



Transcrição

Bem-vindo ao Curso de Linux II: Programas, processos e pacotes!

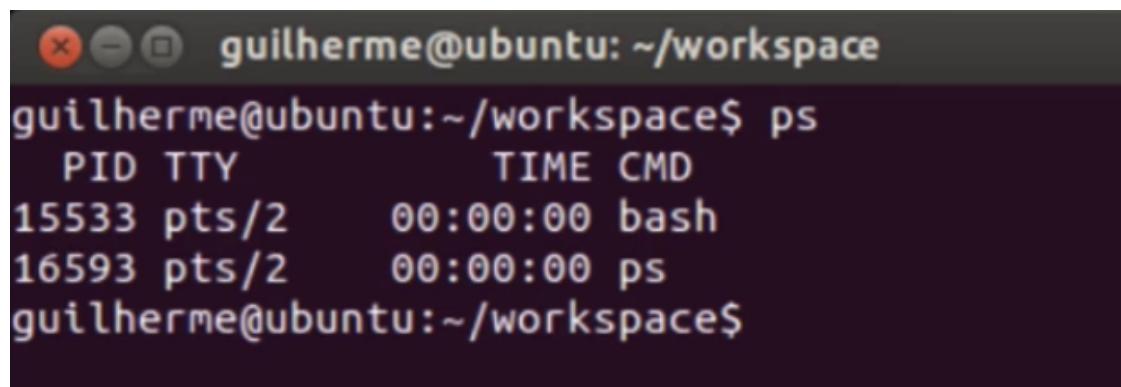
Neste curso, que é continuação do [Curso de Linux I: Conhecendo e utilizando o terminal](https://cursos.alura.com.br/course/linux-ubuntu) (<https://cursos.alura.com.br/course/linux-ubuntu>) iremos aprender diversas coisas:

- Descobrir quais são os processos que estão sendo executados em nossa máquina e como finalizar tais processos;
- Executar programas no *background* ou no *foreground*;
- Configurar programas para ter permissão de execução;
- Procurar arquivos no Sistema Operacional;
- Gerenciar novos usuários;
- Configurar as permissões de acesso aos arquivos;
- Configurar variáveis de ambiente;
- Instalar novos programas do zero;
- Inicializar e parar serviços que rodam junto com o início da máquina;
- Compilar e instalar um programa do zero a partir de seu código fonte;
- Fazer o acesso remoto às máquinas que rodam Linux;

Vamos começar?

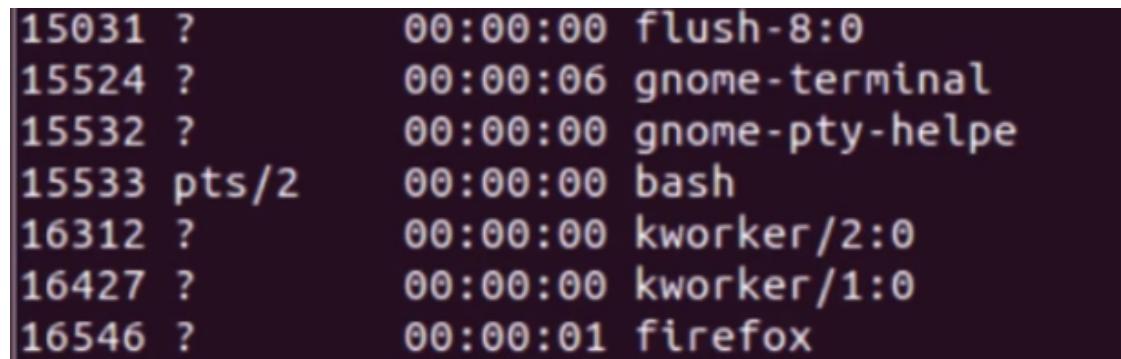
Cada comando que invocamos é executado pelo CPU. Alguns são

finalizados rapidamente como o `ls` e o `mkdir`. Outros demoram mais tempo para parar de executar, alguns exemplos são o `firefox` e o `gedit`. Perceba que se, por exemplo, abrirmos o *Firefox* ou o editor de texto *gedit*, eles ficam em execução. Para sabermos quais processos estão sendo executados no momento, usamos o comando `ps` no Terminal.



```
guilherme@ubuntu: ~/workspace
guilherme@ubuntu:~/workspace$ ps
  PID TTY          TIME CMD
15533 pts/2    00:00:00 bash
16593 pts/2    00:00:00 ps
guilherme@ubuntu:~/workspace$
```

Perceba que só serão mostrados os processos do *bash* (Terminal) atual naquele instante e a execução do comando `ps`. Queremos saber de **todos** os processos em **todo** o sistema, então fazemos `ps -e`. Nos aparecerá uma lista extensa.



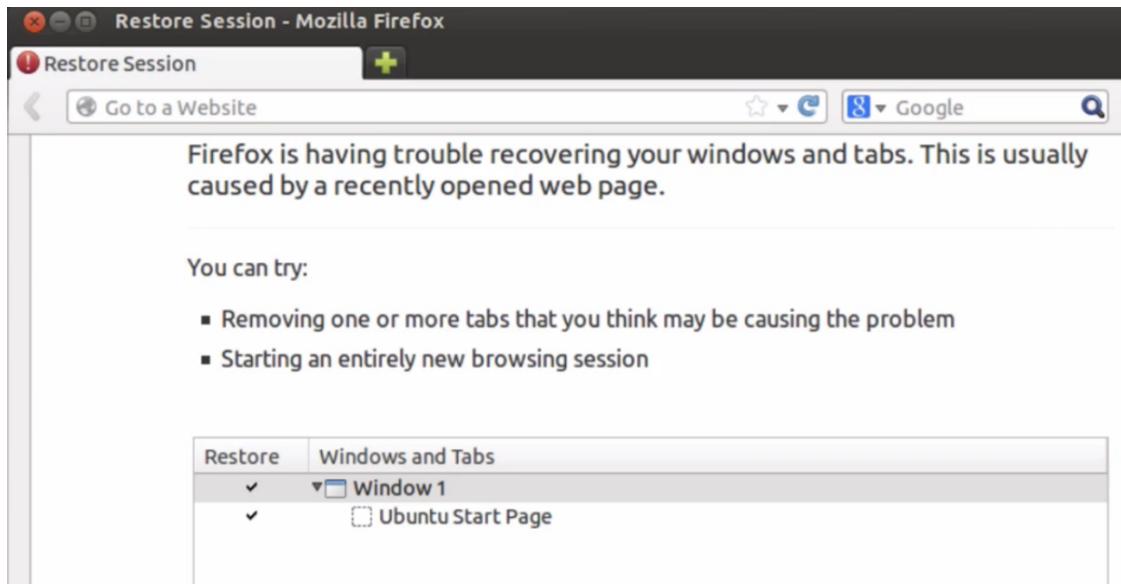
```
15031 ?        00:00:00 flush-8:0
15524 ?        00:00:06 gnome-terminal
15532 ?        00:00:00 gnome-pty-help
15533 pts/2    00:00:00 bash
16312 ?        00:00:00 kworker/2:0
16427 ?        00:00:00 kworker/1:0
16546 ?        00:00:01 firefox
```

Note que o *Firefox* ainda está em execução. A listagem dos processos exibe não só o nome do processo mas também seu ID. Podemos utilizar o ID para finalizarmos um programa usando o comando `kill + ID`, para encerrarmos o *Firefox*, neste caso, fazemos:

`kill 16546`

[COPIAR CÓDIGO](#)

Alguns programas, quando abertos novamente, após matarmos seu processo, mostrarão uma mensagem de alerta para dizer que foram fechados de uma maneira diferente da comum (*clicando no botão de fechar, por exemplo*) e tentam se recuperar, *Firefox* por exemplo, mostra uma mensagem semelhante a da imagem a seguir:



Muitas vezes não conseguimos matar o processo quando o programa trava e para termos certeza de que conseguiremos finalizá-lo, forçando a finalização, podemos usar o modificador `-9` para o comando `kill` da seguinte forma:

```
kill -9 16546
```

[COPIAR CÓDIGO](#)

A diferença para do `kill` com e sem o modificador `-9` é simples: sem o modificador, o comando solicita o fechamento do programa, dando uma chance para o programa se encerrar sozinho, com o modificador essa chance não existe, o processo é encerrado imediatamente.

