

LINUX

INTRODUÇÃO E PRIMEIROS COMANDOS

LISTA DE FIGURAS

Figura 1.1 – Terminal de comandos Microsoft Power Shell	8
Figura 1.2 – Opções de configuração do Microsoft Word	9
Figura 1.3 – O belíssimo gerenciador de janelas Gnome	10
Figura 1.4 – <i>Prompt</i> de comandos bash	12
Figura 1.5 – <i>Prompt</i> de comandos bash	13
Figura 1.6 – Exemplo do comando su sendo utilizado para torna-se root.....	14
Figura 1.7 – Exemplo do comando su sendo utilizado para torna-se outro usuário	15
Figura 1.8 – Exemplo do comando apt com o parâmetro search.....	16
Figura 1.9 – Exemplo do comando apt com o parâmetro install.....	16
Figura 1.10 – Exemplo do comando apt com o parâmetro install.....	17
Figura 1.11 – Exemplo de comando utilizando sudo.....	18
Figura 1.12 – Exemplo de comando passwd em execução	19
Figura 1.13 – Exemplo de comando useradd em execução	20
Figura 1.14 – Exemplo de comando groupadd em execução	20
Figura 1.15 – Exemplo de comando usermod em execução	21
Figura 1.16 – Exemplo de comando ls (simples).....	23
Figura 1.17 – Exemplo de comando ls com parâmetros “l” e “a”	23
Figura 1.18 – Exemplo de comando ls com parâmetros “l”, “a” e “h” e diretório / ..	25
Figura 1.19 – Exemplo de comando pwd	25
Figura 1.20 – Exemplo de comando cd.....	26
Figura 1.21 – Exemplo de comando mkdir	26
Figura 1.22 – Exemplo de comando cp copiando diretório	30
Figura 1.23 – Exemplo de comando cat exibindo o conteúdo do arquivo /proc/locks	31
Figura 1.24 – Exemplo de comando head exibindo as dez primeiras linhas do arquivo /proc/cpuinfo	31
Figura 1.25 – Exemplo de comando tail exibindo as dez últimas linhas do arquivo /var/log/messages	32
Figura 1.26 – Exemplo de comando less exibindo o arquivo /var/log/messages ..	33
Figura 1.27 – Exemplo de comando locate usado para localizar o arquivo Imagem1.png	33
Figura 1.28 – Exemplo de comando grep buscando a palavra fiap dentro do arquivo /var/log/auth.log.....	34
Figura 1.29 – Exemplo de comando grep buscando o termo FIAP ON em /home ..	35
Figura 1.30 – Exemplo de comando tar empacotando e comprimindo arquivos..	36
Figura 1.31 – Exemplo de comando tar empacotando e comprimindo arquivos..	37
Figura 1.32 – Exemplo de comando wget baixando arquivos da Internet	38
Figura 1.33 – Exemplo de comando wget retomando um download pela metade ..	39
Figura 1.34 – Exemplo de execução do comando ifconfig	40
Figura 1.35 – Desligando e ligando uma interface de rede com o comando ifconfig	41
Figura 1.36 – Exemplo de mudança de ip e máscara usando o comando ifconfig	42
Figura 1.37 – Exemplo de execução do comando route	42
Figura 1.38 – Exemplo de execução do comando nmap	44

Figura 1.39 – Exemplo de execução do comando nmap com análise aprofundada
..... 44

EMANIP

LISTA DE QUADROS

Quadro 1.1 –Diretórios em um sistema Linux	22
--	----

EMANIP

LISTA DE LINHAS DE COMANDO

Linha de comando 1.1 – Exemplo do comando man	13
Linha de comando 1.2 – Exemplo de uso do parâmetro –Help.....	14
Linha de comando 1.3 – Exemplo de apt sendo utilizado para atualizar repositórios	15
Linha de comando 1.4 – Exemplo de apt sendo utilizado para atualizar repositórios	15
Linha de comando 1.5 – Exemplo de apt sendo utilizado para instalar o pacote sudo	17
Linha de comando 1.6 – Adicionando o usuário hpoyatos ao grupo sudo	17
Linha de comando 1.7 – Exemplo de uso do parâmetro -p no comando mkdir ...	27
Linha de comando 1.8 – Exemplo do comando rm apagando um arquivo.....	27
Linha de comando 1.9 – Exemplo do comando rm apagando um diretório vazio	28
Linha de comando 1.10 – Exemplo do comando rm apagando um diretório cheio	28
Linha de comando 1.11 – Exemplo do comando cp copiando um arquivo	29
Linha de comando 1.12 – Exemplo do comando cp copiando um arquivo, renomeando-o	29
Linha de comando 1.13 – Exemplo do comando mv usado para mover um diretório	30
Linha de comando 1.14 – Exemplo do comando mv usado para renomear um arquivo	30
Linha de comando 1.15 – Exemplo do comando unzip descompactando arquivos zip	35
Linha de comando 1.16 – Exemplo do comando unzip testando um arquivo zip .	35
Linha de comando 1.17 – Exemplo do comando tar empacotando e comprimindo arquivos txt.....	37
Linha de comando 1.18 – Instalação do pacote net-tools utilizando apt e sudo ..	39
Linha de comando 1.19 – Adicionar uma nova rota de rede com o comando route	43
Linha de comando 1.20 – Adicionar uma nova rota de rede com o comando route	43

SUMÁRIO

1 INTRODUÇÃO E PRIMEIROS COMANDOS	7
1.1 Por que aprender comandos em um mundo de interfaces gráficas?	7
1.2 Algumas considerações antes de começar	11
1.3 Comandos essenciais	12
1.3.1 man ou parâmetro - Help.....	13
1.3.2 su – Tornando-se outro usuário	14
1.3.3 apt – Gerenciamento pacotes de um sistema .deb	15
1.3.4 sudo – Rodando comandos como se fosse o superusuário	17
1.4 Gerenciamento de usuários	19
1.4.1 passwd – Modificar a senha do usuário	19
1.4.2 useradd – Adicionar um usuário	20
1.4.3 groupadd – Adicionar um grupo	20
1.4.4 usermod – Modificando um usuário	20
1.5 Manipulação de arquivos e pastas	21
1.5.1 Raiz de diretórios	21
1.5.2 ls – Listar arquivos e diretórios	23
1.5.3 pwd – Mostrar diretório ou pasta atual	25
1.5.4 cd – Mudar de diretório ou pasta	26
1.5.5 mkdir – Criar um diretório	26
1.5.6 rm – Apagar um arquivo ou diretório	27
1.5.7 cp – Copiando arquivos e diretórios	29
1.5.8 mv – Movendo ou renomeando arquivos e diretórios	30
1.5.9 cat – Exibe o conteúdo de um arquivo	31
1.5.10 head – Exibir linhas iniciais de um arquivo	31
1.5.11 tail – Exibir linhas finais de um arquivo	32
1.5.12 less – Exibe o conteúdo de um arquivo	32
1.5.13 locate – Localiza arquivos	33
1.5.14 grep – Localiza termos dentro de arquivos	34
1.5.15 unzip – Descompactar arquivos zip	35
1.5.16 tar – (Des)empacotar e(des)comprimir arquivos	35
1.5.17 wget – Baixando arquivos da internet	38
1.6 Configuração de rede	39
1.6.1 ifconfig – Configuração de redes	40
1.6.2 route – Estabelecendo rotas de rede	42
1.6.3 nmap – Escaneamento de portas	43
REFERÊNCIAS	46
GLOSSÁRIO	47

1 INTRODUÇÃO E PRIMEIROS COMANDOS

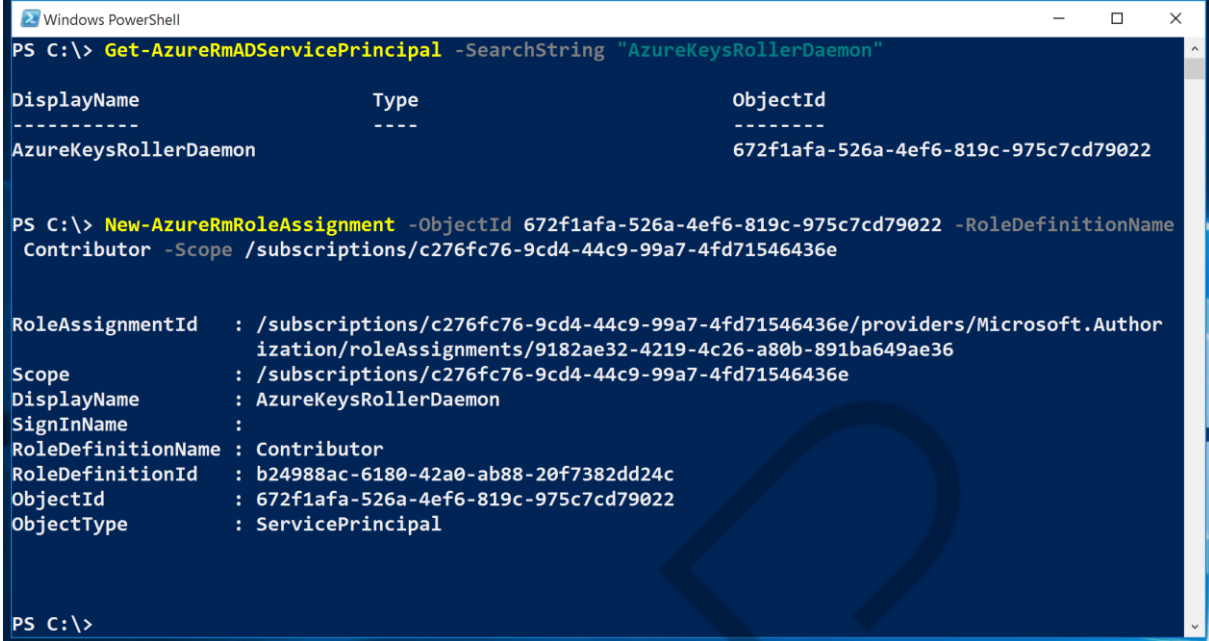
Aprender comandos para o sistema operacional Linux é indispensável para qualquer profissional de TI, seja este um programador, administrador de rede ou um especialista em segurança. Neste capítulo, abordamos os principais comandos deste sistema operacional, formando um verdadeiro “canivete suíço” para desbravar o sistema do pinguim.

Sem o conhecimento dos comandos abordados a seguir, o profissional fica extremamente limitado em relação às possibilidades de configuração deste sistema, além do conhecimento de como proteger seu ambiente computacional.

1.1 Por que aprender comandos em um mundo de interfaces gráficas?

O título desta seção é provocativo, e talvez seja uma dúvida genuína: Por que devo aprender comandos Linux em um mundo dominado por interfaces gráficas? Este tipo de questionamento é muito comum, especialmente vindo de usuários do sistema operacional Windows.

Em primeiro lugar, o terminal de comandos anteriormente conhecido como MS-DOS nunca foi abandonado pelo sistema Windows; na verdade, este sistema conta com uma nova versão deste terminal de comandos, o Power Shell.



```
PS C:\> Get-AzureRmADServicePrincipal -SearchString "AzureKeysRollerDaemon"

DisplayName          Type          ObjectID
-----
AzureKeysRollerDaemon
672f1afa-526a-4ef6-819c-975c7cd79022

PS C:\> New-AzureRmRoleAssignment -ObjectId 672f1afa-526a-4ef6-819c-975c7cd79022 -RoleDefinitionName Contributor -Scope /subscriptions/c276fc76-9cd4-44c9-99a7-4fd71546436e

RoleAssignmentId : /subscriptions/c276fc76-9cd4-44c9-99a7-4fd71546436e/providers/Microsoft.Authorization/roleAssignments/9182ae32-4219-4c26-a80b-891ba649ae36
Scope             : /subscriptions/c276fc76-9cd4-44c9-99a7-4fd71546436e
DisplayName       : AzureKeysRollerDaemon
SignInName        :
RoleDefinitionName : Contributor
RoleDefinitionId   : b24988ac-6180-42a0-ab88-20f7382dd24c
ObjectId          : 672f1afa-526a-4ef6-819c-975c7cd79022
ObjectType        : ServicePrincipal

PS C:\>
```

Figura 1.1 – Terminal de comandos Microsoft Power Shell
Fonte: *Microsoft Docs WebSite* (2018)

Existem situações nas quais o uso de um terminal de comandos se torna mais prático do que a interface gráfica. Quer um exemplo? Você já teve curiosidade de acessar as opções de configuração de seu Microsoft Word? (Disponível no menu Arquivo, item “Opções”). Ele possui várias seções de configurações, e cada uma delas conta com uma infinidade de caixas de textos, caixas de seleção e botões.

