**SISTEMAS OPERACIONAIS – 2024**

**GUIA RÁPIDO DE GERENCIMENTO DE ARQUIVOS NO LINUX – PARTE 1**

Prof. Ana Paula Costacurta

**INTRODUÇÃO**

Gerenciar arquivos e pastas no Linux usando a linha de comando é importante. Um arquivo é um conjunto de dados com um nome e algumas características, como a data em que foi modificado. Por exemplo, quando você transfere fotos do seu telefone para o computador e dá a elas nomes significativos, cria arquivos de imagem. Diretórios são como pastas para organizar arquivos. Pense neles como gavetas dentro de gavetas, onde você pode colocar mais gavetas.

Usar a linha de comando no Linux é a maneira mais eficiente de lidar com arquivos. As ferramentas disponíveis tornam as tarefas mais rápidas do que usar um programa visual. Alguns comandos importantes são: ls (listar), mv (mover), cp (copiar), pwd (mostrar diretório atual), find (procurar), touch (criar ou atualizar), rm (remover), rmdir (remover diretório vazio), echo (mostrar texto) e mkdir (criar diretório). Esses comandos ajudam a organizar e gerenciar arquivos e pastas de forma simples e direta.

**CRIANDO DIRETÓRIOS**

Para criar diretórios, usamos o comando **mkdir**. Vamos ver como criar um novo diretório dentro do nosso diretório pessoal:

Primeiro, vamos para o nosso diretório pessoal usando o comando abaixo:

$ cd ~

Agora, vamos supor que estamos /home/codespace, para verificar em qual diretório estamos utilizamos o comando abaixo:

$ pwd

/home/codespace

Para listamos o conteúdo desse diretório, suponha que temos os diretórios java nvm, usamos o comando abaixo:

$ ls

java nvm

Na sequência, vamos criar um diretório chamado “Aula\_SO\_2024” dentro da pasta workspaces do raiz /, utilizando o comando abaixo:

$ ls /

bin boot dev etc go home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var vscode workspaces

$ mkdir /workspaces/Aula\_SO\_2024

$ cd /workspaces

$ ls

Aula\_SO\_2024

$ cd Aula\_SO\_2024

$ pwd

/workspaces/Aula\_SO\_2024

Entramos nesse diretório com cd Aula\_SO\_2024 e verificamos novamente com pwd, que agora estamos em /workspaces/Aula\_SO\_2024.

|  |
| --- |
| **IMPORTANTE:** |
| O sistema de arquivos do Linux diferencia maiúsculas de minúsculas nos nomes de arquivos e diretórios, ao contrário do Microsoft Windows. Isso significa que nomes como /etc/ e /ETC/ se referem a diretórios distintos. Experimente os comandos a seguir:  $ cd /  $ ls  bin boot dev etc go home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var vscode workspaces  $ cd ETC  bash: cd: ETC: No such file or directory  $ pwd  /  $ cd etc  $ pwd  /etc |

Durante esta lição, todos os comandos serão executados neste diretório ou em seus subdiretórios. Para voltar facilmente ao diretório da lição de qualquer lugar no sistema de arquivos, usamos o comando **cd /workspaces/Aula\_SO\_2024/**.

|  |
| --- |
| **DICA:** |
| Ao listar o conteúdo de um diretório com o comando ls, as opções -la são usadas para mostrar todos os arquivos e diretórios, incluindo os ocultos, e exibir detalhes sobre eles em um formato longo. Aqui está o significado de cada opção:   * -l indica que queremos uma saída em formato longo, o que significa que serão exibidos detalhes sobre cada arquivo ou diretório, como permissões, número de links, proprietário, grupo, tamanho e data de modificação. * -a indica que queremos mostrar todos os arquivos, incluindo os ocultos, que começam com um ponto (.) no nome. |

Agora que dominamos a criação de diretórios, é hora de dar o próximo passo e aprender a criar arquivos.

**CRIANDO ARQUIVOS**

Vamos falar sobre como criar arquivos. Geralmente, os programas criam arquivos para armazenar dados. Podemos criar um arquivo vazio usando o comando **touch**. Se usarmos o **touch** em um arquivo que já existe, ele não mudará o conteúdo, apenas atualizará o horário da última modificação.