Podemos mostrar mais detalhes sobre os processos que estão em execução, além de seu nome e ID. Para isso fazemos `ps -ef`, este

comando mostra a localização dos programas, o instante em que eles foram inicializados e outras informações. A imagem abaixo ilustra a saída desse comando.

root	15031	2	0	22:06	?	00:00:00	[flush-8:0]
1000	15524	1	0	22:19	?	00:00:06	gnome-terminal
1000	15532	15524	0	22:19	?	00:00:00	gnome-pty-helper
1000	15533	15524	0	22:19	pts/2	00:00:00	bash
root	16312	2	0	22:39	?	00:00:00	[kworker/2:0]
root	16427	2	0	22:44	?	00:00:00	[kworker/1:0]

Porém ainda fica difícil encontrar um programa que estejamos procurando com essa listagem enorme de processos. Podemos usar o comando `ps -ef | grep NomeDoPrograma` para filtrar os resultados da listagem:

```
ps -ef | grep firefox
```

[COPIAR CÓDIGO](#)

O comando `grep` é utilizado para filtragem de dados dada uma entrada, neste caso a entrada é o resultado do comando `ps`. O `|` é um redirecionador de saídas de programas para programas, ou seja, ele redireciona a saída do comando `ps` para ser usada como entrada para o comando `grep`. Por isso, são mostrados apenas as linhas que possuam a palavra `firefox`:

```
guilherme@ubuntu:~/workspace$ ps -ef | grep firefox
1000      16792      1  1 22:51 ?        00:00:02 /usr/lib/firefox/firefox
1000      16903  15533  0 22:54 pts/2    00:00:00 grep --color=auto firefox
```

O comando `grep` também serve para pesquisarmos linhas com palavras específicas dentro de um arquivo de texto, no caso do arquivo `google.txt` que criamos no [curso anterior](#) (<https://cursos.alura.com.br/course/linux-ubuntu>), podemos pesquisar as linhas que tenham a palavra `California` da seguinte forma:

```
grep California google.txt
```

[COPIAR CÓDIGO](#)

O *grep* será um comando muito poderoso, pois o utilizaremos sempre que precisarmos encontrar os processos que queremos encerrar ou obter informações.



02

Importante: Ambiente e Arquivos



28%

ATIVIDADES
2 DE 7

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



ABRIR
CADERNO

No [curso Linux I: Conhecendo e utilizando o terminal](https://www.alura.com.br/course/linux-ubuntu) (<https://www.alura.com.br/course/linux-ubuntu>). foi criado um diretório chamado `workspace`, no qual existem alguns arquivos e diretórios. Também foi criado o arquivo `work.zip`, que é um arquivo compactado desse diretório.

Em alguns momentos deste curso iremos precisar do diretório `workspace` e seu conteúdo, ou do arquivo `work.zip`. Caso você não os possua, basta realizar o download no link a seguir:

<https://s3.amazonaws.com/caelum-online-public/linux2/work.zip>
[\(https://s3.amazonaws.com/caelum-online-public/linux2/work.zip\)](https://s3.amazonaws.com/caelum-online-public/linux2/work.zip)

Após realizar o download, descompacte o arquivo no diretório do seu usuário. Você pode utilizar o comando `unzip`.

Considerando que o seu diretório atual é o diretório do seu usuário (`/home/nome_do_seu_usuario`), você deve passar o caminho do diretório onde o arquivo `work.zip` ficou após o download. No meu

L

36.3k xp

a

caso, ele se encontra em Downloads :



\$ unzip Downloads/work.zip

COPIAR CÓDIGO



28%

ATIVIDADES
2 DE 7

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD

Você precisará de um Linux para fazer o curso. Se ainda não tem, veja [esse exercício no curso básico de Linux I](#) (<https://cursos.alura.com.br/course/linux-ubuntu/task/23412>).



36.3k xp

a



05

Filtrando a lista de processos



28%

ATIVIDADES
5 DE 7

Abra o navegador web Firefox. Agora faça o que é necessário para que no terminal você consiga listar apenas os processos que tiverem ligação com o programa firefox.

FÓRUM DO
CURSO

Opinião do instrutor

VOLTAR
PARA
DASHBOARD

Para obter a lista de processos relacionados com o firefox, redirecionamos a saída do comando `ps -ef` para o comando `grep`, que filtra as linhas após fazer uma pesquisa por um determinado termo que passamos. O comando final fica:



```
$ ps -ef | grep firefox
lucas    12549  1185 18 14:57
lucas    12616  9931  0 14:57 |
```

[COPIAR CÓDIGO](#)

36.3k xp

a



28%

ATIVIDADES
5 DE 7

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



36.3k xp

a



07

Encerrando processos



28%

Abra algum programa do seu Linux, como o navegador web firefox ou o editor gedit.

ATIVIDADES
7 DE 7

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



Execute o comando necessário, passando as informações que ele necessita para encerrar o processo do programa de uma forma que ele consiga decidir se será realmente encerrado.

Abra novamente o programa e agora utilize o comando necessário para encerrá-lo de forma que você tenha certeza que o programa não irá se recuperar e será de fato encerrado.



Opinião do instrutor

Utilizamos o comando `kill` para enviar um sinal para um processo. Os processos utilizam sinais para se comunicar entre si. Sinais também são utilizados pelo Linux para interferir no funcionamento dos processos.



36.3k xp



Exemplos de sinais são o `STOP` e o `CONT`,



que podem ser utilizados, respectivamente, para interromper a execução de um processo e retornar à execução do processo que foi interrompido anteriormente.



28%

ATIVIDADES
7 DE 7

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



Para utilizar o comando `kill`, passamos o sinal que desejamos enviar ao processo seguido do identificador único do processo, o `pid`. Imagine que temos um processo com `pid = 11163` no nosso sistema:

```
$ kill -STOP 11163  
$ kill -CONT 11163
```

[COPIAR CÓDIGO](#)

Para encerrar um processo de forma que ele possa realizar algumas tarefas antes de ser encerrado, utilizamos o sinal `TERM`.

Quando não indicamos nenhum sinal para o comando `kill`, é o sinal `TERM` que é executado por padrão. Portanto, o comando que podemos utilizar para encerrar o processo de uma maneira mais educada é:



36.3k xp

a

```
$ kill -TERM 11269
```

[COPIAR CÓDIGO](#)



Em alguns momentos mais críticos precisamos encerrar um processo à força. Precisamos "matar" o processo. Nesse caso utilizamos o sinal KILL , que é também representado com o número -9 . Então para garantir que o programa será encerrado imediatamente, fazemos:

ATIVIDADES
7 DE 7

\$ kill -9 11364

[COPIAR CÓDIGO](#)

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



36.3k xp

a



Transcrição

Os processos consomem uma determinada quantidade de espaço na CPU, mas como podemos descobrir esse consumo? Para isso o Linux disponibiliza um comando chamado `top`, que mostra a situação dos processos, do processador e da memória, além de outras informações em uma única tela.

```
top - 22:54:58 up 3:36, 2 users, load average: 0.02, 0.06, 0.07
Tasks: 187 total, 1 running, 186 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 0.6 sy, 0.0 ni, 99.1 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 2064676 total, 1253024 used, 811652 free, 78532 buffers
KiB Swap: 2095100 total, 0 used, 2095100 free, 769340 cached

PID USER      PR NI VIRT   RES   SHR S %CPU %MEM     TIME+ COMMAND
1188 root      20  0 189m  67m  20m S  1.3  3.4  5:47.86 Xorg
1904 guilherm  20  0 236m  59m  23m S  1.0  2.9  4:12.32 compiz
1928 guilherm  20  0 14692 1584 1232 S  0.7  0.1  1:24.36 prl_wmouse_d
13439 mysql    20  0 312m  34m 6368 S  0.3  1.7  0:07.08 mysqld
 1 root       20  0 3856 2196 1348 S  0.0  0.1  0:01.60 init
 2 root       20  0     0     0 S  0.0  0.0  0:00.01 kthreadd
 3 root       20  0     0     0 S  0.0  0.0  0:00.88 ksoftirqd/0
 5 root      0 -20     0     0 S  0.0  0.0  0:00.00 kworker/0:0H
 7 root      0 -20     0     0 S  0.0  0.0  0:00.00 kworker/u:0H
 8 root      rt  0     0     0 S  0.0  0.0  0:00.33 migration/0
 9 root      20  0     0     0 S  0.0  0.0  0:00.00 rcu_bh
10 root      20  0     0     0 S  0.0  0.0  0:02.46 rcu_sched
11 root      rt  0     0     0 S  0.0  0.0  0:00.16 watchdog/0
```

Esta lista de processos está organizada pelo uso da CPU, e se atualiza dependendo desse uso. Experimente abrir uma janela do navegador Firefox, por exemplo, e navegar para algum site. Deverá notar uma elevação no consumo de memória e CPU pelo Firefox e depois aos poucos uma queda neste mesmo consumo.

