Capítulo 2 Comandos GNU/Linux

2.1. Objetivos

- Conhecer os comandos básicos do sistema

Uma coisa importante antes de mexer no Linux é que ele é Case Sensitive. Ou seja, ele diferencia letras maiúsculas de letras minúsculas. Então, se eu tenho um arquivo chamado:

teste.txt

Ele é diferente do arquivo que chama:

Teste.txt

O Linux tem 2 tipos de comandos, os internos e os externos.

Os comandos internos são comandos que estão dentro do próprio Shell, que é o interpretador de comandos. São os comandos essenciais do sistema. A grande vantagem dos comandos internos é a velocidade, pois não precisam ser procurados no disco rígido, nem criam processos. Exemplo de comando interno:

cd, alias

Comandos externos estão localizados em diretórios específicos no disco rígido, como /bin e /sbin. Para um comando externo funcionar o Linux precisa consultar o disco rígido. A maioria dos comandos Linux é externa. Exemplo de comando externo:

ls, cp, rm, mv, mkdir, rmdir.

Mas como saber se um comando é interno ou externo afinal?

Basta procurá-lo no disco rígido!

Dica para descobrir a localização de um comando no disco rígido:

```
$ which comando
```

Exemplo:

```
$ which ls
/bin/ls
```

Se um usuário comum tentar descobrir a localização de um comando do superusuário (root), ele não vai descobrir, pois o comando which faz sua busca baseado em uma variável de ambiente chamada PATH.

Prática: Crie um diretório chamado "teste" em seu /home. Por exemplo /home/maria

```
$ cd /home/maria
$ pwd
/home/maria
$ mkdir teste
```

É possível criar um diretório se estiver em outro diretório, mas sera preciso indicar o caminho completo onde ficará esse novo diretório. Exemplo:

```
$ pwd
/tmp
$ mkdir /home/maria/teste
```

Dica: Pode usar a tecla TAB do teclado para auto-completar um comando, assim não precisa ter que fica digitando ele todo.

```
$ mkd<TAB>
```

Dica: Se esta no diretório /home/maria e vai para o diretório /tmp com:

```
$ cd /tmp
```

Se digitar o comando abaixo volta para o diretório que estava antes:

```
$ cd -
```

Para ir direto para o home do usuário corrente sem colocar o caminho completo, faça:

```
$ cd ~
```

A vantagem é que não sera preciso mais digitar todo o caminho /home/maria para ir até seu diretório pessoal.

Crie os arquivos no diretório teste, cada um chamado assim: arq1, arq2, arq3, sessao1, sessao2 sessao3 sapo satisfação:

```
$ cd /home/maria/teste
$ pwd
/home/maria/teste
$ touch arq1
$ touch arq2
$ touch arq3
$ touch sessao1 sessao2 sessao3 sapo satisfacao
(Posso criar tudo numa linha de comando só)
```

Para alterar a data e hora que esse diretório foi criado, também usa-se o comando touch. Por exemplo:

$ touch aniversario
$ touch -t 200809161940 aniversario
A opção -t é para escolher o tempo que vamos alterar
200809161940 tem o seguinte formato AAAAMMDDHHHH
A - ano, M - mes, D - dia, H - hora

Como é possível criar um arquivo oculto no linux?

$ touch .arq4

E como pode visualizá-lo? Será que o comando ls sozinho consegue listar arquivos ocultos? É necessário passar algum parâmetro?

$ ls -a
.arq4

Ok, se caso mudar de ideia e não for preciso mais deixar o arquivo oculto, use:

$ mv .arq4 arq4 (basta renomear o arquivo)

Para remover o arquivo:

$ rm arq4

Para copiar um diretório e seu conteúdo precisa usar o parâmetro R com comando cp:

```
$ cd /home/maria
$ pwd
/home/maria
$ cp -R teste /tmp
$ cd /tmp
$ ls
teste
$ cd teste
$ ls
arq1 arq2 arq3 sessao1 sessao2 sessao3 sapo satisfacao
```

O que são metacacteres?

