



Introdução a Administração Linux com o Bash

Prof. Michel Sales Bonfim
Disciplina: Administração de SO Linux

Administrador de Sistemas

► Tarefas:

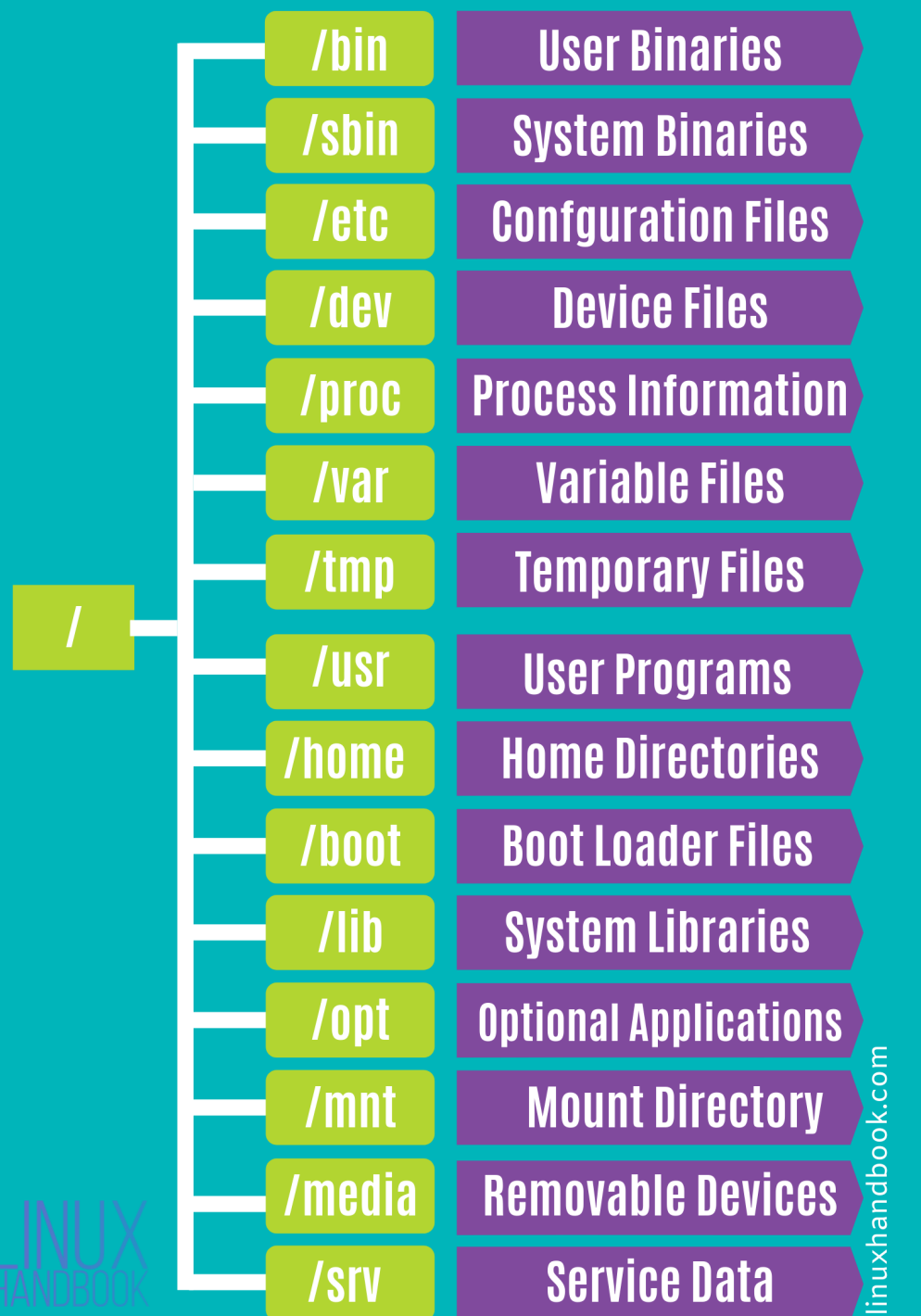
- Gerenciar servidores e serviços;
- Ajudar os usuários com problemas de configuração;
- Recomendar novos softwares;
- Manter a documentação atualizada.

► Para ser um administrador de sistemas Linux é necessário estar familiarizado com as versões Desktop e Server



Dentro do Linux

Estrutura de Diretórios



Estrutura de Diretórios

/

Diretório raiz. Contém todos os arquivos e diretórios do Linux

/bin

Contém arquivos programas do sistema que são usados com frequência pelos usuários.

/sbin

Diretório de programas usados pelo superusuário (root) para administração e controle do funcionamento do sistema.

/etc

Arquivos de configuração de seu computador local.

/dev

Contém arquivos usados para acessar dispositivos (periféricos) existentes no computador.

