



# FUNDAMENTOS DE DESIGN DE SISTEMAS

AULA 2

## CONVERSA INICIAL

O objetivo desta etapa é oferecer um aprofundamento no Ubuntu, conhecendo a sua estrutura de diretórios e a linha de comando do Linux. Os objetivos específicos são:

- conhecer a estrutura de diretórios do Linux;
- aprender os comandos de manipulação de diretórios e arquivos;
- conhecer comandos do sistema;
- conhecer comandos de instalação de pacotes; e
- conhecer permissões, acessos e seus comandos no Linux.

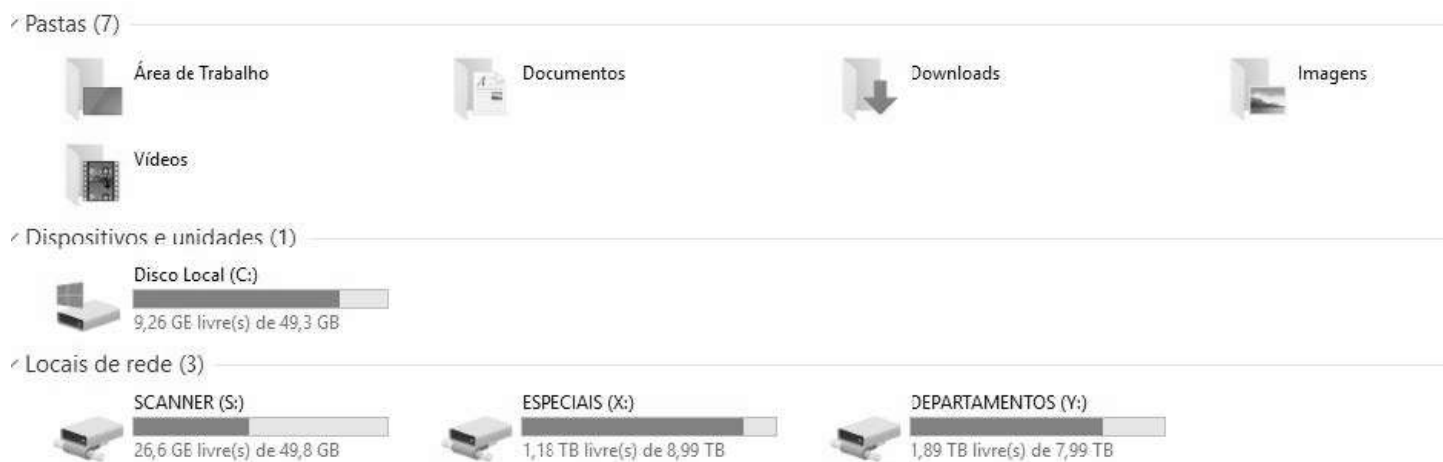
## TEMA 1 – ESTRUTURA DE DIRETÓRIOS DO LINUX

O Windows e o Linux são sistemas operacionais diferentes que utilizam estruturas de arquivos distintas para organizar arquivos e pastas.

No Windows, os arquivos são organizados em uma única estrutura de diretórios e os diretórios são representados como pastas. Cada pasta pode conter arquivos e outras pastas, permitindo que os usuários naveguem pelas estruturas de diretórios para encontrar o arquivo desejado.

O diretório raiz do Windows, por padrão, é o C:\, onde todos os outros diretórios estão alocados. A Figura 1 apresenta um exemplo de diretórios Windows.

Figura 1 – Diretórios do Windows



Já no Linux, a **estrutura de diretórios é hierárquica e baseada em raízes**. A raiz é representada pelo símbolo "/" e todos os outros diretórios são organizados em relação a ela. O Linux usa uma estrutura de diretórios rigorosa, com pastas específicas para cada tipo de arquivo. Vamos conhecer as principais a seguir.

## 1.1 DIRETÓRIO *BIN*

O diretório `"/bin"` no Linux é usado para armazenar arquivos binários executáveis, ou seja, programas que podem ser executados diretamente pelo sistema. É similar ao Arquivos de Programas do Windows, com programas comuns que são frequentemente usados pelos usuários e pelo próprio sistema.