Para vermos um novo comando muito útil, experimente deixar abertas, duas abas executando o comando `top`. Após isso abra mais uma aba e verifique o status dos processos desses comandos com o

comando `ps` , lembra dele?

```
ps -ef | grep top
```

[COPIAR CÓDIGO](#)

Veremos a listagem filtrada exibindo apenas os processos que estão executando o comando `top` . Nenhuma novidade até aqui:

```
1000      16905 15533  0 22:54 pts/2    00:00:00 top
1000      16977 16929  0 22:55 pts/3    00:00:00 top
```

Visto a lista de processos do `top` , como podemos fazer para encerrar todos os processos que estão executando esse comando? Poderíamos usar o comando `kill` fornecendo todos os IDs da seguinte maneira:

```
kill -9 16905 16977
```

[COPIAR CÓDIGO](#)

Ou usando um outro comando que chamado `killall` . Ele permite matar todos os processos de um mesmo programa, ou seja, com um mesmo nome, sem precisar digitar seus ID's, como era feito antes com o `kill` . Exemplo:

```
killall top
```

[COPIAR CÓDIGO](#)

Podemos usar o modificador `-9` igual fazemos com o comando `kill` :

```
killall -9 top
```

[COPIAR CÓDIGO](#)

Assim, serão finalizados todos os processos com o nome `top` . Verifique as abas estavam executando o comando `top` , elas devem ter parado de exibir e atualizar a listagem dos processos em execução.



Para saber mais: outras opções do top



28%

O top possui algumas opções que podemos utilizar para alterar a forma padrão de como as informações são mostradas.

ATIVIDADES
4 DE 4

Para mostrar apenas os processos de um determinado usuário, podemos utilizar a opção `-u`:

FÓRUM DO CURSO

VOLTAR PARA DASHBOARD

`top -u lucas`

[COPIAR CÓDIGO](#)



Para acompanhar as informações de um processo específico, podemos utilizar a opção `-p` passando como argumento o PID do processo:



```
$ ps -e | grep firefox  
19509 ? 00:00:03 firefox  
$ top -p 19509
```

[COPIAR CÓDIGO](#)



36.3k xp



Por padrão, o `top` atualiza a tela com novas informações sobre os processos a cada 3 segundos. Para alterar esse tempo basta pressionar `d` enquanto estiver rodando o `top`, inserir o valor desejado e pressionar a tecla `Enter`:



Change delay from 3,0 to 1

[COPIAR CÓDIGO](#)



28%

O top possui muitas opções. Lembre-se que você pode obter informações sobre um comando consultando a sua documentação:

ATIVIDADES
4 DE 4

`man top`

[COPIAR CÓDIGO](#)

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



36.3k xp

a

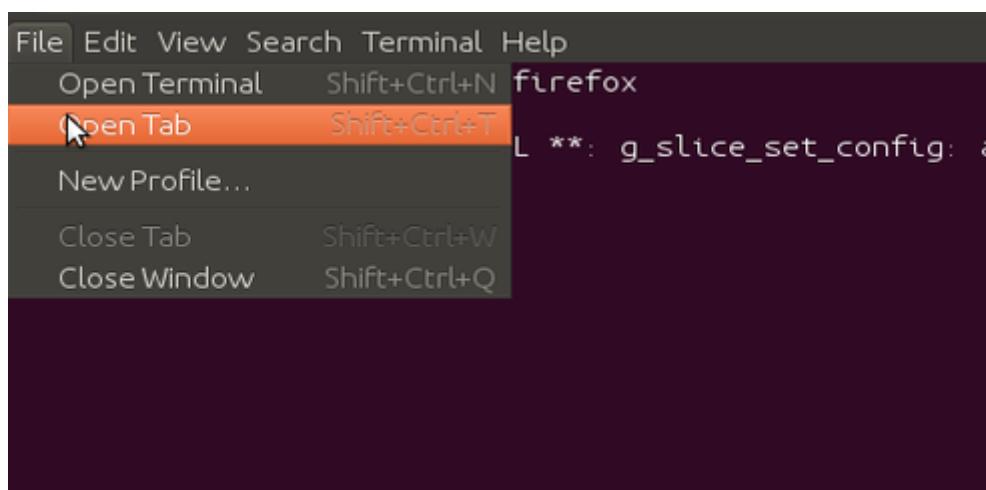


Transcrição

Agora que já sabemos visualizar quais processos estão rodando, vamos nos aprofundar um pouco mais em sua análise. Vamos inicialmente abrir o Firefox pelo terminal. Para isso digitamos no terminal o comando `firefox`.

Quando mandamos executar o Firefox por meio do Terminal, o Terminal fica travado enquanto o Firefox estiver aberto, nós não conseguimos mais digitar quaisquer comandos. Isso acontece porque o Terminal está executando o processo do Firefox dentro dele.

Para podermos continuar trabalhando com os comandos do Terminal, basta abrirmos uma nova aba através do menu superior do Terminal:



Agora podemos voltar a utilizar normalmente o terminal enquanto o outro se ocupa com o Firefox.

Em aulas passadas vimos o comando `ps -ef | grep`, que nos mostra os processos abertos. Um outro comando útil para visualizarmos esses processos de uma maneira gráfica é o `pstree`. Com ele podemos ver quais processos estão em execução e dentro de quais programas esses processos estão executando. Veja por exemplo, que o Firefox está rodando dentro do `bash` (aba) no Terminal:

```
gnome-terminal───bash───firefox───37*[{firefox}]
                  └───bash───pstree
                      └───gnome-pty-help
                          3*[{gnome-terminal}]
```

O processo do Firefox está em execução dentro do Terminal e, com isso, trava o `bash` no qual foi executado. Queremos que ele continue executando, mas no *background* desse `bash`, ou seja, sem travar o Terminal. Apertando `CTRL + Z` nós paramos temporariamente o processo.

```
(process:17483): GLib-CRITICAL **: g_slice_set_config: assertion `sys_page_size
== 0' failed

^Z
[1]+  Stopped                  firefox
```

E para visualizarmos os processos que estão parados, utilizamos o comando `jobs`:

```
[~] ~~~~~
guilherme@ubuntu:~$ jobs
[1]+  Stopped                  firefox
```

Para jogar o Firefox no *background*, ou seja, para ser executado em segundo plano, usamos o comando `bg` + seu número de identificação, neste caso executamos `bg 1` ou só `bg` se houver apenas um programa na lista de processos pausados. Este comando

apresentará uma saída semelhante a listada abaixo:

```
[1]+ firefox &
```

[COPIAR CÓDIGO](#)

E assim nosso *bash* fica destravado, mesmo executando o Firefox. A presença do `&` da saída anterior, significa que o programa está rodando no *background*. E se executarmos novamente o comando `jobs`, veremos algo como:

```
[1]+ Running
```

```
firefox &
```

[COPIAR CÓDIGO](#)

Indicando que o Firefox está realmente executando em *background*. Mas e se quiséssemos fazer o contrário, ou seja, trazer a execução do processo para o *foreground*? Para trazermos o programa para o *foreground* fazemos `fg 1` e assim, teremos nosso *bash* travado novamente.