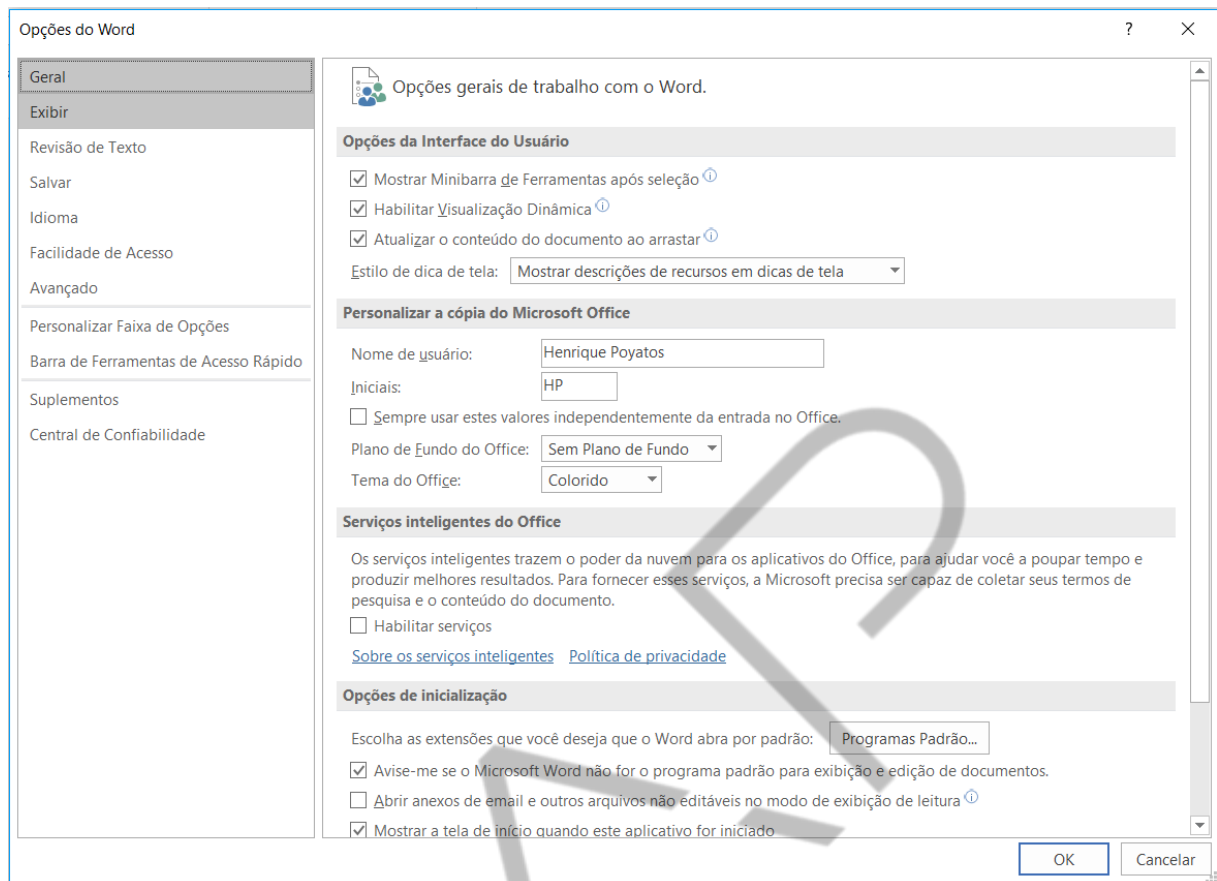


Figura 1.2 – Opções de configuração do Microsoft Word
Fonte: Elaborado pelo autor (2018)

Em um terminal de comandos, estas várias possibilidades de configuração do Microsoft Word seriam substituídas por um único comando com várias opções de parametrização, assim, desde que você saiba o parâmetro correto, poderia alterar a configuração rapidamente, enquanto na interface gráfica várias telas teriam que ser roladas até se localizar a configuração desejada. Talvez você indague, com uma certa razão, que decorar os parâmetros de um comando não é muito fácil. Sim, é verdade, mas você verá no decorrer deste capítulo que isso não é necessário, você aprenderá como procurar ajuda em caso de necessidade.

Também é fato que o sistema operacional Linux conta com belíssimos e eficazes softwares de gerenciamento de janelas (*Windows Managers*, vulgarmente conhecidos como interface gráfica), como é o caso do KDE, Gnome, XFCE, *Window Maker* e vários outros. No entanto, também é realidade que a maioria das implementações de servidor Linux **NÃO CONTA COM UMA INTERFACE GRÁFICA**. Na verdade, para muitos, a implementação de um servidor X e um gerenciador de janelas representa um imenso desperdício dos recursos do servidor.

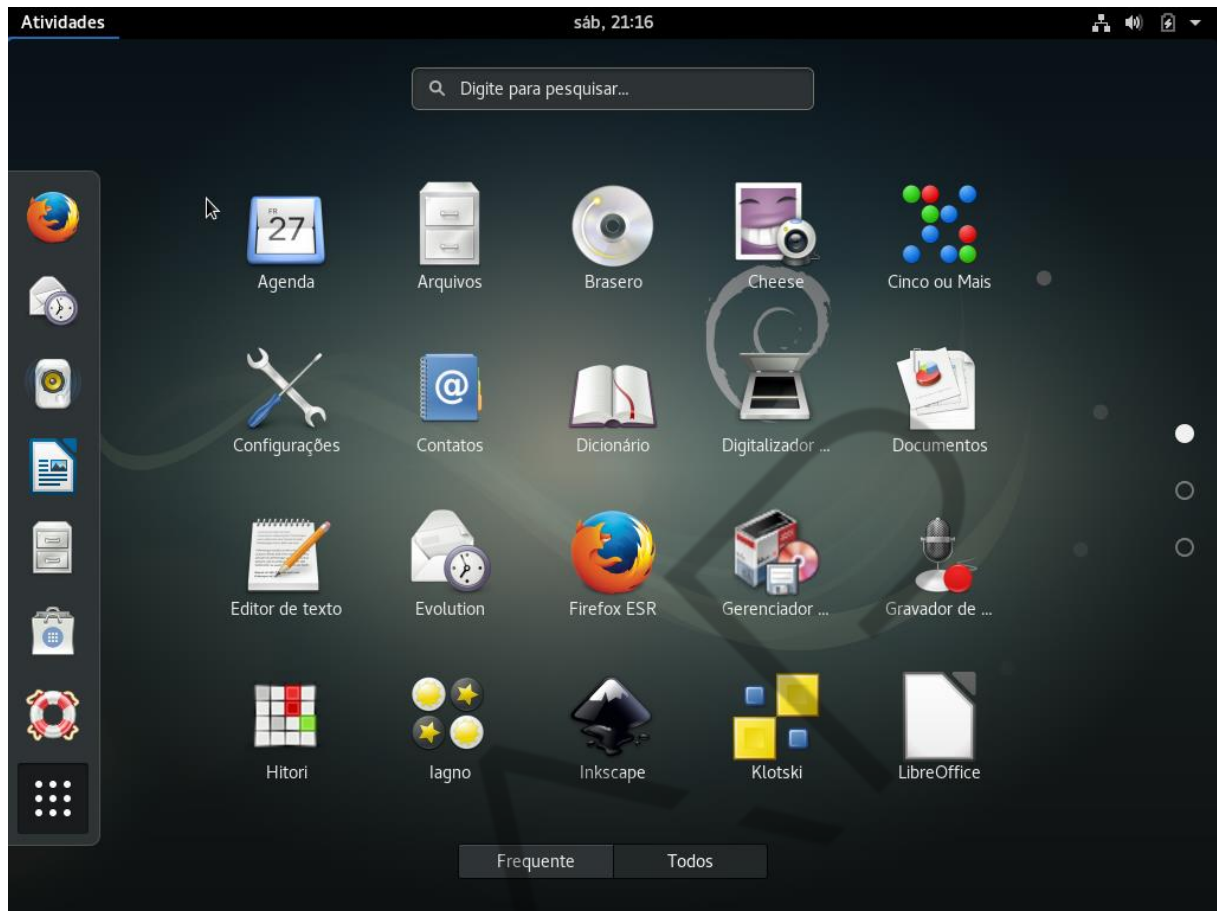


Figura 1.3 – O belíssimo gerenciador de janelas Gnome
Fonte: Elaborado pelo autor (2018)

O terminal de comandos será sempre a forma de interação com um kernel de sistema operacional mais rápida e leve que existe: não é necessária uma placa gráfica poderosa, grandes quantidades de memória RAM ou mesmo um monitor colorido. Na verdade, o terminal dispensa o uso de um monitor, possibilitando um acesso remoto através de SSH, cujo tráfego de dados será sempre infinitamente menor do que um *TeamViewer*, ou seja, nem sequer um bom link de dados é necessário para acessar o terminal de comandos de um servidor remoto.

Sendo assim, desarme-se de qualquer tipo de preconceito a partir de agora: deixe suas armas por aqui e siga conosco adiante nesta jornada, onde abordamos os comandos mais essenciais presentes no sistema operacional Linux. Talvez seu receio seja o desconhecido, mas nada que uma certa dose de dedicação e prática destes comandos não vença. Vamos lá!

1.2 Algumas considerações antes de começar

A variedade de opções disponíveis no software livre é seu grande trunfo, mas, ao mesmo tempo, um calcanhar de Aquiles: se, por um lado, a liberdade oriunda do software de código aberto permite uma infinidade de projetos diferentes, esta mesma variedade pode ser intimidadora para um iniciante.

Em primeiro lugar, o Linux possui diversas versões ou sabores diferentes: Red Hat, Fedora, Debian, Ubuntu, Slackware, Gentoo, Arch Linux, existem centenas de opções. A verdade é que, depois que você começa a conhecer um pouco cada uma delas, descobrirá que elas não são tão diferentes assim; algumas são até mesmo “aparentadas”, como é o caso do Debian e o Ubuntu, as quais poderíamos chamar aqui de distribuição “mãe” e “filha”.

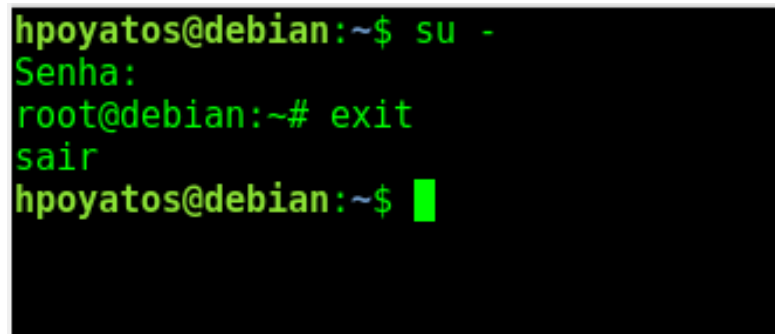
Exatamente por ser a “distribuição mãe” de dezenas de opções disponíveis, e por se manter como uma distribuição absolutamente gratuita e mais livre possível (o que ajuda em sua adoção no mercado), vamos optar pela distribuição Debian Linux para este capítulo de tutorial.

Acredite se quiser, existem também dezenas de terminais de comandos diferentes no Linux: bash, csh, zsh, entre vários outros. O terminal de comandos bash é uma implementação livre do tradicional terminal de comandos do Unix, o *shell*, ou simplesmente “sh”. O nome do terminal, bash, é um acrônimo para **Bourne-Again Shell** ou, em uma tradução livre, “*shell* renascido” (ou “concha renascida”, mas aí fica estranho).

Além de ser o clássico terminal do Unix renascido (diria que o adjetivo certo deveria ser “evoluído”), existe uma boa razão para adotá-lo: 99% de todos os usuários o utilizam; mesmo aqueles que gostam de outros terminais sempre mantêm o **bash** instalado como opção, ou seja, ele sempre estará disponível para uso. Sendo assim, a maioria dos comandos que você aprenderá neste capítulo funciona em outros sabores de Linux e também nos Unix. Nada mal, não é?

O *prompt* de comandos é simbolizado por alguns caracteres iniciais que indicam que o terminal está pronto e aguardando pelo comando do usuário. No MS-DOS era a letra do drive e o diretório em que o usuário está, exemplo: “C:\Windows>”. No bash, geralmente este *prompt* é o nome do usuário, arroba (“@”),

o nome da máquina (ou hostname), dois-pontos (":"), o diretório em que o usuário está, terminado pelo símbolo de cifrão (""). Quando o usuário em questão for o administrador do sistema (superusuário ou *root*), o símbolo de cifrão se transforma em uma cerquinha ("#", sim, cada um chama isso de um jeito, jogo da velha, sustenido, *hash*, enfim):



```
hpoyatos@debian:~$ su -  
Senha:  
root@debian:~# exit  
sair  
hpoyatos@debian:~$ █
```

Figura 1.4 – *Prompt* de comandos bash
Fonte: Elaborado pelo autor (2018)

Para simplificar, vamos encurtá-lo no material didático: o *prompt* será apenas "\$" para o usuário comum e o "#" quando o usuário for o *root*, exceto nas imagens que representarão execuções legítimas.

Outra observação importante: um terminal de comandos Linux é sempre *case-sensitive*, ou seja, letras maiúsculas são diferentes de letras minúsculas. Sendo assim, "ls" é um comando que lista arquivos e diretório no Linux, enquanto "LS" simplesmente não existe.

Por serem muitos, dividimos este capítulo em seções: comandos essenciais, manipulação de arquivos e pastas, gerenciamento de processos, configuração de rede, entre outros, assim, você poderá localizar rapidamente o que precisa.

1.3 Comandos essenciais

Nesta seção, relacionamos os comandos básicos essenciais para se começar.

1.3.1 man ou parâmetro - Help

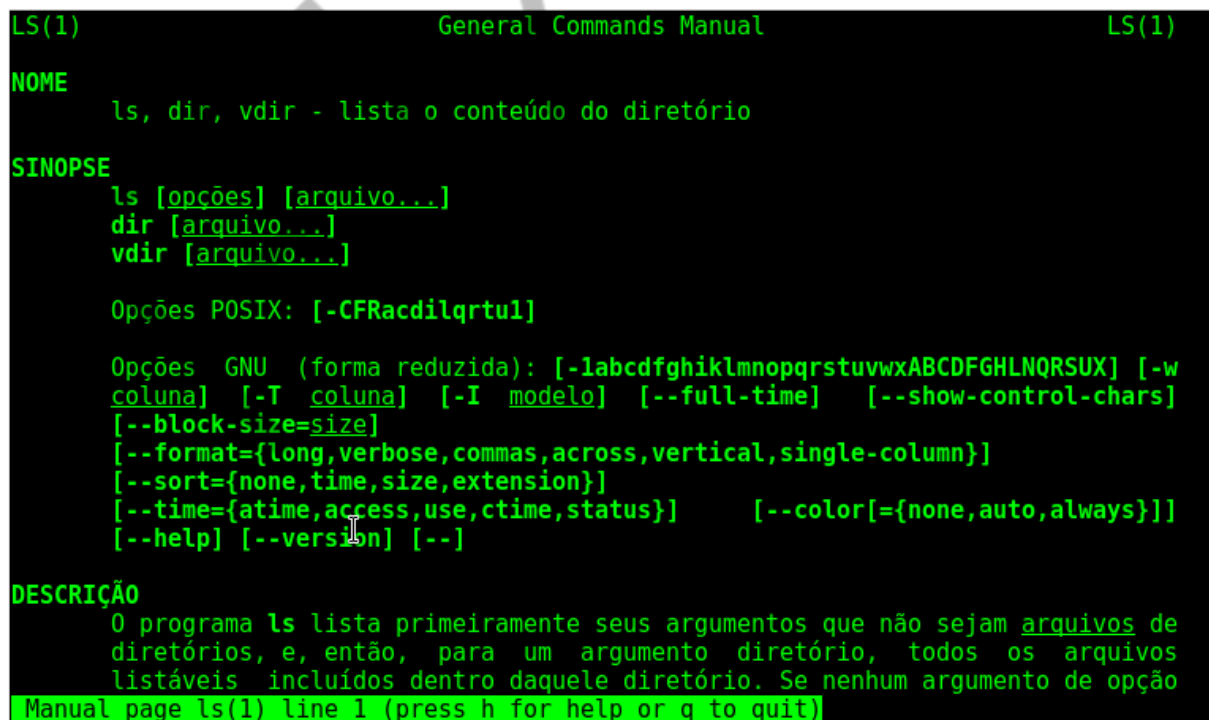
O primeiro comando é o mais essencial de todos e uma lição para todos nós: **NÃO SABE, PEÇA AJUDA!** O comando **man** é a abreviação de **manual**, ou seja, quando digito o comando **man** seguido do comando que desejo aprender, ele me mostra o manual completo dele: para que serve, quais são os parâmetros e, em alguns casos, até exemplos de uso!