Além disso, o `"/bin"` é um dos diretórios incluídos no caminho de pesquisa do sistema, o que significa que o sistema automaticamente procura nesse diretório quando o usuário executa um comando no terminal. Isso garante que os programas comuns estejam sempre disponíveis para o usuário, independentemente da pasta atual em que ele está trabalhando.

## 1.2 DIRETÓRIO *BOOT*

O diretório `"/boot"` no Linux é responsável por armazenar arquivos necessários para o processo de inicialização do sistema. Esses arquivos incluem o *kernel* do Linux, arquivos de configuração e drivers de dispositivos.

Durante o processo de inicialização, o sistema carrega os arquivos contidos no `"/boot"` para configurar e inicializar o hardware. Depois, carrega o *kernel* do sistema. O diretório `"/boot"` é

importante, pois ele é necessário para que o sistema operacional funcione corretamente, considerando aqui que arquivos corrompidos ou ausentes podem impedir o processo de inicialização. Por isso, é importante que os arquivos no `"/boot"` sejam mantidos atualizados e protegidos contra danos ou modificações acidentais.

### 1.3 DIRETÓRIO *DEV*

O diretório `"/dev"` (*devices*) no Linux é usado para representar dispositivos de *hardware* presentes no sistema. Nele, cada dispositivo é representado por um arquivo especial, conhecido como arquivo de dispositivo, que permite que o sistema e os programas acessem o dispositivo de *hardware*. Os arquivos de dispositivo são usados para realizar operações de leitura e escrita no dispositivo, bem como para controlar o seu comportamento.

O diretório `"/dev"` é importante porque é usado como um ponto central para o acesso a todos os dispositivos de hardware no sistema, incluindo discos rígidos, unidades de CD-ROM, dispositivos de entrada, como teclados e mouses, entre outros. Além disso, os arquivos de dispositivo no `"/dev"` são usados por vários programas de sistema para a realização de tarefas, como montar partições de disco ou acessar dispositivos de rede.

A nomenclatura de discos no Linux é baseada na convenção de letras maiúsculas para discos rígidos internos e letras minúsculas para dispositivos removíveis. Cada dispositivo de armazenamento é representado por um arquivo especial no diretório `"/dev"`, como `"/dev/sda"` para o primeiro disco rígido interno, `"/dev/sdb"` para o segundo, e `"/dev/sdc"` para dispositivos removíveis, como pen drives. Além disso, as partições de um disco rígido são identificadas com um número adicional, como `"/dev/sda1"` para a primeira partição do disco rígido `"/dev/sda"`. Essa convenção é usada pelo sistema operacional para identificar e acessar dispositivos de armazenamento de maneira consistente.

### 1.4 DIRETÓRIO *ETC*

O diretório `"/etc"` no Linux é usado para armazenar arquivos de configuração do sistema e de programas instalados no sistema. Os arquivos de configuração controlam como o sistema operacional e os programas se comportam, incluindo configurações de rede, definições de usuário, informações de autenticação e outros detalhes importantes.

O diretório `"/etc"` é acessado frequentemente por administradores de sistema, para a realização de tarefas como configurar serviços de rede, adicionar ou remover usuários, ou ainda alterar configurações de programas instalados. Além disso, os arquivos de configuração no `"/etc"` são usados pelo sistema operacional durante o processo de inicialização para definir as configurações necessárias para o funcionamento correto do sistema.

## 1.5 DIRETÓRIO *HOME*

O diretório `"/home"` no Linux é usado para armazenar os arquivos pessoais e de configuração de cada usuário do sistema. Cada usuário tem o seu próprio diretório dentro do `"/home"`, geralmente com o mesmo nome do usuário. Por exemplo, `"/home/vinicius"` para o usuário `"vinicius"`. Esses diretórios são usados para armazenar arquivos pessoais, como documentos, imagens, músicas e outros, bem como arquivos de configuração específicos para o usuário.

