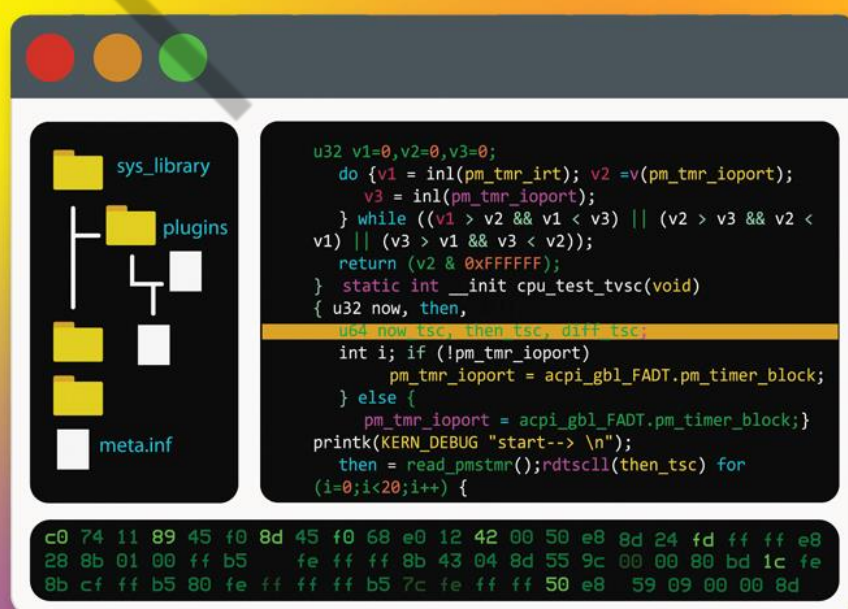


# LINUX FUNDAMENTOS

## ADMINISTRAÇÃO DE SISTEMA E

# EDITOR DE TEXTO

FILIPPI PIRES



# 2

## LISTA DE FIGURAS

Figura 2.1 – Exemplo de execução do comando ps.....	6
Figura 2.2 – Exemplo de execução do top .....	7
Figura 2.3 – Exemplo de execução do comando psaux .....	8
Figura 2.4 – Exemplo de execução do comando psaux   grep gnome-chess.....	8
Figura 2.5 – Exemplo de execução do comando kill.....	9
Figura 2.6 – Exemplo de execução do comando killall .....	10
Figura 2.7 – Exemplo de execução do comando gnome-chess travando o terminal de comandos.....	11
Figura 2.8 – Exemplo de execução do comando gnome-chess rodando em <i>background</i> .....	12
Figura 2.9 – Exemplo de execução do comando screen -ls.....	13
Figura 2.10 – O pinguim remoto.....	14
Figura 2.11 – Um RaspberryPi 3 model B.....	15
Figura 2.12 – Exemplo de execução do comando SSH.....	16
Figura 2.13 – Exemplo de execução do comando SSH ao sair do <i>host</i> remoto .....	16
Figura 2.14 – Adicionando um host no arquivo /etc/hosts .....	17
Figura 2.15 – Exemplo do comando SSH utilizando <i>hostname</i> .....	18
Figura 2.16 – Exemplo do comando scp transferindo um arquivo local para um servidor remoto.....	18
Figura 2.17 – Exemplo do comando scp transferindo um arquivo de um servidor remoto para o local.....	19
Figura 2.18 – Instalação do serviço de SSH.....	19
Figura 2.19 – Subindo o serviço de SSH.....	20
Figura 2.20 – Tela inicial do vi criando arquivo.txt .....	22
Figura 2.21 – Editando um arquivo com vi usando [i].....	23
Figura 2.22 – Editando um arquivo com vi usando [a].....	23
Figura 2.23 – Editando um arquivo com vi usando [A] .....	24
Figura 2.24 – Editando um arquivo com vi usando [o].....	24
Figura 2.25 – Exemplo de documento no vi .....	25
Figura 2.26 – Exemplo de documento no vi com numeração de linhas.....	26
Figura 2.27 – Exemplo para acessar uma linha específica no vi .....	26
Figura 2.28 – Exemplo de como copiar e colar uma linha no vi .....	27
Figura 2.29 – Exemplo de como copiar um trecho do arquivo para outro .....	28
Figura 2.30 – Exemplo de busca por palavra-chave no vi .....	29
Figura 2.31 – Exemplo de substituição de termos no vi.....	30

## LISTA DE LINHAS DE COMANDO

Linha de comando 2.1 – Exemplo do comando kill congelando o processo de PID 2829.....	9
Linha de comando 2.2 – Exemplo do comando vi criando ou editando arquivo.txt ..	22
Linha de comando 2.3 – Texto digitado em arquivo.txt .....	25
Linha de comando 2.4 – Exemplo desubstituição da palavra “mais” por “menos” no vi .....	29

EXEMPLO

## SUMÁRIO

2 ADMINISTRAÇÃO DE SISTEMA E EDITOR DE TEXTO .....	5
2.1 Gerenciamento de processos .....	5
2.1.1 PS – listar processos .....	5
2.1.2 Top – monitor de processos .....	6
2.1.3 Kill e Killall – matando processos .....	7
2.1.4 & - Rodando processos em background .....	11
2.1.5 Screen – criando terminais virtuais .....	12
2.2 Acesso a máquinas remotas com SSH .....	13
2.2.1 SSH – acesso outra máquina remotamente .....	14
2.2.2 SCP – Realizando transferências de arquivo seguras .....	18
2.2.3 Subindo um serviço SSH .....	19
2.2.4 Tornando o SSH ainda mais seguro .....	20
2.3 VI: o melhor editor de texto .....	21
2.3.1 Entrando e saindo do VI .....	22
2.3.2 Os modos de comando e de edição de texto .....	23
2.3.3 Salvando o arquivo .....	24
2.3.4. Numeração de linhas e como ir direto para uma delas .....	24
2.3.5 Copiando e colando textos .....	27
2.3.6 Apagando textos .....	28
2.3.7 Localizando e substituindo palavras .....	29
CONCLUSÃO .....	31
REFERÊNCIAS .....	32
GLOSSÁRIO .....	33

## 2 ADMINISTRAÇÃO DE SISTEMA E EDITOR DE TEXTO

Retomaremos nosso aprendizado dos comandos essenciais e abordaremos, neste capítulo, o gerenciamento de processos, a possibilidade de acesso remoto com SSH e o processador de texto vi.

### 2.1 Gerenciamento de processos

Um programa de computador é algo inerte, adormecido em um disco rígido de um equipamento. Quando chamamos um comando ou clicamos em um lançador de programa em uma interface gráfica, ele se torna um processo. Ou seja, um processo é um programa em execução.

No entanto, em alguns momentos, processos podem travar ou consumir recursos demais de uma máquina. Os comandos a seguir permitem a um administrador monitorar os processos em andamento, verificando o consumo de recursos e podendo, até mesmo, destruir processos.

#### 2.1.1 PS – listar processos

O comando “ps” permite listar os processos atuais de um sistema e é, geralmente, acompanhado pelos parâmetros: “-a”, que mostra os processos de todos os usuários (permissão de root para tal), “-u”, que exibe uma lista detalhada contendo o usuário dono do processo e “-x”, que exibe processos que não estejam associados ao terminal de comandos (pois cada processo tem uma origem, ao usar o parâmetro, podemos ver processos da interface gráfica, por exemplo, que é considerada outro terminal). A Figura “Exemplo de execução do comando ps” mostra a execução do comando `ps -au`, pois a adição de “-x” tornaria a lista longa demais:

```

hpoyatos@debian:~$ sudo ps -au
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
Debian-+  528  0.0  0.1 201400  5728 tty1    Ssl+  07:20   0:00 /usr/lib/gdm3/gdm-x-session gnom
root      534  0.0  1.0 339072 41260 tty1    Sl+   07:20   0:00 /usr/lib/xorg/Xorg vt1 -displayf
Debian-+  562  0.0  0.3 620764 12528 tty1    Sl+   07:20   0:00 /usr/lib/gnome-session/gnome-ses
Debian-+  584  0.0  4.0 1777036 162292 tty1    Sl+   07:20   0:04 /usr/bin/gnome-shell
Debian-+  612  0.0  0.7 1159620 28640 tty1    Sl+   07:21   0:00 /usr/lib/gnome-settings-daemon/g
hpoyatos  722  0.0  0.1 201400  5792 tty2    Ssl+  07:21   0:00 /usr/lib/gdm3/gdm-x-session --ru
root      724  0.2  1.3 363428 53920 tty2    Sl+   07:21   0:14 /usr/lib/xorg/Xorg vt2 -displayf
hpoyatos  732  0.0  0.3 620920 12860 tty2    Sl+   07:21   0:00 /usr/lib/gnome-session/gnome-ses
hpoyatos  810  0.6  5.5 1824532 225060 tty2    Rl+   07:21   0:43 /usr/bin/gnome-shell
hpoyatos  909  0.0  0.9 1144416 37256 tty2    Sl+   07:21   0:00 /usr/lib/gnome-settings-daemon/g
hpoyatos  932  0.0  0.2 445036 11920 tty2    SNl+  07:21   0:00 /usr/lib/tracker/tracker-miner-a
hpoyatos  933  0.0  0.3 595832 15976 tty2    SNl+  07:21   0:00 /usr/lib/tracker/tracker-extract
hpoyatos  935  0.0  0.4 505968 16528 tty2    SNl+  07:21   0:00 /usr/lib/tracker/tracker-miner-f
hpoyatos  936  0.0  1.7 920076 71644 tty2    Sl+   07:21   0:01 /usr/bin/gnome-software --gappli
hpoyatos  939  0.0  0.3 418032 13420 tty2    SNl+  07:21   0:00 /usr/lib/tracker/tracker-miner-u
hpoyatos  940  0.0  1.2 1015060 51412 tty2    Sl+   07:21   0:00 /usr/lib/evolution/evolution-ala
hpoyatos  949  0.0  0.3 495392 12904 tty2    Sl+   07:21   0:00 /usr/lib/gnome-settings-daemon/g
hpoyatos 1172  0.0  0.1 20992  4736 pts/0    Ss    07:21   0:00 bash
root     1725  0.0  0.0  55552  3820 pts/0    S+    09:19   0:00 sudo ps -au
root     1726  0.0  0.0  38304  3376 pts/0    R+    09:19   0:00 ps -au
hpoyatos@debian:~$

```

Figura 2.1 – Exemplo de execução do comando ps

Fonte: Elaborado pelo autor (2018)

### 2.1.2 Top – monitor de processos

O comando **top** não é apenas um monitor de processos, ele permite obter informações importantes, como o total de processos em andamento, seus estados no momento (executando, em espera, parados e *zombies*), uso de CPU, memória RAM e memória de swap.