Se eu desejar, por exemplo, saber como funciona o clássico comando “ls”, basta digitar:

```
$ man ls
```

Linha de comando 1.1 – Exemplo do comando man
Fonte: Elaborado pelo autor (2018)

Tenho como resultado o manual do comando **ls**, com uma breve descrição do que faz o comando, sinopse, descrição detalhada, parâmetros e exemplos de uso. Se o Linux foi instalado em português (ou os pacotes de idiomas pt-br foram instalados posteriormente), o manual vem em nosso idioma!



```
LS(1)                                General Commands Manual                                LS(1)
NOME
    ls, dir, vdir - lista o conteúdo do diretório
SINOPSE
    ls [opções] [arquivo...]
    dir [arquivo...]
    vdir [arquivo...]
Opções POSIX: [-CFRacdilqrtul]
Opções GNU (forma reduzida): [-labcdfghiklmnopqrstuvwABCDEFGHILNQRSUX] [-w
coluna] [-T coluna] [-I modelo] [--full-time] [--show-control-chars]
[--block-size=size]
[--format={long,verbose,commas,across,vertical,single-column}]
[--sort={none,time,size,extension}]
[--time={atime,access,use,ctime,status}]      [--color[]={none,auto,always}]
[--help] [--version] [--]
DESCRIÇÃO
    O programa ls lista primeiramente seus argumentos que não sejam arquivos de
    diretórios, e, então, para um argumento diretório, todos os arquivos
    listáveis incluídos dentro daquele diretório. Se nenhum argumento de opção
Manual page ls(1) line 1 (press h for help or q to quit)
```

Figura 1.5 – Prompt de comandos bash
Fonte: Elaborado pelo autor (2018)

O comando de manual prende o terminal, portanto, navegue pelo manual com as setas para cima e para baixo e, quando estiver satisfeito, pressione “q” (*quit*) para sair, e o comando será encerrado.

Outra possibilidade é digitar o comando desejado seguido por `--help` (traço-traço help):

```
$ ls --help
```

Linha de comando 1.2 – Exemplo de uso do parâmetro `--help`
Fonte: Elaborado pelo autor (2018)

Uma versão resumida do manual é apresentada e o *prompt*, liberado. Utilize as teclas [CTRL]+[PAGE UP] para subir e ler a saída, enquanto [CTRL]+[PAGE DOWN] rola o texto para baixo.

1.3.2 su – Tornando-se outro usuário

Por razões de segurança, é recomendável que o usuário sempre opere com seu próprio usuário, com permissões restritas. No entanto, em alguns momentos, é necessário torna-se outro usuário, costumeiramente, o superusuário root.

O comando `su` é o empregado para a mudança de usuário. Em geral, ele é atribuído como sendo o comando “para se tornar root”, mas, na realidade, ele permite que nos tornemos qualquer usuário do sistema, basta digitar `su` e, na sequência, o nome do usuário que queremos ser. Caso nada seja digitado, o padrão torna-se root.

```
hpoyatos@debian:~$ su
Senha:
root@debian:/home/hpoyatos#
```

Figura 1.6 – Exemplo do comando `su` sendo utilizado para torna-se root
Fonte: Elaborado pelo autor (2018)

Por padrão, o comando `su` solicitará a senha do usuário que você deseja se tornar. Se informada corretamente, o *prompt* muda de usuário antes do símbolo @ (arroba). Se o *prompt* estiver em sua versão simples e não deixar o usuário explícito, você pode usar ainda o comando `whoami` (*Who am I?* ou “Quem sou eu?” em bom português).

```
hpoyatos@debian:~$ su
Senha:
root@debian:/home/hpoyatos# whoami
root
root@debian:/home/hpoyatos# su outrousuuario
outrousuuario@debian:/home/hpoyatos$ whoami
outrousuuario
outrousuuario@debian:/home/hpoyatos$
```

Figura 1.7 – Exemplo do comando su sendo utilizado para torna-se outro usuário
Fonte: Elaborado pelo autor (2018)

No exemplo da Figura “Exemplo do comando su sendo utilizado para torna-se outro usuário”, o comando su foi usado para torna-se um usuário chamado “outrousuuario”. Repare que a senha deste usuário não é necessária, pois, com “o todo-poderoso” **root**, tudo é possível.

1.3.3 apt – Gerenciamento pacotes de um sistema .deb

Já que nos tornamos o root, vamos aproveitar a oportunidade para mostrar como operar o comando apt, responsável pelo gerenciamento de pacotes em um sistema do tipo .deb (Debian e seus descendentes). Este comando não funciona em sistemas baseados em .rpm (Red Hat, Fedora, CentOS, OpenSUSE etc.) e outros (como Slackware ou Gentoo).

```
# apt update
```

Linha de comando 1.3 – Exemplo de apt sendo utilizado para atualizar repositórios
Fonte: Elaborado pelo autor (2018)

O comando apresentado na Linha de comando **# apt update** atualiza a referência dos repositores, ou seja, ele baixa a lista de pacotes disponíveis para download com suas versões. Com isso, o sistema pode comparar as versões instaladas com as disponíveis, permitindo determinar quais pacotes poderiam ser atualizados.

```
# apt upgrade
```

Linha de comando 1.4 – Exemplo de apt sendo utilizado para atualizar repositórios
Fonte: Elaborado pelo autor (2018)

O comando na Linha de comando **# apt upgrade** exibe a lista de pacotes que estão atualizados e confirma se desejamos realmente atualizá-los. Em caso positivo, apt baixa os pacotes e atualiza-os automaticamente.

O comando `apt` possui um parâmetro “search”, que permite buscar por pacotes antes de instalá-los. Observe o que acontece quando rodo o comando procurando pela palavra “trezor” (Curiosidade: `trezor` é uma carteira do tipo hardware para criptomoedas):

```
root@debian:/home/hpoyatos# apt search trezor
Sorting... Pronto
Full Text Search... Pronto
python-keepkey/stable 0.7.3-1 all
  library for communicating with KeepKey Hardware Wallet

python-trezor/stable 0.7.6-1 all
  library for communicating with TREZOR Bitcoin HW wallet

root@debian:/home/hpoyatos#
```

Figura 1.8 – Exemplo do comando `apt` com o parâmetro `search`
Fonte: Elaborado pelo autor (2018)

Já o parâmetro “install” permite instalar pacotes disponíveis nos repositórios. Para que a operação seja feita com sucesso, você precisa estar logado como o superusuário `root`.

```
root@debian:/home/hpoyatos# apt install sudo
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os seguintes pacotes foram instalados automaticamente e já não são necessários:
  libhidapi-hidraw0 libhidapi-libusb0 python-ecdsa python-hid python-mnemonic
  python-pbkdf2 python-protobuf
Utilize 'apt autoremove' para os remover.
Os NOVOS pacotes a seguir serão instalados:
  sudo
0 pacotes atualizados, 1 pacotes novos instalados, 0 a serem removidos e 0 não a
tualizados.
É preciso baixar 1.055 kB de arquivos.
Depois desta operação, 3.108 kB adicionais de espaço em disco serão usados.
Obter:1 http://ftp.br.debian.org/debian stretch/main amd64 sudo amd64 1.8.19p1-2
.1 [1.055 kB]
Baixados 1.055 kB em 2s (425 kB/s)
A seleccionar pacote anteriormente não seleccionado sudo.
(Lendo banco de dados ... 129928 ficheiros e directórios actualmente instalados.
)
A preparar para desempacotar .../sudo_1.8.19p1-2.1_amd64.deb ...
A descompactar sudo (1.8.19p1-2.1) ...
Configurando sudo (1.8.19p1-2.1) ...
A processar 'triggers' para systemd (232-25+deb9u1) ...
A processar 'triggers' para man-db (2.7.6.1-2) ...
root@debian:/home/hpoyatos#
```

Figura 1.9 – Exemplo do comando `apt` com o parâmetro `install`
Fonte: Elaborado pelo autor (2018)

A Figura “Exemplo do comando apt com o parâmetro install” mostra a instalação do pacote **sudo**. Rode este comando também, pois **sudo** não está disponível por padrão no sistema Debian, e trata-se de um comando muito útil.

```
# apt install sudo
```

Linha de comando 1.5 – Exemplo de apt sendo utilizado para instalar o pacote sudo

Fonte: Elaborado pelo autor (2018)

Por fim, o parâmetro “remove” pode ser utilizado para desinstalar pacotes. Observe o que acontece quando executo o comando para desinstalar `python-trezor` (o pacote precisa ser instalado primeiro com `apt install python-trezor`):

```
root@debian:/home/hpoyatos# apt remove python-trezor
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os seguintes pacotes foram instalados automaticamente e já não são necessários:
  libhidapi-hidraw0 libhidapi-libusb0 python-ecdsa python-hid python-mnemonic
  python-pbkdf2 python-protobuf
Utilize 'apt autoremove' para os remover.
Os pacotes a seguir serão REMOVIDOS:
  python-trezor
0 pacotes atualizados, 0 pacotes novos instalados, 1 a serem removidos e 0 não a
tualizados.
Depois desta operação, 329 kB de espaço em disco serão liberados.
Você quer continuar? [S/n] s
(Lendo banco de dados ... 129959 ficheiros e directórios actualmente instalados.
)
A remover python-trezor (0.7.6-1) ...
root@debian:/home/hpoyatos#
```

Figura 1.10 – Exemplo do comando apt com o parâmetro install

Fonte: Elaborado pelo autor (2018)

1.3.4 sudo – Rodando comandos como se fosse o superusuário

Tornar-se o usuário root e voltar a ser seu próprio usuário o tempo todo pode ser cansativo e, por descuido, o usuário pode ficar como **root** tempo demais. O comando **sudo** pode ser útil nestes casos.

A primeira coisa é instalar o sudo com o comando **# apt install sudo** como o usuário root. Logo depois, rode o seguinte comando (também como root):

```
# usermod -aG sudo hpoyatos
```

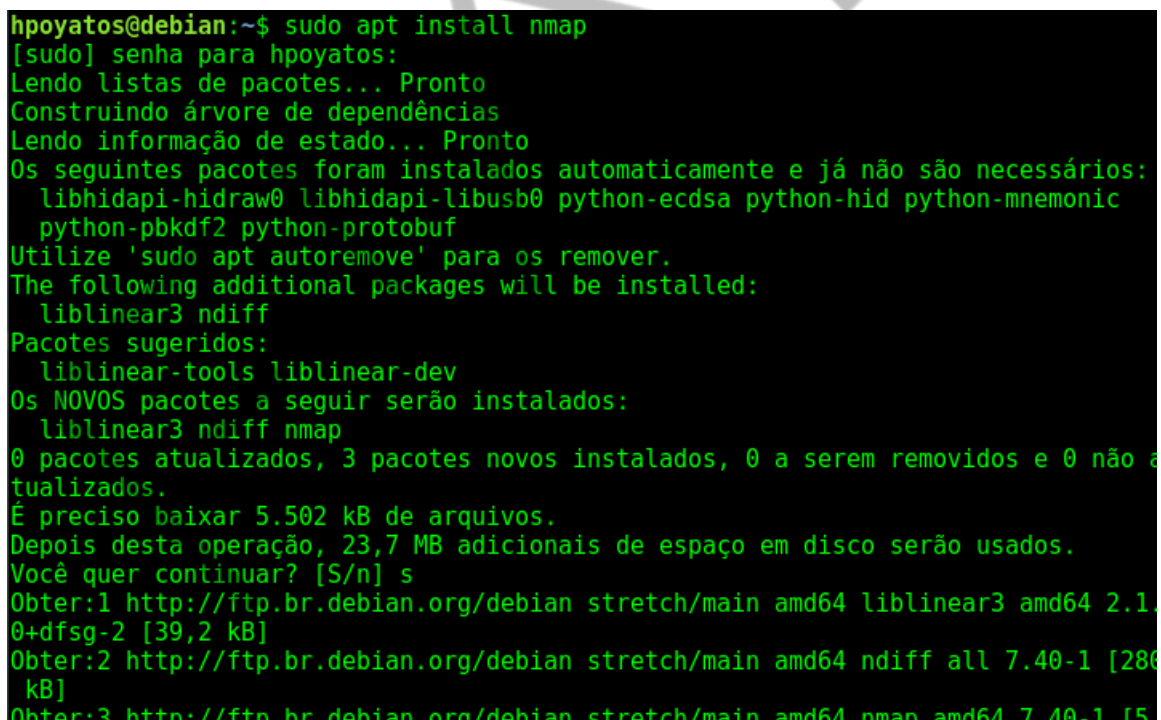
Linha de comando 1.6 – Adicionando o usuário hpoyatos ao grupo sudo

Fonte: Elaborado pelo autor (2018)

A Linha de comando **# usermod -aG sudo hpoyatos** adiciona o usuário “hpoyatos” ao grupo de usuários sudo, que foi criado quando instalei o pacote. Troque “hpoyatos” pelo nome de usuário desejado. Por razões de segurança, apenas os membros do grupo **sudo** poderão usar o comando. O nome do grupo muda de distribuição Linux para distribuição, em alguns casos, este grupo é o **wheel**. Conferir o arquivo /etc/sudoers com o comando **# cat /etc/sudoers** pode esclarecer este ponto.

Antes de rodar o comando seguinte, deslogue e logue novamente, pois a alocação de grupos do usuário é feita no início da sessão do usuário, seja no modo texto ou com a interface gráfica.