Também podemos encerrar a execução do programa de vez, para isso basta que apertemos as teclas: `CTRL + C`.

Mas perceba que é um pouco trabalhoso jogarmos um programa para *background*: temos que abri-lo, fazê-lo parar e dar o comando necessário. Podemos fazer isso direto, ao abrirmos o programa da seguinte forma:

```
firefox &
```

[COPIAR CÓDIGO](#)

Dessa forma já abrimos o programa em *background* e temos o

terminal livre para continuarmos com demais comandos necessários para nossas atividades.

Nesta aula vimos os seguintes comandos:

- `jobs` : mostra os processos que estão sendo executados dentro do *bash*;
- `fg` e `bg` : jogam os processos para *foreground* e *background*, respectivamente;
- `[programa] &` : abre o [programa] diretamente em *background*;
- `pstree` : mostra todos os processos em um gráfico de árvore.



05

Praticando



28%

ATIVIDADES
5 DE 5

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



ABRIR
CADERNO

Para treinar um pouco os comandos que vimos neste capítulo, abra o navegador *Firefox* de modo que o programa fique em *foreground*. Feito isso, faça com que o programa continue executando no *background*.

Agora abra o editor *gedit* e faça com que ele seja executado no *background* desde o começo.

Por último, primeiro faça com que o *Firefox* volte a ser executado em *foreground* e encerre o programa. Faça o mesmo com o *gedit*.



36.3k xp



Opinião do instrutor

Para iniciar o *Firefox*, basta digitar o comando `$ firefox` no terminal. Nesse momento o terminal não estará livre para o uso, e para fazer com que o `firefox` seja executado em `background` , utilizamos `Ctrl + z` e `bg :`



```
$ firefox  
^Z  
[1]+ Stopped
```



28%

```
$ bg  
[1]+ firefox &
```

[COPIAR CÓDIGO](#)

ATIVIDADES
5 DE 5

Agora o nosso bash está livre para o uso.

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



Para fazer com que o *gedit* já inicie executando em *background*, utilizamos o & após o comando:

```
$ gedit &  
[2] 6421
```

[COPIAR CÓDIGO](#)

Para trazer o firefox de volta para o *foreground*, podemos passar o número do processo que o comando *jobs* retorna.

Pois se utilizarmos apenas fg o processo que será colocado em *foreground* será o último da lista.

Para encerrar o processo, utilizamos Ctrl + c .



36.3k xp



```
$ jobs  
[1] - Running  
[2]+ Running  
$ fg 1
```

firefox

^C

[COPIAR CÓDIGO](#)



28%

ATIVIDADES
5 DE 5

Por fim, para encerrar o `gedit`, que agora é o único processo que restou dos que foram iniciados no nosso `bash`, podemos simplesmente fazer `fg` e `Ctrl + c`.

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



36.3k xp

a



Transcrição

Até agora nós usamos os programas através do Terminal, programas como o *Firefox*, o *VI* e o *gedit*. Digitamos seus nomes e eles abrem. Vamos aprender nessa aula como ***criar*** nosso próprio programa, O **bash** suporta uma linguagem que nos permite criar nossos próprios programas, nossos próprios ***scripts***. Vamos primeiramente criar um arquivo de *gedit*, já em *background*, com o nome `dorme` :

```
gedit dorme &
```

[COPIAR CÓDIGO](#)

Dentro do editor, escrevemos o pequeno *script* que exibirá uma mensagem de `vou dormir` e depois de 5 segundos, exibe a mensagem `terminei de dormir`, assim teremos:

```
echo "Vou dormir"  
sleep 5  
echo "Terminei de dormir"
```

[COPIAR CÓDIGO](#)

Podemos executar nosso programa digitando o comando `dorme` , ou seja, o nome do programa, porém, caso façamos isso, teremos uma mensagem de erro, algo como: `dorme: command not found` . Isso porque o bash não conseguiu encontrar nosso programa, podemos realizar a execução direta do programa pedindo para o *bash* executá-lo da seguinte forma com comando `sh` :

sh dorme

[COPIAR CÓDIGO](#)

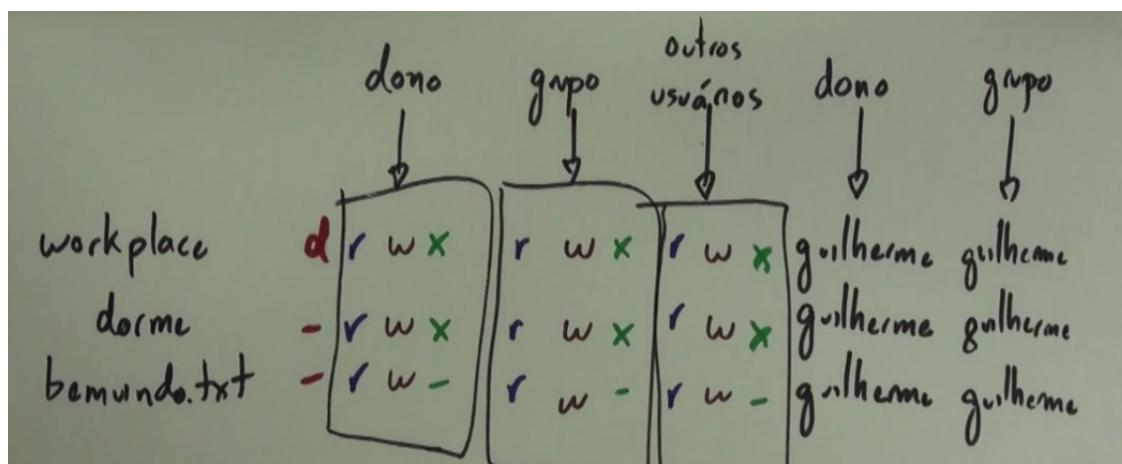
Da forma que escrevemos o programa, no Terminal irá aparecer a primeira mensagem `Vou dormir` e, depois de 5 segundos, a segunda mensagem `Terminei de dormir` e o programa encerra-se.

```
guilherme@ubuntu:~$ sh dorme
Vou dormir
Terminei de dormir
guilherme@ubuntu:~$
```

Perceba que para programas como o *Firefox* ou o *gedit* não é necessário digitar o comando `sh` antes, mesmo sendo eles scripts ou programas binários, veremos como fazer isso, porém, precisamos falar um pouco sobre *permissões*.

Permissões

Os arquivos no Linux podem ter permissões para **leitura (r)**, **escrita (w)** e **execução (x)**. Essas permissões são distribuídas para o dono do arquivo, ao grupo de usuários e também para outros usuários. Os **diretórios (d)** sofrem das mesmas regras. A imagem abaixo ilustra bem as permissões e suas distribuições.



Com o comando `ls -l`, podemos verificar a listagem de arquivos com todas as informações de permissões discutidas. Para o arquivo `dorme` por exemplo, temos as seguintes:

```
-rw-rw-r-- 1 guilherme guilherme 52 Jun 12 10:48 dorme
```

O primeiro dígito `-` informa que este item não é um diretório, os três seguintes indicam que o usuário dono do arquivo tem as permissões de leitura e escrita, o mesmo se repete para o grupo e por último, vemos que publicamente, ou seja, para outros usuários só há a permissão de leitura. Depois das informações de permissão temos o nome do usuário dono do arquivo e o grupo ao qual o usuário pertence.

Se quisermos que o arquivo `dorme` possa ser executado, ele precisará dessa permissão. Então vamos adicionar a permissão de execução para o *script* `dorme` para todos os usuários com a ajuda do comando `chmod`:

```
chmod +x dorme
```

[COPIAR CÓDIGO](#)

Dessa maneira adicionamos a capacidade de execução ao arquivo `dorme` (para retirá-la usamos `-x`), utilizando o comando `ls -l` novamente podemos verificar se a mudança realmente foi aplicada:

```
-rwxrwxr-x 1 guilherme guilherme 52 Jun 12 10:48 dorme
```

Perceba que a permissão de execução foi aplicada tanto para dono e grupo, quanto para outros usuários. Ainda não podemos simplesmente digitar o nome do *script* por questões de localização, então, para executá-lo teremos que usar `./`, para indicar que o

script está salvo no diretório atual:

```
./dorme
```

[COPIAR CÓDIGO](#)

Assim como temos o `+x` para adicionar permissão de execução, temos também o `+r` e o `+w`. Podemos também combinar essas permissões da seguinte forma: `chmod +rwx`.