Estrutura de Diretórios

/usr

Contém todos os arquivos, executáveis, bibliotecas, fontes da maioria dos programas do sistema. Por este motivo, a maioria dos arquivos nele contidos são somente leitura (para o usuário normal)

'/usr/bin' contém comandos básicos do usuário

'/usr/sbin' contém comandos adicionais para o administrador

'/usr/lib' contém as bibliotecas do sistema

'/usr/share' contém documentação ou comum a todas as bibliotecas, por exemplo **'/usr/share/man'** contém o texto da página de manual

Estrutura de Diretórios

/home

O diretório inicial contém diretórios pessoais para os usuários.

Ao criar um usuário em seu sistema Linux , é prática geral criar um diretório pessoal para o usuário: /home/bob

O diretório pessoal contém os dados do usuário e os arquivos de configuração específicos do usuário.

Estrutura de Diretórios

/boot

Contém arquivos necessários para a inicialização do sistema.

/lib

Contém as bibliotecas necessárias aos binários nos diretórios /bin e /sbin.

/proc

Contém informações sobre os processos em execução e parâmetros do kernel.

/var

Arquivos de dados variáveis. Armazenam dados em tempo de execução, como: registro do sistema, rastreamento de usuários, caches e outros arquivos (logs)

/tmp

Diretório para armazenamento de arquivos temporários criados por programas.

Estrutura de Diretórios

/opt

É usado para instalar/armazenar arquivos de aplicativos de terceiros que não estão disponíveis no repositório da distribuição.

/media

Quando você conecta uma mídia removível, como um disco USB, cartão SD ou DVD, um diretório é criado automaticamente no diretório /media para eles. Você pode acessar o conteúdo da mídia removível neste diretório.

/mnt

É semelhante ao diretório /media, mas em vez de montar automaticamente a mídia removível, mnt é usado pelos administradores do sistema para montar manualmente um sistema de arquivos.

/srv

Contém dados de serviços fornecidos pelo sistema

/root

Diretório do usuário root.

Command Line Interface

Why is knowing the command line important?



Flexibility and mobility!

By understanding the foundation of Linux, you have the ability to work on ANY Linux distribution. This could mean one company with a mixed environment or a new company with a different Linux distribution.

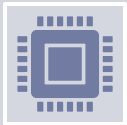
Command Line Interface (CLI)



Uma interface baseada em texto para o computador.



A CLI depende principalmente da entrada do teclado.



Tudo o que o usuário deseja que o computador faça é transmitido digitando comandos em vez de clicar em ícones.



Pode-se dizer que quando um usuário clica em um ícone, o computador está dizendo ao usuário o que fazer, mas, quando o usuário digita um comando, está dizendo ao computador o que fazer.

Command Line Interface (CLI)

CLI é difícil?

Vantagens do uso de um console

- Interagir com um console geralmente é mais rápido do que usar uma interface gráfica.
- Em um console, você pode executar lotes de comandos, sendo, portanto, ideal para a automação de tarefas para pipelines de integração contínua.
- Você pode usar um console para interagir com recursos de nuvem e outros recursos.
- Você pode armazenar comandos e scripts em um arquivo de texto e usar um sistema de controle do código-fonte.
 - Esse é provavelmente um dos maiores benefícios, porque seus comandos são repetíveis e auditáveis.
 - Em muitos sistemas, especialmente sistemas governamentais, tudo precisa ser rastreado e avaliado ou *auditado*.
 - As auditorias abrangem tudo, de alterações de banco de dados a alterações feitas por um script.

O que é o Bash?



Estrutura para automatizar tarefas administrativas no Linux

○ Bash consiste em duas partes:

- Shell de linha de comando;
- Linguagem de script

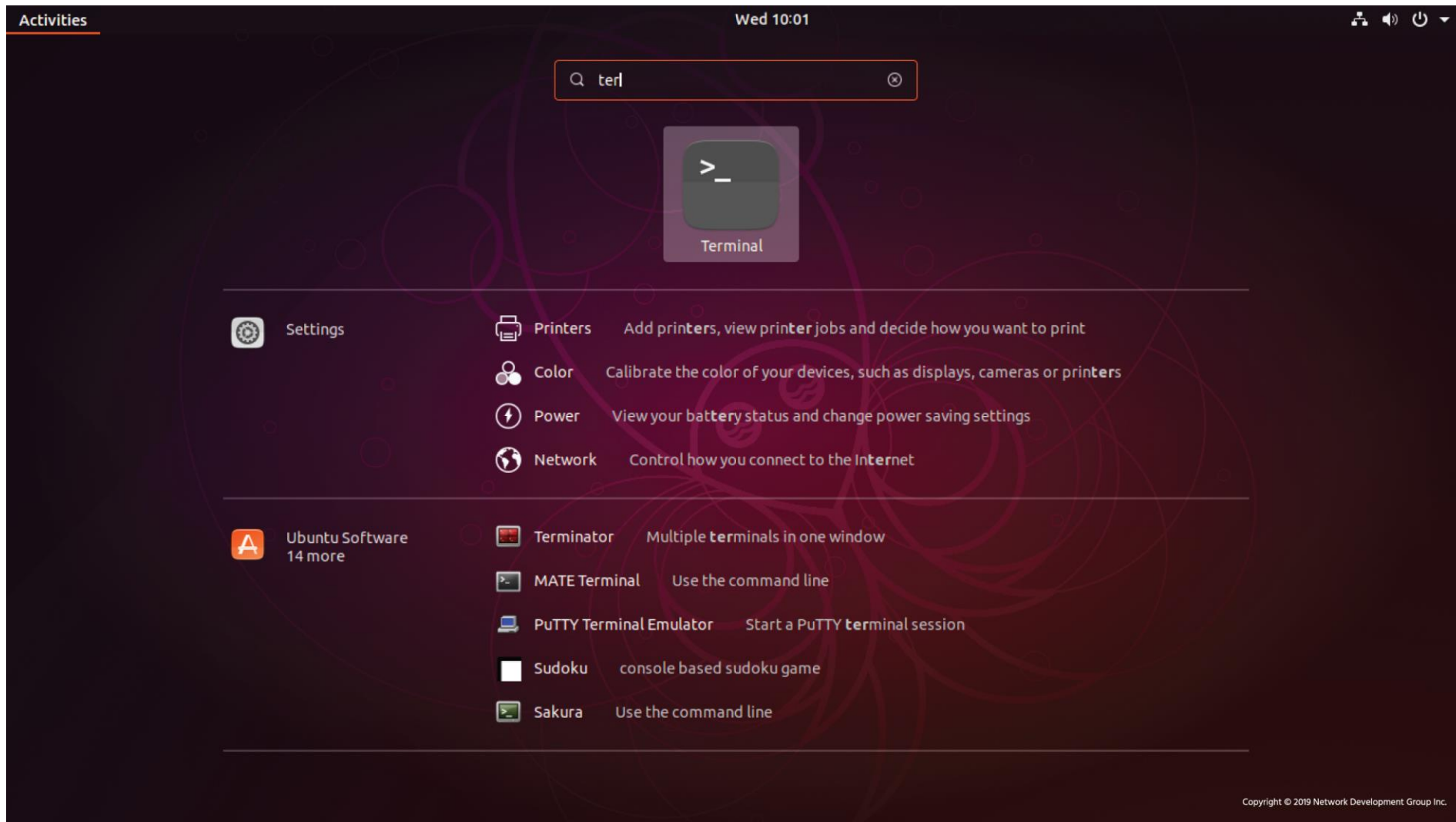
Você pode:

- Executar comandos em computadores locais ou remotos.
- Realizar tarefas como gerenciar usuários e automatizar fluxos de trabalho.
- Gerenciamento de recursos de nuvem e CI/CD (integração contínua e entrega contínua)