São caracteres que possuem significado especial para o Shell. Os principais metacaracteres são:

```
*
?
[]  (listas)
[a-z]  (listas)
```

```
$ cd /home/maria/teste
$ ls
arq1 arq2 arq3 sessao1 sessao2 sessao3 sapo satisfacao

$ ls arq*
arq1 arq2 arq3

$ ls sess*
sessao1 sessao2 sessao3

$ ls *1
arq1 sessao1

Lista:
$ ls sessao[13]
sessao1 sessao3

$ ls sessao[1-3]
sessao1 sessao2 sessao3

[1-3] = [123]
```

Prática:

Lembre que os arquivos que temos no diretório são:
arq1 arq2 arq3 sessao1 sessao2 sessao3 sapo satisfacao

Liste apenas os arquivos que começam com a letra "s", a segunda letra pode ser de "a" até "e" e tem que terminar com a letra "o".

```
$ ls s[a-e]*o
```

Crie um diretório chamado arquivos em /tmp

```
$ cd /tmp
$ mkdir arquivos
$ cd arquivos

$ touch não nao
$ ls n[ãa]o
não nao

$ touch teste Teste
$ ls [Tt]este
Teste teste

$ touch barata batata
$ ls ba?ata
barata batata
```

•Asterisco corresponde a todos os caracteres;

•O ponto de interrogação corresponde a um único caracter;

O que o comando ls faz quando usa junto à ele o parâmetro -l? Mostra detalhes de um arquivo. Exemplo:

```
$ cd /home/maria
$ ls -l
drwxr-xr-x 2 linux linux 1024 2007-07-11 15:37 teste
-rw-r--r-- 1 linux linux 1569 2007-07-09 11:09 instrucoes_iniciais
-rw-r--r-- 1 linux linux 112669 2007-07-09 12:52 intro_firewall.pdf
```

O que são esses detalhes apresentados pelo ls -l?

Esse comando apresenta detalhes como tipo de um arquivo, permissões, dono e grupo de um arquivo, tamanho, data e hora de criação/modificação e nome. O primeiro caractere da saída indica o tipo de um arquivo:

- arquivo texto comum
d diretório
l link

Para remover o diretório teste:

```
$ cd /home/maria
$ pwd
/home/maria
$ rm -rf teste
```

Dica: Você pode usar vários comandos numa linha só ao invés de digitar ENTER a cada comando.

Exemplo:

$ cd /home/maria; mkdir guarda_roupa; cd guarda_roupa; touch camiseta bermuda;

O ponto-e-vírgula nesse caso funciona como ENTER.

Pode-se usar também **&&** entre os comandos em uma linha, mas nesse caso, é necessário que o comando anterior tenha sido executado com sucesso para que os demais também sejam executados.

$ cd /home/maria && mkdir guarda_roupa && cd guarda_roupa && touch camiseta bermuda

Digamos que não tenha um usuário “maria” no computador. Ou seja, não existe uma pasta maria dentro do diretório /home. Entre no home do usuário

```
$ cd ~
```

E depois execute esses comandos:

```
$ cd /home/maria; mkdir guarda_roupa; cd guarda_roupa; touch camiseta bermuda;
```

Aparecerá uma mensagem dizendo que não existe o diretório maria.

```
bash: cd: /home/maria: Arquivo ou diretório inexistente
```

De fato, ele reclama pois não conseguiu executar o primeiro comando. Porém faça um pwd e verifique que ele executou os demais comandos da linhas:

```
$pwd
/home/tux/guarda_roupa
```

Ok, vai ser necessário voltar um diretório e apagar essa pasta que foi criada.

```
$ cd ..
$ rm -r guarda_roupa
```

Agora tente mais uma vez, porém agora usando && ao invés de ;

O mesmo erro é mostrado:

```
bash: cd: /home/maria: Arquivo ou diretório inexistente
```

Porém agora, os demais comandos não são executados. Assim que foi encontrado um erro, a linha de comandos foi interrompida:

```
$pwd
/home/tux/
```

Existe os comandos para localizar arquivos:

Por exemplo, o **comando locate** e o **comando find**

O comando **find** é um comando que procura arquivos pelo nome ou pelas suas características. Inicie uma pesquisa em um arquivo, mostrando um diretório específico. Por exemplo:

```
$ find /etc -name passwd
```

Explicando o comando:

Esta sendo mostrado que ele vai procurar em todo o diretório /etc/ arquivos que tem no nome passwd. Então ele irá mostrar todos os arquivos que tem nome ou parte dele com a palavra passwd. Outro exemplo:

```
$ find / -name *.sh
```

Esta indicando que ele irá buscar a partir do / (barra), ou seja, no sistema inteiro qualquer arquivo * (asterisco) que tem a extensão.sh

Como isso faz parte do nome dos arquivos, usa-se a opção -name também. No caso do comandos find, quando usado a expressão -name ele procura por nomes de arquivos que foi indicado nos comandos.

