivo de usuários `passwd`, com uma mensagem, "qual usuário deseja pesquisar" e retorne apenas esse usuário cadastrado e em qual linha do arquivo ele se encontra.

144. Desenvolva um script que: Faça um script que faça variações com seu nome utilizando todas as vogais e envie para um arquivo de texto no `/tmp` e gerando o arquivo ele já abre onde foi gerado no gedit

145. Desenvolva um script que: Conte quantos votaram sim e quantos votaram não na PEC de acordo com os dados do site: http://g1.globo.com/politica/noticia/2016/10/saiba-como-votou-cada-deputado-no-segundo-turno-da-pec-241.html mostre o resultado da contagem em um arquivo PEC241.txt

146. Vamos trabalhar com esse word list, lista de palavras: https://raw.githubusercontent.com/paritytech/wordlist/master/res/wordlist.txt

147. Coloque a `url` acima como uma variável URL, essa $URL receberá o valor da wordlist.txt, caso queira fazer download do arquivo, fique à vontade. Então teremos URL="http:*". Faça um script que gere um arquivo que contenha apenas as palavras com começam coma letra `ab`

148. Faça o mesmo script que anterior, porém, use variável, para que o filtro da busca seja externo da palavra que você quer procurar, sendo uma variável de entrada `read` (input), com a mensagem: Digite a palavra que deseja Localizar:_

149. Desenvolva um script que: Busque a palavra ou conjunto de palavras e gere um arquivo com todas elas convertidas em maiúsculo

150. Faça um script que gere um arquivo com todos os encontros vocálicos contidos na wordlist, esse arquivo deverá ter a seguinte mensagem no final: Foram Encontros <x> palavras com encontros vocálicos e mostre a lista abaixo.

151. Desenvolva um script que: gere um arquivo de e-mails do site: https://www.fflch.usp.br/docentes apenas e-mail

152. Faça um script que conte quantos mestres e quantos doutores já no site: https://ww2.uft.edu.br/index.php/ensino/76-ensino/10908-corpo-docente

153. Faça um script que, mostre quais os cursos da UFT, na url: https://ww2.uft.edu.br/index.php/ensino/76-ensino/10908-corpo-docente, lembre-se de mostrar um valor único para cada curso, ao final exporte para um novo arquivo de texto chamado cursos.txt

```
Só estou te passando as chaves do castelo, abrir a porta é com você.
```

12 Operadores

Aqui, apresentaremos os principias operadores usados no Bash que servem para comparar dois ou mais argumentos.

12.1 Operadores de string

Serve para comparar duas expressões constituídas por strings

Operadores de strings	
Operador	Descrição
=	Verifica se as strings são iguais
!=	Verifica se não diferentes
-n	Verifica se não é vazia

-z	Verifica se é vazia

12.2 Operadores Numéricos

São usados para comparar números. Veja a tabela abaixo.

Operadores Numéricos				
Operador		Operador		Descrição
-eq	ou	=	Verifica	Se dois números são iguais
-ne		!=		Se dois números são diferentes
-gt		>		Se um é maior que o outro
-lt		<		se um é menor que o outro
-ge		>=		se é maior ou igual ao outro
-le		<=		se é menor ou igual a outro

12.3 Operadores de Arquivos.

Podem ser usados pra se obter informações sobre arquivos.

Operadores de Arquivos		
Operador		Descrição
-f	Verifica	se é arquivo regular
-d		se é um diretório
-r		se o arquivo tem permissões de leitura
-w		se o arquivo tem permissões de escrita
-x		se o arquivo tem permissões de execução
-s		se o arquivo tem um tamanho maior que zero
-u		se o bit SUID está ativo
-g		se o bit SGID está ativo
-k		se o sticky bit está ativo

12.4 Operadores lógicos

São usados para comparar expressões usando as regras da boa e velha lógica.

