Primeiro Contato - Arquivos - Terminal - Comandos

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro

Prof. Edwar Saliba Júnior

Janeiro de 2025

Usuários - Explicações Básicas

- O Linux possui basicamente dois tipos de usuários:
  - o usuário normal que é representado pelos caractere:

    $
  - e o superusuário, que tem permissões avançadas dentro do sistema operacional e é representado pelo caractere:

    #

Arquivos - Explicações Básicas

- É onde gravamos nossos dados;
- cada arquivo deve ser identificado por um nome;
- o Linux é *case-sensitive*, ou seja, diferencia letras maiúsculas de minúsculas e
- um arquivo oculto no GNU/Linux é identificado por um "." no início do nome (por exemplo, `.bashrc`). Arquivos ocultos não aparecem em listagens normais de diretórios. Para vê-los deve-se usar o comando:

```
$ ls -a
```

Extensão dos Arquivos

- A extensão serve para identificar o tipo do arquivo. A extensão são as letras após um "." no nome de um arquivo, explicando melhor:
  - `relatório.txt` - o ".txt" indica que o arquivo é um arquivo de texto;
  - `script.sh` - arquivo de *Script* (interpretado por /bin/sh), ou seja um executável;
  - `system.log` – arquivo de registro de programas do sistema;
  - `arquivo.gz` - arquivo compactado pelo utilitário gzip e
  - `index.html` - página de Internet (formato Hypertext).

- A extensão de um arquivo também ajuda a saber o que precisamos fazer para abri-lo. Por exemplo, o arquivo `relatório.txt` é um texto simples e podemos ver seu conteúdo através do comando `cat`, já o arquivo `index.html` contém uma página de Internet e precisaremos de um navegador para poder visualizá-lo (como o Firefox, Chrome, Lynx, Konqueror e etc.).

- A extensão, na maioria dos casos, não é requerida pelo sistema operacional GNU/Linux, mas é conveniente o seu uso para determinarmos facilmente o tipo de arquivo e que programa precisaremos usar para abri-lo.

Tipos de Arquivos

- Texto:
  - seu conteúdo é compreendido pelas pessoas. Um arquivo texto pode ser uma carta, um *script*, um programa de computador escrito pelo programador, um arquivo de configuração e etc. e

- Binário:
  - seu conteúdo somente pode ser entendido por computadores. Contém caracteres incompreensíveis para pessoas normais.

Diretórios

- Diretório é o local utilizado para armazenar arquivos, para fim de organização e localização e

- tanto o diretório como o arquivo são *case-sensitive* (o diretório `/teste` é completamente diferente do diretório `/Teste`).

Diretório Raiz

- Este é o diretório principal do sistema. Dentro dele estão todos os outros diretórios do sistema e

- o diretório Raiz é representado por uma "/", assim se você executar o comando:

  ```
  $ cd /
  ```

  você estará acessando o diretório raiz.