Encabeçando a lista, estão sempre os processos que consomem mais recursos de sistema e como podemos ver na figura “Exemplo de execução do top”, os processos associados à interface gráfica (gnome-shell, Xorg etc.) estão sempre no topo:

```
top - 09:27:51 up 2:07, 1 user, load average: 0,00, 0,01, 0,00
Tasks: 127 total, 1 running, 126 sleeping, 0 stopped, 0 zombie
%Cpu(s): 6,4 us, 0,3 sy, 0,0 ni, 93,3 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem : 4051036 total, 3007780 free, 546880 used, 496376 buff/cache
KiB Swap: 4192252 total, 4192252 free, 0 used. 3281896 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
810	hpoyatos	20	0	1824532	225224	84472	S	4,3	5,6	0:48.08	gnome-shell
724	root	20	0	360996	51484	28964	S	2,3	1,3	0:15.82	Xorg
1166	hpoyatos	20	0	601272	33604	25084	S	0,3	0,8	0:03.24	gnome-terminal-
1756	hpoyatos	20	0	44908	3744	3116	R	0,3	0,1	0:00.23	top
1	root	20	0	139000	6824	5268	S	0,0	0,2	0:00.82	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.05	ksoftirqd/0
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/0:0H
7	root	20	0	0	0	0	S	0,0	0,0	0:00.13	rcu_sched
8	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	migration/0
10	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	lru-add-drain
11	root	rt	0	0	0	0	S	0,0	0,0	0:00.02	watchdog/0
12	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kdevtmpfs
14	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	netns
15	root	20	0	0	0	0	S	0,0	0,0	0:00.00	khungtaskd

Figura 2.2 – Exemplo de execução do top  
 Fonte: Elaborado pelo autor (2018)

Para sair do comando top (ele trava o terminal), digite **q** (*quit*, sair).

### 2.1.3 Kill e Killall – matando processos

Eventualmente, um administrador de sistema precisa interromper processos em andamento de maneira abrupta, seja por estarem consumindo recursos indesejados, seja por representarem algum risco ao sistema.

Para demonstrar seu funcionamento, aproveitei a interface gráfica e abri o aplicativo *gnome-chess*, que faz parte do pacote padrão para desktop do Ubuntu. Ao executar \$ psaux, o processo é um dos últimos exibidos (

Figura 2.3 – Exemplo de execução do comando psaux”):

```

hpoyatos@debian: ~
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
hpoyatos 1125 0.0 0.9 1144416 37668 tty2 Sl+ 16:31 0:01 /usr/lib/gnome-
hpoyatos 1142 0.0 0.7 852216 32248 ? Ssl 16:31 0:00 /usr/lib/evolut
hpoyatos 1147 0.0 0.3 445032 12252 tty2 SNl+ 16:31 0:00 /usr/lib/tracke
hpoyatos 1149 0.0 0.4 597928 16256 tty2 SNl+ 16:31 0:00 /usr/lib/tracke
hpoyatos 1151 0.0 0.3 579624 16184 tty2 SNl+ 16:31 0:00 /usr/lib/tracke
hpoyatos 1152 0.0 2.3 944012 93568 tty2 Sl+ 16:31 0:01 /usr/bin/gnome-
hpoyatos 1153 0.0 0.3 418032 13616 tty2 SNl+ 16:31 0:00 /usr/lib/tracke
hpoyatos 1154 0.0 1.3 1088992 52944 tty2 Sl+ 16:31 0:00 /usr/lib/evolut
hpoyatos 1160 0.0 0.4 396028 18836 ? Ssl 16:31 0:00 /usr/lib/tracke
hpoyatos 1172 0.0 0.3 495392 12624 tty2 Sl+ 16:31 0:00 /usr/lib/gnome-
hpoyatos 1184 0.0 0.6 809088 24640 ? Sl 16:31 0:00 /usr/lib/evolut
hpoyatos 1197 0.0 0.1 187292 4716 ? Sl 16:31 0:00 /usr/lib/dconf/
hpoyatos 1198 0.0 0.5 777356 20880 ? Sl 16:31 0:00 /usr/lib/evolut
hpoyatos 1200 0.0 0.5 709256 22580 ? Ssl 16:31 0:00 /usr/lib/evolut
hpoyatos 1216 0.0 0.5 799392 21956 ? Sl 16:31 0:00 /usr/lib/evolut
hpoyatos 1295 0.0 0.1 193752 5464 ? Ssl 16:31 0:00 /usr/lib/gvfs/g
root 2718 0.0 0.1 20472 4408 ? S 18:01 0:00 /sbin/dhclient
root 2752 0.0 0.0 0 0 ? S 18:50 0:00 [kworker/0:1]
root 2754 0.0 0.0 0 0 ? S 18:55 0:00 [kworker/0:0]
hpoyatos 2778 0.2 0.8 600956 32964 ? Ssl 18:56 0:00 /usr/lib/gnome-
hpoyatos 2784 0.0 0.1 20992 4900 pts/0 Ss 18:56 0:00 bash
hpoyatos 2829 0.3 0.8 482820 33724 tty2 Sl+ 18:57 0:00 gnome-chess
hpoyatos 2839 0.0 0.0 38304 3164 pts/0 R+ 18:59 0:00 ps aux
hpoyatos@debian:~$

```

Figura 2.3 – Exemplo de execução do comando psaux

Fonte: Elaborado pelo autor (2018)

Trata-se de um bom momento para demonstrar a possibilidade de combinar comandos: a execução do comando `ps`, especialmente com os parâmetros “aux”, pode ficar longa e confusa. O símbolo “|” (ou pipe) permite combinar um comando a outro. Observe o que acontece quando combino o comando `psaux` com o comando `grep`, que permite busca por palavras-chave:

```

hpoyatos@debian: ~
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
hpoyatos@debian:~$ ps aux | grep gnome-chess
hpoyatos 2829 0.0 0.8 482820 33724 tty2 Sl+ 18:58 0:00 gnome-chess
hpoyatos 2894 0.0 0.0 12784 1000 pts/0 S+ 19:29 0:00 grep gnome-chess
hpoyatos@debian:~$

```

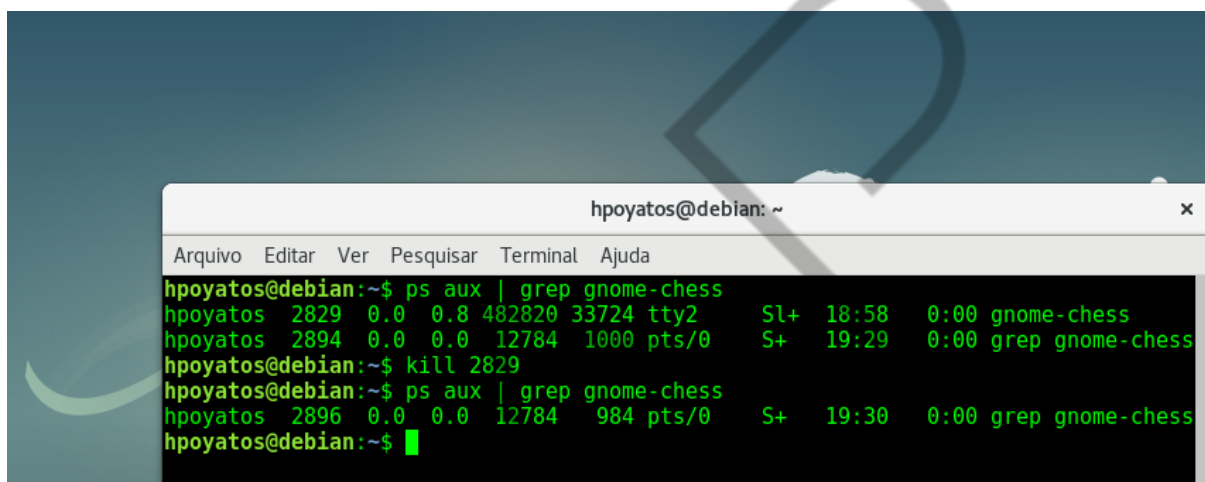
Figura 2.4 – Exemplo de execução do comando psaux | grep gnome-chess

Fonte: Elaborado pelo autor (2018)



A lista de processos mostra apenas duas linhas: o processo `gnome-chess` e a própria execução do comando `grep` (que, afinal, possui a palavra-chave). Muito interessante, não é?

Para “matar” o processo `gnome-chess`, usamos o sugestivo comando `kill` (que em inglês significa “matar” mesmo), acompanhado do identificador do processo (PID), informado na segunda coluna do comando `psaux` (no exemplo da Figura 2.4 “Exemplo de execução do comando `psaux | grep gnome-chess`”, o PID é 2829):

A terminal window titled 'hpoyatos@debian: ~' with a menu bar (Arquivo, Editar, Ver, Pesquisar, Terminal, Ajuda). The terminal shows the following commands and output:

```
hpoyatos@debian:~$ ps aux | grep gnome-chess
hpoyatos 2829 0.0 0.8 482820 33724 tty2 Sl+ 18:58 0:00 gnome-chess
hpoyatos 2894 0.0 0.0 12784 1000 pts/0 S+ 19:29 0:00 grep gnome-chess
hpoyatos@debian:~$ kill 2829
hpoyatos@debian:~$ ps aux | grep gnome-chess
hpoyatos 2896 0.0 0.0 12784 984 pts/0 S+ 19:30 0:00 grep gnome-chess
hpoyatos@debian:~$
```

Figura 2.5 – Exemplo de execução do comando `kill`

Fonte: Elaborado pelo autor (2018)

Além do fato do tabuleiro de xadrez ser fechado abruptamente, ao repetir o comando de listagem (`ps`) à procura de `gnome-chess`, o processo não existe mais; repare que não foi necessário o uso do `sudo`, uma vez que o processo tinha meu próprio usuário como dono (primeira coluna do comando `psaux`).

Embora o comando se chame `kill`, ele pode ser utilizado para mandar sinais diferentes ao processo. Por exemplo, o uso do parâmetro `-SIGSTOP` no comando `kill` não destruiria o processo, apenas o pausaria ou congelaria. Sendo assim, um processo com a compressão de um arquivo (comando `tar`) que demorasse muito tempo não precisaria ser necessariamente interrompido, caso precisasse liberar recursos de sistema. O parâmetro `-SIGCONT` descongela o processo, que retoma seu processamento exatamente de onde parou.

