Ao fim dessa parte, você será capaz de:

* Utilizar o comando ***find***;
* Utilizar os operadores > (maior), >> (maior maior) e | (pipe) no terminal;
* Alterar as permissões de arquivos e diretórios;
* Verificar o status e enviar sinais a processos UNIX;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Vídeo 01

**Operadores > (maior), >> (maior maior) e | (pipe)**

**MANIPULANDO CONTEÚDOS DE UM ARQUIVO**

O **COMANDO *cat*** consegue receber valores do usuário como inputs.

**OBS:** O atalho ***Ctrl + D*** fecha a execução do comando cat.

O símbolo **>** indica que iremos **atribuir valores** a um determinado arquivo.

Ex.: *cat* ***>*** *cores.txt*

Ao dar o *ENTER,* o terminal vai esperar que eu digite os valores. Então digito:

*amarelo*

*vermelho*

*azul*

E ao fim damos um *Ctrl + D*.

Assim, as três cores irão ser armazenadas dentro do *arquivo.txt*.

Pode dar um *cat cores.txt* para confirmar.

**ADICIONANDO VALORES DENTRO DO ARQUIVO**

Usar **>>**

Ex.: *cat* ***>>*** *cores.txt*

Isso permite que o usuário digite novos valores.

Ex.:

*marrom*

*preto*

*laranja*

*(Ctrl + D)*

Assim, as novas cores serão adicionadas à lista, abaixo de "*azul*".

**JUNTAR VALORES DE DOIS ARQUIVOS**

Ex1.: *cat animais >> animais\_2*

Copia o conteúdo de um para o outro.

Ex2.: *cat cores.txt frutas.txt > tudo.txt*

Armazena o conteúdo dos dois arquivos será unido em t*udo.txt.*

Ex3.: *cat cores.txt nomes.txt | sort cores\_nomes.txt*

Armazena o conteúdo de forma alfabética no novo arquivo.

**ORDENAR VALORES EM ORDEM ALFABÉTICA**

Comando ***sort***

Ex.: *sort < cores.txt*

Exibe as cores em ordem alfabética.

OBS: O sinal de menor (<) indica que o terminal irá **apenas exibir** o conteúdo do arquivo **sem alterar** os valores.

**ORDENAR VALORES E ARMAZENÁ-LOS EM OUTRO ARQUIVO**

*sort < cores.txt > coresOrdenadas.txt*

**SABER QUAIS USUÁRIOS ESTÃO USANDO O SISTEMA E QUAIS TEM PERMISSÃO?**

Comando ***who***

Ex.: *who*

Sim, só o comando e mais nada. Isso já exibe os usuários ativos.

**USAR MAIS DE UM COMANDO EM UMA LINHA**

Uso do | (pipe)

O **|** canaliza a saída do primeiro comando para a entrada do segundo.

Ex1.: *who | sort*

Isso exibe os usuários em forma alfabética.

Ex2.: *who | wc -l*

Isso irá contar quantos usuários há no sistema. No meu caso, a resposta é 1.

Ex3.: Considere que queremos uma lista com todos os nomes de um arquivo que **contenham a letra "F"** e que nosso resultado esteja em **ordem alfabética.**

Para isso podemos realizar o comando abaixo:

*grep F nomes.txt | sort*

Vídeo 02

**Controle de acesso aos arquivos no Unix.**

Ao digitar **o Comando *ll*** (LL) o terminal exibe o seguinte:

*drwxrwxr-x 2 vitor vitor 4096 ago 12 11:15 ./*

*drwxr-xr-x 5 vitor vitor 4096 ago 12 11:06 ../*

*-rw-rw-r-- 1 vitor vitor 24 ago 12 11:16 nomes.txt*

**Comando *ls -l*** | Mostra os diretórios e também outras infos que o comando ls não exibe.

E o que são essas informações exibidas?

Elas indicam **o tipo de acesso que cada usuário tem.** O Unix suporta um grande número de usuários diferentes.