Operadores Lógicos				
Operador		Operador		Descrição
!	ou		Usado para	Negar a expressão lógica
-a		&&		Fazer um and lógico em duas expressões
-o		||		Fazer um or lógico em duas expressões

13 Estrutura de Controle

Assim como qualquer outra linguagem de programação, o Bash também apresenta estruturas condicionais e de repetição.

```
É parte da cura o desejo de ser curado. Sêneca
```

13.1 Estruturas condicionais – if, else e elif

As declarações condicionais são usadas nos scripts para decidir qual parte do programa deve ser executada em função de uma condição, a instrução if, avalia uma expressão lógica para tomar uma decisão, temos sua estrutura básica assim:

```
if condição
then
      instrução
      instrução
...
else
      instrução
      instrução
...
fi
```

A sintaxe é um pouco diferente da linguagem a qual deve estar acostumado, como o C por exemplo, pois fácil notar que não há necessidade de usas chaves para limitar os blocos, mesmo que cada bloco possua mais de uma instrução, para fechar o bloco do if usa-se o fi.

Aqui também é importante destacar uma coisa, o verdadeiro vale 0 e o falso vale 1, digamos que esse pequeno detalhe causa muitos erros nos scripts, dê atenção a isso para não errar, assim, quando o bash retorna 0 é porque a instrução foi bem-sucedida, diferente de zero ocorreu um erro na instrução. Abaixo um exemplo sem interatividade com o usuário, pois os valores são digitados no script

```
2   #!/bin/bash
3   variavel=3
4   if (($variavel>10))
5   then
6       echo "você digitou um numero maior que 10"
7   else
8       echo " voce digitou o numero '$variavel'"
9   fi
```

Abaixo um script com interatividade com usuário, pois os valores de entrada são esperados para iniciar o script.

```
2   #!/bin/bash
3   echo "Digite um valor para comparação"
4   read variavel
5   if (($variavel>10))
6   then
7       echo "você digitou um numero maior que 10"
8   else
9       echo " voce digitou o numero '$variavel'"
10  fi
```

Quando atendida a condição `else` o bloco correspondente é executado e os demais ignorados, veja o exemplo abaixo, copie e faça o teste modificando a hora do sistema.

```
2   #!/bin/bash
3   HORARIO=`date +%H`
4   echo "Hora local: $HORARIO - horas"
5   if [ $HORARIO -lt 12 ]
6   then
7       echo "bom dia, $LOGNAME"
8   elif [ $HORARIO -ge 18 ]
9   then
10      echo "boa noite, $LOGNAME"
11  else
12      echo "boa tarde, LOGNAME"
13  fi
14
15  # LEMBRA DOS OPERADORES #
16
17  # eq = verifica se dois numeros são iguais
18  # ne = verifica se os numeros são diferentes
19  # gt = verifica se um numero é maior que o outro
20  # lt = verifica se um numero é menor que o outro
21  # ge = verifica se um numero é maior  ou igual a outro
22  # le = verifica se um numero é menor ou igual a outro
```

13.2 Estruturas condicionais – case

A estrutura case permite selecionar uma entre várias opções de ação, baseando-se em um valor variável é usado no lugar do if, caso este tenha inúmeras condições.

```
2  #!/bin/bash
3  opcao=nao
4  case $opcao in
5      Sim | sim ) echo "Você concordou!";;
6      Nao | nao ) echo "Que pena, não concordou";;
7            *   ) echo " nem sim, nem não";;
8  esac
9
```

14 Estruturas de repetição

Usadas para repetir uma série de comandos contidos em sua declaração, assim, permitindo o script executar mais de uma vez um conjunto de comandos, ou, blocos de comandos, de acordo com uma condição ou contador.

14.1 Estrutura de repetição – for

O for aceita mais de uma forma. Vejamos no exemplo a seguir que o i recebe uma lista de valores que vem depois da declaração in, sendo um valor de casa vez, dessa forma, para cada valor da lista, o loop for será executada uma vez, vejamos:

```
2  #!/bin/bash
3  for i in 1 2 3 4 5
4  do
5      echo "$i"
6  done
```

```
2  #!/bin/bash
3  for i in {1..10}
4  do
5      mkdir "$i"
6  done
```

```
2  #!/bin/bash
3  for i in {1..10000}
4  do
5      echo "$i" >> lista.txt
6  done
```

```
Dê o poder ao homem, e descobrirá quem ele realmente é. Maquiavel
```