```
$ kill -SIGSTOP 2829
```

Linha de comando 2.1 – Exemplo do comando `kill` congelando o processo de PID 2829

Fonte: Elaborado pelo autor (2018)

Esclarecendo, a omissão do parâmetro manda um sinal de SIGKILL que, na prática, significa a destruição do processo.

Há uma variação do comando kill, conhecida como killall (ou “matar todos”), que deve ser acompanhado pelo nome do processo (e não o PID). Isso acontece pois, como o nome sugere, o comando pode destruir todos os processos que possuam o mesmo nome.

Para a demonstração, usei o servidor web mais utilizado do mundo, o Apache Web Server; caso precise de um servidor web, não deixe de instalá-lo com o comando `$ sudo aptinstall apache2`. O apache é útil neste exemplo, pois como todo serviço escalonável, ele cria subprocessos conforme a necessidade de atender a muitas requisições de rede. Logo “de cara”, seu processo principal gera outros dois processos, assim, temos três processos chamados apache2 (Figura 2.6):

```
hpoyatos@debian:~$ sudo /etc/init.d/apache2 start
[ ok ] Starting apache2 (via systemctl): apache2.service.
hpoyatos@debian:~$ ps aux | grep apache2
root      3793  0.0  0.1 75608 4368 ?        Ss   19:44   0:00 /usr/sbin/apache
2 -k start
www-data  3796  0.0  0.1 364768 4060 ?        Sl   19:44   0:00 /usr/sbin/apache
2 -k start
www-data  3797  0.0  0.1 364768 4060 ?        Sl   19:44   0:00 /usr/sbin/apache
2 -k start
hpoyatos  3868  0.0  0.0 12784   952 pts/0    S+   19:44   0:00 grep apache2
hpoyatos@debian:~$ sudo killall apache2
hpoyatos@debian:~$ ps aux | grep apache2
hpoyatos  3880  0.0  0.0 12784   948 pts/0    S+   19:44   0:00 grep apache2
hpoyatos@debian:~$
```

Figura 2.6 – Exemplo de execução do comando killall  
Fonte: Elaborado pelo autor (2018)

O primeiro comando do exemplo acima (`$ sudo /etc/init.d/apache2 start`) é a inicialização do daemon do apache2, o gerenciador de serviço; ou seja, é assim que um serviço no Linux é inicializado. Ao executar `$ psaux | grep apache2`, repare que possuo três processos, sendo um principal e dois subprocessos (embora não saiba dizer qual é qual). O comando `killall apache2` é acompanhado pelo `sudo`, pois o serviço pertence ao usuário `www-data` e, como podemos ver, ao listar os processos novamente, o comando faz o serviço sem deixar vestígios.

Outra forma de interromper processos é fechar o terminal no qual eles rodam. Processos não possuem apenas um usuário dono, mas um terminal (seja ele gráfico ou não) associado a eles.

#### 2.1.4 & - Rodando processos em background

Todo comando que executa em um terminal, ao seu final, devolve o *prompt* de comando para a execução de novos comandos, mas até o seu encerramento, ele “tratava” o terminal, não permitindo novas interações do usuário. Embora seja um fenômeno comum em comandos de modo texto, esse comportamento torna-se evidente em aplicativos gráficos, como o `gnome-chess`:

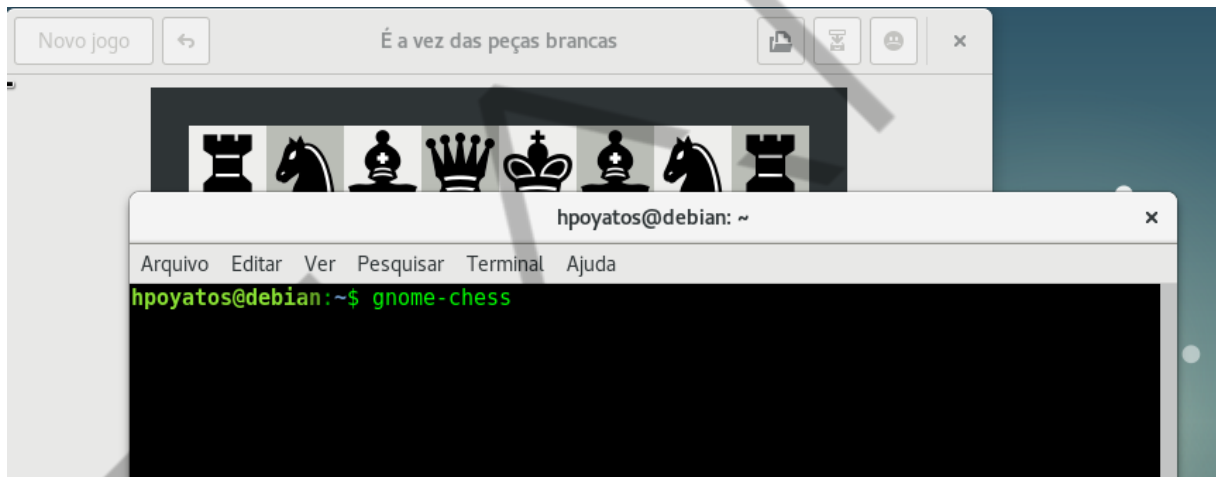


Figura 2.7 – Exemplo de execução do comando `gnome-chess` travando o terminal de comandos  
Fonte: Elaborado pelo autor (2018)

Isso acontece porque o processo está rodando em *foreground* ou primeiro plano. Podemos solicitar a execução de um comando em segundo plano ou *background*, desde que sua linha de comando seja acompanhada por um “&” (ê comercial) em seu final:

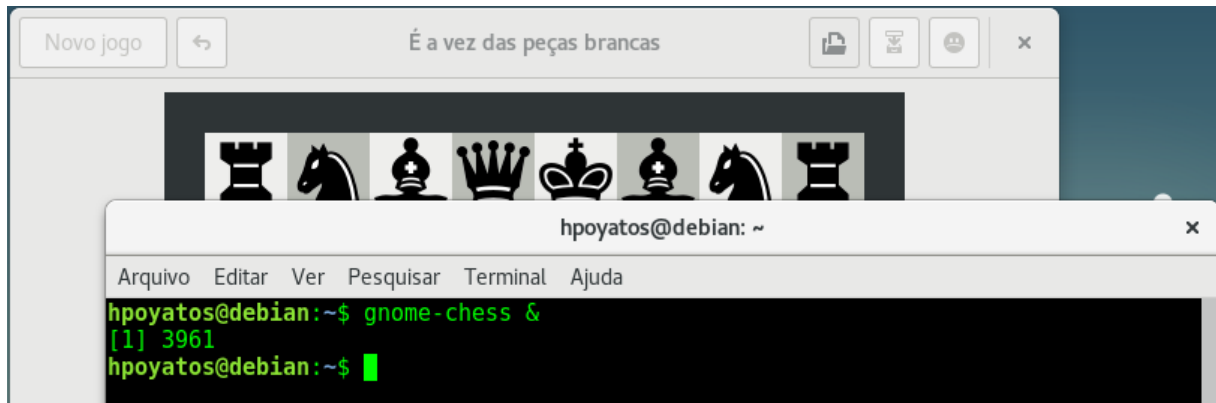


Figura 2.8 – Exemplo de execução do comando `gnome-chess` rodando em *background*  
Fonte: Elaborado pelo autor (2018)

Repare, na Figura 2.8, que o *prompt* de comando é devolvido, informando antes o identificador do processo (PID) que, neste caso, é 3961. No entanto, o fato do processo rodar em segundo plano não significa que este está livre do terminal; o processo continua atrelado ao terminal que, se for fechado, leva o tabuleiro de xadrez com ele.

### 2.1.5 Screen – criando terminais virtuais

Processos sempre estarão associados a terminais, isto é um fato. No entanto, há formas de criar terminais virtuais que podem ser abandonados e retomados em um segundo momento. Isso é particularmente interessante nos casos em que precisamos rodar comandos muito longos, como a realização de um *backup* (como tar ou outro comando) ou o *download* de um arquivo muito grande (com o bom e velho *wget*). Imagine, ainda, que você queira iniciar o comando fisicamente no servidor da empresa e deseja verificar a execução horas depois, remotamente, de sua casa? O comando *screen* pode ajudar nesse caso.

Não deixe de instalá-lo com `$sudo apt install screen`. O comando `screen -S [nome do terminal]` cria um terminal virtual que pode ser abandonado como [CTRL+A], seguida da tecla [D]:

```
hpoyatos@debian:~$ screen -S terminal3
[detached from 4337.terminal3]
hpoyatos@debian:~$ screen -ls
There are screens on:
  4337.terminal3  (17-02-2018 20:19:50)  (Detached)
  4301.terminal2  (17-02-2018 20:19:02)  (Detached)
  4294.terminal1  (17-02-2018 20:18:59)  (Detached)
3 Sockets in /run/screen/S-hpoyatos.
hpoyatos@debian:~$
```

Figura 2.9 – Exemplo de execução do comando screen -ls  
Fonte: Elaborado pelo autor (2018)

A Figura 2.9 Exemplo de execução do comando screen -ls”, mostra um terceiro terminal que foi criado e abandonado com [CTRL+A] e [D]. Ao executar screen -ls, podemos observar três terminais disponíveis. O comando \$ screen -r terminal2 que permite retornar ao terminal 4301.terminal2, e digitar “exit” nesse terminal, que encerra sua execução.

## 2.2 Acesso a máquinas remotas com SSH

Não há necessidade mais atual do que o acesso remoto a uma máquina. A obrigação de estar fisicamente presente em um terminal, em frente ao equipamento que se precisa manipular é absolutamente incompatível como os dias de hoje, uma realidade totalmente *Cloud Computing*, ou seja, por vezes, sequer sabemos em qual gabinete (ou quais) localiza-se o sistema que estamos utilizando. Há décadas, temos servidores sem monitores, teclados e mouses para chamar de seus, sendo necessário acessá-los de outro equipamento (bem, se voltar alguns anos mais, o mouse sequer existia). Terminais sem processamento, apenas com o monitor e teclado, eram destinados para tal fim (os chamados “terminais burros”).



Figura 2.10 – O pinguim remoto  
Fonte: Lynda.com (2016), adaptado por FIAP (2018)

Tão lendário quanto esse paradigma é o serviço que possibilitava isso no Unix, o **telnet**. Ele leva esse nome por possibilitar, ainda, um terminal via linha discada (sim, você pode ligar para seu servidor). Infelizmente, o serviço tornou-se obsoleto pela falta de segurança: os comandos trafegam abertamente em uma rede, sem criptografia, o que tornava a comunicação suscetível a um ataque conhecido como *sessionhijacking*, no qual o atacante sequestra a sessão como servidor, ao derrubar o cliente e se passar por ele (afinal, ele ficou um bom tempo “escutando” e sabe exatamente como o cliente se comporta).

O SSH (*Secure Shell* ou *shell* seguro) é a evolução do telnet, permitindo toda a praticidade de um acesso remoto com uma diferença: um túnel criptografado, garantindo a privacidade e evitando ataques.

### 2.2.1 SSH – acesso outra máquina remotamente

O comando `SSH` é o cliente para acesso a terminais remotos e seu uso é muito simples: ele deve ser seguido do usuário que desejo usar para conectar, “@”

(arroba) o nome ou IP do host. Caso o serviço de SSH não esteja em sua porta padrão (a 22), o parâmetro “-p” pode ser adicionado para indicar o número da porta.

Na demonstração a seguir, utilizei um **RaspberryPi** como *host* remoto. O equipamento está munido com um Raspian OS, que é um derivado do Debian específico para este tipo de *hardware*. O sistema remoto possui um serviço de SSH devidamente configurado com o OpenSSH, o mais utilizado do mercado, e está disponível na porta padrão do serviço – a porta 22. O endereço IP fixo do equipamento é local, em 192.168.0.100.



Figura 2.11 – Um RaspberryPi 3 model B  
Fonte: Raspberrypi.org (2018)

Ao acessar pela primeira vez o host remoto, este apresenta sua *fingerprint* que, na prática, é a apresentação de sua chave pública. O serviço de SSH utiliza uma criptografia assimétrica, também conhecida como chave público-privada, da qual cada host gera um par de chaves, distribuindo a chave pública e guardando a privada para si. As informações do terminal trafegam criptografadas, utilizando a chave privada do host de origem e a pública do remoto, e só podem ser descriptografadas pela combinação inversa (a chave pública do host de origem e a privada do remoto) tornando esse tipo de acesso prático e seguro.



```
hpoyatos@debian:~$ ssh pi@192.168.0.100
The authenticity of host '192.168.0.100 (192.168.0.100)' can't be established.
ECDSA key fingerprint is SHA256:KDxCNmmIVYp7edSLHgoEDE6d61ZuuWg74jfn45+WByw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.100' (ECDSA) to the list of known hosts.
pi@192.168.0.100's password:
Linux raspberrypi 4.9.59-v7+ #1047 SMP Sun Oct 29 12:19:23 GMT 2017 armv7l

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Feb 17 20:29:39 2018 from 192.168.0.10
pi@raspberrypi:~ $
```

Figura 2.12 – Exemplo de execução do comando SSH  
Fonte: Elaborado pelo autor (2018)

A Figura 2.12 “Exemplo de execução do comando SSH”, mostra que a chave pública do Raspberry é aceita e, na sequência, é solicitada a senha do usuário “pi” em 192.168.0.100. Ao digitar a senha correta, uma mensagem de boas-vindas é apresentada e podemos perceber a mudança do *prompt* de comandos (agora pi@raspberrypi:~ \$)

Absolutamente todos os comandos aprendidos aqui (desde que instalados no servidor remoto) são válidos como se estivéssemos sentados na frente do equipamento. Na prática, não faz a menor diferença. Ao terminar, podemos digitar “exit” e retornar ao host original:

```
hpoyatos@debian:~$ ssh pi@192.168.0.100
The authenticity of host '192.168.0.100 (192.168.0.100)' can't be established.
ECDSA key fingerprint is SHA256:KDxCNmmIVYp7edSLHgoEDE6d61ZuuWg74jfn45+WByw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.100' (ECDSA) to the list of known hosts.
pi@192.168.0.100's password:
Linux raspberrypi 4.9.59-v7+ #1047 SMP Sun Oct 29 12:19:23 GMT 2017 armv7l

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

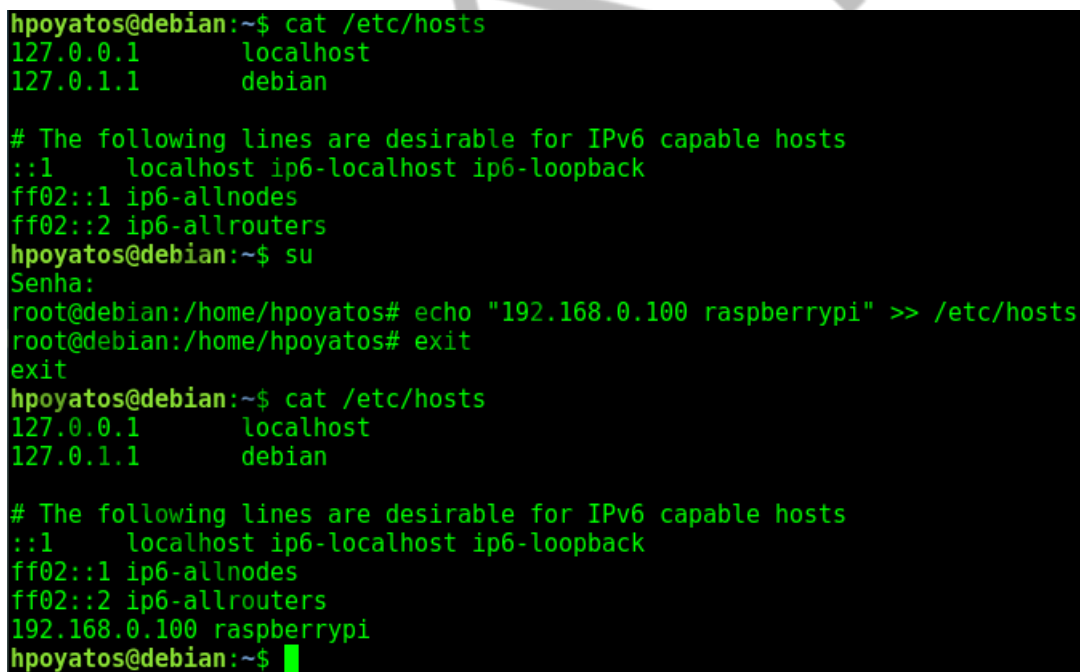
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Feb 17 20:29:39 2018 from 192.168.0.10
pi@raspberrypi:~ $
pi@raspberrypi:~ $
pi@raspberrypi:~ $
pi@raspberrypi:~ $ exit
logout
Connection to 192.168.0.100 closed.
hpoyatos@debian:~$ █
```

Figura 2.13 – Exemplo de execução do comando SSH ao sair do *host* remoto  
Fonte: Elaborado pelo autor (2018)



Você pode tornar o acesso mais prático, definindo os hosts no Linux usando o arquivo `/etc/hosts`; mesmo que o sistema esteja com um servidor DNS (*Domain Name System*) configurado e traduzindo *hostnames* em IPs, o sistema sempre olha para o `/etc/hosts` primeiro.

O exemplo da Figura 2.14 mostra a adição de um novo host no arquivo `/etc/hosts`, sem a necessidade de um processador de textos. Para tal, faz-se necessário tornar-se o superusuário `root`, pois o arquivo é protegido. O comando `echo` serve para “ecoar” algo no terminal, ou seja, tudo o que é digitado entre aspas seria repetido no terminal. Os símbolos “>>” (maior-maior) redirecionam a saída padrão do comando `echo` para dentro de um arquivo, incluindo o resultado do comando ao respectivo final. Tome cuidado apenas para não usar um sinal de maior só (“>”), pois o efeito é o mesmo, mas apaga o arquivo `/etc/hosts` inteiro no processo. A execução do comando `cat` mostra o antes e depois do arquivo:

A terminal window on a Debian system. The user 'hpoyatos' runs 'cat /etc/hosts' showing the current content: 127.0.0.1 localhost, 127.0.1.1 debian, and IPv6 entries. Then they run 'su' to become root. Root runs 'echo "192.168.0.100 raspberrypi" >> /etc/hosts' to append the new entry. Root then runs 'exit' to return to the user prompt. Finally, the user runs 'cat /etc/hosts' again, and the new entry '192.168.0.100 raspberrypi' is visible at the bottom of the file.