A sintaxe é **\$ sudo [comando que desejo rodar com privilégios]**; o terminal solicitará a senha do seu próprio usuário e, caso ele seja membro do grupo sudo, o comando será executado com os privilégios de superusuário:

A terminal window with a black background and green text. The prompt is 'hpoyatos@debian:~\$'. The user enters 'sudo apt install nmap'. The terminal shows the process of installing nmap, including checking package lists, dependencies, and disk space. It lists additional packages to be installed (liblinear3, ndiff) and suggested packages (liblinear-tools, liblinear-dev). It shows the download progress for these packages from the Debian stretch repository. The output is as follows:

```
hpoyatos@debian:~$ sudo apt install nmap
[sudo] senha para hpoyatos:
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os seguintes pacotes foram instalados automaticamente e já não são necessários:
  libhidapi-hidraw0 libhidapi-libusb0 python-ecdsa python-hid python-mnemonic
  python-pbkdf2 python-protobuf
Utilize 'sudo apt autoremove' para os remover.
The following additional packages will be installed:
  liblinear3 ndiff
Pacotes sugeridos:
  liblinear-tools liblinear-dev
Os NOVOS pacotes a seguir serão instalados:
  liblinear3 ndiff nmap
0 pacotes atualizados, 3 pacotes novos instalados, 0 a serem removidos e 0 não a
tualizados.
É preciso baixar 5.502 kB de arquivos.
Depois desta operação, 23,7 MB adicionais de espaço em disco serão usados.
Você quer continuar? [S/n] s
Obter:1 http://ftp.br.debian.org/debian stretch/main amd64 liblinear3 amd64 2.1
0+dfsg-2 [39,2 kB]
Obter:2 http://ftp.br.debian.org/debian stretch/main amd64 ndiff all 7.40-1 [280
kB]
Obter:3 http://ftp.br.debian.org/debian stretch/main amd64 nmap amd64 7.40-1 [5
```

Figura 1.11 – Exemplo de comando utilizando sudo

Fonte: Elaborado pelo autor (2018)

Na Figura “Exemplo de comando utilizando sudo”, solicitei a instalação do pacote nmap, muito útil para verificar as portas de serviços abertas de um servidor (veja mais adiante no capítulo). Sudo solicitou a senha do meu usuário (hpoyatos) e prosseguiu com a instalação como se meu usuário fosse o **root**.

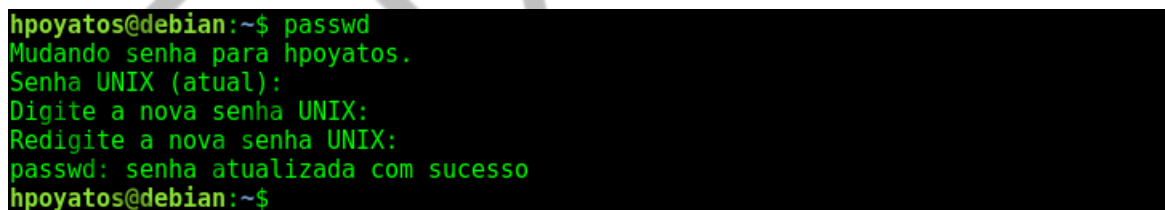
O procedimento é mais seguro do que tornar-se **root** o tempo todo com o comando **su**, neste caso, os privilégios são usados pontualmente, conforme a necessidade.

1.4 Gerenciamento de usuários

Os comandos a seguir permitem realizar algumas operações como trocar a senha do usuário, criar um usuário ou grupo e modificar usuários existentes.

1.4.1 passwd – Modificar a senha do usuário

Para modificar a senha do usuário, o comando **passwd** (abreviação da palavra inglesa *password* ou senha) pode ser utilizado. Ao digitar **passwd** no terminal, este, por sua vez, solicita a senha atual e, posteriormente, a nova senha e a confirmação da nova senha. Repare que o terminal de comandos do Linux não exibe as senhas digitadas ou mesmo máscaras como ********, ele não exibe absolutamente nada e um usuário de primeira viagem pode achar que existe algo errado.



```
hpoyatos@debian:~$ passwd
Mudando senha para hpoyatos.
Senha UNIX (atual):
Digite a nova senha UNIX:
Redigite a nova senha UNIX:
passwd: senha atualizada com sucesso
hpoyatos@debian:~$
```

Figura 1.12 – Exemplo de comando **passwd** em execução
Fonte: Elaborado pelo autor (2018)

Caso o usuário não saiba a própria senha, é possível passar o nome de usuário logo após o comando **passwd** e trocar a senha de outro usuário; se você for o usuário **root**, a senha atual não será solicitada. Sendo assim, **\$ sudo passwd root** pode ser usado para trocar a senha do próprio superusuário, portanto, muito cuidado a quem você concede altos privilégios.

1.4.2 useradd – Adicionar um usuário

Quando for necessário criar um usuário no sistema Linux, o comando **useradd** deve ser utilizado. Utilize os parâmetros “-d” para informar o diretório pessoal deste usuário, e “-p” para informar a senha (embora você possa rodar o comando **\$ sudo passwd [usuario]** logo na sequência):

```
hpoyatos@debian:~$ sudo useradd fiap -d /home/fiap -p senha_usuario_fiap
hpoyatos@debian:~$
```

Figura 1.13 – Exemplo de comando useradd em execução
Fonte: Elaborado pelo autor (2018)

Embora não fique explícito, por padrão o comando **useradd** cria um grupo de mesmo nome que o usuário e o coloca como membro. Procure outras informações sobre o comando com **\$ man useradd**, pois existem parâmetros para criar usuários temporários, adicioná-lo previamente em outros grupos, entre outras possibilidades.

1.4.3 groupadd – Adicionar um grupo

O comando **groupadd** permite criar um grupo de usuários. Para tal, basta digitar o comando groupadd acompanhado do nome deste novo grupo (e sudo, pois criar grupos precisa de privilégio especial):

```
hpoyatos@debian:~$ sudo groupadd fiapon
hpoyatos@debian:~$
```

Figura 1.14 – Exemplo de comando groupadd em execução
Fonte: Elaborado pelo autor (2018)

1.4.4 usermod – Modificando um usuário

Embora seja possível alterar os arquivos /etc/passwd (configurações do usuário), /etc/shadow (as senhas dos usuários) e /etc/group (configurações de grupo) com os privilégios de um root, a prática não é recomendada, pois qualquer falha na alteração destes arquivos pode comprometer o acesso dos usuários ao sistema.

Sendo assim, o comando **usermod** (de *user modification* ou modificação de usuário) e os parâmetros “-aG” são geralmente usados em conjunto e permitem adicionar o usuário em novos grupos sem, no entanto, retirá-lo dos grupos dos quais já faz parte.

O parâmetro “-d” permite estabelecer um novo diretório pessoal para o usuário, “-L” (de *lock*) bloqueia um usuário, enquanto “-U” (de *unlock*) pode desbloqueá-lo facilmente. O uso de “-e” permite estabelecer uma data de expiração para a conta do usuário, que é automaticamente bloqueada após esta data.

A Figura “Exemplo de comando usermod em execução” adiciona o recém-criado usuário **fiap** em dois novos grupos (**sudo** e **fiapon**) e estabelece que o usuário ficará ativo até o final do ano de 2018:

```
hpoyatos@debian:~$ sudo usermod -aG fiapon,sudo -e 2018-12-31 fiap
[sudo] senha para hpoyatos:
hpoyatos@debian:~$
```

Figura 1.15 – Exemplo de comando usermod em execução
Fonte: Elaborado pelo autor (2018)

1.5 Manipulação de arquivos e pastas

Navegar pelo sistema de arquivos de um sistema operacional organizando arquivos em diretórios (ou pastas) é uma das necessidades primárias em um sistema operacional, sendo, assim, vejamos alguns comandos úteis nesta tarefa.

1.5.1 Raiz de diretórios

Antes de abordar os comandos, no entanto, precisamos falar um pouco sobre a raiz de diretórios do sistema operacional Linux. Como a maioria dos usuários conhece a árvore de um sistema Windows, tentarei explicar fazendo um paralelo entre eles, quando for possível.

Diretório	Descrição
/	É o diretório-raiz, onde tudo começa. Ele seria equivalente ao C:\ do Windows, mas diferente do sistema operacional de Redmond, um sistema *nix tem sempre uma raiz única; o Windows cria uma raiz (ou diretório inicial) para cada volume, sempre identificando com uma letra (D:\, E:\, A:\, e por aí vai). Outros volumes no Linux são “pendurados” nesta raiz única iniciada em “/”, e isso traz uma série de vantagens. O termo exato, na

	verdade, é que os “volumes são montados”, mas isso é um assunto para outro capítulo.
/bin	É o diretório dos binários no Linux. Os comandos que podem ser executados de alguma forma por qualquer usuário ficam por aqui.
/boot	Armazena os arquivos necessários para realizar o boot do sistema Linux.
/dev	“dev” é uma abreviação da palavra inglesa <i>devices</i> , ou dispositivos; diferente do Windows que possuímos uma lista de dispositivos como esta: em sistemas *nix, os dispositivos viram arquivos, fazendo parte do sistema de arquivos. Sendo assim, periféricos, como mouses, impressoras, webcams, e dispositivos embarcados, como HDs e suas partições, tudo vira um arquivo em /dev.
/etc	Os arquivos de configuração ficam aqui, especialmente os de serviços da máquina; quando a configuração é específica para um determinado usuário, geralmente ficam em pastas ocultas (iniciadas com “.”, ponto) no diretório do usuário.
/home	Armazena os diretórios de usuários, sendo equivalente ao C:\Usuários ou C:\Users. Cada usuário possui seu próprio diretório (Por exemplo, meu usuário “hpoyatos” possui o /home/hpoyatos).
/lib e /lib64	Equivalente ao C:\Windows\System e C:\Windows\System32, são os diretórios onde ficam as bibliotecas de sistema; as bibliotecas em 64bits ficam em /lib64, enquanto as de 32bits ficam em /lib.
/lost+found	Diretório de “achados e perdidos” do sistema. Em uma eventual falha no sistema de arquivos, o comando fsck deve ser utilizado e, caso ache arquivos ou fragmentos dele que ele não sabe onde guardar, eles ficam armazenados por aqui.
/media e /mnt	Diretórios de “montagem” de volumes, como pendrives, CD-ROMs e DVD-ROMs.
/opt	Diretório opcional, pode ser utilizado como diretório alternativo de instalação de programas.
/proc	Diretório virtual de informações do sistema; através de seus pseudo arquivos, podemos obter informações sobre o processador, uso de memória, entre outros.
/root	Diretório pessoal do superusuário ou root.
/run	Os processos em andamento (rodando ou <i>run</i>) geram arquivos .pid que ficam armazenados por aqui.
/sbin	Os binários (comandos) específicos para uso do superusuário (<i>root</i>) ficam armazenados aqui.
/srv	Dados de serviços em execução ficam armazenados neste diretório.
/sys	(apenas para kernels 2.6.x ou superiores) armazena os módulos para equipamentos USB.
/tmp	Diretório de arquivos temporários, que podem ser gerados por serviços ativos; é equivalente ao C:\Windows\Temp.
/usr	É aqui que os programas são instalados, assim sendo, seria equivalente ao C:\Arquivos de programas ou C:\Program Files.
/var	O diretório var armazena informações de tamanhos variados; é nele, por exemplo, que os arquivos que armazenam dados em um banco de dados ficam.
/var/log	Como também é uma informação de tamanho variado, os <i>logs</i> de sistema ficam neste subdiretório.

Quadro 1.1 –Diretórios em um sistema Linux

Fonte: Elaborado pelo autor (2018)

1.5.2 ls – Listar arquivos e diretórios

O comando **ls** permite exibir informações dos arquivos e subdiretórios de um determinado diretório:

```
hpooyatos@debian:~$ ls
Área de trabalho  Arquivo2.txt  Documentos  Imagens  Música  Vídeos
Arquivol.txt      Arquivo3.txt  Downloads   Modelos  Público
```

Figura 1.16 – Exemplo de comando ls (simples)
Fonte: Elaborado pelo autor (2018)

Em sua execução simples, exibirá apenas o nome dos arquivos e diretórios. Graças ao padrão de cores, podemos diferenciar o que é arquivo (em verde) do que é diretório (em azul). Não se baseie, no entanto, em extensões de arquivos: diferente do Windows, elas são completamente dispensáveis. A extensão “txt” nos três arquivos do exemplo é apenas para fácil identificação e para seu maior conforto.

