

**INSTITUTO
FEDERAL**
São Paulo

PFDA1 - PRÁTICAS E FERRAMENTAS
DE DESENVOLVIMENTO DE
SOFTWARE

THIAGO J. INOCÊNCIO¹

¹ Departamento de Informática e Turismo, Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, São Paulo, Brasil

SUMÁRIO

1	Terminal e Linha de Comando	3
1.1	Shell Script	3
1.2	Diretório	4
1.3	Comandos	5

LISTA DE FIGURAS

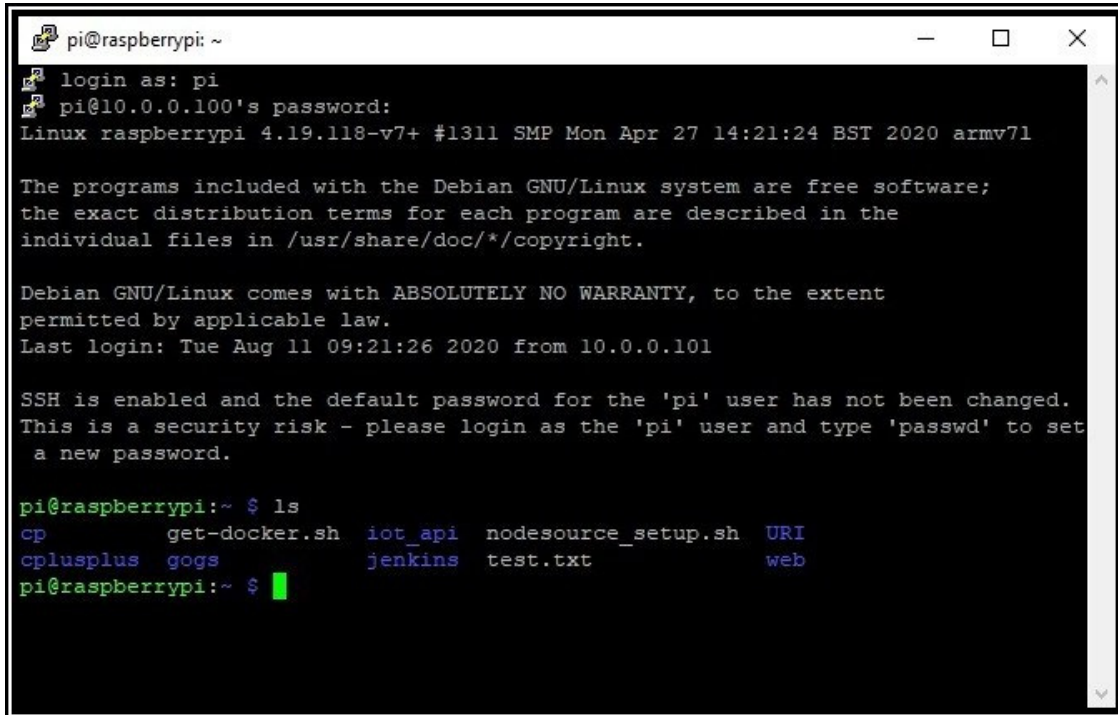
Figura 1	Ilustração de um terminal de linha de comando	3
Figura 2	Árvore de diretório de um sistema Linux/UNIX	4
Figura 3	Exemplo da execução do comando <code>pwd</code>	6
Figura 4	Exemplo da execução do comando <code>ls</code> para listagem de arquivos e pastas	6
Figura 5	Exemplo da execução do comando <code>ls</code> com parâmetro <code>-la</code>	8

LISTA DE TABELAS

Tabela 1	Comandos para operações em diretórios	5
Tabela 2	Parâmetros úteis do comando <code>ls</code>	7
Tabela 3	Explicação do retorno do comando <code>ls</code> com parâmetro <code>-la</code>	7
Tabela 4	Explicação do retorno do comando <code>ls</code> com parâmetro <code>-la</code>	8