O diretório `"/home"` é uma parte importante da estrutura de arquivos do Linux, pois permite que cada usuário tenha os seus próprios arquivos e configurações em separado dos demais usuários e do sistema operacional. Isso oferece separação de responsabilidades entre usuários e protege contra a perda de dados ou configurações importantes, caso ocorra algum problema com o sistema operacional ou com outros usuários. Além disso, a estrutura de diretórios do `"/home"` permite que os usuários tenham acesso aos seus arquivos a partir de qualquer conta, desde que estejam logados no sistema.

O diretório equivalente ao `"/home"` no Windows é geralmente o diretório `"C:\Users"`. Assim como o `"/home"` no Linux, o diretório `"C:\Users"` também armazena os arquivos pessoais e de configuração para cada usuário do sistema. Cada usuário tem o seu próprio diretório dentro do `"C:\Users"`.

Em termos de funcionamento, o diretório `"C:\Users"` no Windows e o diretório `"/home"` no Linux servem para a mesma finalidade: separam os arquivos pessoais e de configuração de cada usuário para fornecer segurança e facilidade de acesso aos arquivos.

## 1.6 OUTROS DIRETÓRIOS LINUX

Por fim, vejamos mais alguns diretórios importantes do Linux Ubuntu.

- **root**: diretório raiz para o usuário administrador (*root*); armazena arquivos e configurações exclusivas para o usuário *root*.
- **lib**: diretório que armazena bibliotecas compartilhadas usadas pelo sistema operacional e pelos aplicativos.
- **media**: diretório usado para montar dispositivos externos, como discos rígidos externos ou dispositivos de mídia removíveis.
- **mnt**: diretório similar ao `"/media"`, usado para montar dispositivos de armazenamento temporários.
- **opt**: diretório usado para armazenar aplicativos adicionais que não fazem parte do sistema operacional padrão – o Google Chrome, por exemplo, usa este diretório.
- **proc**: diretório virtual que fornece informações sobre o sistema e o processo em execução.
- **run**: diretório usado para armazenar informações de tempo de execução para o sistema e os aplicativos.
- **sbin**: diretório que armazena comandos binários usados para manter e configurar o sistema operacional – é a pasta *bin* do super usuário.
- **tmp**: diretório usado para armazenar arquivos temporários para o sistema e os aplicativos.
- **usr**: diretório que armazena aplicativos, documentação e outros arquivos compartilhados usados pelo sistema e pelos usuários.
- **var**: diretório que armazena arquivos que variam, como logs de sistema, arquivos de e-mail e outros arquivos de dados dinâmicos.

## TEMA 2 – COMANDOS DE MANIPULAÇÃO DE DIRETÓRIOS

O terminal é a interface de linha de comando que permite aos usuários interagir com o sistema operacional Linux. Através do terminal, os usuários podem executar comandos para realizar tarefas como gerenciar arquivos, instalar softwares, gerenciar processos e realizar configurações do sistema.

O terminal do Linux é uma ferramenta poderosa e flexível, amplamente utilizada por administradores de sistemas e desenvolvedores. Além de ser mais rápido e eficiente em comparação a interfaces gráficas, o terminal permite aos usuários automatizar tarefas repetitivas e executar tarefas em lote com facilidade. Além disso, muitas distribuições Linux incluem recursos adicionais, como autocompletar comandos, histórico de comandos e *shell* diferentes para escolha.

Vamos começar a conhecer os comandos de manipulação de diretórios em Linux.

## 2.1 COMANDO *LS*

O comando *ls* é um dos comandos mais básicos e amplamente utilizados no terminal Linux. A sigla *ls* significa *list* (listar). Ela é usada para listar os arquivos e diretórios em um diretório. A sintaxe básica do comando *ls* é:

```
ls [opções] [caminho/diretório]
```

Alguns dos principais parâmetros do comando *ls* incluem:

- *-l*, de arquivos e data de modificação;
- *-a*, que mostra todos os arquivos, incluindo aqueles ocultos que começam com um ponto (.); e
- *-h*, que exibe o tamanho dos arquivos em uma formatação humanamente legível.