02

Criando um script



32%

ATIVIDADES
2 DE 5FÓRUM DO
CURSOVOLTAR
PARA
DASHBOARDMODO
NOTURNOABRIR
CADERNO

Neste capítulo vimos como criar o nosso próprio *script* e executá-lo. Agora vamos criar um *script* que realizará o *backup* do diretório `workspace` em um arquivo `.zip`. O arquivo será criado na pasta atual em que o *script* for executado.

Neste caso, o nosso *backup* é local, estamos realizando uma cópia para o mesmo hd. Para ter uma maior segurança seria interessante realizar um *backup* remoto.

Crie um diretório chamado `scripts`, dentro do diretório do seu usuário. O *script* `realizabackup` deverá ser criado dentro desse diretório.

Abra o editor de textos `gedit` de forma que ele seja executado no *background*. Crie um arquivo chamado `realizabackup`. O arquivo deverá ter o seguinte conteúdo:

```
zip backup.zip -qr ~/workspace,  
echo "Backup realizado com sucesso!"
```

[COPIAR CÓDIGO](#)

L

40.8k xp

a



32%

ATIVIDADES
2 DE 5FÓRUM DO
CURSOVOLTAR
PARA
DASHBOARD

Vamos utilizar o comando `zip` para criar um arquivo compactado. A opção `-q`, indica que queremos fazer isso de maneira silenciosa, sem imprimir muitas informações no terminal. A opção `-r` indica que desejamos fazer a operação de forma recursiva, compactando o diretório `workspace` e subdiretórios também.

O `~` em `~/workspace/` é um atalho para representar o diretório do usuário. Sempre que quisermos nos referir ao diretório do usuário (`/home/nome_do_usuario/`), podemos utilizar o `~`. Dessa forma independente de qual seja o nosso diretório atual, o comando `zip` saberá onde o diretório `workspace` se encontra.

Por fim, vamos imprimir uma mensagem informando que o *backup* foi concluído com sucesso.

Teste o *script* e veja se o arquivo `.zip` é gerado da forma correta.



40.8k xp





04

Listando as permissões



35%

Execute um `ls -l` no nosso script para listar as permissões:

ATIVIDADES
4 DE 5

```
$ ls -l realizabackup  
-rw-rw-r-- 1 lucas lucas 52 Jai
```

[COPIAR CÓDIGO](#)

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD

Repare nos caracteres que a saída mostra no que se refere as permissões.



Reflita sobre seus significados e veja a *Opinião do Instrutor* para a nossa explicação.



Opinião do instrutor



41.0k xp

As informações sobre as permissões são listadas nos primeiros caracteres da saída do `ls -l`. O primeiro caractere, no nosso caso um `-`, indica que se trata de um arquivo. Caso fosse um diretório seria exibida a letra `d`.

a

No restante dos caracteres, quando



35%

ATIVIDADES
4 DE 5

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



41.0k xp

a

encontramos um - significa que não temos uma determinada permissão. O primeiro conjunto de três caracteres representa as permissões do dono do arquivo, o segundo, as permissões do grupo e o terceiro, as permissões dos outros usuários.

No nosso script, o dono e o grupo têm permissão para leitura(r) e escrita(w), mas não para execução. Já os outros usuários têm apenas permissão de leitura.



05

Modificando as permissões



37%

ATIVIDADES
5 DE 5

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD

MODO
NOTURNO

ABRIR
CADERNO

Utilize o comando necessário para que seja adicionada a permissão de execução ao script `realizabackup`, fazendo com que ele possa ser executado sem utilizar o comando `sh`.

Aproveite para testar a execução do script.

Após executar o script, realize um teste: remova a permissão de escrita do arquivo. Tente abrir o script no editor de texto e perceba que se fizer alguma modificação no arquivo, não conseguirá salvar, pois no momento não possui permissão para escrita.



Opinião do instrutor



41.1k xp

Para alterar a permissão de um arquivo, utilizamos o comando `chmod`. No nosso caso, desejamos adicionar a permissão de execução ao arquivo:

```
$ chmod +x realizabackup  
$ ls -l realizabackup
```

-rwxrwxr-x 1 lucas lucas 52 Jan



[COPIAR CÓDIGO](#)



37%

Agora, possuímos permissão para execução, e podemos fazer:

ATIVIDADES
5 DE 5

./realizabackup

[COPIAR CÓDIGO](#)

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



```
$ chmod -w realizabackup  
$ ls -l realizabackup  
-r-xr-xr-x 1 lucas lucas 52 Jan
```

[COPIAR CÓDIGO](#)



Com a permissão de escrita removida, conseguimos apenas visualizar o arquivo. Ao abrir o script em um editor de texto, se fizermos modificações, não seremos capazes de salvá-las.



41.1k xp

a



Transcrição

Utilizamos diversos programas até este agora e até criamos o nosso próprio *script*, mas onde será que estão, por exemplo, o *Firefox*, o *gedit* ou o *VI*? O modo mais simples de procurar por arquivos é utilizando o comando `locate` :

`locate firefox`

[COPIAR CÓDIGO](#)

Podemos perceber que o Terminal retornará muito rapidamente uma lista extensa de arquivos que possuem esse nome.

```
/usr/share/lightdm-remote-session-uccsconfigure/firefox-uccsconfigure.desktop  
/usr/share/lightdm-remote-session-uccsconfigure/firefox-uccsconfigure.sh  
/usr/share/lintian/overrides/firefox  
/usr/share/locale-langpack/en_AU/LC_MESSAGES/unity_firefox_extension.mo  
/usr/share/locale-langpack/en_CA/LC_MESSAGES/unity_firefox_extension.mo  
/usr/share/locale-langpack/en_GB/LC_MESSAGES/unity_firefox_extension.mo  
/usr/share/man/man1/firefox.1.gz  
/usr/share/pixmaps/firefox.png
```

O Linux verifica a todo momento quando criamos, renomeamos, etc, um arquivo, e atualiza seu banco de dados interno com a localização de todos os arquivos no sistema operacional, por isso a busca por nome de arquivo é tão rápida.

Tal atualização ocorre de tempos em tempos, mas se quisermos forçar essa atualização podemos executar o comando `updatedb`. Esse *update* é executado globalmente no Sistema Operacional e para que possamos realizar essa atualização, precisamos executar esse

comando como *super usuário*.

Para executar um programa como *super usuário*, ou seja, como usuário principal da máquina usamos o comando `sudo` antes do comando a ser executado, por exemplo:

```
sudo updatedb
```

[COPIAR CÓDIGO](#)

Quando executarmos o comando dessa forma, o Terminal pedirá uma senha, que é a do seu usuário atual e logo depois disso executará a ação como *root* (o *super usuário* da nossa máquina) que neste caso será atualizar o banco de dados interno com a localização dos arquivos no sistema operacional. Apesar disso ser possível, lembre-se, a atualização acontece periodicamente de forma automática.



03

Localizando arquivos



42%

Vamos agora buscar um arquivo no nosso sistema. Primeiramente, atualize a base de dados que o `locate` utiliza.

ATIVIDADES
3 DE 3

Faça uma busca pelo script `realizabackup`, criado anteriormente.

FÓRUM DO CURSO

VOLTAR PARA DASHBOARD



Opinião do instrutor



Para atualizar a base de dados utilizada pelo `locate`, fazemos:

```
$ sudo updatedb  
[sudo] password for lucas:
```

[COPIAR CÓDIGO](#)



41.3k xp



Para encontrar o arquivo `realizabackup` utilizamos o `locate` e passamos o nome do arquivo:



```
$ locate realizabackup  
/home/lucas/scripts/realizabac
```

[COPIAR CÓDIGO](#)



42%

ATIVIDADES
3 DE 3

Para indicar que queremos localizar qualquer arquivo com extensão .txt , utilizamos o caractere * :

```
$ locate *.txt
```

[COPIAR CÓDIGO](#)

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



MODO
NOTURNO



ABRIR
CADERNO



41.3k xp

a



Transcrição

Aprendemos anteriormente como buscar por arquivos e programas no sistema operacional com o comando `locate`, Porém o Terminal retorna uma lista extensa de arquivos que possuem em seu nome, parte do texto que usamos para pesquisar. Mas, qual é o arquivo que será executado quando digitamos o comando `vi`, `gedit` ou mesmo `firefox`?