1 TERMINAL E LINHA DE COMANDO

Um terminal (linha de comando) é uma interface utilizada para se comunicar com o sistema operacional. A partir de um terminal, o usuário pode enviar comandos ao sistema operacional para que o computador realize alguma tarefa. A Figura 1 mostra um terminal.



```
pi@raspberrypi: ~
login as: pi
pi@10.0.0.100's password:
Linux raspberrypi 4.19.118-v7+ #1311 SMP Mon Apr 27 14:21:24 BST 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Aug 11 09:21:26 2020 from 10.0.0.101

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $ ls
cp      get-docker.sh  iot_api  nodesource_setup.sh  URI
cplusplus  gogs          jenkins  test.txt              web
pi@raspberrypi:~ $
```

Figura 1: Terminal de linha de comando.

Embora seja possível interagir com o sistema operacional por meio de uma interface gráfica¹, muitas funcionalidades avançadas são mais facilmente realizadas utilizando um terminal. Por exemplo, configuração de rede, administração de contas de usuários etc. A Figura X mostra o mesmo diretório exibido na Figura 1 só que a partir de uma interface gráfica.

Podemos citar alguns exemplos de comandos que podem ser utilizados em terminais de sistemas UNIX/Linux, como: **date**, **whoami**, **pwd** etc. Nas próximas seções aprenderemos mais sobre comandos via terminal.

1.1 Shell Script

De acordo com Jargas [2] um script é uma lista de comandos para serem executados em sequência (um comando após o outro). Um roteiro predefinido de comandos e parâmetros.

Um Script Shell é um arquivo com uma lista de comandos que são executados em sequência e são interpretados por programas do tipo shell. No linux os interpretadores Shell mais conhecidos são o sh e o bash. No Windows podemos citar o PowerShell, entretanto nem todos os comandos que existem no Shell de sistemas UNIX são compatíveis com o PowerShell de sistemas Windows. O Algoritmo 1.1 mostra um script shell básico para exibir a mensagem 'Hello World':

¹ Ambiente visual com janelas

Algorithm 1: Sheel Script básico para exibir a mensagem Hello World

```

1  #!/bin/bash
2  echo "Hello World"

```

Conforme visto no exemplo, a primeira linha de todo script sheel deve conter o local do sistema que está o interpretador sheel. No caso desse interpretador shell utilizado foi o *bash* que está localizado no diretório */bin/bash*.

1.2 Diretório

Diretório² é o local utilizado para armazenar conjuntos de arquivos para melhor organização e localização dentro de um sistema de arquivos. Em alguns sistemas como os Linux/UNIX os nomes de diretórios são *Case Sensitive*, ou seja, há diferenciação da nomenclatura de uma pasta chamada Documentos e DOCUMENTOS. A Figura 2 mostra um exemplo da árvore de diretórios de um sistema Linux/UNIX. O primeiro diretório / mostrado no topo da figura é conhecido como diretório raiz, a partir dele é iniciado uma hierárquica de subdiretórios (i.e bin, boot, dev etc) cada um com seus arquivos e subdiretórios. Como veremos na seção 1.3.2, existem diversos comandos que o usuário do sistema precisa saber para navegar e manipular os diretórios e seus arquivos.