Um exemplo de uso do comando *ls* é listar os arquivos e diretórios em um diretório específico:

```
ls ~/Downloads
```

Este comando irá exibir todos os arquivos e diretórios na pasta de *downloads* do usuário corrente.

## 2.2 COMANDO *CD*

A sigla *cd* significa *change directory* (mudar diretório), sendo usada para mudar o diretório de trabalho atual. A sintaxe básica do comando *cd* é:

```
cd [caminho/diretório]
```

Alguns dos principais parâmetros do comando *cd* incluem:

- *"."* (ponto), que representa o diretório atual;
- *".."* (ponto ponto), que representa o diretório pai (um nível acima do diretório atual); e
- *"~"* (til), que representa o diretório home do usuário corrente.

Um exemplo de uso do comando *cd* é mudar para o diretório *home* do usuário corrente:

```
cd ~
```

## 2.3 COMANDO *MKDIR*

O comando *mkdir* é utilizado para criar novos diretórios em sistemas operacionais Linux. A sigla *mkdir* significa *make directory* (criar diretório). A sintaxe básica do comando *mkdir* é:

```
mkdir [nome_do_diretorio]
```

Alguns dos principais parâmetros do comando *mkdir* incluem:

- *-p*, que permite criar diretórios aninhados, ou seja, diretórios dentro de outros diretórios; e
- *-m*, que permite definir as permissões para o novo diretório.

Um exemplo de uso do comando *mkdir* é criar um diretório chamado *novo\_diretorio*, como:

```
mkdir novo_diretorio
```

## 2.4 COMANDO *RMDIR*

O comando *rmdir* é utilizado para remover diretórios vazios em sistemas operacionais Linux. A sigla *rmdir* significa *remove directory* (remover diretório). A sintaxe básica do comando *rmdir* é:

```
rmdir [nome_do_diretorio]
```

Alguns dos principais parâmetros do comando *rmdir* incluem:

- *-p*, que permite remover diretórios aninhados, ou seja, diretórios dentro de outros diretórios; e
- *-v*, que exibe mensagens na tela sobre as ações realizadas pelo comando.

Um exemplo de uso do comando *rmdir* é remover um diretório chamado *diretorio\_a\_ser\_removido*

```
rmdir diretorio_a_ser_removido
```

Este comando irá remover o diretório *diretorio\_a\_ser\_removido* se ele estiver vazio. Se o diretório não estiver vazio, o comando apresentará uma mensagem de erro.

## 2.5 COMANDO *RM*



O comando *rm* é utilizado para remover arquivos e diretórios. A sigla *rm* significa *remove* (remover). A sintaxe básica do comando *rm* é:

```
rm [nome_do_arquivo_ou_diretório]
```

Alguns dos principais parâmetros do comando *rm* incluem:

- *-r* ou *-R*, que permite remover diretórios e seus conteúdos recursivamente;
- *-f*, que força a remoção sem confirmação ou mensagem de erro; e
- *-v*, que exibe mensagens na tela sobre as ações realizadas pelo comando.

Um exemplo de uso do comando *rm* é remover um arquivo chamado *arquivo\_a\_ser\_removido*:

```
rm arquivo_a_ser_removido
```

Este comando irá remover o arquivo *arquivo\_a\_ser\_removido* e não exibirá nenhuma mensagem na tela. Observe que, ao usar o comando *rm*, não é possível desfazer a ação – portanto, é importante tomar cuidado ao usá-lo.

## 2.6 COMANDO *PWD*

O comando *pwd* (*print working directory*) é utilizado para imprimir o caminho completo da pasta atual no sistema. A sintaxe é simples, sem a necessidade de parâmetros.

Ao executar esse comando, o terminal irá mostrar o caminho absoluto da pasta atual. Por exemplo, se o usuário estiver na pasta */home/user/docs*, o comando *pwd* retornará */home/user/docs*. Esse comando é útil para verificar em que pasta o usuário se encontra no momento e também para construir caminhos absolutos a partir da pasta atual.