Estrutura de Subdiretórios

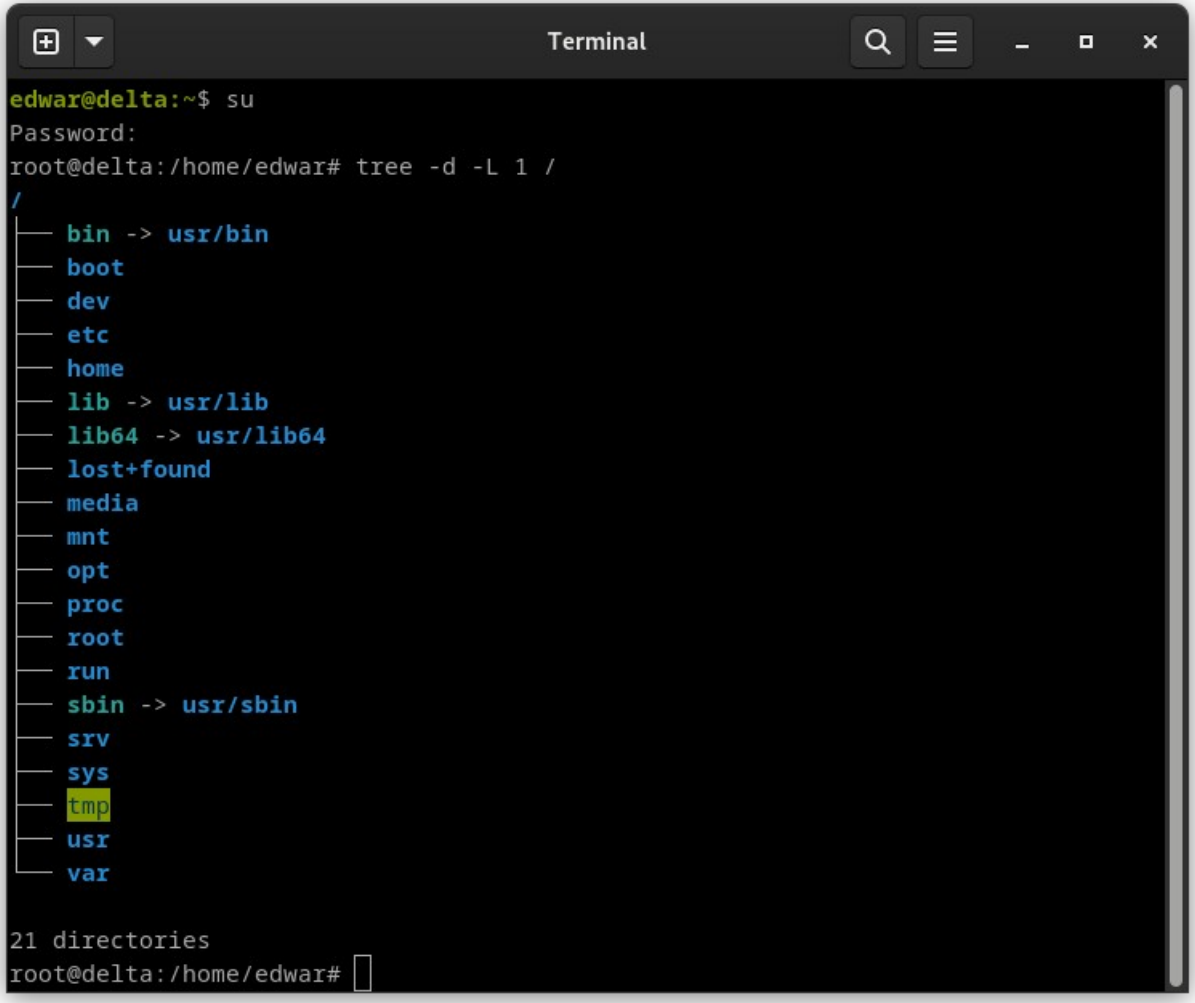
- A estrutura de diretórios e subdiretórios do Linux pode variar um pouco em cada distribuição.

- A execução do comando:

  # tree -d -L 1 /

  cria uma árvore de diretórios básica (apenas diretórios ligados à raiz) (Mota Filho, 2013, 186). Na distribuição Debian, o resultado poderá ser visto no *slide* a seguir.

Estrutura de Subdiretórios



Estrutura de Diretórios do Linux

A estrutura de diretórios dos sistemas Unix é regulada por uma norma chamada *Filesystem Hierarchy Standard* (FHS), que se encontra disponível em:

https://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html

Estrutura de Diretórios do Linux

`/bin`

contém arquivos executáveis de uso geral (binário ou não) que podem ser acessados por qualquer usuário. No entanto, neste diretório devem estar somente os executáveis necessários durante a recuperação em panes no sistema (também podendo se usados fora destes momentos). Executáveis de uso geral que não sejam necessários em momentos de pane deverão ser colocados em `/usr/bin`;

`/boot`

contém arquivos que compõem o Kernel (núcleo) e os arquivos que controlam a inicialização do sistema, como parte do gestor de *boot* conhecido como GRUB;

`/dev`

contém arquivos que servem de ligação com os dispositivos de *hardware* (*devices*) do computador. Esses arquivos são tecnicamente conhecidos como *pipes* (tubos), justamente por conectarem o sistema operacional ao *hardware*;

`/etc`

contém a grande maioria dos arquivos de configuração do sistema operacional e dos serviços de redes, além de arquivos executáveis ligados à inicialização do sistema e dos seus serviços;

`/home`

contém arquivos e configurações dos usuários do sistema operacional. Cada usuário possui um subdiretório dentro deste diretório, sendo que a única exceção é o usuário root, que possui um diretório exclusivo no diretório raiz;