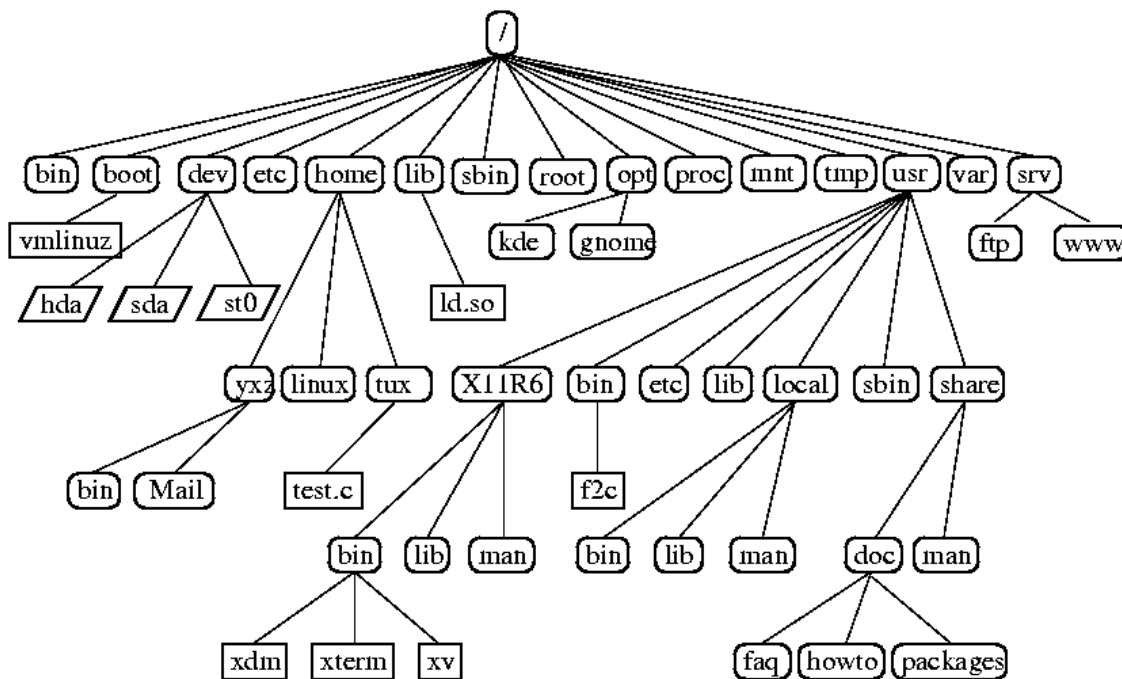


Figura 2: Árvore de diretório de um sistema Linux/UNIX [1]

Os caminhos na estrutura de diretórios são os caminhos que o usuário deve percorrer para chegar até um arquivo ou diretório desejado. Na figura 2 que mostra a hierarquia de diretórios de um sistema Linux/UNIX podemos chegar até o diretório *home/tux* fazendo o caminho que começa no diretório raiz */* e vai até */home/linux*. Quando é feito um caminho partindo-se da origem (*/*) chamamos de um caminho (path) absoluto. Por exemplo, o caminho */home/tux* é um caminho absoluto que parte do diretório raiz até o diretório */home/tux*.

Caso o caminho não inicie da raiz (*/*) e sim de outro diretório chamamos de caminho relativo, pois o caminho é relativo a um determinado diretório. Por exemplo, se estivermos no diretório */usr/local* e

² De acordo com [3] o termo diretório é sinônimo de pasta

precisamos acessar o diretório `/usr/local/bin`, pelo caminho absoluto podemos acessar a pasta usando `cd /usr/local/bin`, no entanto, como estamos a um diretório `bin` na hierarquia de diretórios, tudo que precisamos fazer é utilizar o caminho relativo do diretório atual (local) e digitar `cd bin`.

1.3 Comandos

Comandos são muito utilizados em ambientes de execução de linha de comando. Como vimos na seção 1.1 é possível criar arquivos (scripts) contendo listagens de comandos para serem executados pelo sistema. Essas listagens de comandos são utilizadas pelos administradores de sistemas para executar tarefas repetitivas de maneira automatizada.

Na próxima seção, apresentaremos os principais comandos utilizados para navegação em diretórios.