## TEMA 3 – COMANDOS DE GERENCIAMENTO DE PACOTES

O gerenciamento de pacotes é uma das principais características do sistema operacional Linux. Trata-se de um dos aspectos mais importantes na administração de sistemas. O sistema de gerenciamento de pacotes permite que os usuários instalem, atualizem, removam e gerenciem facilmente os programas e bibliotecas instalados no sistema.

Os pacotes no Ubuntu são arquivos *.deb* que contêm informações sobre a instalação, a configuração e as dependências de software. O gerenciador de pacotes do Ubuntu, o *apt*, é responsável por baixar e instalar os pacotes necessários para o sistema, além de garantir que as dependências sejam resolvidas corretamente. Além disso, o *apt* também permite que os usuários atualizem todo o sistema, ou apenas pacotes específicos, de maneira fácil e segura.

### 3.1 COMANDO *APT*

A sintaxe básica é:

```
apt [opções] [comando] [pacote1] [pacote2] ...
```

Os principais parâmetros incluem:

- *update*: atualiza a lista de pacotes disponíveis;
- *upgrade*: atualiza os pacotes já instalados para sua versão mais recente;
- *install*: instala um pacote;
- *remove*: remove um pacote;
- *list*: descobre se o pacote está instalado ou não, e qual a sua versão; e
- *search*: procura por um pacote no repositório.

Um exemplo de uso é:

```
sudo apt update  
sudo apt install vlc
```

Este exemplo atualiza a lista de pacotes disponíveis e instala o pacote *vlc*, referente ao VLC Player. É importante lembrar que o comando *sudo* é necessário para executar comandos como administrador.

### 3.2 COMANDO *DPKG*

O comando *dpkg* é um gerenciador de pacotes para sistemas operacionais baseados em Debian, como o Ubuntu. É útil para instalar pacotes fora do repositório padrão do sistema. O próprio Google Chrome é um exemplo que precisa ser instalado dessa maneira. A sintaxe básica é:

```
dpkg [opções] nome_do_pacote
```

Alguns dos principais parâmetros são:

- -i (instalar): instala um pacote específico;
- -r (remover): remove um pacote específico;
- -l (listar): lista todos os pacotes instalados; e
- -S (buscar): busca por um pacote específico.

## TEMA 4 – COMANDOS DE PROCESSOS

Em um sistema operacional Linux, **um processo é uma instância em execução de um programa ou aplicação**. Cada processo tem a sua própria identificação única (PID), espaço de endereçamento virtual, informações de status e recursos alocados, como memória RAM e CPU. Quando um usuário executa um comando ou inicia um aplicativo, um novo processo é criado.

Processos podem estar em *background* ou em *foreground*, termos usados para descrever a prioridade e a forma como os processos são executados em um sistema operacional Linux. Processos em *foreground* estão sendo executados atualmente, recebendo a entrada do teclado e saída de tela. Já os processos em *background* estão sendo executados em segundo plano, sem interação com o usuário. Processos em *background* são úteis para tarefas que precisam ser executadas, mas não precisam de atenção imediata, como *downloads* ou *backup*.

Já um *job* é uma tarefa específica executada no terminal do Linux. Os *jobs* são basicamente processos executados no terminal, como a execução de um comando de *background* ou a suspensão de um processo. Os *jobs* são identificados por um número de *job* (JID). O usuário pode acessar o status dos *jobs* ativos, suspender ou retomar a execução usando os comandos de gerenciamento de *jobs* do *shell*.

### 4.1 COMANDO *PS*

O comando *ps* no Linux é utilizado para exibir informações sobre processos ativos no sistema. A sintaxe básica do comando é:

```
ps [OPÇÕES]
```

Alguns dos principais parâmetros utilizados com o comando *ps* incluem:

- -a: mostra todos os processos, incluindo aqueles de outros usuários;
- -e: mostra todos os processos, como o parâmetro -a;
- -f: exibe a árvore de processos; e
- -u: exibe informações sobre os processos de um determinado usuário.