Mas se souber o tamanho do arquivo, só que sabe o nome dele, faça a seguinte busca:

```
$ find . -size -1024k
```

Com esse comando, esta indicando que ele tem que procurar (find), dentro do diretório que o usuario se encontra (.) arquivos e diretórios com tamanho (size) menor que 1024k

Como pode ser feito para pesquisar aquivos com mais de 1024k em todo o meu sistema?

```
$ find / -size +1024k
```

*Outro comando para localização, é o comando **locate**.*

Esse comando é muito utilizado e tem como finalidade de localizar arquivos e/ou diretórios no sistema também. O comando locate faz a sua busca em uma base de dados que o próprio usuário tem que criar. Geralmente só o root pode executar o comando para criar a base de dados.

Mas qual é o comando para criar essa base de dados?

```
# updatedb
```

Esse comando irá demorar um pouco, porque ele vai ler todo o sistema e atualizar a base de dados. Quanto mais coisas estiver na máquina, mais esse comando vai demorar. Depois que atualizar a base de dados, pode-se fazer a busca por um arquivo qualquer. Por exemplo:

```
$ locate passwd
```

Saída do comando:

Ele mostrará todos os locais no sistema que tem algum arquivo ou diretório chamado passwd. A documentação padrão do Linux é chamada de man pages. Dentro dessa documentação, estão todos os comandos do sistema operacional.

Essas informações são instaladas automaticamente quando esta instalando o sistema operacional. O man é um dos recursos mais importantes, porque ele tem as informações mais completas de um comando.

Ele explica para que aquele comando serve, mostra os parâmetros e explica para que esses parâmetros servem. É uma das maneiras mais utilizadas para pesquisar comandos.

O comando man apresenta todos os tópicos do manual do Linux. Além do que ele está presente em qualquer Unix, e os seus comandos são praticamente os mesmos em todos os sistemas. O man é dividido em 8 sub-seções. Elas são:

1 - comandos de usuários - comandos que podem ser executados a partir de um shell
2 - chamadas do sistema - funções executadas pelo kernel
3 - bibliotecas de funções - funções da biblioteca libc
4 - formato de arquivo especiais - drivers e hardware
5 - arquivos de configuração - formato de arquivos e convenções
6 - jogos e demonstrações
7 - pacotes de macro e convenções - sistema de arquivos, protocolos de rede, códigos ASCII etc
8 - comandos de administração do sistema - comandos que o root pode executar

Exemplos:

```
$ man ls
```

Para sair do man, aperte q, de quit.

Como eu faria para saber informações sobre o comando passwd?

$ man 1 passwd

E se é preciso saber informações sobre o arquivo de configuração passwd:, como ficaria o comando?

$ man 5 passwd

Para saber mais sobre expressões regulares:

$ man regex

Um link interessante para olhar depois:

http://www.linuxmanpages.com/

Tem também o comando **apropos**. O apropos trabalha com um banco de dados, buscando uma descrição do comando.

Ele é muito usado em situações que você quer executar um comando, mas não lembra o nome ou a funcionalidade desse comando.

Imagine que é preciso ver quais os compiladores que existe na minha máquina, porque eu não lembro o nome do compilador que eu posso usar para rodar os meus programas em C.

Como é que poderia tentar procurar por esse compilador?

 # apropos compiler

Saída do comando:

```
tic (1) - the terminfo entry-description compiler
xsubpp (1) - compiler to convert Perl XS code into C code
zic (8) - time zone compiler
```

Se fizer um teste com a palavra "copy", quantas opção ele trás?

```
# apropos copy
```

Depois, existe também o comando **whatis**. Esse comando apresenta uma breve descrição sobre o comando que usar e não lembra se é ele mesmo.

```
$ whatis ls
$ whatis touch
$ whatis mkdir
```

O comando **whatis** também faz a busca da descrição dos comandos em um banco de dados. E o comando whatis não tem páginas de manual. Para abrir a ajuda de um comando, usamos a seguinte sintaxe:

```
$ ls --help
$ mkdir --help
$ cat--help
```