Podemos criar alguns arquivos usando o comando touch:

$ cd /workspaces/Aula\_SO\_2024

$ touch exercicio1

$ ls

exercicio1

Como o **touch** cria arquivos vazios, não veremos nada, quando executar o comando **cat**:

$ cat exercicio1

Assim podemos usar o “**echo** >” para criar um arquivos de texto simples:

$ echo Hello World > exercicio1

$ cat exercicio1

Hello World

O comando **echo** mostra texto na linha de comando. O símbolo > instrui o sistema a escrever a saída de um comando no arquivo especificado, em vez de mostrá-la no terminal. Isso significa que o texto "Hello World" é escrito no arquivo exercicio1. É importante ter cuidado com o >, pois se já houver um arquivo com o mesmo nome, ele será sobrescrito.

**RENOMEANDO ARQUIVOS**

O comando **mv** serve para mover e renomear arquivos. Para isso crie alguns arquivos para praticar. Neste momento, você já deve estar familiarizado com estes comandos:

$ cd /workspaces/Aula\_SO\_2024

$ mkdir exercicio2

$ touch exer11 exer22

$ echo texto3 > texto3

$ cat texto3

$ echo texto40 > texto4

$ cat texto4

$ ls

exer11 exer22 texto3 texto4

Suponha que o nome do arquivo exer22 está com um erro de digitação; o correto seria exercicio2. Para corrigir o nome vamos usar o comando **mv.** Ao renomear um arquivo, o primeiro argumento é o nome atual, o segundo é o novo nome:

$ mv exer22 exercicio2

$ ls

exer11 exercicio2 texto3 texto4

Tenha cuidado ao usar o comando mv. Se o novo nome já estiver atribuído a um arquivo existente, ele será sobrescrito. Use a opção -i para que mv avise caso você esteja prestes a sobrescrever um arquivo existente. Utilize y se desejar substituir.

$ mv exer11 exercicio2

mv: overwrite 'exercicio2'? y

**MOVENDO ARQUIVOS**

Os arquivos são movidos de um diretório para outro com o comando mv.

$ cd /workspaces/Aula\_SO\_2024/exercicio2

$ mkdir d1 d2

$ ls

d1 d2 exer22 exercicio2 texto3 texto4

Vamos mover exercicio2 para d1:

$ mv exercicio2 d1

$ ls

d1 d2 exer22 texto3 texto4

$ cd d1

$ ls

exercicio2

Também podemos usar o mv para mover e renomear diretórios. Renomeie d2 como dir2:

$ mv d2 dir1

$ ls

d1 dir2 exer22 texto3 texto4

**EXCLUINDO ARQUIVOS E DIRETÓRIOS**

**‘** O comando **rm** serve para excluir arquivos e diretórios, ao passo que o comando **rmdir** pode excluir apenas diretórios. Vamos limpar o diretório d1 excluindo exercicio2:

$ cd /workspaces/Aula\_SO\_2024/exercicio2

$ ls

d1 dir2 exer22 texto3 texto4

$ rm exer22

$ rmdir dir2

$ ls

d1 texto3 texto4

|  |
| --- |
| DICA |
| o comando rm tem uma opção -i para exibir um aviso antes de executar qualquer ação. |

**COPIANDO ARQUIVOS E DIRETÓRIOS**

O comando cp é usado para copiar arquivos e diretórios.

$ ls

d1 texto3 texto4

$ cp texto3 d1

Se o último argumento for um diretório, o cp criará uma cópia dos argumentos anteriores dentro do diretório. Como no caso de mv, vários arquivos podem ser especificados ao mesmo tempo, desde que o destino seja um diretório.

Quando os dois operandos de cp são arquivos e os dois arquivos existem, o cp sobrescreve o segundo arquivo com uma cópia do primeiro arquivo.

|  |
| --- |
| DICA: |
| O comando cp, por padrão, só funciona em arquivos individuais. Para copiar um diretório, usamos a opção -r. Lembre-se de que a opção -r fará com que cp também copie o conteúdo do diretório que está sendo copiado |

**Exercícios Guiados**

1. O que -v faz em mkdir, rm e cp?

2. O que acontece se você acidentalmente tentar copiar três arquivos na mesma linha de comando para um arquivo que já existe em vez de um diretório?

3. O que acontece quando você usa mv para mover um diretório para dentro de si mesmo?

4. Como você excluiria todos os arquivos no diretório atual que começa com old?

**Exercícios Exploratórios**

**REFERENCIAS**

<https://learning.lpi.org/pt/learning-materials/010-160/2/2.4/2.4_01/>

https://ubuntu.com/tutorials/command-line-for-beginners#1-overview