`/lib`

contém os módulos do kernel, *drivers* de dispositivos e as bibliotecas utilizadas no momento da inicialização (*boot*) do sistema. Bibliotecas compartilhadas pelos programas do sistema e módulos do kernel;

`/lost+found`

após recuperações de arquivo do sistema, os arquivos encontrados no disco e que tenham perdido o vínculo com o seu *inode* serão colocados dentro desse diretório - geralmente arquivos corrompidos por falta de energia. Os arquivos que vão para esta pasta terão seus nomes originais modificados para um número sequencial;

`/media`

ponto de montagem de dispositivos diversos do sistema (rede, pendrives, CD-ROM em distribuições mais novas);

(adaptado de Mota Filho, 2012, p. 187-188)

Estrutura de Diretórios do Linux

`/mnt`

abreviatura de *mount*. É um ponto de montagem de arquivos de sistemas localizado em dispositivos de armazenamento não removíveis como o HD;

`/opt`

esse é um diretório destinado aos programas sque não fazem parte da distribuição em questão (*optional*). É um diretório pouco utilizado, pois a maioria das distribuições não segue a filosofia que o envolve. Similar ao diretório `/usr/local`;

`/proc`

diretório virtual. Na verdade, um arquivo de sistema virtual. Ele não contém arquivos e sim referências a informações dinâmicas dos sistema (*procedures*), geradas constantemente pelo Kernel;

`/root`

pasta que pertence ao usuário root (administrador do sistema). O root é o único usuário que tem seu subdiretório no diretório raiz;

`/sbin`

similar ao /bin, contém arquivos executáveis. No entanto, esses arquivos são destinados à administração e à manutenção do sistema em momentos de pane. Arquivos executáveis de administração do sistema que não sejam essenciais em momentos de panes deverão ser colocados em `/usr/sbin`;

`/tmp`

é um diretório utilizado pelo sistema para gravar informações temporárias e dados durante processamentos. Pro norma, esse diretório tem seu conteúdo totalmente apagado a cada *boot* do sistema;

`/sys`

o objetivo desse diretório é abrigar as informações que serão servidas pela máquina, como *sites*, arquivos para FTP e etc. Não é muito utilizado pela maioria das distribuições, pelo mesmo motivo que ocorre com o `/opt`;

`/usr`

USR é a sigla para *Unix Shared Resources*. Esse diretório, em situações normais, contem a maior parte dos arquivos do sistema. O objetivo do `/usr` é guardar dados que possam ser compartilhaods, que sejam estáticos e que estejam no modo somente leitura. O exemplo disso é o diretório `/usr/shar/doc`, que contém diversos documentos sobre programas e comandos. Dentro deste diretório existem diretórios como: `/usr/bin`, `/usr/sbin` e `/usr/lib`, que contêm arquivos essenciais para o funcionamento dos sistema operacional, mesmo em situações críticas ou após desastres;

`/var`

contém arquivos com conteúdo variável, como *logs*, *spool* de impressoras, caixas postais em servidores de *e-mail* e etc.

(adaptado de Mota Filho, 2012, p. 188-190)

Diretório Atual

- É o diretório em que nos encontramos no momento. Você pode executar:

  ```
  $ pwd
  ```

  para verificar qual é seu diretório atual e

- o diretório atual também é identificado por um "." (ponto).

Diretório Home

- Também chamado de diretório de usuário. Em sistemas GNU/Linux cada usuário (inclusive o root) possui seu próprio diretório onde poderá armazenar seus programas e arquivos pessoais;

- este diretório está localizado em `/home/[login]`, neste caso se o seu login for "joao" o seu diretório home será `/home/joao`. O diretório home também é identificado por um ~ (til), você pode executar o comando:

  ```
  $ ls /home/joao
  ```

  como

  ```
  $ ls ~
  ```

- para listar os arquivos de seu diretório home e o diretório home do usuário root, na maioria das distribuições GNU/Linux, está localizado em /root.

Diretório Superior

- O diretório superior (*Upper Directory*) é identificado por `..` (ponto ponto);

- caso você esteja no diretório `/usr/local` e queira listar os arquivos do diretório `/usr`, você poderá executar:

  ```
  $ ls ..
  ```

  este recurso, `..`, também pode ser usado para copiar e mover arquivos/diretórios e etc.