```
hpoyatos@debian:~$ cat /etc/hosts
127.0.0.1    localhost
127.0.1.1    debian

# The following lines are desirable for IPv6 capable hosts
::1         localhost ip6-localhost ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
hpoyatos@debian:~$ su
Senha:
root@debian:/home/hpoyatos# echo "192.168.0.100 raspberrypi" >> /etc/hosts
root@debian:/home/hpoyatos# exit
exit
hpoyatos@debian:~$ cat /etc/hosts
127.0.0.1    localhost
127.0.1.1    debian

# The following lines are desirable for IPv6 capable hosts
::1         localhost ip6-localhost ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
192.168.0.100 raspberrypi
hpoyatos@debian:~$
```

Figura 2.14 – Adicionando um host no arquivo `/etc/hosts`  
Fonte: Elaborado pelo autor (2018)

A partir de agora, o equipamento remoto também pode ser acessado pelo *hostname* definido em `/etc/hosts` (Figura 2.15Exemplo do comando SSH utilizando *hostname*):

```
hpoyatos@debian:~$ ssh pi@raspberrypi
The authenticity of host 'raspberrypi (192.168.0.100)' can't be established.
ECDSA key fingerprint is SHA256:KDXCNmmIVYp7edSLHgoEDE6d61ZuuWg74jfn45+WByw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'raspberrypi' (ECDSA) to the list of known hosts.
pi@raspberrypi's password:
Linux raspberrypi 4.9.59-v7+ #1047 SMP Sun Oct 29 12:19:23 GMT 2017 armv7l

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Feb 17 20:55:33 2018 from 192.168.0.10
pi@raspberrypi:~ $
pi@raspberrypi:~ $
pi@raspberrypi:~ $
pi@raspberrypi:~ $ exit
logout
Connection to raspberrypi closed.
hpoyatos@debian:~$
```

Figura 2.15 – Exemplo do comando SSH utilizando *hostname*  
Fonte: Elaborado pelo autor (2018)

### 2.2.2 SCP – Realizando transferências de arquivo seguras

O serviço de **SSH** não possibilita apenas o controle total de um servidor remoto, ele permite, ainda, a transferência de arquivos utilizando esse mesmo canal seguro. O comando **SCP** (*SecureCopy* ou cópia segura) pode ser utilizado para copiar arquivos inteiros de um servidor a outro, sem a necessidade de apelar para o antigo e ultrapassado FTP.

A sintaxe do comando é `scp [origem] [destino]`, referindo-se ao arquivo local da mesma forma que o comando `cp` funciona, e o local remoto da forma que o SSH funciona.

Vamos a um exemplo, para simplificar. Na Figura 2.16, estando no diretório pessoal do usuário (`/home/hpoyatos`), aponto para o arquivo `Imagem1.png` dentro do subdiretório `Imagens`, copiando-o para o diretório `/tmp` do RaspberryPi:

```
hpoyatos@debian:~$ scp Imagens/Imagem1.png pi@raspberrypi:/tmp
pi@raspberrypi's password:
Imagem1.png                               100%   0   0.0KB/s   00:00
hpoyatos@debian:~$
```

Figura 2.16 – Exemplo do comando `scp` transferindo um arquivo local para um servidor remoto  
Fonte: Elaborado pelo autor (2018)

Na Figura 2.17 “Exemplo do comando scp transferindo um arquivo de um servidor remoto para o local”, repare que podemos realizar o inverso: peço pelo arquivo Imagem1.png, que sei estar no diretório /tmp do RaspberryPi, e o copio para meu diretório atual (representando pelo símbolo “.”, o ponto):

```
hpoyatos@debian:~$ scp pi@raspberrypi:/tmp/Imagem1.png .
pi@raspberrypi's password:
Imagem1.png                               100%   0   0.0KB/s   00:00
hpoyatos@debian:~$ ls -la Imagem1.png
-rw-r--r-- 1 hpoyatos hpoyatos 0 fev 18 19:14 Imagem1.png
hpoyatos@debian:~$
```

Figura 2.17 – Exemplo do comando scp transferindo um arquivo de um servidor remoto para o local  
Fonte: Elaborado pelo autor (2018)

### 2.2.3 Subindo um serviço SSH

Falamos, até então, em como acessar um serviço SSH disponível, mas como configurar um serviço como esse na máquina em que estamos?

Na Figura 2.18 – Instalação do serviço de “”, começo pelo primeiro comando, ao instalar o serviço com `$ sudo apt install SSH`:

```
hpoyatos@debian:~$ sudo apt install ssh
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 linux-image-4.9.0-4-amd64 python-ecdsa
  python-hid python-mnemonic python-pbkdf2 python-protobuf
Utilize 'sudo apt autoremove' para os remover.
Os NOVOS pacotes a seguir serão instalados:
  ssh
0 pacotes atualizados, 1 pacotes novos instalados, 0 a serem removidos e 0 não a
tualizados.
É preciso baixar 189 kB de arquivos.
Depois desta operação, 208 kB adicionais de espaço em disco serão usados.
Obter:1 http://ftp.br.debian.org/debian stretch/main amd64 ssh all 1:7.4p1-10+de
b9u2 [189 kB]
Baixados 189 kB em 1s (181 kB/s)
A seleccionar pacote anteriormente não seleccionado ssh.
(Lendo banco de dados ... 135729 ficheiros e directórios actualmente instalados.
)
A preparar para desempacotar .../ssh_1%3a7.4p1-10+deb9u2_all.deb ...
A descompactar ssh (1:7.4p1-10+deb9u2) ...
Configurando ssh (1:7.4p1-10+deb9u2) ...
```

Figura 2.18 – Instalação do serviço de SSH  
Fonte: Elaborado pelo autor (2018)

O comando mostrado na próxima figura, por sua vez, é utilizado para subir o serviço através do seu *daemon*, localizado em `/etc/init.d/ssh`:

```
hpoyatos@debian:~$ sudo /etc/init.d/ssh start
[sudo] senha para hpoyatos:
[ ok ] Starting ssh (via systemctl): ssh.service.
hpoyatos@debian:~$
```

Figura 2.19 – Subindo o serviço de SSH  
Fonte: Elaborado pelo autor (2018)

## 2.2.4 Tornando o SSH ainda mais seguro

Embora seja um serviço que conte com criptografia e possua uma configuração inicial razoavelmente segura, o acesso remoto provido pelo SSH é muito visado por eventuais invasores, justamente pelo seu potencial de manipulação remota (embora não seja, nem de longe, a única maneira de invadir uma máquina).

Com o intuito de proteger ainda mais o serviço, seguem algumas dicas rápidas de como tornar o SSH mais seguro do que já é:

- **Use senhas seguras em seus usuários:** o ataque mais banal que se procura fazer é um ataque de força bruta (ou *brute force attack*), no qual se conecta ao serviço e tenta-se adivinhar a senha, testando todas as combinações possíveis. Sem muita dificuldade, o atacante pode automatizar o processo, usando todas as palavras do vocabulário de um idioma, conhecido como ataque de dicionário, ou seja, senhas muito simples são facilmente descobertas e garantem um acesso simples e rápido. Não deixe isso acontecer.
- **Configure o serviço:** por falar em força bruta, o arquivo `/etc/ssh/sshd_config` permite uma série de parametrizações interessantes, como determinar o número de tentativas máximas para senhas. Garanta que o usuário *root* não possa se autenticar e restrinja o serviço para as interfaces de rede em que ele precise realmente ficar disponível. Você pode restringir o acesso apenas para *hosts* conhecidos (aqueles dos quais você possui as chaves públicas). Não será prático adicionar novos *hosts*, mas torna-se mais seguro. Aprenderemos, na sequência, sobre o editor de textos *vi* e como alterar arquivos com ele.

- **Restrinja os usuários que podem usar SSH:** nem todo mundo precisa ter acesso remoto. Para fazê-lo, adicione, no arquivo `/etc/ssh/sshd_config`, as diretivas ***allowuser***, para especificar os usuários permitidos (separe-os por vírgula), ou ***allowgroup***, para liberar grupos. As diretivas ***denyuser*** e ***denygroup*** permitem definir quais usuários e grupos estão proibidos de usá-lo.
- **Troque a porta de serviço:** no mesmo arquivo `/etc/ssh/sshd_config`, você pode trocar o número da porta de serviço por outro número que não seja o óbvio 22.
- **Apenas IPs permitidos:** aprenderemos sobre configuração de *firewall* em um capítulo de uma etapa seguinte; quando acontecer, você pode usar o comando ***iptables*** para definir quais IPs podem acessar a porta de serviço na qual o SSH está.
- **Identifique-se primeiro:** outro truque muito interessante, de que trataremos em outro capítulo, é a possibilidade de implementar um *Port Knocking*, que em tradução livre significa “batidas na porta”; configuramos o *firewall* para liberar a porta de serviço apenas quando pacotes de dados são enviados para portas específicas em uma determinada sequência; podemos até mesmo configurar a sequência para fechar a porta ao sair! Muito seguro, não é?

### 2.3 VI: o melhor editor de texto

O título desta seção é tendencioso e está longe de ser imparcial, mas é, também, uma verdade: embora existam outros editores de texto, como o **emacs** ou o **nano** (que é bem fácil de usar), o **vi** (lê-se vi-ai) ou **vim** (***Vi Improved*** ou “vi melhorado”) possui uma infinidade de recursos interessantes e poderosos.

Praticamente todas as letras minúsculas e maiúsculas disparam comandos no editor; portanto, para que este tutorial não fique muito complexo (e chato), vou me restringir às funcionalidades mais comuns e úteis!

### 2.3.1 Entrando e saindo do VI

Para entrar no vi, basta digitar, no terminal, `$ vi`. Normalmente, o comando é acompanhado com o nome do arquivo que se quer criar ou editar, conforme podemos ver na Linha de comando “Exemplo do comando vi criando ou editando arquivo.txt”, a seguir:

```
$ vi arquivo.txt
```

Linha de comando 2.2 – Exemplo do comando vi criando ou editando arquivo.txt

Fonte: Elaborado pelo autor (2018)

A tela do editor é muito simples, como podemos ver na Figura 2.20:



Figura 2.20 – Tela inicial do vi criando arquivo.txt

Fonte: Elaborado pelo autor (2018)

Para sair do editor, você deve digitar: `":q"` (dois pontos e a letra q minúscula, de *quit*). Sugiro acostumar-se com a combinação das teclas [ESC], [:], [q] e [ENTER], pois você pode estar em modo de edição de texto, e a tecla [ESC] fará voltar ao modo de comandos.

### 2.3.2 Os modos de comando e de edição de texto

O vi conta com dois modos: um modo de comandos, no qual as teclas que você digitar dispararão comandos no editor, e o modo de edição de texto, no qual tudo aquilo que você digitar fará, literalmente, parte do documento.

Existem várias maneiras de migrar do modo de comando para o modo de edição de texto, vou me restringir às quatro mais interessantes:

1) A mais simples é a tecla [i], através da qual entramos em modo de edição de texto **no ponto em que o cursor estiver no momento**. Na Figura 2.21 Editando um arquivo com vi usando [i], digite [i] e a frase “Vi é muito legal!”, seguido de [ESC], para voltar ao modo de comandos:

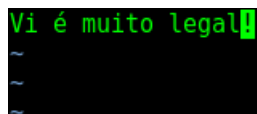


Figura 2.21 – Editando um arquivo com vi usando [i]  
Fonte: Elaborado pelo autor (2018)

2) Outra forma é a tecla [a], através da qual entramos em modo de edição de texto **uma posição após o ponto em que o cursor estiver no momento**. Na Figura 2.21, digite [a] com o cursor posicionado em “!” e, dessa maneira, podemos continuar a frase exatamente de onde havia parado; digite “E muito rápido, também.”, seguido de [ESC], para voltar ao modo de comandos:

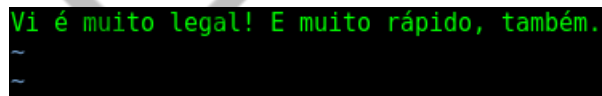


Figura 2.22 – Editando um arquivo com vi usando [a]  
Fonte: Elaborado pelo autor (2018)

3) O terceiro modo é editar com [A], e entramos em modo de edição **no final da frase em que ‘estamos**. Para a demonstração da Figura 2.23 “Editando um arquivo com vi usando [A], retorne com [←] até chegar ao início da frase, embora [^] nos posicione imediatamente no início da frase. Ao digitar [A], somos enviados imediatamente ao final da frase, em modo de edição, e digite “E cheio de truques!”, seguido de [ESC], para voltar ao modo de comandos:

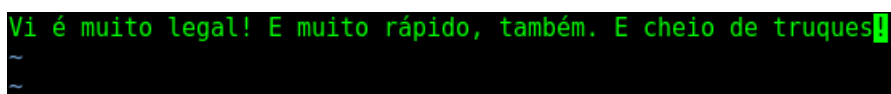


Figura 2.23 – Editando um arquivo com vi usando [A]  
Fonte: Elaborado pelo autor (2018)

4) Por último, podemos usar [o] e entrar no modo de edição **na linha seguinte àquela em que estamos**. Para a demonstração da Figura 2.24 Editando um arquivo com vi usando [o]”, basta digitar [o] e a frase “Acho que estou pegando o jeito!”, seguido de [ESC], para voltar ao modo de comandos:

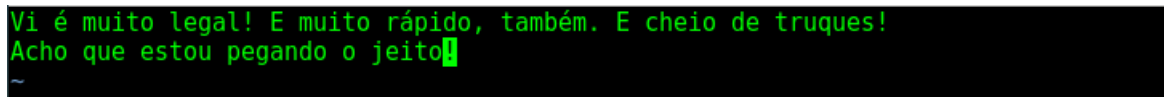


Figura 2.24 – Editando um arquivo com vi usando [o]  
Fonte: Elaborado pelo autor (2018)

E acredite, existem ainda outras maneiras. Quanto a retornar ao modo de comandos, acho que você já entendeu como funciona.

### 2.3.3 Salvando o arquivo

Para salvar o arquivo, você deve estar em modo de comandos, e pode utilizar as teclas [:], [w] (de *write*) e [ENTER]. Para salvar e sair do arquivo, você pode combinar as letras “w” e “q”, usando a sequência [:], [w], [q] e [ENTER] (:wq), embora eu prefira [:], [x] e [ENTER], que faz o serviço com uma tecla a menos.

Quer sair sem salvar? Utilize [:], [q], [!] e [ENTER] (ou :q!) em modo de comando.

### 2.3.4. Numeração de linhas e como ir direto para uma delas

Para os demais exemplos, vamos aumentar o tamanho do arquivo, como podemos ver na Figura 2.25 Exemplo de documento no vi”:



```
Vi é muito legal! E muito rápido, também. E cheio de truques!
Acho que estou pegando o jeito!

Ele parece complicado mas nada que uma boa prática não resolva!

Por ser muito rápido e leve, é ideal para editar arquivos em linha de comando.
Mas, se eu quisesse alguém rápido e leve, escolheria o nano! Além dessas
duas características, o vi é extremamente poderoso, e neste tutorial
você verá apenas um lampejo disso!

Quem sabe você não toma gosto e se aprofunda depois, descobrindo mais e mais
atalhos?

Talvez um dia alguém lhe fale do emacs, mas o vi é mais legal, vai por mim!
(E o prof. Poyatos continua alimentando um dos Flame Wars ou "Fla-Flus" mais
antigos da tecnologia... vi x emacs!)

~
~
~
~
~

"arquivo.txt" 16 lines, 713 characters
```

Figura 2.25 – Exemplo de documento no vi  
Fonte: Elaborado pelo autor (2018)

Segue o texto no *box*, caso queira me acompanhar:

```
Vi é muito legal! E muito rápido, também. E cheio de truques!
Acho que estou pegando o jeito!

Ele parece complicado, mas nada que uma boa prática não resolva!

Por ser muito rápido e leve, é ideal para editar arquivos em linha de comando.
Mas, se eu quisesse alguém rápido e leve, escolheria o nano! Além dessas
duas características, o vi é extremamente poderoso e, neste tutorial,
você verá apenas um lampejo disso!

Quem sabe você não toma gosto e se aprofunda depois, descobrindo mais e
maisatalhos?

Talvez, um dia, alguém lhe fale do emacs, mas o vi é mais legal, vai por mim!
(E o prof. Poyatos continua alimentando um dos Flame Wars ou "Fla-Flus" mais
antigos da tecnologia... vi x emacs!)
```

Linha de comando 2.3 – Texto digitado em arquivo.txt  
Fonte: Elaborado pelo autor (2018)

Não ver a numeração das linhas pode ser um problema em algumas situações. Imagine no caso de editar um código-fonte cujo depurador apontou um erro na linha 11? Para habilitar as linhas, digite a sequência [:], "set nu" e [ENTER] (ou :set nu), no modo de comandos:

```
1 Vi é muito legal! E muito rápido, também. E cheio de truques!
2 Acho que estou pegando o jeito!
3
4 Ele parece complicado mas nada que uma boa prática não resolva!
5
6 Por ser muito rápido e leve, é ideal para editar arquivos em linha de comando.
7 Mas, se eu quisesse alguém rápido e leve, escolheria o nano! Além destas
8 duas características, o vi é extremamente poderoso, e neste tutorial
9 você verá apenas um lampejo disso!
10
11 Quem sabe você não toma gosto e se aprofunda depois, descobrindo mais e mais
12 atalhos?
13
14 Talvez um dia alguém lhe fale do emacs, mas o vi é mais legal, vai por mim!
15 (E o prof. Poyatos continua alimentando um dos Flame Wars ou "Fla-Flus" mais
16 antigos da tecnologia... vi x emacs!)
```

Figura 2.26 – Exemplo de documento no vi com numeração de linhas  
Fonte: Elaborado pelo autor (2018)

Se quisermos nos dirigir rapidamente à linha 11, podemos usar a tecla [↓] dez vezes até chegar lá. Podemos usar as setas para navegar para qualquer parte do documento, **desde que estejamos em modo de comando**: usar as setas em modo de edição sujará seu documento com os caracteres mais bizarros possíveis e, se meu aviso veio tarde demais, a seção **2.3.6 Apagando textos** é para você.

Ir até a linha 11 é fácil, mas e se fosse a linha 2853? Melhor aprender a ir direto para onde precisa, você não acha? Experimente digitar [:], "11" e [ENTER]:

```
1 Vi é muito legal! E muito rápido, também. E cheio de truques!
2 Acho que estou pegando o jeito!
3
4 Ele parece complicado mas nada que uma boa prática não resolva!
5
6 Por ser muito rápido e leve, é ideal para editar arquivos em linha de comando.
7 Mas, se eu quisesse alguém rápido e leve, escolheria o nano! Além destas
8 duas características, o vi é extremamente poderoso, e neste tutorial
9 você verá apenas um lampejo disso!
10
11 Quem sabe você não toma gosto e se aprofunda depois, descobrindo mais e mais
12 atalhos?
13
14 Talvez um dia alguém lhe fale do emacs, mas o vi é mais legal, vai por mim!
15 (E o prof. Poyatos continua alimentando um dos Flame Wars ou "Fla-Flus" mais
16 antigos da tecnologia... vi x emacs!)
```

Figura 2.27 – Exemplo para acessar uma linha específica no vi  
Fonte: Elaborado pelo autor (2018)

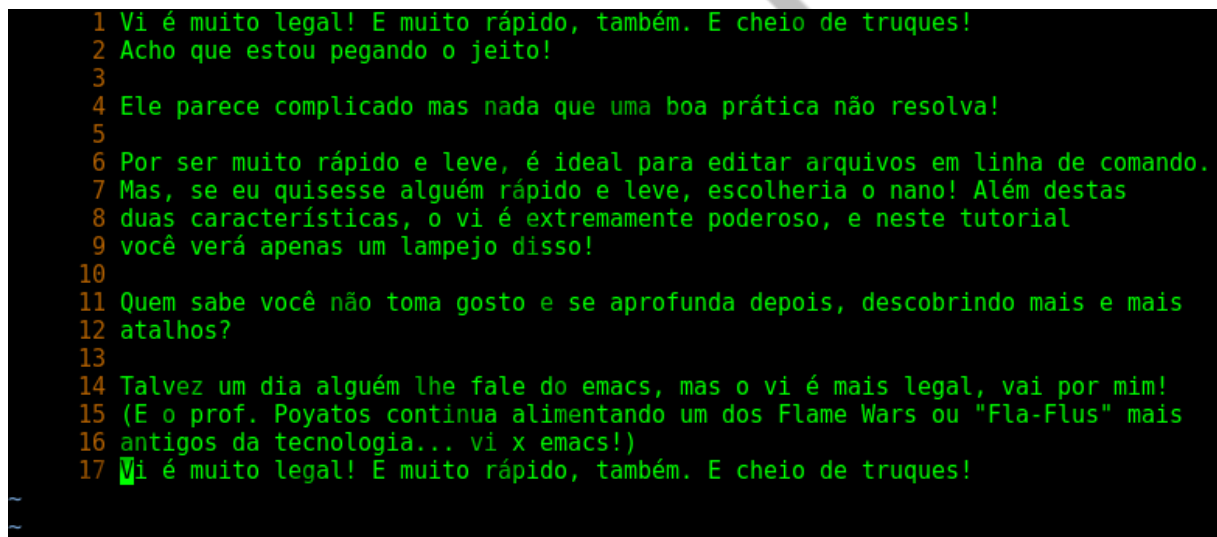
Não negue, você está sorrindo neste exato momento.

### 2.3.5 Copiando e colando textos

Não seríamos ninguém sem a possibilidade de copiar e colar textos. Você não usaria o **vi** se isso não fosse possível (bem, nem eu), e, talvez, muitos falem mal do **vi** justamente por não saberem como fazer.

Vamos começar copiando uma linha inteira. Vá para a primeira linha (:1 em modo de comandos resolve) e digite [y][y] (de *yank* ou “arrancar”; por quê? Não sei) no modo de comando. Embora não tenha avisado, acredite, ele copiou a primeira linha.

Vá, agora, para a última linha do arquivo (experiente [G] em modo de comando) e digite [p] (de *paste* ou colar):



```
1 Vi é muito legal! E muito rápido, também. E cheio de truques!
2 Acho que estou pegando o jeito!
3
4 Ele parece complicado mas nada que uma boa prática não resolva!
5
6 Por ser muito rápido e leve, é ideal para editar arquivos em linha de comando.
7 Mas, se eu quisesse alguém rápido e leve, escolheria o nano! Além destas
8 duas características, o vi é extremamente poderoso, e neste tutorial
9 você verá apenas um lampejo disso!
10
11 Quem sabe você não toma gosto e se aprofunda depois, descobrindo mais e mais
12 atalhos?
13
14 Talvez um dia alguém lhe fale do emacs, mas o vi é mais legal, vai por mim!
15 (E o prof. Poyatos continua alimentando um dos Flame Wars ou "Fla-Flus" mais
16 antigos da tecnologia... vi x emacs!)
17 Vi é muito legal! E muito rápido, também. E cheio de truques!
```

Figura 2.28 – Exemplo de como copiar e colar uma linha no vi  
Fonte: Elaborado pelo autor (2018)

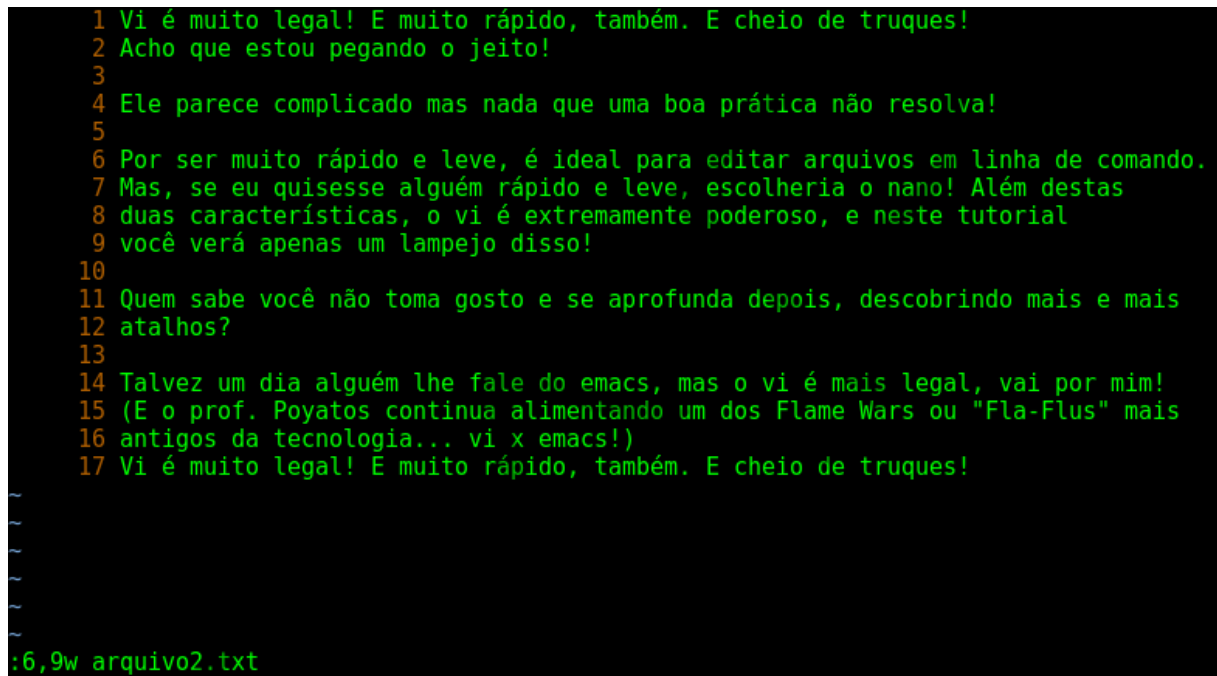
A propósito, um único [y] também copiaria, mas ele copia um parágrafo (ou melhor, até achar uma linha em branco); ou seja, na primeira linha e copiaria duas linhas, enquanto com o cursor na linha 6 teria copiado quatro.

Para copiar um número específico de linhas, basta colocar a quantidade antes do comando, ou seja, digitar 10yy em modo de comando fará o vi copiar dez linhas, sendo a linha em que você está e as nove seguintes.

Você pode copiar trechos de linha também. As teclas yw (de *word* ou palavra) copiarão uma única palavra (ou seja, até o vi achar um *caracter* de espaço), y\$ copiará do ponto em que o cursor esteja até o final da linha, enquanto y^ faz o

contrário, copiará de onde estivermos até o início da linha. Talvez você conheça **expressões regulares** e tenha reconhecido o “^” para início de sentença e o “\$” para final de sentença; caso contrário, recomendo, é sempre muito útil!

E se quisermos copiar um trecho do documento para outro arquivo? Na Figura 2.29 “Exemplo de como copiar um trecho do arquivo para outro”, podemos ver como copiar das linhas 6 a 9 de um arquivo para outro, que chamei de arquivo2.txt:



```
1 Vi é muito legal! E muito rápido, também. E cheio de truques!
2 Acho que estou pegando o jeito!
3
4 Ele parece complicado mas nada que uma boa prática não resolva!
5
6 Por ser muito rápido e leve, é ideal para editar arquivos em linha de comando.
7 Mas, se eu quisesse alguém rápido e leve, escolheria o nano! Além destas
8 duas características, o vi é extremamente poderoso, e neste tutorial
9 você verá apenas um lampejo disso!
10
11 Quem sabe você não toma gosto e se aprofunda depois, descobrindo mais e mais
12 atalhos?
13
14 Talvez um dia alguém lhe fale do emacs, mas o vi é mais legal, vai por mim!
15 (E o prof. Poyatos continua alimentando um dos Flame Wars ou "Fla-Flus" mais
16 antigos da tecnologia... vi x emacs!)
17 Vi é muito legal! E muito rápido, também. E cheio de truques!