O comando **ls** é geralmente executado com dois parâmetros muito úteis: “-l”, que exibe uma lista longa ou detalhada de informações, e o “-a”, que vem de *all*, ou seja, traga **todos** os arquivos e diretórios, mesmo aqueles que são ocultos:

```
hpooyatos@debian:~$ ls -la
total 76
drwxr-xr-x 15 hpooyatos hpooyatos 4096 fev  3 22:17 .
drwxr-xr-x  3 root      root      4096 fev  3 20:47 ..
drwxr-xr-x  2 hpooyatos hpooyatos 4096 fev  3 21:14 Área de trabalho
-rw-r--r--  1 hpooyatos hpooyatos   0 fev  3 22:17 Arquivol.txt
-rw-r--r--  1 hpooyatos hpooyatos   0 fev  3 22:17 Arquivo2.txt
-rw-r--r--  1 hpooyatos hpooyatos   0 fev  3 22:17 Arquivo3.txt
-rw-r--r--  1 hpooyatos hpooyatos 220 fev  3 20:47 .bash_logout
-rw-r--r--  1 hpooyatos hpooyatos 3526 fev  3 20:47 .bashrc
drwx-----  9 hpooyatos hpooyatos 4096 fev  3 21:38 .cache
drwx----- 10 hpooyatos hpooyatos 4096 fev  3 21:37 .config
drwxr-xr-x  2 hpooyatos hpooyatos 4096 fev  3 21:14 Documentos
drwxr-xr-x  2 hpooyatos hpooyatos 4096 fev  3 21:14 Downloads
drwx-----  3 hpooyatos hpooyatos 4096 fev  3 21:37 .gnupg
-rw-----  1 hpooyatos hpooyatos 314 fev  3 21:14 .ICEauthority
drwxr-xr-x  2 hpooyatos hpooyatos 4096 fev  3 21:14 Imagens
drwx-----  3 hpooyatos hpooyatos 4096 fev  3 21:14 .local
drwxr-xr-x  2 hpooyatos hpooyatos 4096 fev  3 21:14 Modelos
drwxr-xr-x  2 hpooyatos hpooyatos 4096 fev  3 21:14 Música
-rw-r--r--  1 hpooyatos hpooyatos 675 fev  3 20:47 .profile
drwxr-xr-x  2 hpooyatos hpooyatos 4096 fev  3 21:14 Público
drwx-----  2 hpooyatos hpooyatos 4096 fev  3 21:37 .ssh
drwxr-xr-x  2 hpooyatos hpooyatos 4096 fev  3 21:14 Vídeos
hpooyatos@debian:~$
```

Figura 1.17 – Exemplo de comando ls com parâmetros “l” e “a”
Fonte: Elaborado pelo autor (2018)

Repare que vários outros arquivos e diretórios fizeram parte da listagem agora; seguindo o padrão do Unix, arquivos e diretórios ocultos são precedidos por “.” (ponto). Uma grande variedade de informações é trazida nesta lista, sendo:

- **Primeira coluna – Tipo de arquivo e permissionamento:** contendo sempre 10 caracteres, os primeiros deles se referem ao tipo de arquivo: um arquivo comum é representado por um traço “-“, e um diretório pela letra “d”. Você verá ainda as letras “l” (arquivo de link), “b” e “c” (para dispositivos, que transmitem por bloco ou “b” e caracter “c”). Os nove caracteres restantes devem ser lidos em conjuntos de três em três: os três primeiros são as permissões do dono do arquivo; os três do meio permissões do grupo dono do arquivo; e os três finais, dos demais. As letras “rwx” simbolizam leitura (**r** de **read**), escrita (**w** de **write**) e execução (**x** de **execute**).
- **Segunda coluna – Quantidade de arquivos:** indica a quantidade de arquivos ou diretórios aquele elemento possui. Repare que arquivos possuem sempre o número “1”, enquanto diretórios, geralmente, têm valores maiores. Na imagem de exemplo, “.config” possui 10 arquivos ou subdiretórios dentro dele.
- **Terceira coluna:** exibe o nome do dono do arquivo. Quando o nome não estiver disponível, exibe o número identificador do usuário (uid).
- **Quarta coluna:** mostra o nome do grupo dono do arquivo; quando o nome não estiver disponível, exibe o número identificador do grupo (gid).
- **Quinta coluna:** exibe o tamanho em bytes do arquivo. Ao acrescentar o parâmetro “h” (`ls -lah`), o tamanho fica mais legível para humanos, sendo apresentado em megabytes, gigabytes ou a unidade de medida mais próxima.
- **Sexta coluna:** mostra a data da última modificação no arquivo ou diretório.
- **Sétima coluna:** o nome do arquivo ou diretório propriamente dito.

Quando o diretório do qual queremos lista não é mencionado, o comando automaticamente exibe o conteúdo do diretório atual (em meu caso, o diretório de

meu usuário, /home/hpoyatos). O diretório desejado pode ser indicado logo depois dos parâmetros (ou antes, por que não?):

```
hpoyatos@debian:~$ ls -lah /
total 96K
drwxr-xr-x 23 root root 4,0K fev  3 20:20 .
drwxr-xr-x 23 root root 4,0K fev  3 20:20 ..
drwxr-xr-x  2 root root 4,0K fev  3 20:19 bin
drwxr-xr-x  3 root root 4,0K fev  3 20:21 boot
drwx----- 2 root root 4,0K fev  3 20:20 .cache
drwxr-xr-x 17 root root 2,9K fev  3 20:56 dev
drwxr-xr-x 118 root root 12K fev  3 22:01 etc
drwxr-xr-x  3 root root 4,0K fev  3 20:47 home
lrwxrwxrwx  1 root root  29 fev  3 19:46 initrd.img -> boot/initrd.img-4.9.0-4-amd64
lrwxrwxrwx  1 root root  29 fev  3 19:46 initrd.img.old -> boot/initrd.img-4.9.0-4-amd64
drwxr-xr-x 15 root root 4,0K fev  3 20:20 lib
drwxr-xr-x  2 root root 4,0K fev  3 19:45 lib64
drwx----- 2 root root 16K fev  3 19:45 lost+found
drwxr-xr-x  3 root root 4,0K fev  3 19:45 media
drwxr-xr-x  2 root root 4,0K fev  3 19:45 mnt
drwxr-xr-x  2 root root 4,0K fev  3 19:45 opt
dr-xr-xr-x 140 root root  0 fev  3 22:10 proc
drwx----- 3 root root 4,0K fev  3 21:42 root
drwxr-xr-x 20 root root 620 fev  3 22:01 run
drwxr-xr-x  2 root root 4,0K fev  3 20:55 sbin
drwxr-xr-x  2 root root 4,0K fev  3 19:45 srv
```

Figura 1.18 – Exemplo de comando ls com parâmetros “l”, “a” e “h” e diretório /

Fonte: Elaborado pelo autor (2018)

1.5.3 pwd – Mostrar diretório ou pasta atual

Exibe o diretório atual (pwd – *print work directory*):

```
hpoyatos@debian:~$ pwd
/home/hpoyatos
hpoyatos@debian:~$
```

Figura 1.19 – Exemplo de comando pwd

Fonte: Elaborado pelo autor (2018)

É um comando pertinente especialmente em situações em que o *prompt* de comandos não exibe o diretório atual. No exemplo em questão, possui um *prompt* que mostra o diretório, mas de maneira resumida: “~” será sempre o diretório-padrão ou home do usuário que está sendo utilizado.

1.5.4 cd – Mudar de diretório ou pasta

O comando é um acrônimo de *change directory* e permite mudar o diretório atual. O diretório pode ser mencionado tanto com seu caminho completo ou absoluto (/home/hpoyatos) quanto o caminho relativo, que recebe este nome porque é relativo ao diretório que estamos naquele momento. Ou seja, `cd hpoyatos` (sem a barra antes do hpoyatos), estando no diretório /home, possibilitará acessar o subdiretório hpoyatos, /home/hpoyatos.

O comando `cd..` indica o acesso ao diretório superior ou pai (simbolizado pelo ponto-ponto). A chamada do comando `cd` sem nenhuma informação adicional resulta na troca para o diretório-padrão do usuário, ou seja, equivale ao comando `cd ~` ou, no meu caso, `cd /home/hpoyatos`.

```
hpoyatos@debian:~$ pwd
/home/hpoyatos
hpoyatos@debian:~$ cd /tmp
hpoyatos@debian:/tmp$ cd /home
hpoyatos@debian:/home$ cd hpoyatos/
hpoyatos@debian:~$ cd ../
hpoyatos@debian:/home$
```

Figura 1.20 – Exemplo de comando cd
Fonte: Elaborado pelo autor (2018)

1.5.5 mkdir – Criar um diretório

O comando **mkdir** é uma abreviação da frase inglesa “*make a directory*”, que significa literalmente “criar um diretório”, veja um exemplo de seu funcionamento:

```
hpoyatos@debian:~$ mkdir Nova\ Pasta
hpoyatos@debian:~$
```

Figura 1.21 – Exemplo de comando mkdir
Fonte: Elaborado pelo autor (2018)

Repare que caracteres especiais (como o espaço “ ” ou a própria barra invertida) são permitidos, desde que sejam precedidos por uma barra invertida. Relembro que o Linux é *case-sensitive*, sendo, assim, a pasta criada no exemplo foi “Nova Pasta” e não “nova pasta” (e muito menos NoVa PaStA).

Outra consideração é o fato de que o comando não me congratulou por ter escrito o comando corretamente; não acontece uma mensagem “Diretório criado

com sucesso”. O sistema operacional só apresenta uma mensagem em caso de fracasso, ou seja, não espere um “parabéns” por ter acertado o comando (o Linux provavelmente acredita que “não fiz mais do que minha obrigação”).

Um parâmetro muito útil para acompanhar o `mkdir` é o “-p”, especialmente em situações que eu esteja criando subdiretórios, veja um exemplo:

```
$ mkdir -p novapasta/subpasta/subsubpasta
```

Linha de comando 1.7 – Exemplo de uso do parâmetro -p no comando `mkdir`
Fonte: Elaborado pelo autor (2018)

Presumindo que eu esteja em meu diretório de usuário (`/home/hpoyatos`), o comando em questão criou uma pasta “Nova Pasta2”, um subdiretório “subpasta” dentro desta e um subdiretório “subsubpasta” dentro de “subpasta”. Sem o parâmetro “-p”, o comando apresentaria um erro, pois ele tentaria localizar a “Nova Pasta2” e não encontraria, ou seja, `mkdir Nova\Pasta2/subpasta/subsubpasta` (sem o “-p”) só criaria o diretório “subsubpasta”, tendo como condição `/home/hpoyatos/Nova Pasta2/subpasta` como um caminho existente e válido.

1.5.6 `rm` – Apagar um arquivo ou diretório

O comando `rm` é uma abreviação para o verbo inglês *to remove* e permitirá que apaguemos um arquivo ou um diretório. Embora exista o comando `rmdir`, restrito apenas para remoção de diretórios, vamos nos ater ao comando que apaga qualquer um dos dois.

```
$ rm /home/hpoyatos/Arquivo1.txt
```

Linha de comando 1.8 – Exemplo do comando `rm` apagando um arquivo
Fonte: Elaborado pelo autor (2018)

O comando apresentado remove o arquivo de texto “Arquivo1.txt” do diretório de usuário `/home/hpoyatos`. Repare que utilizei o caminho absoluto que, embora seja mais longo, possui menos risco de fazer algo errado. Posso rodar o comando `rm Arquivo1.txt` também, desde que eu tenha como diretório atual

/home/hpoyatos; no entanto, se eu estiver em outro diretório que também possui um Arquivol.txt, a confusão está criada.

Posso apagar diretórios, desde que o diretório em questão esteja completamente vazio (a regra vale para `rmdir` também). Ou seja:

```
$ rm /home/hpoyatos/Nova\ Pasta2/subpasta/subsubpasta
```

Linha de comando 1.9 – Exemplo do comando `rm` apagando um diretório vazio

Fonte: Elaborado pelo autor (2018)

Na Linha de comando `$ rm /home/hpoyatos/Nova\ Pasta2/subpasta/subsubpasta`, apenas o diretório “subsubpasta” seria apagado da existência, mantendo “Nova Pasta2” e “Nova Pasta2/subpasta” intactos. Apagar “Nova Pasta2” com o comando `rm ~/Nova\ Pasta2` (o til abrevia /home/hpoyatos !) não daria certo, pois o diretório não está vazio neste momento, ele tem “subpasta” dentro dele.

Para facilitar, informo que o comando `rm` possui dois parâmetros muito usados: “-r” significa “recursivo”, ou seja, se ao apagar “Nova Pasta2” ele tiver uma resistência, ele entra os subdiretórios e os apaga primeiro, e seus subsubdiretórios caso existam, em um verdadeiro efeito cascata. Como se trata de um comando perigoso, o Linux fará uma pergunta para configurar cada diretório que eu tentar apagar, esperando um “y” indicando “yes” ou “sim”. Como apagar um diretório com outros 30 dentro dele se torna uma tarefa muito chata, a maioria adiciona um “-f” de *force*, ou seja, “força uma barra” e não me peça confirmação, e aí o comando fica realmente perigoso:

```
$ rm -rf /home/hpoyatos/Nova\ Pasta2
```

Linha de comando 1.10 – Exemplo do comando `rm` apagando um diretório cheio

Fonte: Elaborado pelo autor (2018)

A Linha de comando `$ rm -rf /home/hpoyatos/Nova\ Pasta2` apaga o diretório “Nova Pasta2” e absolutamente tudo que estiver dentro dele, sem dó nem piedade; havia apenas um subdiretório “subpasta” vazio dentro dele, mas poderiam ter outros 5 milhões de arquivos que não faria a menor diferença (na verdade, faria, mas o comando demoraria um pouquinho).

“Se eu me arrepender, eu restauro da Lixeira”, não, não restaura!: Lixeira é para usuários de interface gráfica que não sabem o que estão fazendo. Você sabe o que está fazendo, ou pelo menos deveria.

O comando só permite que eu apague arquivos e diretórios dos quais eu tenha permissão de fazê-lo, ou seja, o dano pode ser contido, a não ser que você seja o superusuário root.

Ou seja, o comando `# rm -rf /`, que não vou colocar em um quadro porque não quero que você execute, apagaria, com privilégios totais, o sistema de arquivos inteiro. Ok, na verdade, ele apagaria todos os arquivos e diretórios que encontrasse pela frente, até apagar o arquivo `/bin/rm`, para, no momento seguinte, não saber mais o que fazer (acredite, eu já rodei). No entanto, o estrago generalizado está feito, sendo, assim, “grandes poderes trazem grandes responsabilidades”.

1.5.7 cp – Copiando arquivos e diretórios

O comando **cp** é uma abreviatura do verbo inglês *to copy* e permite copiar arquivos e diretórios, solicitando sempre qual o arquivo copiado e, logo depois, para onde ele será copiado:

```
$ cp /home/hpoyatos/Arquivo2.txt /tmp
```

Linha de comando 1.11 – Exemplo do comando cp copiando um arquivo
Fonte: Elaborado pelo autor (2018)

No exemplo em Linha de comando **\$ cp /home/hpoyatos/Arquivo2.txt /tmp**, o arquivo Arquivo2.txt foi copiado para outro diretório, o diretório de arquivos temporários/tmp. É importante que você tenha permissão de escrita no diretório destino e, neste caso, /tmp permite a qualquer usuário de sistema escrever arquivos.

```
$ cp /home/hpoyatos/Arquivo2.txt /tmp/NovoNome2.txt
```

Linha de comando 1.12 – Exemplo do comando cp copiando um arquivo, renomeando-o
Fonte: Elaborado pelo autor (2018)

O segundo exemplo em Linha de comando **\$ cp /home/hpoyatos/Arquivo2.txt /tmp/NovoNome2.txt** faz a mesma operação, mas

ao copiar renomeia o novo arquivo para “NovoNome2.txt”. A primeira execução preservou o nome original, ou seja, `/tmp/Arquivo2.txt` existe.