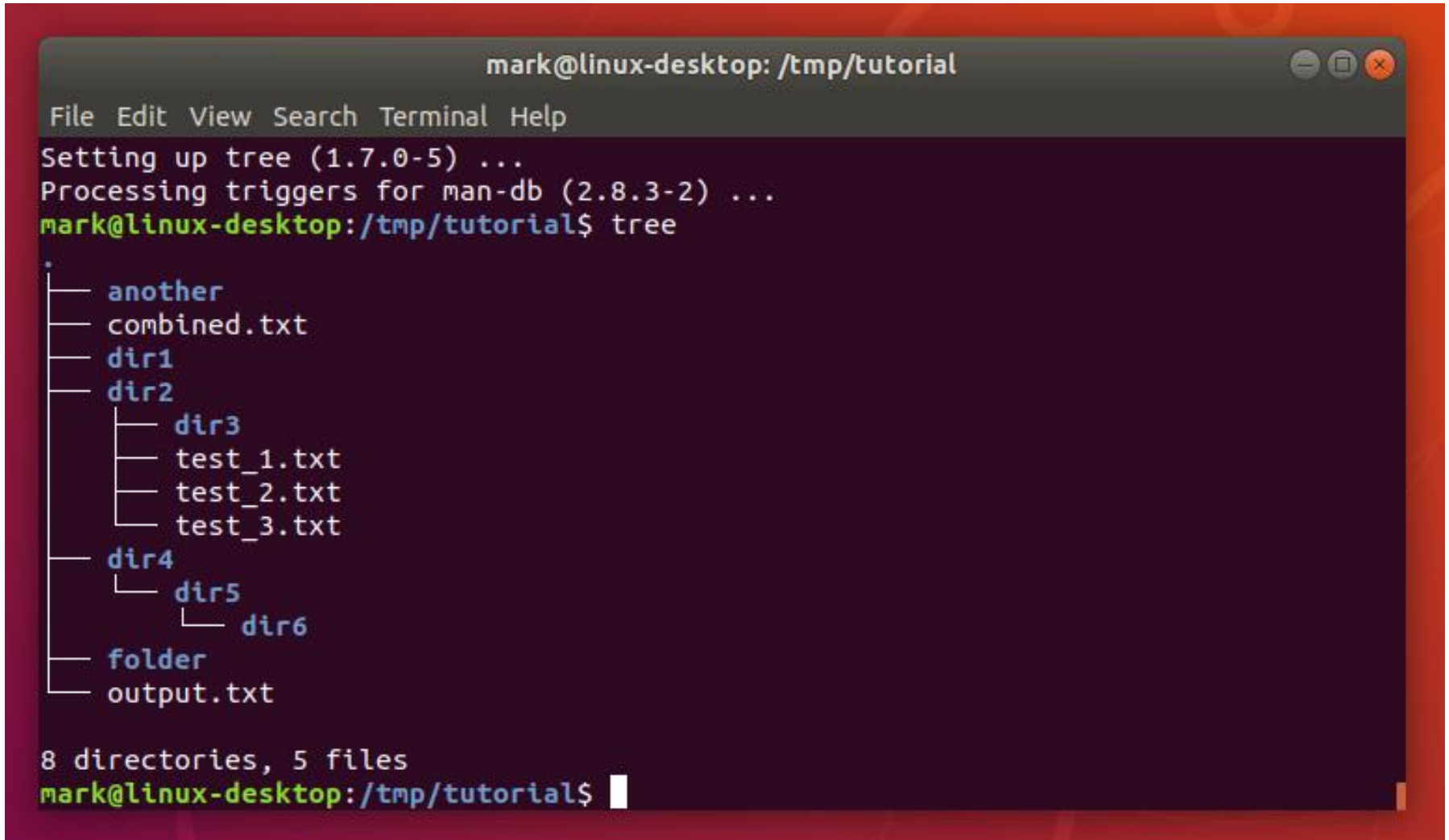
Bash fornece muitos recursos populares, como:

- **Histórico** de comandos e **edição inline**;
- **Scripting**:
 - A capacidade de colocar comandos em um arquivo e então interpretar o arquivo, resultando na execução de todos os comandos;
 - Este recurso também possui alguns recursos de programação, como **instruções condicionais e de repetição** e a capacidade de criar **funções**.
- **Aliases**: A capacidade de criar apelidos curtos para comandos mais longos.
- **Variáveis**: usadas para armazenar informações do shell Bash e do usuário. Essas variáveis podem ser usadas para modificar o funcionamento dos comandos e recursos, bem como fornecer informações vitais do sistema.

Acessando o CLI (pela GUI - Desktop)



Acessando o CLI (pela GUI - Desktop)



```
mark@linux-desktop: /tmp/tutorial
File Edit View Search Terminal Help
Setting up tree (1.7.0-5) ...
Processing triggers for man-db (2.8.3-2) ...
mark@linux-desktop:/tmp/tutorial$ tree
.
├── another
├── combined.txt
├── dir1
├── dir2
│   ├── dir3
│   │   ├── test_1.txt
│   │   ├── test_2.txt
│   │   └── test_3.txt
│   ├── dir4
│   │   └── dir5
│   │       └── dir6
├── folder
└── output.txt

8 directories, 5 files
mark@linux-desktop:/tmp/tutorial$
```

Acessando o CLI (Servidores)

```
Ubuntu 18.04 Ubuntu tty2
```

```
login do Ubuntu:
```

```
Ubuntu 18.04 Ubuntu tty2
```

```
login do Ubuntu: processar
```

```
Senha:
```

```
Os programas incluídos no sistema Ubuntu são softwares gratuitos;  
os termos exatos de distribuição para cada programa estão descritos no  
arquivos individuais em /usr/share/doc/*/copyright.
```

```
O Ubuntu vem com ABSOLUTAMENTE NENHUMA GARANTIA, na medida permitida por  
lei aplicável.
```

```
Bem vindo ao Ubuntu 18.04 LTS (GNU/Linux 4.4.0-72-genérico x86_64)
```

```
* Documentação: https://help.ubuntu.com/
```

```
212 pacotes podem ser atualizados.
```

```
91 atualizações são atualizações de segurança.
```

```
sue@ubuntu:~$w
```