1.3.1 Comandos de Navegação

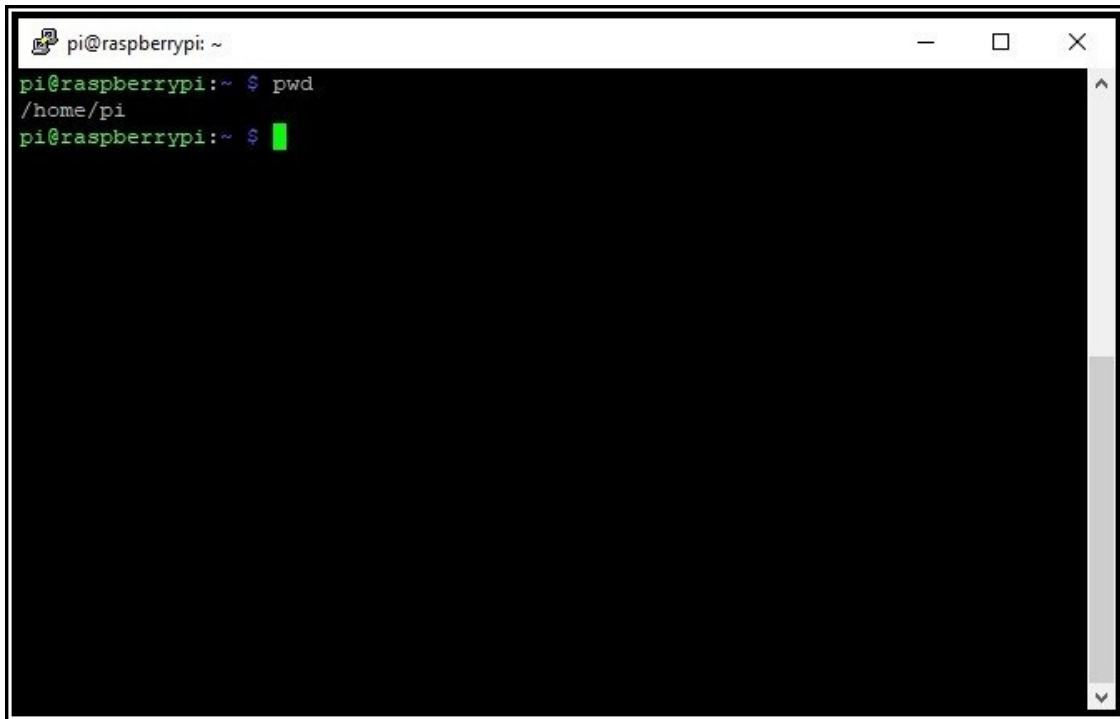
A Tabela 2 mostra os principais comandos para navegação em diretórios.

Tabela 1: Comandos para operações em diretórios

pwd	Mostra o diretório atual
ls	Lista os arquivos e diretórios do diretório atual <i>dir</i>
cd <i>dir</i>	Acessa o diretório <i>dir</i>

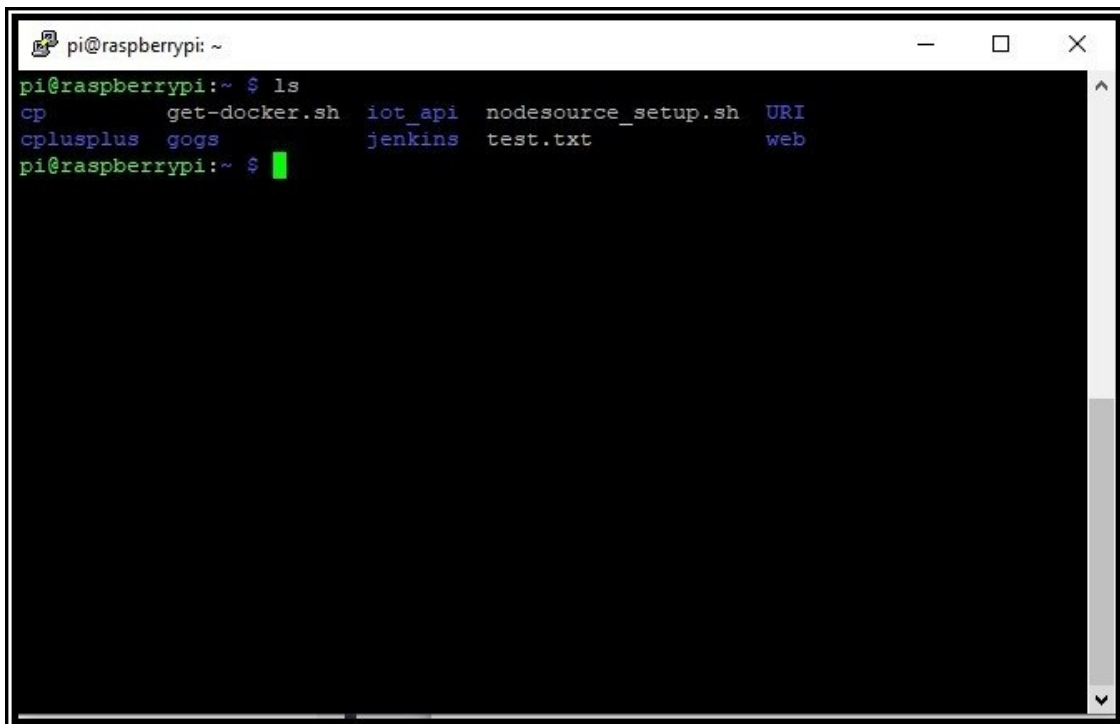
Ao digitar o comando **pwd** no terminal será exibido em qual diretório o usuário se encontra. Esse comando é útil quando o usuário precisa saber o diretório atual antes de executar algum comando. Como pode ser observado na Figura 3, após a execução do comando o retorno no terminal será o diretório atual, neste exemplo `/home/pi`.