Copiar diretórios inteiros é perfeitamente possível, mas precisamos de ajuda de três parâmetros: “-R” indica recursividade, ou seja, permite copiar o diretório e tudo contido dentro dele; o parâmetro “-v” vem da palavra em inglês *verbose* e é muito comum em comandos Linux, neste caso, mostrará quais arquivos dentro do diretório foram copiados; conheça também “-p”, que preserva as permissões originais do arquivo copiado:

```
hpoyatos@debian:~$ cp -Rvp /home/hpoyatos/Imagens /tmp
'/home/hpoyatos/Imagens' -> '/tmp/Imagens'
'/home/hpoyatos/Imagens/Imagem1.png' -> '/tmp/Imagens/Imagem1.png'
hpoyatos@debian:~$
```

Figura 1.22 – Exemplo de comando `cp` copiando diretório
Fonte: Elaborado pelo autor (2018)

O comando apresentado na Figura “Exemplo de comando `cp` copiando diretório” copiou o diretório `Imagens` com todo seu conteúdo (graças ao “-v”, vemos que apenas um arquivo constava, “`Imagem1.png`”) para o diretório `/tmp`.

1.5.8 `mv` – Movendo ou renomeando arquivos e diretórios

O comando **`mv`** é uma abreviatura do verbo inglês *to move* e permite mover arquivos e diretórios de um ponto a outro, solicitando sempre a origem e depois o destino:

```
$ mv /tmp/Imagens /opt
```

Linha de comando 1.13 – Exemplo do comando `mv` usado para mover um diretório
Fonte: Elaborado pelo autor (2018)

No exemplo na Linha de comando **`$ mv /tmp/Imagens /opt`**, o diretório `Imagens/` presente no diretório de arquivos temporários `/tmp` foi transferido para o diretório `/opt`; para dar certo, o usuário que realiza a operação precisa ter permissão de escrita no diretório destino. O processo para arquivos é idêntico. Para renomear o arquivo, o procedimento é similar; basta informar o nome atual do arquivo e, na sequência, o novo nome (faça o mesmo para diretórios):

```
$ mv /tmp/Arquivo2.txt /tmp/Arquivo1.txt
```

Linha de comando 1.14 – Exemplo do comando `mv` usado para renomear um arquivo
Fonte: Elaborado pelo autor (2018)

1.5.9 cat – Exibe o conteúdo de um arquivo

O comando **cat** pode ser utilizado para exibir o conteúdo de um arquivo texto no terminal de comandos e deve ser acompanhado pelo caminho do arquivo.

```
hpoyatos@debian:~$ cat /proc/locks
1: POSIX ADVISORY READ 1115 08:01:659858 128 128
2: POSIX ADVISORY READ 1115 08:01:670631 1073741826 1073742335
3: POSIX ADVISORY READ 1122 08:01:659858 128 128
4: POSIX ADVISORY READ 1122 08:01:670631 1073741826 1073742335
5: POSIX ADVISORY READ 1116 08:01:659858 128 128
6: POSIX ADVISORY READ 1116 08:01:670631 1073741826 1073742335
7: POSIX ADVISORY READ 1118 08:01:659858 128 128
8: POSIX ADVISORY READ 1118 08:01:670631 1073741826 1073742335
9: POSIX ADVISORY READ 1123 08:01:659858 128 128
10: POSIX ADVISORY READ 1123 08:01:670631 1073741826 1073742335
11: FLOCK ADVISORY WRITE 362 00:12:11565 0 EOF
hpoyatos@debian:~$
```

Figura 1.23 – Exemplo de comando cat exibindo o conteúdo do arquivo /proc/locks
Fonte: Elaborado pelo autor (2018)

1.5.10 head – Exibir linhas iniciais de um arquivo

Como o nome indica, o comando **head** permite exibir o cabeçalho do arquivo, ou seja, é o comando **cat** que mostra apenas as linhas iniciais do arquivo; pode ser muito útil para arquivos muito grandes dos quais você só precisa visualizar as primeiras linhas.

A Figura 1.23 seguir utiliza o parâmetro “-n”, que permite determinar o número de linhas iniciais que desejo ver do arquivo e está sendo usado para exibir as dez primeiras linhas do arquivo **/proc/cpuinfo** (que informa dados do processador):

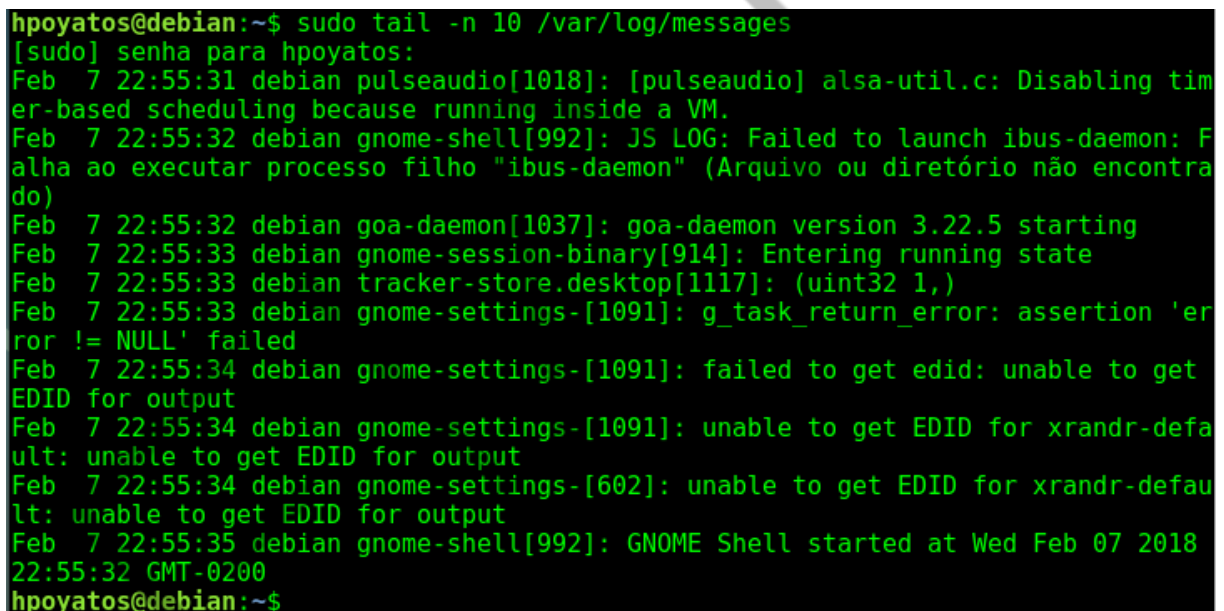
```
hpoyatos@debian:~$ head -n 10 /proc/cpuinfo
processor       : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 61
model name    : Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz
stepping      : 4
cpu MHz       : 2394.458
cache size    : 4096 KB
physical id   : 0
siblings      : 1
hpoyatos@debian:~$
```

Figura 1.24 – Exemplo de comando head exibindo as dez primeiras linhas do arquivo /proc/cpuinfo
Fonte: Elaborado pelo autor (2018)

1.5.11 tail – Exibir linhas finais de um arquivo

O comando **tail** faz o inverso do comando **head**, e a palavra significa em inglês “cauda”: exibe as últimas linhas do arquivo desejado. O parâmetro “-n” também pode ser usado aqui para definir o número de linhas finais que desejo exibir.

Muito utilizado para a leitura de arquivos de log, que geralmente são muito grandes e, em vários casos, desejamos apenas os últimos acontecimentos do sistema. O exemplo da Figura "Exemplo de comando tail exibindo as dez últimas linhas do arquivo /var/log/messages mostra as dez últimas linhas do principal arquivo de log de sistema, /var/log/messages (com sudo, pois são necessários privilégios especiais):



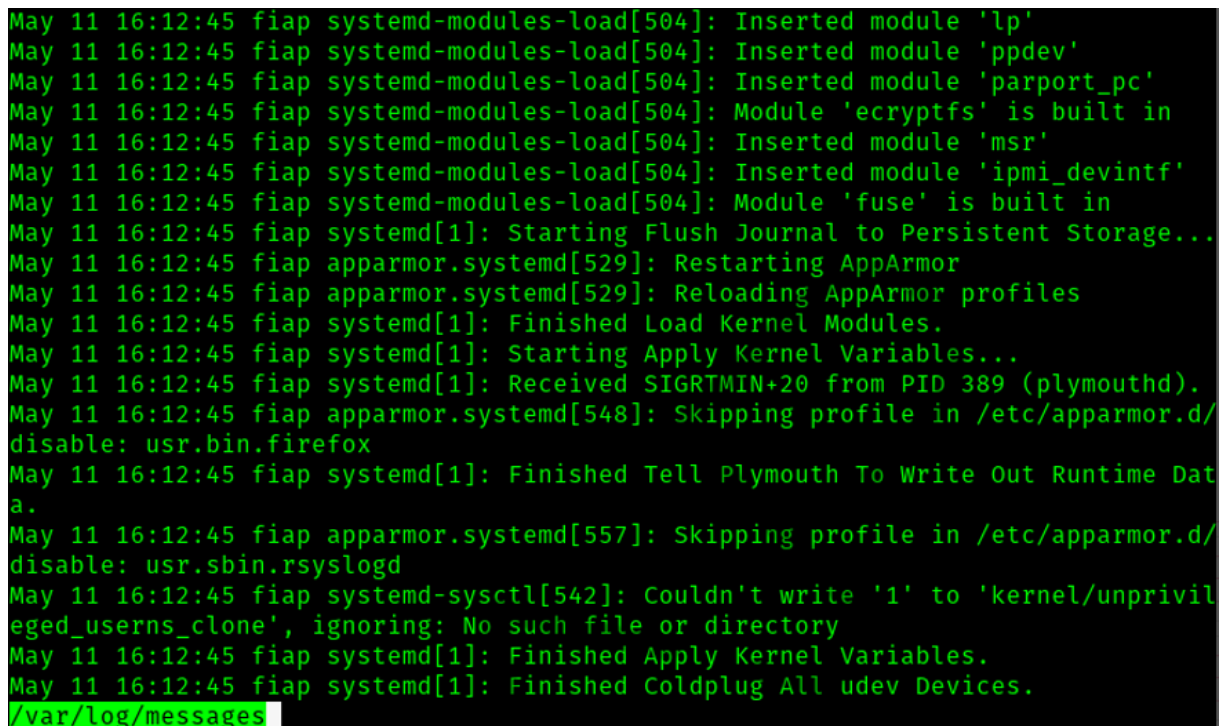
```
hpoyatos@debian:~$ sudo tail -n 10 /var/log/messages
[sudo] senha para hpoyatos:
Feb  7 22:55:31 debian pulseaudio[1018]: [pulseaudio] alsa-util.c: Disabling tim
er-based scheduling because running inside a VM.
Feb  7 22:55:32 debian gnome-shell[992]: JS LOG: Failed to launch ibus-daemon: F
alha ao executar processo filho "ibus-daemon" (Arquivo ou diretório não encontra
do)
Feb  7 22:55:32 debian goa-daemon[1037]: goa-daemon version 3.22.5 starting
Feb  7 22:55:33 debian gnome-session-binary[914]: Entering running state
Feb  7 22:55:33 debian tracker-store.desktop[1117]: (uint32 1,)
Feb  7 22:55:33 debian gnome-settings-[1091]: g_task_return_error: assertion 'er
ror != NULL' failed
Feb  7 22:55:34 debian gnome-settings-[1091]: failed to get edid: unable to get
EDID for output
Feb  7 22:55:34 debian gnome-settings-[1091]: unable to get EDID for xrandr-defa
ult: unable to get EDID for output
Feb  7 22:55:34 debian gnome-settings-[602]: unable to get EDID for xrandr-defau
lt: unable to get EDID for output
Feb  7 22:55:35 debian gnome-shell[992]: GNOME Shell started at Wed Feb 07 2018
22:55:32 GMT-0200
hpoyatos@debian:~$
```

Figura 1.25 – Exemplo de comando tail exibindo as dez últimas linhas do arquivo /var/log/messages
Fonte: Elaborado pelo autor (2018)

1.5.12 less – Exibe o conteúdo de um arquivo

O comando **less** pode ser muito útil para visualizar arquivos muito grandes, pois, com ele, podemos navegar no conteúdo do arquivo com as setas para cima e para baixo, pois a exibição com **cat** é de difícil navegação (para cat, use [CTRL]+[PAGE UP] e [CTRL]+[PAGE DOWN]). Como o comando less trava o terminal, digite “q” (de quit) para sair do comando. A Figura “Exemplo de comando

less exibindo o arquivo `/var/log/messages`” é o resultado do comando `$ sudo less /var/log/messages`:



```
May 11 16:12:45 fiap systemd-modules-load[504]: Inserted module 'lp'
May 11 16:12:45 fiap systemd-modules-load[504]: Inserted module 'ppdev'
May 11 16:12:45 fiap systemd-modules-load[504]: Inserted module 'parport_pc'
May 11 16:12:45 fiap systemd-modules-load[504]: Module 'ecryptfs' is built in
May 11 16:12:45 fiap systemd-modules-load[504]: Inserted module 'msr'
May 11 16:12:45 fiap systemd-modules-load[504]: Inserted module 'ipmi_devintf'
May 11 16:12:45 fiap systemd-modules-load[504]: Module 'fuse' is built in
May 11 16:12:45 fiap systemd[1]: Starting Flush Journal to Persistent Storage...
May 11 16:12:45 fiap apparmor.systemd[529]: Restarting AppArmor
May 11 16:12:45 fiap apparmor.systemd[529]: Reloading AppArmor profiles
May 11 16:12:45 fiap systemd[1]: Finished Load Kernel Modules.
May 11 16:12:45 fiap systemd[1]: Starting Apply Kernel Variables...
May 11 16:12:45 fiap systemd[1]: Received SIGRTMIN+20 from PID 389 (plymouthd).
May 11 16:12:45 fiap apparmor.systemd[548]: Skipping profile in /etc/apparmor.d/
disable: usr.bin.firefox
May 11 16:12:45 fiap systemd[1]: Finished Tell Plymouth To Write Out Runtime Data.
May 11 16:12:45 fiap apparmor.systemd[557]: Skipping profile in /etc/apparmor.d/
disable: usr.sbin.rsyslogd
May 11 16:12:45 fiap systemd-sysctl[542]: Couldn't write '1' to 'kernel/unprivileged_users_nice', ignoring: No such file or directory
May 11 16:12:45 fiap systemd[1]: Finished Apply Kernel Variables.
May 11 16:12:45 fiap systemd[1]: Finished Coldplug All udev Devices.
/var/log/messages
```

Figura 1.26 – Exemplo de comando less exibindo o arquivo `/var/log/messages`

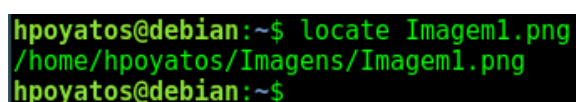
Fonte: Elaborado pelo autor (2018)

1.5.13 locate – Localiza arquivos

Por vezes, é difícil localizar um arquivo armazenado em um sistema de arquivos. Com o objetivo de facilitar esta tarefa, podemos usar o comando **locate**.