Para saber onde os programas estão instalados usamos o comando `which`, seguido do nome do programa que desejamos localizar:

```
guilherme@ubuntu:~$ which firefox  
/usr/bin/firefox  
guilherme@ubuntu:~$ which vi  
/usr/bin/vi  
guilherme@ubuntu:~$ which gedit  
/usr/bin/gedit  
guilherme@ubuntu:~$ █
```

Os programas que executamos antes estão todos dentro de um mesmo diretório chamado `/usr/bin`. Podemos listar todos os programas dentro desta pasta usando o comando `ls` da seguinte forma: `ls /usr/bin`, nos será listado uma lista enorme de programas do Linux.

Já que os programas ficam instalados nesse diretório, vamos colocar aquele arquivo/programa que criamos em aulas passadas, o `dorme`, dentro desse diretório. Isso com a ajuda do comando `mv`:

```
mv dorme /usr/bin
```

[COPIAR CÓDIGO](#)

Teremos um erro:

```
guilherme@ubuntu:~$ mv dorme /usr/bin  
mv: cannot move 'dorme' to '/usr/bin/dorme': Permission denied
```

O erro informa que não há permissão para mover o arquivo `dorme` para dentro do diretório `/usr/bin`. De fato, se verificarmos as permissões para o diretório `usr`, verificaremos que, somente o usuário `root` tem permissão de escrita para o mesmo.

```
guilherme@ubuntu:~$ ls -l /usr  
total 100  
drwxr-xr-x    2 root root 40960 Jun 11 18:42 bin
```

Perceba que para o diretório "bin", quem é dono tem permissão de leitura, escrita e execução, quem estiver no mesmo grupo tem de leitura e execução, e também leitura e execução para outros usuários. Então o problema é: como não somos "`root`" (o principal usuário da máquina), não podemos escrever dentro do diretório.

É preciso um cuidado especial ao realizar tarefas como `root`. Este usuário por ser o principal no sistema operacional, tem permissões para todas as operações, inclusive destrutivas, que farão o sistema parar de funcionar completamente.

Por ser um diretório muito importante, faz sentido não termos permissão de escrita para ele. Mas, mesmo assim, vamos fazer com que nosso programa `dorme` seja movido para dentro dele, uma vez que queremos que o programa seja executado por todos os usuários.

Para isso precisaremos mover o programa para o diretório como

usuário *root*, ou seja, como se nosso usuário padrão fosse o usuário *root*, para isso, antes do comando usamos um comando auxiliar, o *sudo* :

```
sudo mv dorme /usr/bin
```

[COPIAR CÓDIGO](#)

O Terminal pedirá a senha de usuário. Depois disso conseguimos executar esse comando como se fôssemos o usuário principal. Para termos certeza de que conseguimos mover o *dorme* para dentro do diretório *bin*, podemos executar:

```
ls /usr/bin | grep dorme
```

[COPIAR CÓDIGO](#)

Ou até mesmo usar o comando *which* :

```
which dorme
```

[COPIAR CÓDIGO](#)

No primeiro caso, se o programa *dorme* estiver dentro da *bin* ele será listado, e no segundo caso, o caminho */usr/bin/dorme* será exibido. Agora podemos executar o *dorme* sem adicionar o *sh* ou o *./*.

Gerenciando senhas de usuários

Vamos mudar a senha do nosso usuário *guilherme*, para isso usamos o comando *passwd*. O Terminal irá pedir a senha atual, se houver, e depois duas vezes a nova senha.

Esta senha, será para o usuário padrão, para o usuário `guilherme` , caso queiramos trocar ou colocar uma senha para o usuário `root` fazemos uso do comando `sudo` da seguinte forma: `sudo passwd` .

```
guilherme@ubuntu:~$ passwd
Changing password for guilherme.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
guilherme@ubuntu:~$ sudo passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
guilherme@ubuntu:~$ █
```

Podemos nos logar como outro usuário, para isso usamos o comando `su` seguido do nome do usuário com qual queremos logar:

`su root`

[COPIAR CÓDIGO](#)

Assim podemos nos logar como o usuário `root`, basta que digitemos a senha do usuário, podemos descobrir também com qual usuário estamos logado usando o comando `whoami` . E podemos sair da sessão de um usuário usando o comando `exit` .



03

Movendo o script realizabackup



48%

ATIVIDADES
3 DE 5

Se tentarmos mover o arquivo `realizabackup` para o diretório `/usr/bin` usando o comando `$ mv realizabackup /usr/bin`, não iremos conseguir. Por que isso ocorre e como podemos resolver?

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD

Aproveite para mover o arquivo e testar se o script `realizabackup` pode ser executado como um comando independente de qual seja o nosso diretório atual.



Opinião do instrutor

Não conseguimos mover o arquivo pois o dono do diretório é o único que possui permissão de escrita na pasta, e o dono nesse caso é o usuário `root`. Outros usuários, como o que estamos utilizando no momento, não possuem permissão de escrita, e por isso o comando para mover o arquivo não funciona.



41.5k xp

a

`$ ls -l /usr/`



```
total 108  
drwxr-xr-x  2 root root 45056
```

[COPIAR CÓDIGO](#)



48%

ATIVIDADES
3 DE 5

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



MODO
NOTURNO



ABRIR
CADERNO

O usuário `root` é o que costumamos chamar de superusuário - um usuário que é utilizado para realizar tarefas relacionadas à administração do sistema.

Esse usuário não possui qualquer restrição. Para obter o acesso de administrador no nosso sistema, utilizamos o comando `sudo`. A senha que será solicitada é a do nosso usuário, pois ele faz parte do mesmo grupo ao qual pertence o usuário `root`.

É dessa forma que vamos conseguir mover o arquivo `realizabackup` para a pasta `/usr/bin`. Após mover o arquivo, conseguimos executar o script a partir de qualquer diretório:

```
$ sudo mv realizabackup /usr/b:  
[sudo] password for lucas:  
$ ls /usr/bin/ | grep realizab:  
realizabackup  
$ realizabackup  
Backup realizado com sucesso
```

[COPIAR CÓDIGO](#)



41.5k xp

a



48%

ATIVIDADES
3 DE 5

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



41.5k xp

a



Transcrição

Já aprendemos como trocar de usuário usando o comando `su`. Mas como adicionamos um usuário novo na máquina? Basta usarmos o comando `adduser` seguido do nome do usuário, como por exemplo:

`adduser jose`

[COPIAR CÓDIGO](#)

```
root@ubuntu:/home# adduser jose
Adding user `jose' ...
Adding new group `jose' (1002) ...
Adding new user `jose' (1002) with group `jose' ...
Creating home directory `/home/jose' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for jose
Enter the new value, or press ENTER for the default
      Full Name []: Jose
      Room Number []:
      Work Phone []:
      Home Phone []:
      Other []:
Is the information correct? [Y/n] y
```

Depois de escolhermos a senha para o usuário "José", o Terminal pede algumas outras informações, como nome completo, telefone, etc.

Já sabemos que podemos trocar de usuário com o comando `su`, fornecendo a senha deste usuário, mas há um caso especial. Se estivermos como o usuário principal, o `root`, podemos trocar para outro sem precisarmos digitar senha. Essa é uma das razões de ser perigoso darmos acesso ao `root` para qualquer pessoa.

Outro ponto importante a ser destacado com o qual teremos que ter cuidado é que, independente do usuário no qual estamos conectados, sempre temos permissão de leitura para o diretório de outro usuário. Então vamos ver como bloquear essa leitura em nossos diretórios pessoais.