Diretório Anterior

- O diretório anterior é identificado por "-" (sinal de subtração). É útil para retornar ao último diretório usado;

- exemplo: se você estiver no diretório `/usr/local` e executar:

```
$ cd /lib
```

você pode retornar facilmente para o diretório `/usr/local` usando o comando:

```
$ cd -
```

Nomeando Arquivos e Diretórios

- No GNU/Linux os arquivos e diretórios podem ter o tamanho de até 255 bytes. Lembrando que existem caracteres que são chamados de multi-byte, ou seja, que ocupam mais de um byte;

- podem possuir uma extensão (um conjunto de letras separadas do nome do arquivo por um ".")e

- os programas executáveis do GNU/Linux, ao contrário dos programas de DOS e Windows, não são executados a partir de extensões `.exe`, `.com` ou `.bat`. O GNU/Linux (como todos os sistemas POSIX) usa a permissão de execução de arquivo para identificar se um arquivo pode ou não ser executado.

Comando `mkdir`

- Comando para criar um novo diretório;
- por exemplo:

`$ mkdir Teste` – cria o diretório "Teste", dentro do diretório atual;

`$ mkdir Teste/Cartas` – cria o diretório "Cartas" dentro do diretório "Teste". Isto, sem sair do diretório atual e

`$ mkdir ../Documentos/Cartas` – sobe um nível na árvore de diretórios, entra no diretório "Documentos" e cria o diretório "Cartas" dentro dele. Tudo isto, sem sair do diretório atual;

Comando `rmdir`

- Comando para apagar um diretório;
- para funcionar o diretório tem que estar vazio, por exemplo:
  - `$ rmdir Teste` – apaga o diretório "Teste" que está dentro do diretório atual;
  - `$ rmdir Teste/Cartas` – apaga o diretório "Cartas" que está dentro do diretório "Teste". Isto, sem sair do diretório atual;
  - `$ rmdir ../Documentos/Cartas` – sobe um nível na árvore de diretórios, entra no diretório "Documentos" e apaga o diretório "Cartas" que está dentro dele. Tudo isto, sem sair do diretório atual
- se o diretório não estiver vazio:
  - `$ rm -R Teste` – apaga o diretório "Teste" e todos os subdiretórios e arquivos que existirem dentro dele, use com moderação!

Comandos `ls` e `cd`

- Talvez estes sejam os comandos mais usados no Terminal. E servem para:

  `ls` – para mostrar arquivos de diretórios e

  `cd` - para mudar de diretório.

Comando `ls`

- Um comando pode receber opções e parâmetros, ou seja algumas opções.

- Por exemplo:
  - as opções são usadas para controlar como o comando será executado, para fazer uma listagem mostrando o dono, o grupo e o tamanho dos arquivos, você deve executar o comando: `ls -l`
  - opções podem ser passadas ao comando através de um "`-`" ou "`—`";
  - "`-`" seguido por letras; podem ser usadas mais de uma opção com um único hífen. O comando "`ls -l -a`" pode ser escrito assim: "`ls -la`"
  - "`--`" seguido por um nome. Também chamado de opção extensa. O comando "`ls --all`" é equivalente ao comando: "`ls -a`".

- Pode ser usado tanto "-" como "--", mas há casos em que somente "-" ou "--" está disponível.

Comando `ls`

- Parâmetros:
  - um parâmetro identifica o caminho, origem, destino, entrada padrão ou saída padrão que será passada ao comando;
  - se você digitar: "`ls /usr/share/doc/copyright`" então `/usr/share/doc/copyright` será o parâmetro passado ao comando `ls`, neste caso queremos que seja listado os arquivos do diretório `/usr/share/doc/copyright`;
  - se o nome de um comando é digitado errado, então é mostrada a mensagem `command not found` (comando não encontrado).

- Por exemplo: "`ls -la /usr/share/doc`", `ls` é o comando, `-la` é a opção passada ao comando e `/usr/share/doc` é o diretório passado como parâmetro ao comando `ls`.

Terminal Virtual

- Terminal é o teclado e tela conectados em seu computador;
- "Terminal Virtual" é uma interface de texto independente que permite que você interaja diretamente com o kernel (núcleo) do sistema operacional, sem a necessidade de um ambiente gráfico;
- se estiver usando o ambiente gráfico, você deve segurar `CTRL + ALT` enquanto pressiona uma tecla de `F2` a `F6` para entrar em um terminal e
- o GNU/Linux possui mais de 63 terminais virtuais. Mas apenas 6 estão disponíveis inicialmente por motivos de economia de memória RAM.
- Observação: as teclas correspondentes aos terminais virtuais pode variar um pouco de distribuição para distribuição. As teclas de F2 a F6 são válidas na distribuição Debian.