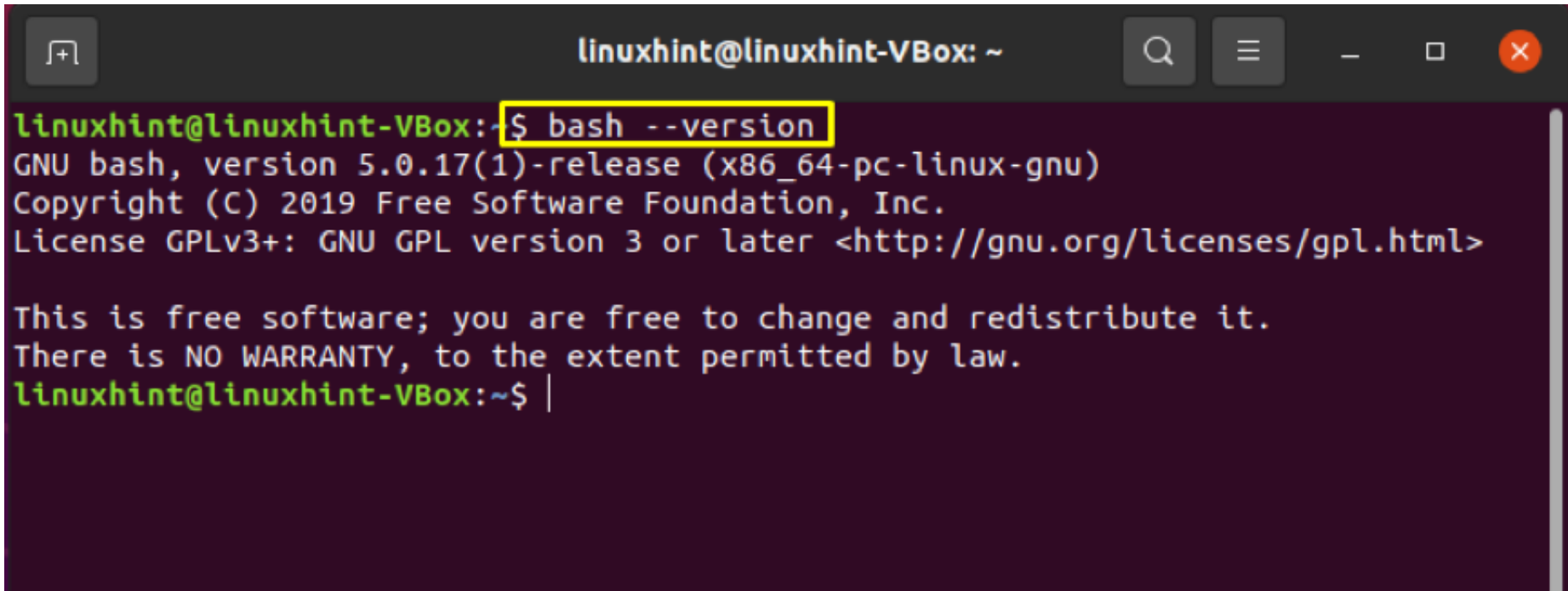
```
17:27:22 até 14 min, 2 usuários, média de carga: 1,73, 1,83, 1,69
```

```
USUÁRIO TTY DE LOGIN @ IDLE JCPU PCPU O QUE
```

```
processar tty2 20:08 14,35 0,05s 0,00sw
```

Vamos testar um primeiro comando?

Qual a versão do bash?

A terminal window titled 'linuxhint@linuxhint-VBox: ~' with standard window controls. The prompt is 'linuxhint@linuxhint-VBox:~\$' and the command 'bash --version' is entered and highlighted with a yellow box. The output shows the GNU bash version 5.0.17(1)-release for x86_64-pc-linux-gnu, copyright information for 2019, and the GPL license details. It also includes a disclaimer about free software and warranty.

```
linuxhint@linuxhint-VBox:~$ bash --version
GNU bash, version 5.0.17(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
linuxhint@linuxhint-VBox:~$ |
```

Prompt do Bash

Nome de usuário:

```
administrador de sistema @localhost : ~ $
```

Nome do sistema:

```
sysadmin@ localhost : ~ $
```

Diretório atual:

```
sysadmin@localhost : ~ $
```

O ~ símbolo é usado como abreviação do diretório inicial do usuário. Normalmente, o diretório inicial do usuário está no /home diretório e é nomeado após o nome da conta do usuário; por exemplo, /home/sysadmin

Comandos

Comandos

Programa de software que, quando executado na CLI, executa uma ação no computador

comando [opções] [argumentos]

[opções]:modifica o comportamento de um comando.

[argumentos]:especificar algo para o qual o comando agirá (como nome de arquivo ou nome de usuário)

OBSERVAÇÕES:

Comandos podem ser usados sozinho, sem nenhuma entrada adicional;

Comandos, opções e argumentos são **case-sensitive**.