O comando **ls** é muito utilizado para visualizar os arquivos e pastas presentes dentro de um diretório. A Figura 4 mostra a execução do documento `ls` em um terminal. O comando **ls** pode ser usado também com uma série de parâmetros. A Tabela 2 mostra alguns parâmetros úteis para serem utilizando com comando **ls**. Após a execução do comando **ls -la** (cf. Tabela 2), A Figura 5 mostra um exemplo da execução do comando `ls` com esses parâmetros. Na Tabela 4 é apresentado uma descrição do retorno da execução do comando `ls -la` conforme observado na Figura 5.

A terminal window titled 'pi@raspberrypi: ~' with standard window controls. The prompt is 'pi@raspberrypi:~ \$'. The command 'pwd' has been entered and executed, resulting in the output '/home/pi'. The prompt is now 'pi@raspberrypi:~ \$' with a green cursor.

```
pi@raspberrypi:~ $ pwd
/home/pi
pi@raspberrypi:~ $
```

Figura 3: Exemplo da execução do comando pwd

A terminal window titled 'pi@raspberrypi: ~' with standard window controls. The prompt is 'pi@raspberrypi:~ \$'. The command 'ls' has been entered and executed, resulting in a two-column listing of files and directories: 'cp', 'get-docker.sh', 'iot_api', 'nodesource_setup.sh', and 'URI' in the first column; 'cplusplus', 'gogs', 'jenkins', 'test.txt', and 'web' in the second column. The prompt is now 'pi@raspberrypi:~ \$' with a green cursor.

```
pi@raspberrypi:~ $ ls
cp      get-docker.sh  iot_api  nodesource_setup.sh  URI
cplusplus  gogs          jenkins  test.txt              web
pi@raspberrypi:~ $
```

Figura 4: Exemplo da execução do comando ls para listagem de arquivos e pastas

O comando **cd** é utilizado para navegação entre diretórios e para isso o usuário precisa ter permissão de leitura do diretório que ele pretende acessar. A sintaxe do comando **cd** é a seguinte: **cd** *[diretório]*, sendo *[diretório]* o nome do diretório que se deseja acessar.

Tabela 2: Parâmetros úteis do comando **ls**

ls -a [diretório]	Lista todos os itens incluindo os ocultos de um <i>diretório</i> . Caso o parâmetro [diretório] não seja fornecido, o diretório . (atual) será utilizado.
ls -l [diretório]	Parecido com o comando anterior (ls -a [diretório]), porém exibe informações detalhadas linha a linha sobre os arquivos e pastas.
ls -la [diretório]	Parecido com o comando anterior (ls -l [diretório]), porém exibe informações detalhadas linha a linha sobre os arquivos e pastas visíveis e ocultos.
ls -F [diretório]	Mostra os arquivos e pastas em [diretório] e adiciona uma barra \ quando o item for do tipo diretório
ls -lh [diretório]	Mostra o tamanho dos arquivos em Kbytes, Mbytes e Gbytes presentes em [diretório]
ls -n [diretório]	Mostra a identificação numérica de usuários e grupos ao invés dos nomes.
ls -r [diretório]	reverte a ordem da listagem.
ls -S [diretório]	ordena pelo tamanho do arquivo.
ls -t [diretório]	ordena pela data do arquivo.
ls -X [diretório]	ordena pela extensão do arquivo.