Curingas (*wildcards*)

- Curingas são caracteres que podem ser utilizados em alguns comandos, ou seja, podem substituir outros caracteres ou sequências de caracteres.

- Este é um recurso que permite que você faça a filtragem do que será listado, copiado, apagado e etc.

- Alguns tipos de curingas do GNU/Linux:

  ***** - faz referência a um nome completo ou restante de um arquivo e/ou diretório;

  ? - faz referência a uma letra na posição em que estiver.

Curingas

- `[]` - serve para definir possibilidades de caracteres. Os caracteres circunflexo (`^`) e exclamação (`!`) podem ser usados como negação (apenas para um caractere). O caractere hífen (`-`) pode ser utilizado para definir intervalos seguindo a tabela ASCII, exemplos:
  - **[ABCde15]** – quaisquer dos caracteres citados, considerando a caixa dos mesmos no caso de letras;
  - **[a-z]** – qualquer caractere de **a** a **z** em caixa baixa (minúsculos);
  - **[0-9]** – qualquer número de **0** até **9**;
  - **[a-zA-Z0-9]** – qualquer letra independentemente da caixa ou algarismo;
  - **[!t]** – qualquer caractere exceto o **t**.

- A procura de caracteres é *case-sensitive*. Assim, se você deseja que sejam localizados todos os caracteres alfabéticos você deve usar **`[a-zA-Z]`**, como mostrado acima.

- Caso a expressão seja precedida por um **^**, faz referência a qualquer caractere exceto o da expressão.
  - Por exemplo: **[^abc]** faz referência a qualquer caractere exceto **a**, **b** e **c**.

Curingas

- Exemplos:
  - Vamos dizer que existam 5 arquivos no diretório `/home/edwar/teste`: **`teste1.txt`**, **`teste2.txt`**, **`teste3.txt`**, **`teste4.new`** e **`teste5.new`**.

  - Caso deseje listar todos os arquivos do diretório `/home/edwar/teste` você pode usar o curinga `*` de três formas diferentes:

```
$ cd /home/edwar/teste
$ ls *
-----------------------
$ ls /home/edwar/teste*
-----------------------
$ ls /home/edwar/teste/*
```

  - Não tem muito sentido usar o comando `ls` com `*` porque todos os arquivos serão listados se o `ls` for usado sem nenhum curinga.

Curingas

- Exemplos:
  - agora para listar todos os arquivos **teste1.txt**, **teste2.txt** e **teste3.txt** com exceção de **teste4.new** e **teste5.new**, podemos usar inicialmente 3 métodos:

```
$ ls *.txt
$ ls teste?.txt
$ teste*.txt
```

  - os três comandos trazem o mesmo resultado.

Curingas

- Continuando o exemplo anterior:
  - usando o comando `ls teste[1-3].txt`, que pega todos os arquivos que começam com o nome `teste`, tenham qualquer caractere entre o número `1-3` no lugar da 6ª letra e terminem com `.txt`. Neste caso se obtém uma filtragem mais exata, pois o curinga `?` especifica qualquer caractere naquela posição e `[]` especificam um intervalo de números e/ou letras.

- Agora para listar somente **`teste4.new`** e **`teste5.new`** podemos usar os seguintes comandos:
  - `$ ls *.new` - que lista todos os arquivos que terminam com `.new`
  - `$ ls teste?.new` - que lista todos os arquivos que começam com `teste`, contenham qualquer caractere na posição do curinga `?` e terminem com `.new` e
  - `$ ls teste[4,5].*` - que lista todos os arquivos que começam com `teste` contenham o número 4 ou 5 naquela posição e terminem com qualquer extensão.