Exemplo

Listar os arquivos de um diretório.

```
sysadmin@localhost : ~ $ ls  
Desktop Documentos Downloads Música Imagens Modelos públicos Vídeos
```

Com argumentos

```
sysadmin@localhost : ~ $ ls /etc/ppp  
ip-down.d ip-up.d
```


Exemplo

Com opções

```
sysadmin@localhost : ~ $ ls -l
total 0
drwxr-xr-x 1 sysadmin sysadmin 0 29 de janeiro 20:13 Desktop
drwxr-xr-x 1 sysadmin sysadmin 0 29 de janeiro 20:13 Documentos
drwxr-xr-x 1 sysadmin sysadmin 0 29 de janeiro 20:13 Downloads
drwxr-xr-x 1 sysadmin sysadmin 0 29 de janeiro 20:13 Música
drwxr-xr-x 1 sysadmin sysadmin 0 29 de janeiro 20:13 Imagens
drwxr-xr-x 1 sysadmin sysadmin 0 29 de janeiro 20:13 Público
drwxr-xr-x 1 sysadmin sysadmin 0 jan 29 20:13 Modelos
drwxr-xr-x 1 sysadmin sysadmin 0 29 de janeiro 20:13 Vídeos
```

Observe que, no comando acima, `-l` há uma letra “L” minúscula. Uma maneira fácil de lembrar isso é `-l` usar um mnemônico (código de programação fácil de memorizar) para listagem longa.

Fazendo combinações de opções e argumentos

```
sysadmin@localhost : ~ $ ls -lh /usr/bin/perl  
-rwxr-xr-x 2 root root 11K 4 de fevereiro de 2018 /usr/bin/perl
```

date

- Permite ver/modificar a Data e Hora do Sistema.

date MesDiaHoraMinuto[Ano.Segundos]

Onde:

MesDiaHoraMinuto[Ano.Segundos] São respectivamente os números do mês, dia, hora e minutos sem espaços. Opcionalmente você pode especificar o Ano (com 2 ou 4 dígitos) e os Segundos.

Observação:

Você precisa estar como usuário root para modificar a data e hora.

date

- Exemplo:
 - Se quiser mudar a Data para 25/12 e a hora para 08:15 digite: ***date 12250815***
 - Se quiser mudar a Data para 20/11/2010 e a hora para 22:15:40 digite: ***date 112022152010.40***

Histórico de comandos

- Quando um comando é executado no terminal, ele é armazenado em uma lista de histórico.
- Isso foi projetado para facilitar a execução do mesmo comando, eliminando posteriormente a necessidade de redigitar o comando inteiro.
- Pressionar a tecla **Seta para cima** ↑ exibe o comando anterior na linha de prompt. Pressionar a **tecla Enter** executa o comando exibido novamente.
- Ou podemos usar o seguinte comando:

history

Histórico de comandos

```
sysadmin@localhost:~$ date
```

```
Wed Dec 12 04:28:12 UTC 2018
```

```
sysadmin@localhost:~$ ls
```

```
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
```

```
sysadmin@localhost:~$ cal 5 2030
```

```
May 2030
```

```
Su Mo Tu We Th Fr Sa
```

```
1 2 3 4
```

```
5 6 7 8 9 10 11
```

```
12 13 14 15 16 17 18
```

```
19 20 21 22 23 24 25
```

```
26 27 28 29 30 31
```

```
sysadmin@localhost:~$ history
```

```
1 date
```

```
2 ls
```

```
3 cal 5 2030
```

```
4 history
```

Histórico de comandos

Se o comando desejado estiver na lista gerada pelo comando de histórico, ele poderá ser executado digitando um ponto de exclamação **!** **caractere** e **depois o número ao lado do comando.**

```
sysadmin@localhost:~$ history
```

```
1  date
```

```
2  ls
```

```
3  cal 5 2030
```

```
4  history
```

```
sysadmin@localhost:~$ !3
```

```
cal 5 2030
```

```
May 2030
```

```
Su Mo Tu We Th Fr Sa
```

```
1 2 3 4
```

```
5 6 7 8 9 10 11
```

```
12 13 14 15 16 17 18
```

```
19 20 21 22 23 24 25
```

Histórico de comandos

```
sysadmin@localhost:~$ history 3
```

```
6  date
7  ls /home
8  history 3
```

```
sysadmin@localhost:~$ !-3
```

```
date
```

```
Wed Dec 12 04:31:55 UTC 2018
```

```
sysadmin@localhost:~$ date
```

```
Wed Dec 12 04:32:36 UTC 2018
```

```
sysadmin@localhost:~$ !!
```

```
date
```

```
Wed Dec 12 04:32:38 UTC 2018
```


shutdown

- Desliga/reinicia o computador imediatamente ou após determinado tempo (programável) de forma segura.

shutdown [opções] [hora] [mensagem]

Onde:

hora Momento que o computador será desligado. Você pode usar HH:MM para definir a hora e minuto, MM para definir minutos, +SS para definir após quantos segundos, ou **now** para imediatamente (equivalente a +0).

mensagem Mensagem que será mostrada a todos os usuários alertando sobre o reinício/desligamento do sistema.

opções

- h Inicia o processo para desligamento do computador.
- r Reinicia o sistema

shutdown

- Exemplo
- ***shutdown -h now*** - Desligar o computador imediatamente.
- ***shutdown -r now*** - Reinicia o computador imediatamente.
- ***shutdown -r 15:00*** - O sistema será reiniciado às 15:00 horas”
- Faz o computador ser reiniciado às 15:00 horas enviando a mensagem: O sistema será reiniciado às 15:00 horas a todos os usuários conectados ao sistema.
- ***shutdown -r 20*** - Faz o sistema ser reiniciado após 20 minutos.