Para entender melhor:

* **drwxrwxr-x:** Permissões (3 primeiras letras para usuário principal, 3 seguintes para grupo e 3 últimas para os outros).
* **2**: Links
* **vitor**: Proprietário
* **vitor**: Grupo
* **4096**: Tamanho
* **ago** **12** **11**:**15**: Data e Hora

**E O QUE SIGNIFICA CADA UM DESSES?**

* **Permissões** => É possível verificar o tipo do item e nível de permissão para Leitura, Escrita e Execução de item ou diretório;
* **Links** => Número de ligações que o item possui, no caso do diretório, número de subdiretórios que possui;
* **Proprietário** => Quem é a pessoa dona, quem criou. É o diretório padrão da pessoa usuária, o home;
* **Grupo** => Grupo ao qual pertence o item ou diretório. Utilizado para dar permissões à outras pessoas;
* **Tamanho** => Em Bytes;
* **Data** **e** **Hora** => Momento em que foi criado ou última modificação;
* **Nome** => Nome do item ou diretório;

**Entendendo o código das Permissões**

* **D** = indica um diretório;
* **-** = arquivo comum (não é diretório).
* **R** = pode ler o arquivo (read);
* **W** = pode editar o conteúdo (write);
* **X** = pode acessar pasta por comando cd (execute);

**Comando *chmod***

Usado para definir qual o tipo de permissão.

Ex.: *chmod u* | Dá permissão ao usuário.

Ex.: *chmod g* | Dá permissão ao grupo.

E na sequência, adicionamos o referido caractere para indicar o **TIPO** DE PERMISSÃO

Ex.: *chmod u-* | Retira permissões.

Ex.: *chmod u+* | Concede permissões

Por fim, adicionamos **a referida parte** a ser alterada.

Ex1.: ***chmod u-rw arquivo.txt* |** Retira a possibilidade do usuário ler e escrever no arquivo.txt.

Ex2.: ***chmod u+rw arquivo.txt*** | Concede as permissões de ler e escrever ao usuário.

Você aprendeu no vídeo como alterar as permissões no modo literal (caracteres), **mas também é possível usar o modo octal.** Para compreendermos esta outra forma, precisamos entender que eles são administrados por meio de **valores**, como descrito abaixo:

* Leitura r - **4**
* Escrita w - **2**
* Execução x - **1**

Esses valores são permissões com base em **bits** de **ligados = 1** e **desligados = 0**,

* rwx = 111 ( 7 | Acesso Total )
* r-- = 100 ( 4 | Somente Leitura )
* -w- = 010 ( 2 | Somente Escrita )
* --x = 001 ( 1 | Somente Execução )
* rw- = 110 ( 6 | Somente Leitura e Escrita )
* r-x = 101 ( 5 | Somente Leitura e Execução )
* -wx = 011 ( 3 | Somente Escrita e Execução )
* --- = 000 ( 0 | Todos Acessos Negados )

A sintaxe para realizar esta alteração continua a mesma que no modo literal, sendo: ***chmod 766 [arquivo ou diretório]***

*Vídeo 03*

Processos dentro do Unix

Para ver a lista de processos

**Comando *ps |*** E o terminal imprime:

*PID TTY TIME CMD*

*7223 pts/0 00:00:00 bash*

*8879 pts/0 00:00:00 ps*

Exemplos de processo

**Comando *sleep*** | Para os processos por um determinado momento

Ex1.: *sleep 5* | fará o processo ser interrompido por 5 segundo.

**Criar processos e deixá-los em background.**

Ex2.: *sleep 10 &* | o & deixa o processo em background por 10 segundos. Se der um ps depois disso, verá que o referido processo está em sleep.

OBS: O comando **Ctrl + Z suspende/pausa** o trabalho em execução.

OBS2.: Já **Ctrl + C elimina** o trabalho em execução.

Todos os processos em background vão para uma lista chamada jobs

**Comando *jobs*** | Exibe todos os processos executados e qual o status.

**Executando outra vez um processo suspenso**

Comando *fg* | foreground

Ex.1: *fg %1* | sendo [1] o número do referido processo a ser executado.

**Para encerrar um processo**

**Comando *kill***

Opção 1: Comando kill para o número

Ex.: *kill %1* | E o processo não existe mais.

Opção 2: kill pelo PID.

Ex.: *kill 7657*

BAIXAR AQUIVOS

COMANDO *wget*

*Ex.: wget link.com.br/12312 &*

O & garante que a instalação será em background

Conteúdo Final

Agora vamos aprender sobre os comandos:

* ***find***
* ***history***
* ***echo .***

**COMANDO *find*** | O ***find*** é um comando para **pesquisar** em diretórios por arquivos ou outras pastas, de acordo com os **parâmetros** passados a ele.

Esses parâmetros podem ser:

* *name*
* *date*
* *size*
* *type*

Caso nenhum atributo seja passado, ele pesquisará tudo que estiver dentro do diretório atual.