Trabalhando com Arquivos

- Vamos imaginar que no diretório Home do usuário "edwar" (`/home/edwar`) existem os seguintes subdiretórios com seus respectivos arquivos:
  - `/home/edwar/teste` – com os aquivos:
    - `carta.odt`
    - `planilha.ods`
    - `apresentacao.odp`

Comando `touch`

- Para criarmos um arquivo tipo texto no diretório `/home/edwar/teste` basta executarmos o comando `touch`. Exemplo, se você estiver:
  - no diretório `/home/edwar/teste` :

```
$ touch texto.txt
```

  - fora do diretório `/home/edwar/teste` :

```
$ touch ~/teste/texto.txt
```

```
$ touch /home/edwar/teste/texto.txt
```

Comando `cp`

- Para copiarmos um arquivo do ou no diretório `/home/edwar/teste` basta executarmos o comando `cp`. Exemplos:
  - copiando um arquivo de outro diretório para o diretório `/home/edwar/teste`, estando neste diretório:

```
$ cp /tmp/texto2.txt .
```

  - copiando um arquivo para fora do diretório `/home/edwar/teste`, estando neste diretório

```
$ cp texto.txt /tmp
```

  - fazendo uma cópia de arquivo no seu próprio diretório:

```
$ cp texto3.txt texto4.txt
```

Comando `mv`

- Para movermos um arquivo do ou no diretório `/home/edwar/teste` basta executarmos o comando `mv`. Exemplos:
  - movendo um arquivo de outro diretório para o diretório `/home/edwar/teste`, estando neste diretório:

    ```
    $ mv /tmp/texto5.txt .
    ```
  - copiando um arquivo para fora do diretório `/home/edwar/teste`, estando neste diretório

    ```
    $ mv texto5.txt /tmp
    ```
  - movendo os dados de um arquivo para outro arquivo no mesmo diretório:

    ```
    $ mv texto3.txt texto4.txt
    ```

    (o arquivo `texto3.txt` será apagado)

Comando `rm`

- Para apagarmos um arquivo no diretório `/home/edwar/teste` basta executarmos o comando `rm`. Exemplos:
  - no diretório `/home/edwar/teste` :

```
$ rm texto.txt
```

  - fora do diretório `/home/edwar/teste` :

```
$ rm ~/teste/texto.txt
```

```
$ rm /home/edwar/teste/texto.txt
```

Comando `cat`

- Para visualizarmos o conteúdo de um arquivo no diretório `/home/edwar/teste` basta executarmos o comando `cat`. Exemplos:
  - no diretório `/home/edwar/teste` :

```
$ cat texto.txt
```

  - fora do diretório `/home/edwar/teste` :

```
$ cat ~/teste/texto.txt
```

```
$ cat /home/edwar/teste/texto.txt
```

Comando `cat`

- O comando `cat` é normalmente utilizado para exibir o conteúdo de arquivos, mas ele possui outros recursos;
- vamos criar uma miniagenda com o comando `cat`;
- execute o comando:

```
$ cat > agenda
```

- digite cinco nomes quaisquer e seus telefones, exemplo:

```
$ Paulo César [Tab] 9111-7788
$ Pedro Henrique [Tab] 9999-7744
$ Maria Emília – 8888-5544
$ Cristiane – 9000-2323
$ Ana – 8989-3465
[Crtl + d]
```

- agora digite o comando `ls` para listar os arquivos do diretório e verá que o arquivo agenda foi criado e
- para finalizar execute o comando `cat agenda` para ver o que está dentro do arquivo.

Comando `cat`

- Agora façamos os seguintes teste, vamos executar os seguintes comandos e observar suas saídas:

```
$ cat -t agenda
$ cat -e agenda
$ cat -et agenda
$ cat -n agenda
```

- os comandos acima nos mostram respectivamente:
  - as tabulações "<Tab>" (^ - caractere que representa a tabulação) que o arquivo agenda possui,
  - os "<Enter>" ($ - caractere que representa o fim de linha) que o arquivo agenda possui,
  - as tabulações e os "<Enter>" ao mesmo tempo e
  - as linhas numeradas.

Comando `touch`

- Se queremos criar um arquivo texto vazio, temos diversos caminhos para fazer isto. Um dos mais práticos e rápidos é usando o comando `touch`, que funciona assim:

```
touch nomeDoArquivo.txt
```

- exemplo:

```
$ touch texto.txt
```

- o comando acima cria o arquivo `texto.txt` no diretório atual.