## 4.2 COMANDO *TOP*

O comando *top* no Linux é um utilitário para monitorar processos. Ele mostra uma lista interativa de processos que estão sendo executados no sistema, incluindo informações como CPU, memória, nome do processo e tempo de execução. Vejamos a sintaxe básica do comando:

```
top [OPÇÕES]
```

Alguns dos principais parâmetros do comando *top* são:

- -d: especifica a frequência de atualização da lista de processos;
- -p: especifica quais processos deseja-se visualizar;
- -u: especifica um usuário específico para exibir processos; e
- -h: mostra ajuda com informações sobre os parâmetros do comando.

## 4.3 COMANDO *JOBS*

O comando *jobs* no Linux é usado para exibir a lista de *jobs* em segundo plano em uma sessão de terminal atual. Ele não apresenta parâmetros. A sua sintaxe é simplesmente *jobs*.

Exemplo de uso:

```
$ sleep 100 &  
$ jobs  
[1]+  Running                  sleep 100 &
```

Neste exemplo, o comando *sleep 100 &* coloca o processo *sleep* em segundo plano, enquanto o comando *jobs* mostra que há um job em segundo plano, com o número 1, que está em execução.

## 4.4 COMANDOS *FG* E *BG*

Os comandos *fg* (*foreground*) e *bg* (*background*) são usados para controlar processos em execução no terminal do Linux. O comando *fg* é usado para trazer um processo em *background* para o *foreground* – ou seja, o processo passa a ser o principal em execução no terminal. Já o comando *bg* é usado para enviar um processo para *background*, quando o processo continua a ser executado, mas agora em segundo plano, permitindo que o usuário execute outros processos no terminal sem interromper o processo em *background*.

## 4.5 COMANDO *KILL*

O comando *kill* do Linux é usado para encerrar ou interromper processos. A sintaxe básica desse comando é:

```
kill [OPÇÕES] PID
```

Onde PID é o número de identificação do processo que se deseja encerrar ou interromper. Alguns dos principais parâmetros utilizados com o comando *kill* são:

- -l: lista os sinais que podem ser enviados para os processos;
- -s: envia o sinal SIG para o processo; e
- -9: envia o sinal de interrupção SIGKILL, que é usado para forçar o encerramento de um processo.

Um exemplo de uso do comando *kill* é:

```
kill -9 1456
```

Neste exemplo, o comando envia o sinal SIGKILL (9) para o processo com PID 1456, forçando o seu encerramento.

## TEMA 5 – COMANDOS DE ACESSO E PERMISSÕES

As permissões de acesso no Linux são uma forma de controlar quem pode acessar arquivos e diretórios em um sistema de arquivos Linux. Isso permite que os administradores de sistemas atuem como gerentes de segurança de arquivos, protegendo arquivos sensíveis e concedendo acesso apenas a usuários confiáveis.

Em um sistema Linux, cada arquivo ou diretório é associado a um usuário proprietário e a um grupo de usuários. O proprietário, também chamado de **dono (owner)** do arquivo ou diretório, é o usuário responsável pelo arquivo ou diretório. Apenas ele pode realizar determinadas operações no arquivo ou diretório, como modificá-lo ou excluí-lo. O *user ID* (identificador de usuário) é um número único atribuído a cada usuário no sistema. Ele identifica de forma única cada usuário, sendo utilizado para controlar o acesso a arquivos e diretórios. Quando um arquivo ou diretório é criado, o usuário atualmente logado é definido como o proprietário e o *user ID* desse usuário é armazenado com o arquivo ou o diretório.