Tabela 3: Explicação do retorno do comando **ls** com parâmetro **-la**

-rw-r--r--	Permissões de acesso ao arquivo/diretório
1	Caso seja um diretório, mostra a quantidade de subdiretórios existentes dentro dele. Caso for um arquivo o valor será 1.
pi	Nome do usuário dono do arquivo/diretório
pi	Nome do grupo ao qual o arquivo/diretório pertence
13857	Tamanho do arquivo em bytes
Jun	Mês que o arquivo/diretório foi criado ou modificado.
5	Dia que o arquivo/diretório foi criado ou modificado.
23:31	horário que o arquivo/diretório foi criado ou modificado.
get-docker.sh	nome do arquivo.

A Tabela X apresenta alguns exemplos de utilização do comando **cd**.

```

pi@raspberrypi:~ $ ls -la
total 112
drwxr-xr-x 13 pi pi 4096 Aug 13 05:13 .
drwxr-xr-x 7 root root 4096 Jun 17 17:57 ..
-rw-r--r-- 1 pi pi 6325 Aug 11 14:38 .bash_history
-rw-r--r-- 1 pi pi 220 May 27 08:10 .bash_logout
-rw-r--r-- 1 pi pi 3523 May 27 08:10 .bashrc
drwxr-xr-x 2 root root 4096 Jul 18 04:09 cp
drwxr-xr-x 2 pi pi 4096 Jun 7 21:07 cplusplus
-rw-r--r-- 1 pi pi 13857 Jun 5 23:31 get-docker.sh
-rw-r--r-- 1 pi pi 72 Jun 5 23:50 .gitconfig
drwxr-xr-x 3 pi pi 4096 Jun 5 20:22 .gnupg
drwxr-xr-x 2 pi pi 4096 Jun 5 23:53 gogs
drwxr-xr-x 3 pi pi 4096 Jun 10 17:28 .local
drwxr-xr-x 2 pi pi 4096 Jun 17 20:50 .local
drwxr-xr-x 3 pi pi 4096 Jun 5 23:39 .local
-rw-r--r-- 1 pi pi 12991 Jun 10 17:34 .local
drwxr-xr-x 4 pi pi 4096 Jun 14 12:28 .npm
-rw-r--r-- 1 pi pi 807 May 27 08:10 .profile
drwxr-xr-x 3 pi pi 4096 Jun 7 21:45 .ssh
-rw-r--r-- 1 pi pi 0 Aug 11 10:50 test.txt
-rw-r--r-- 1 pi pi 0 Aug 13 05:13 text.backup
drwxr-xr-x 3 pi pi 4096 Jun 7 23:20 URI
drwxr-xr-x 3 root root 4096 Jul 18 04:10 web
-rw-r--r-- 1 pi pi 165 Jun 5 22:24 .wget-hsts
pi@raspberrypi:~ $

```

Figura 5: Exemplo da execução do comando ls com parâmetro -la

Tabela 4: Explicação do retorno do comando ls com parâmetro -la

cd .	Mantém no diretório atual, uma vez que . é um aliás que representa o diretório atual.
cd	Ao digitar apenas o comando cd sem o parâmetro [diretório] o terminal retornará ao diretório home do usuário (nesse caso /home/pi)
cd ~	Mesmo retorno do comando anterior (cd).
cd /	Terminal retorna ao diretório raiz sistema.
cd -	Terminal retorna ao último diretório acessado.
cd ..	Retorna um diretório na hierarquia. Por exemplo: /home/pi => /home
cd ../[diretório]	Retorna um diretório na hierarquia e depois acessa o diretório [diretório]

1.3.2 Comandos para manipulação de arquivos e diretórios

mkdir é um comando utilizado para criar um diretório no sistema. Como foi visto na seção 1.2, um diretório é usado para armazenar arquivos e também outros diretórios. A sintaxe do comando **mkdir** é a seguinte: **mkdir** [opções] [caminho/diretório]. Nesse caso:

- *[caminho]*: é o caminho na árvore de diretórios que o novo diretório será criado.
- *diretorio*: nome do diretório que será criado.
- *opções*:
 - **-p** criar toda estrutura de diretórios conforme o caminho informado. Por exemplo, para o comando **mkdir -p /home/thiago/exemplo/novapasta**, caso o diretório */exemplo* não exista, ele será criado se o usuário fornecer a opção **-p**.
 - **-verbose** para cada diretório criado será mostrado uma mensagem sobre a criação. **-p**.