Comando `head` e `tail`

- O comando head exibe o início dos dados de um arquivo. Por padrão ele exibe as 10 primeiras linhas. Sintaxe:

```
$ head nomeDoArquivo.ext
```

- mas usando o parâmetro "-n" podemos limitar o número de linhas a serem exibidas. Suponhamos que queremos exibir somente as 3 primeiras linhas do arquivo, então:

```
$ head -n 3 nomeDoArquivo.ext
```

- o mesmo serve para o comando `tail` (que exibe, por padrão, as 10 últimas linhas do arquivo).

Localizando Arquivos

- Para localizarmos arquivos no sistema operacional podemos utilizar os seguintes comandos:
  - `find <startingdirectory> <opções> <termo de busca>`

    exemplo:

    `$ find . -name \*.bash*`
  - `locate <termo de busca>`

    `exemplo:`

    `$ locate *.bash*` (o comando `locate` não vem instalado por padrão e assim sendo é necessário fazer sua instalação antes de usá-lo)

Ajuda

- Sempre que precisarmos consultar alguma informação sobre algum comando no Linux, podemos fazer três tipos de consulta. Para tanto, tomemos como exemplo o comando `cp` seguinte:

```
$ cp --help
----------------
$ man help
----------------
$ help cp
```

lembrando que nem sempre as três maneiras vão funcionar para todos os comandos.

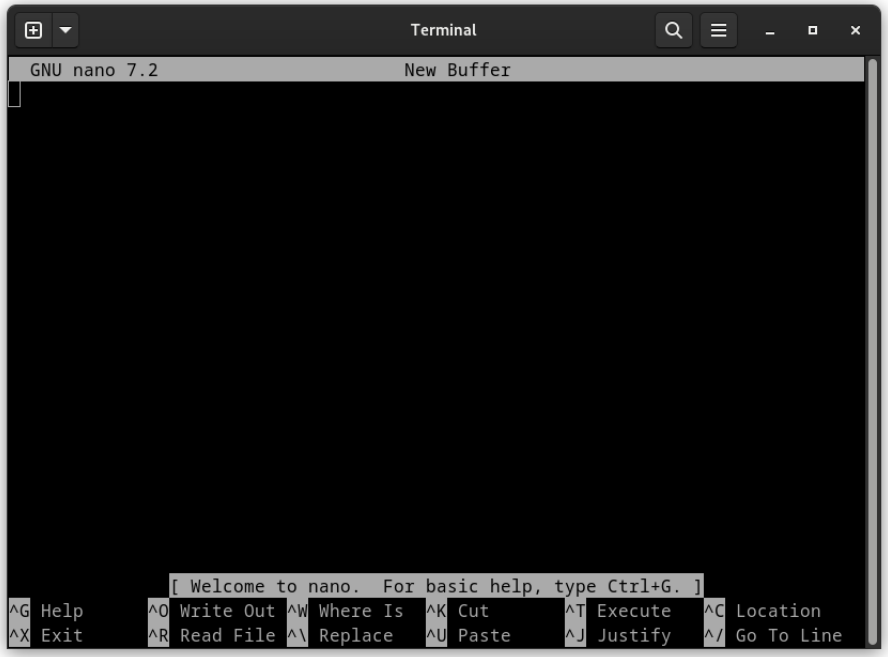
Descrição dos Comandos

- Para visualizar uma descrição simplificada dos comandos, pode-se usar o comando `whatis`, desta forma:

```
$ whatis cp
```

O editor Nano

- O Nano é um software para edição de textos no terminal;
- está disponível em quase todas as distribuições Linux e
- é fácil de usar, pois possui uma interface intuitiva;



Comando `nano`

- Execute o `Nano` e crie um arquivo com o seguinte texto (sem as aspas):

  "Minha agenda de nomes:"

- salve o arquivo com o seguinte nome:

  `titulo.txt`

- essa atividade foi só para você lembrar de como se usa o `Nano`.

Referências

- GUIA FOCA GNU/Linux. Iniciante. Disponível em: <http://www.guiafoca.org/cgs/guia/iniciante/ch-intro.html>. Acesso em: 27 jul. 2017.

- MOTA FILHO, J. E. Descobrindo o Linux. 3. ed. São Paulo: Novatec, 2012.

- ROMERO, D. Começando com Linux: Comandos, serviços e administração. São Paulo: Casa do Código, 2013.

- TRYBE. Como localizar arquivos no Linux. Disponível em: https://blog.betrybe.com/localizar-arquivos-no-linux/. Acesso em: 24 mar. 2025.