:6,9w arquivo2.txt
```

Figura 2.29 – Exemplo de como copiar um trecho do arquivo para outro  
Fonte: Elaborado pelo autor (2018)

### 2.3.6 Apagando textos

Para apagar textos, os procedimentos são os mesmos usados para copiar, mas você substitui o “y” por “d” (de *delete* ou apagar); ou seja, digitar `dd` em modo comando apaga uma única linha, enquanto `10dd` apaga a linha em que você está e mais nove; `dw` apaga a palavra em que você está, `d$` do ponto em que está até o final da linha, enquanto `d^` apaga de onde está até o início. Quer apagar um único carácter? `d→`.

Apagou algo errado? Digite `[u]` e `[ENTER]` (de *undo* ou desfazer); com `[u]` você pode desfazer o último comando, e **APENAS O ÚLTIMO**; Se for o penúltimo, danou-se.

Quer recortar? Use os comandos “d” também. Talvez você se pergunte: “ué, dd não é para apagar uma linha?”. Na verdade, ele recorta a linha. Se você usar o [p] (de *paste*) na sequência, ele realiza um recortar-e-colar. Engenhoso, não?

### 2.3.7 Localizando e substituindo palavras

É indispensável saber fazer uma busca por palavra-chave em um documento. Para fazer isso no vi, digite [/], a palavra que você deseja e [ENTER], como você pode ver na Figura 2.30 “Exemplo de busca por palavra-chave no vi”:

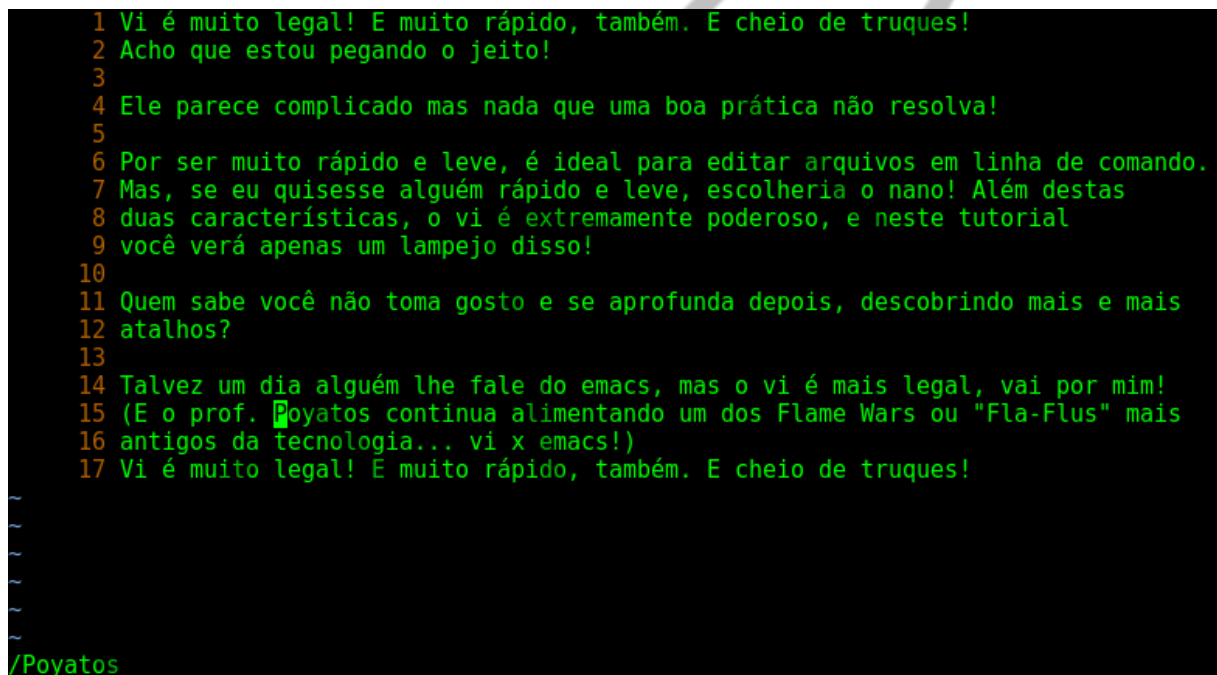
A screenshot of the vi editor interface. The background is black, and the text is green. It shows a document with 17 lines of text. Line 15 contains the word 'mais'. At the bottom of the screen, the search command '/Poyatos' is entered, and the cursor is positioned at the start of line 15, where the word 'Poyatos' appears. The text in the editor is as follows:  
1 Vi é muito legal! E muito rápido, também. E cheio de truques!  
2 Acho que estou pegando o jeito!  
3  
4 Ele parece complicado mas nada que uma boa prática não resolva!  
5  
6 Por ser muito rápido e leve, é ideal para editar arquivos em linha de comando.  
7 Mas, se eu quisesse alguém rápido e leve, escolheria o nano! Além destas  
8 duas características, o vi é extremamente poderoso, e neste tutorial  
9 você verá apenas um lampejo disso!  
10  
11 Quem sabe você não toma gosto e se aprofunda depois, descobrindo mais e mais  
12 atalhos?  
13  
14 Talvez um dia alguém lhe fale do emacs, mas o vi é mais legal, vai por mim!  
15 (E o prof. Poyatos continua alimentando um dos Flame Wars ou "Fla-Flus" mais  
16 antigos da tecnologia... vi x emacs!)  
17 Vi é muito legal! E muito rápido, também. E cheio de truques!  
At the bottom, the prompt is /Poyatos and the cursor is at the beginning of line 15.

Figura 2.30 – Exemplo de busca por palavra-chave no vi  
Fonte: Elaborado pelo autor (2018)

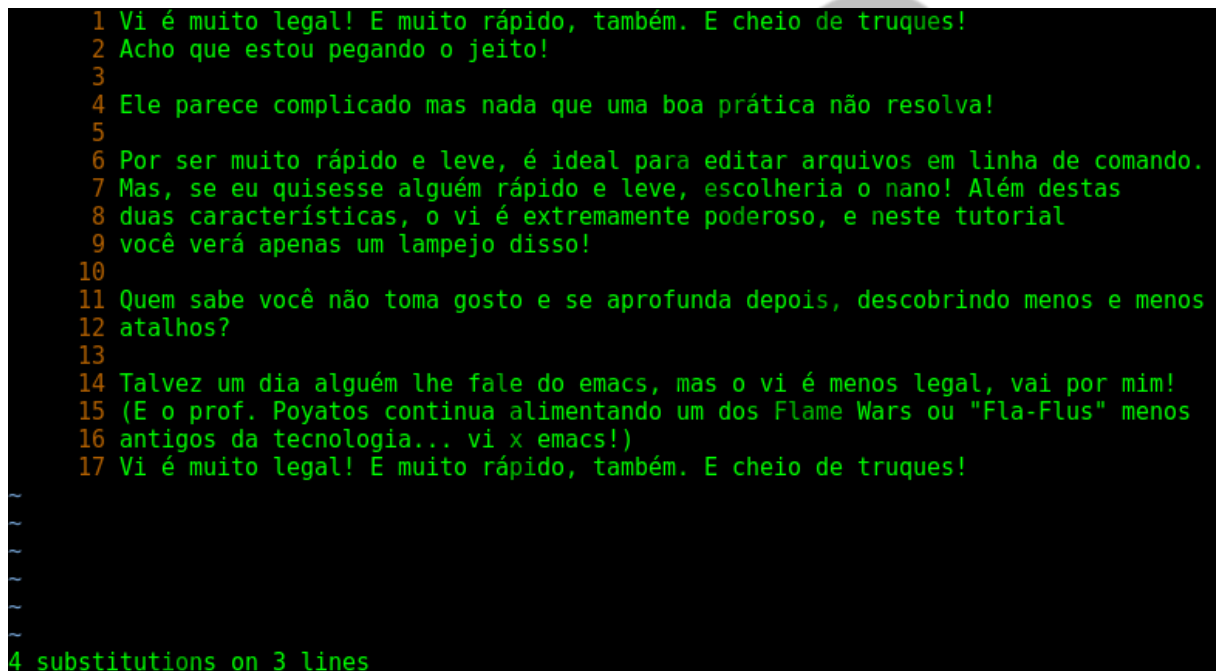
Se for uma palavra-chave por mais de uma ocorrência (neste exemplo, a melhor palavra é justamente “mais”), digite [n] para pular para a próxima ocorrência; para voltar para a anterior, use [N].

E para substituir termos no documento? Na Linha de comando “Exemplo de substituição da palavra “mais” por “menos” no vi” mostro como substituir todas as ocorrências da palavra “mais” por “menos”:

```
:%s/mais/menos/gc
```

Linha de comando 2.4 – Exemplo de substituição da palavra “mais” por “menos” no vi  
Fonte: Elaborado pelo autor (2018)

O “c” ao final do comando é para pedir confirmação para cada substituição; se você remover o “c” do comando, ele troca todas as ocorrências sem a maior cerimônia. A letra “g” antes dela indica que a substituição é **g**lobal, ou seja, para todas as ocorrências; sua ausência faria o **vi** substituir apenas a primeira palavra “mais” que encontrasse, deixando as demais intactas. Veja como fica a execução da linha de comando “Exemplo de substituição da palavra “mais” por “menos” no vi” na figura “Exemplo de substituição de termos no vi”:



```
1 Vi é muito legal! E muito rápido, também. E cheio de truques!
2 Acho que estou pegando o jeito!
3
4 Ele parece complicado mas nada que uma boa prática não resolva!
5
6 Por ser muito rápido e leve, é ideal para editar arquivos em linha de comando.
7 Mas, se eu quisesse alguém rápido e leve, escolheria o nano! Além destas
8 duas características, o vi é extremamente poderoso, e neste tutorial
9 você verá apenas um lampejo disso!
10
11 Quem sabe você não toma gosto e se aprofunda depois, descobrindo menos e menos
12 atalhos?
13
14 Talvez um dia alguém lhe fale do emacs, mas o vi é menos legal, vai por mim!
15 (E o prof. Poyatos continua alimentando um dos Flame Wars ou "Fla-Flus" menos
16 antigos da tecnologia... vi x emacs!)
17 Vi é muito legal! E muito rápido, também. E cheio de truques!

4 substitutions on 3 lines
```

Figura 2.31 – Exemplo de substituição de termos no vi  
Fonte: Elaborado pelo autor (2018)

O texto não fez muito sentido agora, mas você entendeu a ideia.

## CONCLUSÃO

Neste capítulo, abordamos alguns pontos muito importantes no sistema do pinguim. Começamos pelo gerenciamento de processos, indispensável na administração de sistemas, e pudemos perceber o quanto o Linux nos permite gerenciá-los facilmente, disponibilizando diversas ferramentas para tal.

Passamos pelo serviço mais importante do sistema operacional, o ssh. Considero impensável sobreviver sem ele nos dias de hoje. Como configurar um servidor em *cloud* sem ele? Absolutamente impossível.

Por fim, seríamos incapazes de editar arquivos de configuração sem a ajuda com melhor editor de textos em linha de comando do mercado, o vi. A aliança ssh+vi+ os comandos que você aprendeu até agora, o céu é o limite.

## REFERÊNCIAS

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

EMANIP



## GLOSSÁRIO

<b>SSH</b>	<i>Secure Shell</i> , ou <i>shell</i> seguro. Trata-se de 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.
------------	---