Consegue sentir o poder que agora está em suas mãos, claro que talvez você não venha a trabalhar com infraestrutura em si, que necessite especificamente de script em shell, mas, tenho certeza que pelo menos uma vez você terá que gerar um relatório especifico com vários filtros, o que eu desejo é que aqui você entenda que script é só um pontapé inicial, você pode usar o python para resolver muita coisa automática, o Excel ou o Calc, o R para estatística, ou qualquer outra linguagem que venha a surgir ou que já morreu, mas, o conceito é o mesmo, estrutura, laço, condicional, dentre outros, o poder é esse, é primeiro, saber que existe (conhecimento), segundo saber usar e terceiro onde usar. O mundo está

mudando e todas as áreas estão bebendo da computação para automatizar processos, qualifique-se, aproveite essa oportunidade.

14.2 Estrutura de repetição – while

Pode ser usada para executar uma série de instruções enquanto uma for verdadeira. O loop termina quando a condição especifica se tornar falsa e for verificada pelo while.

```
1  #!/bin/bash
2
3  i=1
4
5  while (($i < 10 ))
6  do
7      echo "$i"
8  i=`expr $i + 1`
9  done
```

Bem, existe várias outras estruturas, como o continue, break until, mas, até aqui já vai da pra entender a magnitude do shell script. Vamos finalizar por aqui, para que possamos fazer alguns exercícios e escolher continuar ou parar, a escolha sempre será sua. Como já disse, estou passando as chaves da porta, abri-la será única e exclusivamente por você.

```
O que te preocupa, te escraviza. John Locke
```

14.3 Exercícios Diversos

154. Para começarmos a treinar, vamos nos habituar com questões simples para passarmos as questões usuais, faça um script que calculo o fatorial de um numero variável, o script deverá ser interativo. O fatorial de um número inteiro m não negativo, é indicado por m! (lê-se "m fatorial") e é definido pela relação: $m!=m\cdot(m-1)\cdot(m-2)\cdot(m-3)...3\cdot2\cdot1$, para $m \geq 2$.

O comando ping serve para verificar se uma máquina está ativa na rede ou não, sua forma mais usual é `ping <nome_do_host>` ou `ping <ip_do_host>`, assim, se você der o comando ping google.com por exemplo o ping retornará se o site está ativo ou não, simples assim, claro, não serve apenas para sites, todos os `hosts` de uma rede também são verificados com ping, desde que tenham endereço `IP`.

155. Faça um script que ao executa-lo verifique um endereço ip e retorne se ele está ON | OFF, assim se o endereço a sua escolha estiver ON, será gravado em um log no `/home` do usuário, a data do ping, a hora que foi feito, e a mensagem host on ou off.

156. Um só host fica fácil não, agora faça um script que verifique 10 hosts diferentes, ou seja, 10 ips diferentes, para cada um site deverá ser dado apenas 5 disparos de ping, se o endereço estiver ON/OFF retornar a mensagem host ON ou OFF dentro do log, com data, hora que foi realizado o ping e seu endereço. #para dar apenas 5 disparos no ping use a opção -w timeout

157. Vamos verificar como está a oscilação da nossa rede com o ping e fazer um relatório, tudo via script, dispare o comando ping em algum ip ou site da rede, deixe-o rodando até que o pare manualmente, gere um log no `/home` com esse relatório, mostrando apenas o tempo sem o *ms* (milissegundos)

158. Crie um script que verifica se existe o diretório chamado backup em /tmp, se não existir, ele vai criar o diretório e compactar todo o conteúdo do perfil do usuário e colocar compactador no diretório criado, se existir ele vai apenas compactar e colocar o perfil do usuário no diretório, o arquivo de backup deverá ser nomeado com o nome bk_%dia%mês%hora%minuto%Segundo <variável>.

159. Crie 2 usuários no linux, depois entre em terminais diferentes com usuários diferentes para realização dos testes, primeiro faça um script que verificar se determinado usuário está ou não logado no sistema, se sim, envie a mensagem para um arquivo de log dizendo, o usuário está logado deste: %Horas, ou se não, se o usuário não estiver logado, colocar no log, o usuário não está logado.