No Linux, um **grupo** é uma coleção de usuários com acesso a um ou mais arquivos ou diretórios. Quando um arquivo ou diretório é criado, ele é atribuído a um grupo específico. O *group ID* (GID) é o número único que identifica o grupo ao qual o arquivo ou diretório pertence. O GID pode ser usado para conceder ou restringir o acesso ao arquivo ou diretório para os usuários que fazem parte do grupo. Por exemplo, se um arquivo pertencer a um grupo "projeto", os usuários que fazem parte desse grupo podem receber permissões de leitura, escrita ou execução para esse arquivo, dependendo das permissões definidas.

## 5.1 TIPOS DE PERMISSÕES DE ACESSO

Existem três tipos de permissões de acesso para arquivos e diretórios: leitura, escrita e execução.

A permissão de **leitura (r - read)** permite que o arquivo ou diretório seja lido pelo usuário. Em diretórios, permite listar conteúdo com o comando *ls*, por exemplo.

A permissão de **escrita (w - write)** permite que o arquivo ou diretório seja modificado pelo usuário. Em diretórios, podemos gravar arquivos dentro dele. Ainda, um arquivo ou diretório só pode ser apagado se tiver permissão de escrita.

A permissão de **execução (x - execute)** permite que o arquivo seja executado como um programa. Permite que um diretório seja acessado através do comando *cd*.

Essas permissões são sempre atribuídas a três entidades diferentes: dono, grupo e outros. Cada entidade pode ter permissões de leitura, escrita e execução diferentes para o mesmo arquivo ou diretório.

Vejamos um exemplo de permissões em um arquivo:

```
-rwxr-xr--  vinicius  users  nomeArquivo
```

- 1º caractere: define o tipo do arquivo (um "d" é um diretório; um "l", um link a um arquivo no sistema; um "-" é um arquivo comum).
- (2-4)º caractere: permissões do dono do arquivo (vinicius).
- (5-7)º caractere: permissões do grupo do arquivo (users).
- (8-10)º caractere: permissões de outros usuários ao arquivo.

## 5.2 O ROOT (SUPERUSUÁRIO)

O *root* é o usuário administrador no sistema Linux. Ele tem permissões totais sobre o sistema, incluindo acesso a todos os arquivos e diretórios, bem como a capacidade de executar comandos com privilégios elevados. Isso significa que o *root* pode fazer mudanças significativas no sistema, incluindo instalação de software, configuração de serviços e gerenciamento de usuários. É importante que o uso da conta *root* seja feito com precaução, pois uma ação mal intencionada ou equivocada pode causar problemas graves no sistema. É recomendável que o usuário *root* seja utilizado somente em situações específicas e que a maioria das tarefas sejam realizadas por outro usuário com permissões restritas.

## 5.3 COMANDO CHMOD

O *chmod* é um comando utilizado para mudar as permissões de acesso de um arquivo ou diretório. A sintaxe geral para o comando *chmod* é:

```
chmod [OPÇÕES] MODO ARQUIVO/DIRETÓRIO
```

O quadro apresenta a nomenclatura utilizada nesse comando.

Quadro 1 – Comando *chmod*

Caractere	Significado	Caractere	Significado
u	Usuário	r	Leitura
g	Grupo	w	Escrita
o	Outros	x	Execução

<b>a</b>	<b>Todos</b>	<b>+</b>	<b>Adiciona permissão</b>
		<b>-</b>	<b>Remove permissão</b>

Exemplo de uso: para dar permissões de leitura e escrita para o dono e o grupo, e apenas permissões de leitura para outros usuários em um arquivo chamado *arquivo.txt*, você pode usar o seguinte comando:

```
chmod u+rw,g+rw,o+r arquivo.txt
```

## FINALIZANDO

O terminal do Linux é uma ferramenta essencial para o gerenciamento do sistema operacional. Ele permite aos usuários controlar o sistema através de comandos, oferecendo uma ampla gama de opções, como gerenciamento de arquivos e diretórios, instalação e gerenciamento de pacotes, gerenciamento de processos e gerenciamento de permissões de acesso. É importante que qualquer usuário de Linux saiba utilizar o terminal, pois ele é uma ferramenta poderosa e flexível para resolver problemas, automatizar tarefas e aprimorar a sua eficiência.