Se verificarmos as permissões dos diretórios pessoais de cada usuário usando o comando `ls -l` no diretório '/home', teremos:

```
jose@ubuntu:/home$ ls -l
total 12
drwxr-xr-x 19 guilherme guilherme 4096 Jun 12 11:03 guilherme
drwxr-xr-x  2 jose      jose      4096 Jun 12 12:28 jose
drwxr-xr-x  8 maria    maria    4096 Jun 10 22:17 maria
```

Veremos que para o diretório `jose` como usuário `jose` temos todas as permissões, execução e leitura para os usuários do mesmo grupo e execução e leitura para outros usuários. O que queremos é que os outros usuários não tenham permissão alguma. Para tirarmos tais permissões para os outros usuários, usamos o comando `chmod` da seguinte forma:

```
chmod o-rx jose
```

[COPIAR CÓDIGO](#)

Neste caso, o comando `chmod` é seguido de alguns operadores, que são: `o` para *others*, ou seja, outros usuários, `-` indica uma remoção de permissão, `r` e `x` indicam permissões de leitura e execução. Se quiséssemos tirar permissões do próprio usuário, seria: `u-rx` e para o grupo seria: `g-rx`.

Agora se verificarmos novamente, veremos que o diretório do usuário `jose` não permite mais nenhuma operação para outros usuários:

```
drwxr-xr-x 19 guilherme guilherme 4096 Jun 12 11:03 guilherme
drwxr-x---  2 jose      jose      4096 Jun 12 12:28 jose
```

```
drwxr-xr-x  8 maria      maria      4096 Jun 10 22:17 maria
```



04

Adicionando um novo usuário



58%

Adicione um novo usuário chamado `jose` e retire todas as permissões que os outros usuários tem na pasta do usuário criado.

ATIVIDADES
4 DE 4

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



Dica: Lembre-se que para alterar a permissão da pasta você deve estar logado com o usuário `jose`, que é o dono da pasta, ou com o usuário `root`, que não tem restrições. Outra opção é utilizar o comando com o `sudo` antes, indicando que desejamos executar apenas aquele comando como usuário `root`, essa é uma opção interessante, já que não é recomendado logar como usuário `root` e fazer tudo a partir dele.



Opinião do instrutor

Para adicionar um novo usuário, utilizamos o comando `adduser`:



42.0k xp



```
$ sudo adduser jose
Adding user `jose' ...
Adding new group `jose' (1002)
Adding new user `jose' (1002) \
```



58%

ATIVIDADES
4 DE 4FÓRUM DO
CURSOVOLTAR
PARA
DASHBOARD

42.0k xp

```
Creating home directory `/home/  
Copying files from `/etc/skel'  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully  
Changing the user information  
Enter the new value, or press I to keep  
Full Name []:  
Room Number []:  
Work Phone []:  
Home Phone []:  
Other []:  
Is the information correct? [Y,
```

[COPIAR CÓDIGO](#)

Após a criação do usuário, podemos remover as permissões para a pasta desse usuário utilizando o comando `chmod`.

Como queremos remover as permissões atuais, que são de leitura e execução, vamos utilizar `-rx`. Para indicar que desejamos remover as permissões apenas dos outros usuários que acessarem o diretório, vamos utilizar o `o`:

```
$ su jose  
Password:  
$ cd /home/  
$ chmod o-rx jose/
```

[COPIAR CÓDIGO](#)

Podemos verificar que tudo ocorreu como

esperado:



58%

ATIVIDADES
4 DE 4

[COPIAR CÓDIGO](#)

Agora a pasta do usuário `jose` não possui permissões liberadas para os outros usuários.

VOLTAR
PARA
DASHBOARD



42.0k xp

a



Transcrição

Dentro do diretório workspace vamos criar um novo programa chamado oi , entraremos no diretório usando o comando cd e abriremos o gedit em background:

```
cd workspace
```

[COPIAR CÓDIGO](#)

E depois,

```
gedit oi &
```

[COPIAR CÓDIGO](#)

Dentro do gedit digitaremos o código do programa, este simplesmente imprime uma mensagem de oi , tudo bem? :

```
echo "Oi, tudo bem?"
```

[COPIAR CÓDIGO](#)

Após salvar o programa e fechar o gedit , podemos visualizar as permissões desse programa, onde veremos que não temos permissão de execução:

```
-rwx-rw-r-- 1 guilherme guilherme 21 Jun 12 12:36 oi
```

Para executá-lo, precisamos adicionar a permissão de execução para ele, para isso usamos o comando `chmod` como já vimos anteriormente:

```
chmod +x oi
```

[COPIAR CÓDIGO](#)

Se executarmos o programa com o `./oi` o mesmo funciona:

```
guilherme@ubuntu:~/workspace$ ./oi
oi, tudo bem?
```

Porém, como anteriormente, queremos que ele funcione como os programas normais do sistema, sem que precisemos digitar algo antes do nome dele.

Uma opção é como fizemos com o programa `dorme`: movê-lo para o diretório `/usr/bin`. Mas dessa forma todos os usuários passam a ter acesso a esse programa, o que não queremos. O diretório `bin` existe para armazenar programas globais, aos quais todos os usuários têm acesso. Além disso teríamos que mover todo o programa para esse diretório, o que daria muito trabalho se ele possuisse muitos arquivos.

Precisamos saber onde o `bash` procura os programas e os *scripts* para executar. Esse lugar é o ***Path***, uma **Variável de Ambiente**. Se executarmos o comando `env` conseguimos visualizar todas as variáveis de ambiente. Para visualizarmos apenas o *PATH* podemos fazer `env | grep PATH`, e na listagem, a linha que nos interessa é semelhante a apresentada na imagem a seguir:

```
PATH=/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:
/sbin:/bin:/usr/games:/usr/local/games
```

Perceba que nessa listagem da variável `PATH` encontra-se o diretório `/usr/bin` e que os diretórios são separados por `:`. Estes diretórios são onde o *bash* procura os programas para serem executados.

Vamos adicionar nessa variável o diretório `workspace` . Se conseguirmos fazer isso, para executarmos nosso programa só será necessário digitar seu nome: `oi` . Fazemos isso da seguinte forma:

```
PATH=$PATH:/home/guilherme/workspace
```

[COPIAR CÓDIGO](#)

Isso significa que o novo *PATH* será igual a o *PATH* atual (`$PATH`) adicionado a ele (`:`) o diretório `/home/guilherme/workspace` . Podemos verificar se o diretório `workspace` foi adicionado, verificando novamente a variável `PATH` :

```
PATH=/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/home/guilherme/workspace
```

Agora conseguimos executar o *script* "oi" digitando apenas seu nome, em qualquer diretório que estivermos no *bash*.

```
guilherme@ubuntu:~$ oi  
oi, tudo bem?
```

Porém, se utilizarmos um outro *bash* no Terminal, a variável *PATH* volta a ter os mesmos diretórios por padrão, sem que o `workspace` esteja presente na listagem. O que fizemos no outro *bash* só fica gravado apenas naquela sessão e não globalmente.

Experimente abrir uma nova aba no terminal e verificar a variável *PATH* novamente. Verá que na listagem o diretório `workspace` deixou de existir.

Precisamos de um arquivo o qual diga que, toda vez que abrirmos um *bash*, o *PATH* seja configurado como queremos. Tal arquivo já existe e tem o nome de `.bashrc` . Vamos editá-lo com o `gedit` :

```
gedit .bashrc &
```

[COPIAR CÓDIGO](#)

Esse arquivo é carregado toda vez que abrimos uma aba nova no Terminal. Ele já possui vários comandos. O que vamos fazer é: Ao final dele vamos adicionar a seguinte linha e salvar:

```
PATH=$PATH:/home/guilherme/workspace
```

[COPIAR CÓDIGO](#)

Dessa forma, para toda sessão e todo *bash*, o *PATH* será configurado para possuir o diretório que queríamos, permanentemente.

Experimente verificar novamente, fechar o terminal, abri-lo, tentar executar o programa `oi` , abrir novas abas e verificar a variável `PATH` .



02

Variáveis de ambiente



62%

Execute o comando `env` no seu sistema.