**EXEMPLOS DE USO DO *find***

# Para listar todos os arquivos que terminam em .txt

***find . -name "\*.txt"***

# Para localizar todos os diretórios

***find . -type d***

# Para localizar todos os arquivos

***find . -type f***

# Localizar tanto arquivos quanto diretórios que comecem por algum trecho

***find ./teste -name exemplo\****

Resultado:

*./teste/exemplo.txt*

*./teste/exemplo*

# Localizar somente arquivos que comecem por algum trecho

***find ./teste -type f -name "exemplo\*"***

Resultado:

*./teste/exemplo.txt*

# Localizar somente diretórios que **comecem** por algum trecho

***find ./teste -type d -name "exemplo\*"***

Resultado:

*./teste/exemplo*

# Localizar somente diretórios que **terminem** por algum trecho

***find ./teste -type d -name "\*exemplo"***

O que muda é a posição do asterisco.

**COMANDO *history*** | O ***history*** é um comando que mostra o **histórico de comandos** que você executou no terminal.

A quantidade ou o tamanho desse \_"histórico"\_ podem ser configurados para um número arbitrário de comandos ou para ver todo o histórico.

# Mostra o histórico de comandos

*history*

# Pegar os últimos 10 comandos

*history | tail*

*#* Armazenar histórico em um arquivo

*history | tail -5 >> historico.txt*

Cria um arquivo que contém os últimos 5 processos do histórico.

**COMANDO *echo*** | O ***echo*** é um comando utilizado em scripts ou no terminal para exibir mensagens na tela ou em um arquivo. Funciona como um *printf* em C.

Ex1.: *echo "Este é um teste"*

Output:

*Este é um teste*

E pode ser usado para **colocar textos dentro de arquivos**.

Ex2.: *echo "Este é mais um teste" > teste.txt*

Depois...

*cat teste.txt*

Output:

*Este é mais um teste*

Podemos usar aspas simples também.

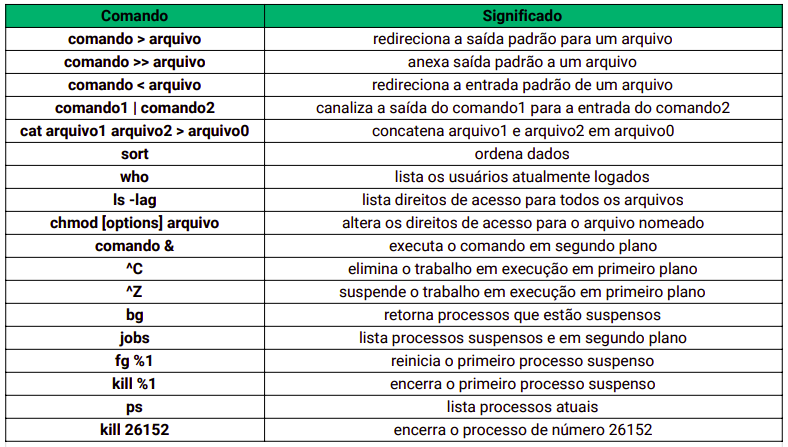
Podemos usar o echo para inserir conteúdo dentro de um arquivo também.

Ex3.: *echo “cachorro” > animais.txt*

PORÉM

Se adicionar uma nova palavra, como “gato”, ela substitui a existente. Para que isso não ocorra, usamos **>>**.

LISTA DE COMANDOS VISTOS HOJE

****

Recapitulando tudo…

Agora que você passou um bom tempo estudando sobre o Unix e seus comandos, sabemos que você já entende que:

* O Unix é um sistema operacional que inspirou muitos outros sistemas, como MacOS, Linux, Solaris etc.;
* O Unix é composto basicamente pelo Shell, Kernel e os programas;
* A estrutura de diretórios do Unix é organizada no formato de uma árvore invertida, de forma hierárquica, iniciando pelo root (raiz);
* Tudo no Unix é um arquivo ou um processo;
* O terminal é seu principal aliado para ter produtividade em ambiente Unix;

Depois de entender como abrir e utilizar o terminal, você aprendeu na prática como utilizar os comandos Bash. Vamos relembrar alguns dos comandos que você aprendeu a usar.

Um bom exercício é, mentalmente, tentar descrever o que cada comando faz.

*ls , mkdir , cd , pwd , cp , mv , rm , rmdir , clear , cat , less , head , tail , grep , wc , > , >> , | , sort , who , man , whatis , apropos , \* , ? , chmod , ps , bg , kill , find , history .*