É possível também criar muitos diretórios com apenas um comando **mkdir**, para isso o usuário precisa utilizar a seguinte sintaxe:

- **mkdir** *[opcoes]* *[caminho/diretório]* *[camibho1/diretorio1]* *[camibho2/diretorio2]*

rm comando utilizado para apagar arquivos e diretórios e seus subdiretorios. A sintaxe do comando **rm** é a seguinte: **rm** *[opcoes]* *[caminho][arquivo/diretório]* *[caminho][arquivo1/diretorio1]*. Nesse caso:

- **[caminho]**: localização do arquivo ou diretório que será apagado. Caso o caminho não seja fornecido, será considerado que o arquivo/diretório esteja no diretório atual do usuário.
- **[arquivo/diretório]** nome do arquivo ou diretório que será apagado
- *Opcoes*:
 - **-i** pergunta antes de remover um arquivo/diretorio.
 - **-v** mostra os arquivos na medida que são removidos.
 - **-r** remove arquivos e diretórios em subdiretorios. Nesse caso ele apaga recursivamente todos os subdiretorios do diretório fornecido no comando **rm**.
 - **-f** remove os arquivos sem perguntar.

Alguns exemplos de uso do comando **rm**:

- **rm aula01.txt** - apaga o arquivo *aula01.txt* no diretório atual.
- **rm -r meusdocumentos** - apaga recursivamente todos os arquivos e subdiretorios do diretório *meusdocumentos*.
- **rm -r meusdocumentos/*** - apaga todos os arquivos e subdiretorios do diretório *meusdocumentos*, mas não apaga o diretório *meusdocumentos*.
- **rm *.txt** - apaga todos os arquivos do diretório atual que tenham a extensao *.txt*.
- **rm -r ../outrosarquivos/*** - sobe um nível na estrutura de diretórios e apaga todos os arquivos e subdiretorios do diretório *outrosarquivos*.
- **rm -r /home/thiago/backup** - apaga todos os arquivos e subdiretorios do diretório */home/thiago/backup* inclusive o próprio diretório.

O comando **cp** é utilizado para copiar arquivos e diretorios. tal comando copia arquivos da *[origem]* para o *[destino]*. Ambos *[origem]* e *[destino]* terão o mesmo arquivo/diretório de *[origem]* após a execução do comando. A sintaxe do comando **cp** é a seguinte: **cp** *[opções]* *[origem]* *[destino]*. Nesse caso:

- **[origem]** é o arquivo ou diretório que será copiado.
- **[destino]** O caminho do arquivo ou diretório que o arquivo/diretório presente em [origem] será copiado.
- **opcoes:**
 - **-i** pergunta antes de substituir um arquivo existente.
 - **-f** substitui tudo sem perguntar.
 - **-R** copia recursivamente os arquivos e sub-diretorios de [origem] para [destino].

Alguns exemplos de utilização do comando **cp** são vistos a seguir:

- **cp** aula01.txt teste02.txt - copia o arquivo aula01.txt para o arquivo aula02.txt. Se o arquivo aula02.txt não existir ele é criado após a execução do comando.
- **cp** aula01.txt /aulas - copia o arquivo aula01.txt para o diretório /aulas.
- **cp *** /aulas - copia todos os arquivos do diretório atual para o diretório /aulas.
- **cp** aulas/* aulasbackup - copia todos os arquivos presentes no diretório aulas para o diretório aulas-backup
- **cp** backupaulas/* . - copia todos os arquivos em backupaulas para o diretório atual.