160. Faça um script para criação de usuários. Primeiro o script deverá ler o /etc/passwd e verificar se o usuário existe, se existir ele retorna a mensagem, usuário já cadastrado, se não, cadastra o usuário e em seguida pede a senha, esse usuário deverá conseguir fazer entrar no sistema, porém, no primeiro login ele deverá alterar a senha padrão.

161. Faça um script que liste todos os usuários que acessam o sistema e que tenham um diretório dentro do /home e envie apenas seus nomes para /tmp/<utilizar forma padrão de geração de log>

162. Lembra quando criou um comando para criar o dia%mês%ano como diretório em um só comando ficando dia dentro do mês e mês dentro do ano. Agora, faça um script que, ao ser executado, verifique se existe o diretório

163. Faça um script que, ao entrar com seu nome e data de nascimento diga sua idade

164. Esse é clássico, faça um script que dê bom dia, tarde, ou noite dependendo do horário

```
Chegamos ao início de um novo começo. Espero que tenha gostado.
Wilmar Borges Leal Junior
```

15 Referências

BOANAN, A. R. **Linux: Fundamentos, Prática & Certificação LPI**. Rido de Janeiro: Alta Book, 2010.

CCM. **Sed - Introdução ao SED**. Disponível em: <https://br.ccm.net/faq/8760-sed-introducao-ao-sed-part-i>. Acesso em: 30 jan. 2019.

COSTA, R. F. **6 recursos interessantes oferecidos pela ferramenta cURLLinux Descomplicado**, 19 abr. 2016. Disponível em:

<https://www.linuxdescomplicado.com.br/2016/04/6-recursos-interessantes-oferecidos-pela-ferramenta-curl.html>. Acesso em: 29 jan. 2019

EDUARDO MORIMOTO, C. **Entendendo e dominando o Linux**. 3. ed. São Paulo: [s.n.].

HAEDER, A.; SCHNEITER, A.; PESSANHA, B. G.; STANGER, J. **Certificação Linux LPI - Rápido e Prático**. 3. ed. [s.l.] Alta Book, 2012.

JARGAS, A. **Canivete Suíço do Shell (Bash)**. Disponível em: <https://aurelio.net/shell/canivete/>. Acesso em: 30 jan. 2019.

LUDOLF, D. **Desbravando o comando touch**. Disponível em: <https://sejalivre.org/desbravando-o-comando-touch/>. Acesso em: 25 jan. 2019.

OLIVEIRA, M. **20 exemplos do comando find**. Disponível em: </2015/07/20-exemplos-do-comando-find.html>. Acesso em: 30 jan. 2019.

PEDRODELFINO. **10 Formas de Utilizar o Comando find do LINUX**. Disponível em: <https://e-tinet.com/linux/comando-find-linux/>. Acesso em: 27 jan. 2019.

RAFAEL H. **Como utilizar comandos Linux Find e LocateHostinger Tutoriais**, 23 maio 2017. Disponível em: <https://www.hostinger.com.br/tutoriais/comandos-linux-find-e-locate/>. Acesso em: 27 jan. 2019

REIS, F. DOS. **Inicialização, init, runlevels, init.d, inittab e telinit - LinuxBóson Treinamentos em Tecnologia**, 2013. Disponível em: <http://www.bosontreinamentos.com.br/linux/16-inicializacao-init-runlevels-init-d-inittab-e-telinit-linux/>. Acesso em: 25 jan. 2019

SILVA, J. C. F. DA. **O comando LS de A a Z**. Disponível em: <https://www.vivaolinux.com.br/artigo/O-comando-LS-de-A-a-Z>. Acesso em: 25 jan. 2019.

STACKEXCHANGE. **Difference between help, info and man command**. Disponível em: <https://unix.stackexchange.com/questions/19451/difference-between-help-info-and-man-command>. Acesso em: 24 jan. 2019.

UNIRIOTEC. **redirecionadores**. Disponível em: <http://www.uniriotec.br/~morganna/guia/redirecionadores.html>. Acesso em: 25 jan. 2019.

WIKIPÉDIA. **cat (Unix)**, 8 ago. 2017. (Nota técnica).