Primeiramente, é necessário instalar o pacote do comando digitando o comando `$sudo apt install locate`. Uma vez instalado, é necessário criar a base para buscas e, para tal, é necessário rodar o comando `$ sudo updatedb` de tempos em tempos.

Para utilizar o comando, digite **locate** e o nome do arquivo que você deseja buscar:



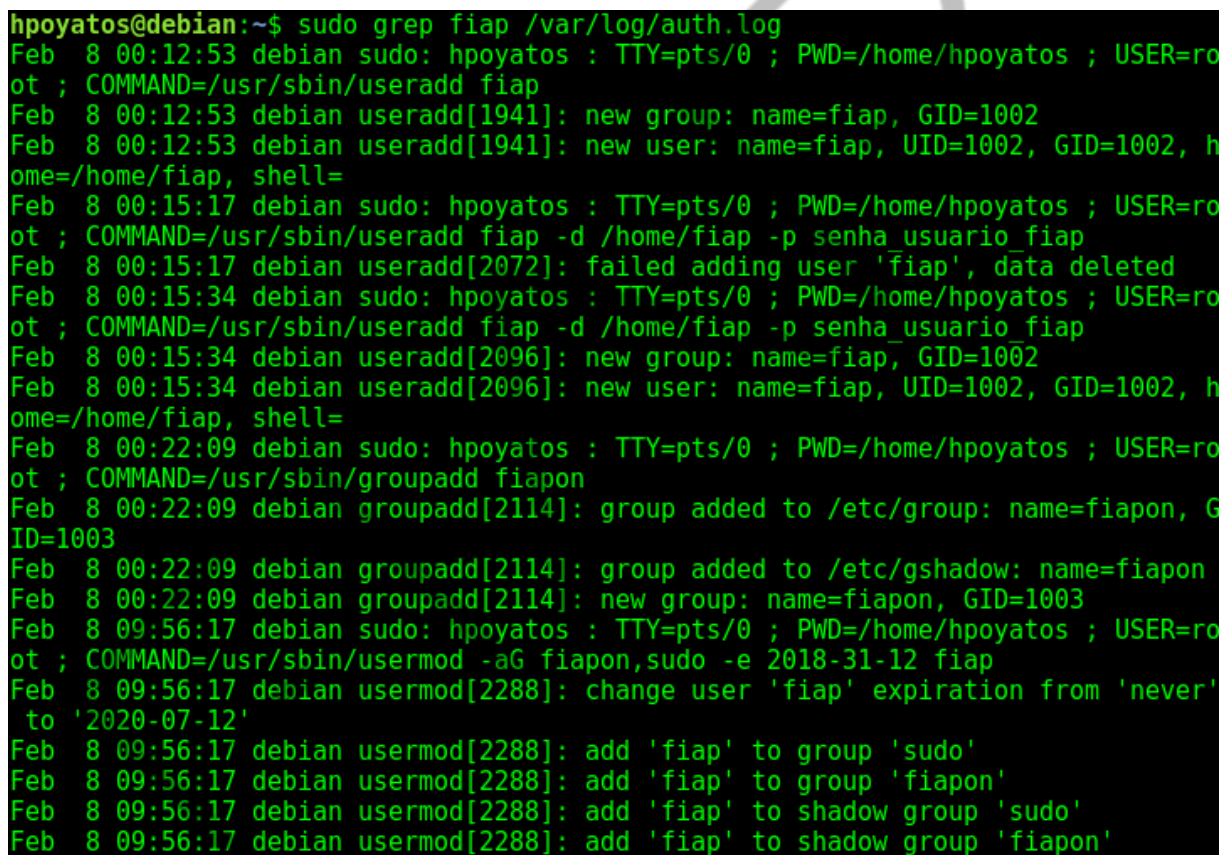
```
hpoyatos@debian:~$ locate Imagem1.png
/home/hpoyatos/Imagens/Imagem1.png
hpoyatos@debian:~$
```

Figura 1.27 – Exemplo de comando locate usado para localizar o arquivo `Imagem1.png`

Fonte: Elaborado pelo autor (2018)

1.5.14 grep – Localiza termos dentro de arquivos

Trata-se de um excelente comando para realizar buscar de palavras-chave dentro de arquivos; basta digitar a palavra-chave logo após o comando, seguido do local que se deseja procurar. Na Figura “Exemplo de comando grep buscando a palavra fiap dentro do arquivo /var/log/auth.log”, o comando foi utilizado para buscar a palavra “fiap” dentro do arquivo /var/log/auth.log, responsável por gerenciamento de usuários:



```
hpooyatos@debian:~$ sudo grep fiap /var/log/auth.log
Feb  8 00:12:53 debian sudo: hpooyatos : TTY=pts/0 ; PWD=/home/hpooyatos ; USER=root ; COMMAND=/usr/sbin/useradd fiap
Feb  8 00:12:53 debian useradd[1941]: new group: name=fiap, GID=1002
Feb  8 00:12:53 debian useradd[1941]: new user: name=fiap, UID=1002, GID=1002, home=/home/fiap, shell=
Feb  8 00:15:17 debian sudo: hpooyatos : TTY=pts/0 ; PWD=/home/hpooyatos ; USER=root ; COMMAND=/usr/sbin/useradd fiap -d /home/fiap -p senha_usuario fiap
Feb  8 00:15:17 debian useradd[2072]: failed adding user 'fiap', data deleted
Feb  8 00:15:34 debian sudo: hpooyatos : TTY=pts/0 ; PWD=/home/hpooyatos ; USER=root ; COMMAND=/usr/sbin/useradd fiap -d /home/fiap -p senha_usuario fiap
Feb  8 00:15:34 debian useradd[2096]: new group: name=fiap, GID=1002
Feb  8 00:15:34 debian useradd[2096]: new user: name=fiap, UID=1002, GID=1002, home=/home/fiap, shell=
Feb  8 00:22:09 debian sudo: hpooyatos : TTY=pts/0 ; PWD=/home/hpooyatos ; USER=root ; COMMAND=/usr/sbin/groupadd fiapon
Feb  8 00:22:09 debian groupadd[2114]: group added to /etc/group: name=fiapon, GID=1003
Feb  8 00:22:09 debian groupadd[2114]: group added to /etc/gshadow: name=fiapon
Feb  8 00:22:09 debian groupadd[2114]: new group: name=fiapon, GID=1003
Feb  8 09:56:17 debian sudo: hpooyatos : TTY=pts/0 ; PWD=/home/hpooyatos ; USER=root ; COMMAND=/usr/sbin/usermod -aG fiapon,sudo -e 2018-31-12 fiap
Feb  8 09:56:17 debian usermod[2288]: change user 'fiap' expiration from 'never' to '2020-07-12'
Feb  8 09:56:17 debian usermod[2288]: add 'fiap' to group 'sudo'
Feb  8 09:56:17 debian usermod[2288]: add 'fiap' to group 'fiapon'
Feb  8 09:56:17 debian usermod[2288]: add 'fiap' to shadow group 'sudo'
Feb  8 09:56:17 debian usermod[2288]: add 'fiap' to shadow group 'fiapon'
```

Figura 1.28 – Exemplo de comando grep buscando a palavra fiap dentro do arquivo /var/log/auth.log

Fonte: Elaborado pelo autor (2018)

O comando **grep** permite buscar termos em diretórios completos. O exemplo na Figura “Exemplo de comando grep buscando o termo FIAP ON em /home” busca pelo termo “FIAP ON” dentro de todos os diretórios contidos em /home, graças ao parâmetro “-r” que indica recursividade (sudo é necessário para vasculhar no diretório de outros usuários:

```
hpoyatos@debian:~$ sudo grep FIAP\ ON /home -r
/home/hpoyatos/Arquivo2.txt:0 FIAP ON é muito legal!
/home/hpoyatos/Arquivo3.txt:0s comandos do sistema operacional Linux são legais!
Aprendi no FIAP ON!
hpoyatos@debian:~$
```

Figura 1.29 – Exemplo de comando grep buscando o termo FIAP ON em /home

Fonte: Elaborado pelo autor (2018)

Repare que o exemplo achou ocorrências em dois arquivos diferentes, e a linha indica em qual arquivo o termo foi localizado.

1.5.15 unzip – Descompactar arquivos zip

Existirão ocasiões em que você precisará descompactar arquivos do tipo “zip”, popularmente conhecido como “deszipar”, e o comando é este:

```
$ unzip -v arquivoZipado.zip
```

Linha de comando 1.15 – Exemplo do comando unzip descompactando arquivos zip

Fonte: Elaborado pelo autor (2018)

O parâmetro “-v” permitirá listar cada arquivo e diretório que ele está descompactando. Sem o parâmetro, o comando faz o trabalho e devolve o *prompt* da maneira mais silenciosa possível.

O parâmetro “-t” é útil e permite testar o arquivo antes de descompactá-lo, assim sendo:

```
$ unzip -tv arquivoZipado.zip
```

Linha de comando 1.16 – Exemplo do comando unzip testando um arquivo zip

Fonte: Elaborado pelo autor (2018)

A Linha de comando **\$ unzip -tv arquivoZipado.zip** exibe todos os arquivos e diretórios de arquivoZipado.zip e testa a integridade dos arquivos sem, no entanto, realmente descompactá-los, o que pode ser muito útil em vários casos.

1.5.16 tar – (Des)empacotar e(des)comprimir arquivos

Comando absolutamente indispensável, tar permite empacotar e desempacotar arquivos, ou seja, vários arquivos se tornam um só e vice-versa. Repare que não usei “compactar” e “descompactar”: esta tarefa fica para outro comando, como é o caso do `gzip` (o mais utilizado) ou do `bzip2` (que compacta

melhor). “Terei que empacotar com um comando e depois comprimir com outro? ” Teria, mas nada que conhecer os parâmetros certos não resolva com um comando só.

Vamos a um exemplo na Figura “Exemplo de comando tar empacotando e comprimindo arquivos”:



```
hpoyatos@debian:~$ ls
Área de trabalho  Arquivo3.txt  Downloads  Modelos  Nova Pasta  Vídeos
Arquivo2.txt      Documentos    Imagens    Música    Público
hpoyatos@debian:~$ tar -czvf exemplo1.tar.gz *
Área de trabalho/
Arquivo2.txt
Arquivo3.txt
Documentos/
Downloads/
Imagens/
Imagens/Imagem1.png
Modelos/
Música/
Nova Pasta/
Público/
Vídeos/
hpoyatos@debian:~$
```

Figura 1.30 – Exemplo de comando tar empacotando e comprimindo arquivos

Fonte: Elaborado pelo autor (2018)

O exemplo começa com um “ls” para sabermos o que existia no diretório atual (no caso /home/hpoyatos). O comando tar pede primeiro o destino (o empacotamento) e depois o destino (os arquivos que serão empacotados ou o local onde serão desempacotados).

O comando em questão (Figura Exemplo de comando tar empacotando e comprimindo arquivos) conta com quatro parâmetros: “-c” indica que o arquivo de pacote está sendo criado, enquanto “-z” aponta que desejo comprimir o arquivo de destino com `gzip`; “-v” é o sempre útil `verbose`, enquanto “-f”, desta vez, não é force, e, sim, qual o nome do arquivo que irei gerar, neste caso, `exemplo1.tar.gz`.

Lembre-se que extensões são opcionais, mas não quer dizer que não devemos usá-las: o arquivo resultante poderia se chamar apenas `exemplo1`, mas seria complicado fazer o processo inverso sem saber a natureza do arquivo; a extensão `.tar` indica que o arquivo é um empacotamento do tipo tar, enquanto a segunda extensão do arquivo, `.gz`, aponta que este foi comprimido com `gzip`. É uma convenção útil, embora existam outros usuários que preferem a extensão `.tgz`.

Chamo a atenção para o fato de o comando ser recursivo: graças ao “-v”, podemos ver que todos os arquivos em `/home/hpoyatos` foram incluídos, e até mesmo arquivos presentes em subdiretório foram considerados, como `/home/hpoyatos/Imagens/Imagem1.png`.


```
$ tar -czvf exemplo2.tgz *.txt
```

Linha de comando 1.17 – Exemplo do comando tar empacotando e comprimindo arquivos txt
Fonte: Elaborado pelo autor (2018)

Este segundo exemplo na Linha de comando **\$ tar -czvf exemplo2.tgz *.txt** filtra os arquivos que serão incluídos no pacote; agora, apenas arquivos de extensão `.txt` serão usados, no caso, `Arquivo2.txt` e `Arquivo3.txt`.

O parâmetro “-j” no lugar do “-z” permitirá usar o compactador bzip2 no lugar do gzip e, neste caso, as extensões aceitáveis para arquivo final seriam `.tar.bz2` ou `.tbz`.

Para fazer o processo inverso:



```
hpoyatos@debian:~$ tar -xvzf exemplo1.tar.gz
Área de trabalho/
Arquivo2.txt
Arquivo3.txt
Documentos/
Downloads/
Imagens/
Imagens/Imagem1.png
Modelos/
Música/
Nova Pasta/
Público/
Vídeos/
hpoyatos@debian:~$
```

Figura 1.31 – Exemplo de comando tar empacotando e comprimindo arquivos
Fonte: Elaborado pelo autor (2018)

Como visto na Figura “Exemplo de comando tar empacotando e comprimindo arquivos”, basta substituir o parâmetro “-c” por “-x” (do verbo inglês *to extract* ou extrair) e não informar os arquivos de origem (como `*` ou `*.txt`), pois, desta vez, a origem é o arquivo de pacote. A exemplo do que vimos em *unzip*, o parâmetro “-t” no lugar de “-x” permite testar a integridade do arquivo de pacote e ver seu conteúdo sem, no entanto, “explodir” o arquivo.