O comando **mv** é utilizado para mover ou renomear um arquivo ou diretório. O processo é semelhante ao do comando **cp** mas no final da execução do comando o arquivo de [origem] é movido para [destino] e depois é apagado de [origem]. A sintaxe do comando **mv** é a seguinte: **mv** [opcoes] [origem] [destino]. Nesse Caso:

- **[origem]** é o arquivo ou diretório que será movido.
- **[destino]** O caminho do arquivo ou diretório que o arquivo/diretório presente em [origem] será movido.
- **opcoes:**
 - **-f** - substitui o arquivo de destino sem perguntar
 - **-v** mostra os arquivos que estão sendo movidos

Alguns exemplos de utilização do comando **mv** são vistos a seguir:

- **mv** aula01.txt aula02.txt - renomeia o arquivo aula01.txt para aula02.txt
- **mv** aulas/* aulasbackup - move todos os arquivos presentes no diretório aulas para o diretório aulasbackup.
- **mv** aulasbackup backupaulas - renomeia o diretório aulasbackup para backupaulas.

O comando **touch** é utilizado principalmente para alterar a data de um arquivo. Também é utilizado com frequência para criar um novo arquivo. A seguir é mostrado a sintaxe do comando: **touch** [opções] [arquivos]. Em nosso caso não estamos preocupados com a alteração da data de um arquivo, mas sim a criação dele, portanto vejamos alguns exemplos:

- **touch** aula01.txt - cria o arquivo aula01.txt
- **touch** aula01.txt aula02.txt aula03.txt - cria os arquivos aula01.txt, aula02.txt e aula03.txt no diretório atual.
- **touch** aulas/aula01.txt - cria o arquivo aula01.txt no diretório aulas.

O comando **cat** mostra o conteúdo de um arquivo binário ou de texto. A sintaxe do comando é a seguinte: **cat** [opcoes] [diretório/arquivo]. Nesse caso:

- **[diretório/arquivo]** - localização do arquivo que se deseja visualizar o conteúdo.
- Opções:
 - **-n** - mostra o número de linhas enquanto o conteúdo do arquivo é listado.

Alguns exemplos de utilização do comando **cat** são vistos a seguir:

- **cat** aula01.txt - mostra o conteúdo do arquivo aula 01.txt
- **cat** aulas/aula01.txt - mostra o conteúdo do arquivo aula01.txt que está dentro do diretório aulas.

O comando **head** mostra as primeiras linhas de um arquivo de texto. A Sintaxe do comando é a seguinte: **head** [opcoes] [diretório/arquivo]. Nesse caso:

- **[diretório/arquivo]** - localização do arquivo que se deseja visualizar o conteúdo.
- Opções
 - **-n** - mostra o número de linhas enquanto o conteúdo do arquivo é listado.

O comando **tail** mostra as últimas linhas de um arquivo de texto. A Sintaxe do comando é a seguinte: **tail** [opcoes] [diretório/arquivo]. Nesse caso:

- **[diretório/arquivo]** - localização do arquivo que se deseja visualizar o conteúdo.
- Opções
 - **-n** - mostra o número de linhas do final do arquivo
 - **-f** mostra continuamente linhas adicionadas no final do arquivo.

REFERENCES

- [1] Bartsch et al. Suse Linux. User Guide. <https://www-uxsup.csx.cam.ac.uk/pub/doc/suse/suse9.0/userguide-9.0/ch24s02.html>. [Online; acessado em 20 agosto de 2020].
- [2] A.M. Jargas. *Shell Script Profissional*. Novatec Editora, 2017. ISBN 9788575225769. URL <https://books.google.com.br/books?id=B2YkDwAAQBAJ>.
- [3] Wikipedia. Diretório (computação) — Wikipedia, the free encyclopedia. [http://pt.wikipedia.org/w/index.php?title=Diret%C3%B3rio%20\(computa%C3%A7%C3%A3o\)&oldid=56988516](http://pt.wikipedia.org/w/index.php?title=Diret%C3%B3rio%20(computa%C3%A7%C3%A3o)&oldid=56988516), 2020. [Online; acessado em 05 Março de 2020].