Tipos de Comandos

Tipo do comando

- ▶ O comando **type** pode ser usado para determinar informações sobre o tipo de comando.

\$ type comando

- ▶ Existem diversas fontes diferentes de comandos no shell da CLI, incluindo:
 - **Comandos internos**
 - **Comandos externos**
 - **Aliases**
 - **Funções**
- ▶ O comando **which** procura a localização de um comando pesquisando a variável **PATH**.

\$ which comando

Aliases

- Um alias pode ser usado para mapear comandos mais longos para sequências de teclas mais curtas.

```
sysadmin@localhost : ~ $ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```

Aliases

```
sysadmin@localhost : ~ $ alias mycal="cal 2019"
```

```
sysadmin@localhost : ~ $ mycal
```

```
2019
```

```
janeiro fevereiro março
```

```
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
```

```
1 2 3 4 5 1 2 1 2 6 7
```

```
8 9 10 11 12 3 4 5 6 7 8 9 3 4 5 6 7 8 9 13 14 15 16 17 18 19 10 11 12 13 14 15 16 10 11 12 13 14
```

```
15 16 20 21 2 2
```

```
23 24 25 26 17 18 19 20 21 22 23 17 18 19 20 21 22 23 27 28
```

```
29 30 31 24 25 26 27 28 24 25 26 27 28 29 30
```

```
31
```

Funções

- ▶ As funções também podem ser construídas usando comandos existentes para criar novos comandos ou para substituir comandos integrados ao shell ou comandos armazenados em arquivos.
- ▶ Normalmente, as funções são usadas para executar vários comandos. Para criar uma função, a seguinte sintaxe é usada:

```
nome_da_função ()  
{ comandos  
}
```

Funções

```
sysadmin@localhost : ~ $ my_report () {  
> ls Documentos  
> data  
> echo "Relatório do diretório de documentos"  
> }
```

```
sysadmin@localhost : ~ $ my_report  
Escola          alfa-terceiro.txt oculto.txt números.txt ortografia.txt  
Trabalho        alfa.txt letras.txt os.csv palavras  
adjetivos.txt animais.txt linux.txt pessoas.csv  
alfa-primeiro.txt food.txt longfile.txt profile.txt  
alpha-second.txt hello.sh newhome.txt red.txt  
Quarta, 13 de outubro 06:54:04 UTC 2021  
Relatório de diretório de documentos
```


Considerações sobre Aliases e Funções

- 1) Os aliases e funções criados dessa forma persistem apenas enquanto o shell está aberto.
- 2) Cada shell tem seus próprios aliases e funções, portanto, os aliases criados em um shell não estarão disponíveis em um novo shell aberto.
- 3) Para que estejam sempre disponíveis, aliases e funções devem ser carregados a partir dos arquivos de inicialização quando o shell é iniciado pela primeira vez.

Ajuda!

- ▶ Para visualizar uma página de manual de um comando, use o comando `man`.

\$ man comando

- ▶ Muitos comandos fornecerão informações básicas, muito semelhantes às SYNOPSIS encontradas nas páginas de manual, simplesmente usando o `--help` como opção do comando.

\$ comando --help

Variáveis

Variáveis

- ▶ Recurso que permite ao usuário ou ao shell armazenar dados.
- ▶ As variáveis recebem nomes e são armazenadas temporariamente na memória.
- ▶ Tipagem fraca e dinâmica.
- ▶ Existem dois tipos de variáveis usadas no shell Bash: **local** e **ambiente**

Variáveis Locais

- ▶ Variáveis locais ou shell existem apenas no shell atual.
- ▶ Eles são frequentemente associados a tarefas baseadas no usuário e estão em minúsculas por convenção.
- ▶ Por convenção, usam-se apenas letras minúsculas

```
sysadmin@localhost:~$ variable1='Something'
```

```
sysadmin@localhost:~$ echo $variable1  
Something
```

Variáveis de Ambiente

- ▶ Também chamadas de **variáveis globais**;
- ▶ Afetam diferentes comandos do Linux
- ▶ O sistema recria automaticamente variáveis de ambiente quando um novo shell é aberto.
- ▶ Exemplos: **PATH** , **HOME** e **HISTSIZE**.