1.5.17 wget – Baixando arquivos da internet

Em certas ocasiões, será necessário baixar arquivos de um algum *host* na Internet. Como fazer isso sem um navegador? Você pode usar o comando `wget`:

```
hpoyatos@debian:~$ wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.15.
tar.xz
--2018-02-05 23:31:34-- https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.15
tar.xz
Resolvendo cdn.kernel.org (cdn.kernel.org)... 151.101.93.176, 2a04:4e42:16::432
Conectando-se a cdn.kernel.org (cdn.kernel.org)[151.101.93.176]:443... conectado
A requisição HTTP foi enviada, aguardando resposta... 200 OK
Tamanho: 102181404 (97M) [application/x-xz]
Salvando em: "linux-4.15.tar.xz"

linux-4.15.tar.xz  50%[=====>          ] 49,18M  1,51MB/s   eta 27s   ^
C
hpoyatos@debian:~$ █
```

Figura 1.32 – Exemplo de comando `wget` baixando arquivos da Internet
Fonte: Elaborado pelo autor (2018)

No exemplo em questão (Exemplo de comando `wget` baixando arquivos da Internet), decidi baixar o código-fonte da versão mais atual do Kernel do Linux (4.15, na ocasião) do site oficial, `kernel.org`. Uma curiosidade: a extensão `.xz` é utilizada para compactador `xz` (do pacote `xz-utils`), considerado mais eficaz que o `bzip2` (se instalado, basta utilizar o parâmetro `-J` no comando `tar` para fazer uso).

Quando metade do arquivo foi baixado, interrompi o processo de *download* com um `[CTRL]+[C]` (que cancela qualquer comando em andamento). Observe o que acontece quando repito o comando incluindo o parâmetro `-c` de *continue* (seta para cima e a tecla `[ENTER]` me poupam de digitá-lo novamente):

```
A requisição HTTP foi enviada, aguardando resposta... 200 OK
Tamanho: 102181404 (97M) [application/x-xz]
Salvando em: "linux-4.15.tar.xz"

linux-4.15.tar.xz   50%[=====>          ] 49,18M 1,51MB/s   eta 27s   ^
C
hpoyatos@debian:~$ wget -c https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.
15.tar.xz
--2018-02-05 23:32:33-- https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.15
.tar.xz
Resolvendo cdn.kernel.org (cdn.kernel.org)... 151.101.93.176, 2a04:4e42:16::432
Conectando-se a cdn.kernel.org (cdn.kernel.org)[151.101.93.176]:443... conectado
.
A requisição HTTP foi enviada, aguardando resposta... 206 Partial Content
Tamanho: 102181404 (97M), 50338012 (48M) restantes [application/x-xz]
Salvando em: "linux-4.15.tar.xz"

linux-4.15.tar.xz   68%[+++++++==>        ] 66,86M 1,74MB/s   eta 17s
```

Figura 1.33 – Exemplo de comando wget retomando um download pela metade
Fonte: Elaborado pelo autor (2018)

O parâmetro permite retomar o download exatamente de onde parou, de sua metade. Muito útil, não é?

Ah, e caso tenha se perguntado se precisa de uma interface gráfica com um navegador web para localizar o endereço desejado, não precisa: instale o navegador *web* para terminal de comando **links**, com o comando `sudo apt-get install links`.

1.6 Configuração de rede

Reconhecer como configurar as interfaces de rede em um sistema operacional Linux possibilitará a comunicação desta máquina com outras em uma rede local ou na Internet, logo, alguns comandos podem ser úteis, como veremos a seguir.

Antes de começar, certifique-se que o pacote `net-tools` esteja instalado em seu Debian, rodando a Linha de comando **\$ sudo apt install net-tools**:

```
$ sudo apt install net-tools
```

Linha de comando 1.18 – Instalação do pacote `net-tools` utilizando `apt` e `sudo`
Fonte: Elaborado pelo autor (2018)

1.6.1 ifconfig – Configuração de redes

O comando `ifconfig` permite a configuração das informações mais básicas de uma interface de rede, como seu IP e a máscara de rede.

```
hpoyatos@debian:~$ sudo ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe9d:7458 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:9d:74:58 txqueuelen 1000 (Ethernet)
    RX packets 4510 bytes 5995562 (5.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 661 bytes 44163 (43.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Loopback Local)
    RX packets 180 bytes 14076 (13.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 180 bytes 14076 (13.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

hpoyatos@debian:~$
```

Figura 1.34 – Exemplo de execução do comando `ifconfig`
Fonte: Elaborado pelo autor (2018)

Observa na Figura “Desligando e ligando uma interface de rede com o comando `ifconfig`” a execução de `$ sudo ifconfig`. Informações sobre meu IPv4, máscara de rede e *broadcast* são apresentadas, assim como meu IPv6 e o endereço *mac address*, além de algumas estatísticas da interface de rede (batizada aqui de `enp0s3`). Interfaces de rede outrora eram padronizadas como `eth0`, `eth1`, `eth2` e assim por diante, mas a verdade é que seus nomes variam de acordo com o driver, seja placa de rede cabeada ou Wi-Fi.

A interface `lo` é o *looback*, sempre presente em qualquer sistema operacional. Seu ip padrão `127.0.0.1` indica como posso apontar para minha própria máquina, mesmo que a interface de rede real esteja inativa.


```
hpoyatos@debian:~$ sudo ifconfig enp0s3 down
hpoyatos@debian:~$ sudo ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Loopback Local)
    RX packets 180 bytes 14076 (13.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 180 bytes 14076 (13.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

hpoyatos@debian:~$ sudo ifconfig enp0s3 up
hpoyatos@debian:~$ sudo ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe9d:7458 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:9d:74:58 txqueuelen 1000 (Ethernet)
    RX packets 4515 bytes 5996582 (5.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 676 bytes 45733 (44.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Loopback Local)
    RX packets 184 bytes 14232 (13.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 184 bytes 14232 (13.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

hpoyatos@debian:~$
```

Figura 1.35 – Desligando e ligando uma interface de rede com o comando ifconfig

Fonte: Elaborado pelo autor (2018)

Conforme podemos ver na Figura “Desligando e ligando uma interface de rede com o comando ifconfig”, `$ sudo ifconfig [nome da interface] down` | `up` permite ligar e desligar a interface de rede facilmente.

Outra possibilidade trazida pelo comando ifconfig é alterar o ip e máscara de rede. Repare, no exemplo da Figura “Desligando e ligando uma interface de rede com o comando ifconfig”, que o IP atual de minha máquina é 10.0.2.15 e a máscara, 255.255.255.0; poderia trocar facilmente ambos para um ip interno de classe B, conforme Figura “Exemplo de mudança de ip e máscara usando o comando ifconfig”:

```

hpooyatos@debian:~$ sudo ifconfig enp0s3
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe9d:7458 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:9d:74:58 txqueuelen 1000 (Ethernet)
    RX packets 15 bytes 1958 (1.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 98 bytes 10721 (10.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

hpooyatos@debian:~$ sudo ifconfig enp0s3 172.16.0.1 netmask 255.255.0.0
hpooyatos@debian:~$ sudo ifconfig enp0s3
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.16.0.1 netmask 255.255.0.0 broadcast 172.16.255.255
    inet6 fe80::a00:27ff:fe9d:7458 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:9d:74:58 txqueuelen 1000 (Ethernet)
    RX packets 15 bytes 1958 (1.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 104 bytes 11811 (11.5 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

hpooyatos@debian:~$ █

```

Figura 1.36 – Exemplo de mudança de ip e máscara usando o comando ifconfig

Fonte: Elaborado pelo autor (2018)

Como podemos ver no exemplo, basta adicionar ao comando ifconfig e o nome da interface o IP desejado, a palavra *netmask* e a máscara de sub-rede, e a interface é facilmente configurada para o novo endereço.

1.6.2 route – Estabelecendo rotas de rede

O comando route serve para visualizar rotas existentes ou mesmo na criação de novas rotas; em um servidor com várias interfaces de redes ligadas a várias sub-redes diferentes, o estabelecimento destas rotas pode permitir melhorias estratégicas de infraestrutura importantes.

```

hpooyatos@debian:~$ sudo route
Tabela de Roteamento IP do Kernel
Destino      Roteador      MáscaraGen.    Opções Métrica Ref    Uso Iface
default      gateway       0.0.0.0        UG      100    0      0 enp0s3
10.0.2.0     0.0.0.0       255.255.255.0  U       100    0      0 enp0s3
link-local   0.0.0.0       255.255.0.0    U       1000   0      0 enp0s3
hpooyatos@debian:~$ █

```

Figura 1.37 – Exemplo de execução do comando route

Fonte: Elaborado pelo autor (2018)

Como podemos ver na Figura “Exemplo de execução do comando route”, a execução simples do comando route mostra as rotas já estabelecidas. Para adicionar uma nova rota, basta usar o parâmetro add seguido do parâmetro -net,

para adicionar uma nova rede; o endereço da rede deve vir na sequência, seguido pela máscara de rede (opcional em algumas situações) e, após a palavra `dev`, a interface em questão:

```
$ sudo route add -net 192.36.73.0 netmask 251.251.251.0 dev enp0s3
```

Linha de comando 1.19 – Adicionar uma nova rota de rede com o comando `route`
Fonte: Elaborado pelo autor (2018)

No exemplo na Linha de comando **\$ sudo route add -net 192.36.73.0 netmask 251.251.251.0 dev enp0s3**, é adicionada uma rota para a rede 192.36.73.*, que dará saída pela interface `enp0s3`; a máscara, neste caso, seria opcional, pois o endereço de rede é uma rede de classe C (que tem a máscara informada como padrão).

Para adicionar um gateway-padrão, a Linha de comando **\$ sudo route add default gw 10.0.2.1** traz um exemplo:

```
$ sudo route add default gw 10.0.2.1
```

Linha de comando 1.20 – Adicionar uma nova rota de rede com o comando `route`
Fonte: Elaborado pelo autor (2018)

1.6.3 nmap – Escaneamento de portas

Um dos grandes riscos para um computador conectado a uma rede são as portas de serviço que se encontram abertas, esperando para atender aos serviços de rede. Muitas destas portas são abertas por serviços que, por muitas vezes, não sabemos o porquê estão ativos e que portas abrem para tais. O comando `nmap` realiza um escaneamento de portas (o chamado *Port Scan*) na máquina-alvo, indicando quais portas estão abertas e como estão abertas.

Recomendo que você realize um escaneamento de portas apenas em máquinas das quais você conta, pois o ato é considerado um ataque de invasão preliminar. Antes de realizar o escaneamento, certifique-se que o comando está instalado; caso não esteja, digite `$ sudo apt install nmap`. Para realizar a análise, digite `nmap` e o IP da máquina em questão:

```
hpoyatos@debian:~$ sudo nmap 127.0.0.1

Starting Nmap 7.40 ( https://nmap.org ) at 2018-02-06 23:11 -02
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000040s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 1 IP address (1 host up) scanned in 1.62 seconds
hpoyatos@debian:~$
```

Figura 1.38 – Exemplo de execução do comando nmap
Fonte: Elaborado pelo autor (2018)

Repare na Figura “Exemplo de execução do comando nmap” que decidi escanear meu próprio equipamento apontando para o *loopback* (127.0.0.1); o escaneamento-padrão detectou apenas uma porta aberta – a 22 –, geralmente destinada para o serviço de ssh (que abordaremos mais adiante).

No entanto, estou presumindo que o serviço em questão é um servidor de ssh, porque esta é a porta destinada a ele; a verdade é que pode ser qualquer outro serviço escutando nesta porta. Para aprimorar esta análise, posso usar o parâmetro `--all-ports`, que, como a tradução indica, vai escanear todas as portas (o escaneamento-padrão despreza algumas). Os parâmetros “-s” e “-V” informam o serviço que está escutando e sua versão, respectivamente. Veja o exemplo:

```
hpoyatos@debian:~$ sudo nmap 127.0.0.1 --allports -sV
[sudo] senha para hpoyatos:

Starting Nmap 7.40 ( https://nmap.org ) at 2018-02-06 23:38 -02
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000060s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.4p1 Debian 10+deb9u2 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.00 seconds
hpoyatos@debian:~$
```

Figura 1.39 – Exemplo de execução do comando nmap com análise aprofundada
Fonte: Elaborado pelo autor (2018)

Posso parar de presumir agora: a análise do nmap mostra que, na porta 22, temos um servidor de OpenSSH e a versão exata (7.4p1); “de quebra”, informou-me a versão do sistema operacional (Debian 10+deb9u2) e a versão do protocolo SSH que está rodando (2.0).

É muita informação pertinente; para se proteger, talvez possa desligar um serviço de rede que não esteja sendo utilizado; para um atacante, a versão exata do OpenSSH, se desatualizada, proporciona o uso de um *exploit* e uma falha de segurança, permitindo uma invasão e até mesmo o controle do sistema com o usuário root. Proteja-se de riscos desnecessários.

Vamos dar uma pausa no mundo dos pinguins, e agora que estamos nos habituando ao mundo dos “consoles”, voltaremos na infraestrutura do nosso cliente, onde já vimos que podemos utilizar tranquilamente o sistema operacional Linux, caso tenha interesse em reduzir custos do projeto que apresentará ao seu cliente.

REFERÊNCIAS

FERREIRA, Rubem E. **Linux – Guia do Administrador do Sistema**. 2. ed. São Paulo: Novatec, 2008.

EMANIP

GLOSSÁRIO

*nix	Termo utilizado para referenciar sistemas operacionais baseados em Unix, como o caso do Linux.
Kernel	É o coração de qualquer sistema operacional, responsável por realizar o gerenciamento de processos, os recursos físicos do computador e implementar o sistema de arquivos. A interação do usuário com o Kernel é sempre indireta, via interface gráfica ou terminal de comandos.
SSH	<i>Secure Shell</i> ou <i>shell</i> seguro é um serviço de acesso remoto que permite realizar comandos em outra máquina (como o antigo <i>telnet</i>), mas que trafega as interações em um túnel criptografado.
TeamViewer	Popular software para acesso remoto em outros equipamentos, exportando a interface gráfica do host remoto para o host que estamos operando; é muito utilizado no sistema operacional Windows.