```
sysadmin@localhost : ~ $ echo $HISTSIZE  
1000
```

```
sysadmin@localhost : ~ $ HISTSIZE=500  
sysadmin@localhost : ~ $ echo $HISTSIZE  
500
```

Variáveis de Ambiente

- ▶ Usa-se o comando **env** para listar as variáveis de ambiente.
- ▶ Caso queira filtrar um variável específica, basta usar o comando **grep** combinado com o **env**

```
sysadmin@localhost:~$ env | grep variable1
```

Variáveis de Ambiente

- ▶ O comando **export** é usado para criar variáveis de ambiente.

```
sysadmin@localhost:~$ export variable2='Else'  
sysadmin@localhost:~$ env | grep variable2  
variable2=Else
```

- ▶ O comando **export** também é usado para transformar uma variável local em uma variável de ambiente.

```
sysadmin@localhost:~$ export variable1  
sysadmin@localhost:~$ env | grep variable1  
variable1=Something
```


Variável de Ambiente

- ▶ Para alterar uma variável de ambiente:

```
sysadmin@localhost:~$ variable1=$variable1 ' '$variable2  
sysadmin@localhost:~$ echo $variable1  
Something Else
```

- ▶ Para remover uma variável:

```
sysadmin@localhost:~$ unset variable2
```

Manipulação de Variáveis

- ▶ As únicas vezes em que uma variável aparece "nua" - **sem o prefixo \$** - é:
 - Quando declarada ou atribuída;
 - Quando unset;
 - Quando exportada;
 - Em uma expressão aritmética entre parênteses duplos ((...))
- ▶ Observe que ***\$variable*** é na verdade uma forma simplificada de ***\${variable}***.
 - Em contextos onde a sintaxe ***\$variable*** causa um erro, a forma mais longa pode funcionar



Citando

Citando

- ▶ As **aspas** são usadas em toda a administração do Linux para informar ao sistema que as informações contidas entre aspas devem ser ignoradas ou tratadas de uma forma muito diferente da que normalmente seriam tratadas.
- ▶ Existem três tipos de aspas que têm um significado especial para o shell Bash:
 - aspas duplas "
 - aspas simples '
 - crase `

Aspas Duplas "

- ▶ Não interpreta alguns metacarateres ou caracteres especiais, incluindo caracteres glob ou curinga (*, ?, [,])
- ▶ Faz substituição de variáveis
- ▶ Faz substituição de comandos

```
sysadmin@localhost:~$ echo "The glob characters are *, ? and [ ]"  
The glob characters are *, ? and [ ]
```

```
sysadmin@localhost:~$ echo "The path is $PATH"  
The path is /usr/bin/custom:/home/sysadmin/bin:/usr/local/sbin:/usr
```

Aspas Simples '

- ▶ Não interpreta todos os metacarateres ou caracteres especiais
- ▶ Não faz substituição de variáveis
- ▶ Não faz substituição de comandos

```
sysadmin@localhost:~$ echo The car costs $100
The car costs 00
sysadmin@localhost:~$ echo 'The car costs $100'
The car costs $100
```

Barra invertida \

- ▶ Técnica alternativa para essencialmente colocar aspas simples em um único caractere. Considere a seguinte mensagem:

O serviço custa \$1 e o caminho é \$PATH

```
sysadmin@localhost : ~ $ echo O serviço custa \"$1 e o caminho é $PATH  
O serviço custa $1 e o caminho é /usr/bin/custom:/home/sysadmin/bin:/usr/lo
```

Crases ou \$(command)

- ▶ São usados para especificar um comando dentro de um comando, um processo chamado **substituição de comando**

```
sysadmin@localhost:~$ echo Today is date  
Today is date
```

```
sysadmin@localhost:~$ echo Today is `date`  
Today is Mon Nov 4 03:40:04 UTC 2018
```


Instruções de Controle

Permitem usar vários comandos ao mesmo tempo ou executar comandos adicionais, dependendo do sucesso de um comando anterior

Ponto e vírgula ;

```
sysadmin@localhost : ~ $ cal 1 2030; cal 2 2030; cal 3 2030

janeiro de 2030
Su Mo Tu We Th Fr Sa
    1 2 3 4 5
 6 7 8 9 10 11 12 13
14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

de fevereiro de 2030
Su Mo Tu We Th Padre Sa
    1 2
 3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23 24
25 26 27 28

de março de 2030
Su Mo Tu We Th Padre Sa
    1 2
 3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```



- ▶ Atua como um "e" lógico

```
sysadmin@localhost : ~ $ ls /etc/ppp && echo sucesso  
ip-down.d ip-up.d  
sucesso
```

```
sysadmin@localhost : ~ $ ls /etc/junk && echo sucesso  
ls: não é possível acessar /etc/junk: Esse arquivo ou diretório não existe
```



- Atua como um "ou exclusivo" lógico

```
sysadmin@localhost : ~ $ ls /etc/ppp || echo falhou
ip-down.d ip-up.d
sysadmin@localhost : ~ $ ls /etc/junk || echo falhou
ls: não é possível acessar /etc/junk: nenhum arquivo ou diretório
falhou
```