ATIVIDADES
2 DE 5

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



MODO
NOTURNO



ABRIR
CADERNO

Opinião do instrutor



As variáveis de ambiente são variáveis globais que podemos utilizar para definir configurações, personalizar nosso terminal.

Quando executamos o comando `env`, podemos ver quais são as variáveis que estão definidas. A variável `HOME`, por exemplo, guarda o caminho para o diretório do usuário.



42.1k xp



A variável `PATH`, guarda informações de



onde estão nossos arquivos executáveis para que possamos executar um comando sem a necessidade de digitar o caminho absoluto.



62%

ATIVIDADES
2 DE 5

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



42.1k xp

a



03

Adicionando um diretório nas variáveis de ambiente



64%

ATIVIDADES
3 DE 5

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



```
$ wc -w *.txt
6 projetos_java.txt
10 projetos_php.txt
16 total
```

[COPIAR CÓDIGO](#)



O `wc` nos informou que o arquivo `projetos_java.txt` possui 6 palavras, o arquivo `projetos_php.txt` possui 10 palavras, e por fim, o total de palavras nos dois arquivos é 16.



42.2k xp

a

Dentro do diretório `scripts`, crie um `script` chamado `contapalavras` que conta a quantidade de palavras nos arquivos `.txt` presentes no diretório atual.



64%

ATIVIDADES
3 DE 5FÓRUM DO
CURSOVOLTAR
PARA
DASHBOARDMODO
NOTURNOABRIR
CADERNO

42.2k xp



O *script* deverá imprimir apenas a última linha que o comando `wc` retorna, representando o total de palavras dos arquivos `.txt` que existem no nosso diretório atual. Esse filtro pode ser feito através do comando `grep`.

O arquivo `contapalavras` irá possuir o seguinte conteúdo:

```
WC -w *.txt | grep total
```

[COPIAR CÓDIGO](#)

Lembre-se de adicionar permissão de execução para o script.

Adicione o diretório `scripts` na variável de ambiente `PATH`, pois é nessa variável que ficam os caminhos dos programas a serem executados. Modifique o que for necessário para que quando um novo terminal for aberto essas configurações não se percam.

Adicione alguns arquivos `.txt` no diretório `scripts`, abra um novo terminal e tente executar o *script* `contapalavras`, para verificar se tudo funcionou.

Observação: Para que o `wc` mostre a linha com o total é necessário que exista mais de



um arquivo .txt no diretório em que executamos o contapalavras .



64%



Opinião do instrutor

ATIVIDADES
3 DE 5

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



Vamos assumir que scripts é nosso diretório atual. Primeiro criaremos o script contapalavras e adicionaremos permissão de execução:

```
$ gedit contapalavras &  
[1] 6295  
$ chmod +x contapalavras
```

[COPIAR CÓDIGO](#)

Para adicionar o diretório scripts nas variáveis de ambiente do usuário, podemos editar o arquivo oculto .bashrc dentro do diretório do usuário.

```
$ cd  
$ gedit .bashrc &
```

[COPIAR CÓDIGO](#)



42.2k xp



Vamos adicionar o conteúdo após a última linha, no fim do arquivo:



PATH=\$PATH:/home/lucas/scripts

[COPIAR CÓDIGO](#)



64%

ATIVIDADES
3 DE 5

Indicamos que a variável de ambiente PATH recebe o conteúdo já presente nela, mais o diretório scripts . Ao abrir um novo terminal, já conseguimos acessar o script contapalavras :

FÓRUM DO
CURSO

\$ contapalavras

16 total

[COPIAR CÓDIGO](#)

VOLTAR
PARA
DASHBOARD



MODO
NOTURNO



ABRIR
CADERNO



42.2k xp

a



Obtendo o número de processos



66%

ATIVIDADES
4 DE 5FÓRUM DO
CURSOVOLTAR
PARA
DASHBOARDMODO
NOTURNOABRIR
CADERNO

Além de contar o número de palavras em um arquivo, o comando `wc` pode também contar o número de caracteres e linhas em um arquivo, ou em uma saída do terminal.

Para isso, podemos utilizar a opção `-c` para caracteres e `-l` para linhas.

Se utilizarmos o comando `ps -e`, que lista todos os processos, podemos passar o retorno do `ps` para o `wc -l` contar quantas linhas o retorno possui. A quantidade de linhas representa a quantidade de processos existentes no nosso sistema.

Dentro do diretório `scripts`, crie um `script` chamado `totaldeprocessos`, com o seguinte conteúdo:

```
ps -e | wc -l
```

[COPIAR CÓDIGO](#)

Lembre-se de adicionar permissão de execução no `script`.



42.3k xp



Tente executar o `script` alterando o seu diretório para um diferente de `scripts`, para testar se tudo funcionou bem.



Opinião do instrutor



66%

ATIVIDADES
4 DE 5

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



```
$ gedit totaldeprocessos &
[1] 6523
$ chmod +x totaldeprocessos
```

[COPIAR CÓDIGO](#)

Agora podemos testar. O número
retornado é o número de processos que o
nossa sistema possui no momento:



```
$ cd
$ totaldeprocessos
240
```

[COPIAR CÓDIGO](#)

Perceba que agora sempre que colocarmos
um script no diretório `scripts` , ele
poderá ser executado independentemente
do diretório em que estejamos, pois
`scripts` agora é um dos diretórios nos
quais o comando será procurado pelo
nossa `bash` .



42.3k xp





66%

ATIVIDADES
4 DE 5

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



42.3k xp

a



05

Alterando a configuração do prompt



67%

ATIVIDADES
5 DE 5

Já vimos que com o devido cuidado, podemos alterar nossas variáveis de ambiente para customizar a forma de interação com o nosso terminal.

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



MODO
NOTURNO



ABRIR
CADERNO

Uma outra variável de ambiente é a `PS1`.

Vamos olhar o conteúdo dela:

```
$ echo $PS1  
\[\e[0;1;32m\]\u@\h:\ \w\$
```

[COPIAR CÓDIGO](#)

Quando estamos utilizando o terminal, algumas informações sempre são mostradas, exemplo: `lucas@lucas-Inspiron-5458:~$`. Essa formatação é definida na variável `PS1`.

Experimente alterar a variável e perceba que a definição do seu prompt será alterada:



42.4k xp

a

```
lucas@lucas-Inspiron-5458:~$ PS1="alura>"  
alura>PS1="alura:"  
alura:
```

[COPIAR CÓDIGO](#)



Opinião do instrutor



67%

Para tudo voltar ao normal, basta fechar o terminal atual e abrir um novo.

ATIVIDADES
5 DE 5

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



MODO
NOTURNO



ABRIR
CADERNO



42.4k xp

a



Transcrição

Nesta aula vamos aprender como instalar um programa via Terminal de Linux. Como exemplo, iremos instalar um servidor FTP.

O Ubuntu nos disponibiliza um sistema de gerenciamento de pacotes chamado **apt**. Para ver as versões atualizadas dos programas que estão disponíveis para instalação fazemos:

```
sudo apt-get update
```

[COPIAR CÓDIGO](#)

Veja que executamos o comando como *root* (`sudo`), isto porque esta é uma tarefa de administração, por isso é necessário que seja feita como *root*, caso não, teremos uma mensagem de permissão negada. Neste passo, o Terminal irá buscar na internet o que existe de novidade nos programas para instalação. Para buscar um programa de servidor FTP podemos fazer:

```
apt-cache search ftp
```

[COPIAR CÓDIGO](#)

Este comando busca na lista de pacotes disponíveis, qualquer programa que se encaixe nesse termo de busca, por isso retorna uma longa lista de programas. Sejamos mais restritos na busca e procuremos um servidor específico:

```
apt-cache search vsftpd
```

[COPIAR CÓDIGO](#)

Como estamos usando uma busca mais restrita, poucos resultados serão retornados pela busca, a imagem abaixo ilustra esses resultados.

```
guilherme@ubuntu:~$ apt-cache search vsftpd
vsftpd - lightweight, efficient FTP server written for security
ccze - A robust, modular log coloriser
ftpd - File Transfer Protocol (FTP) server
yasat - simple stupid audit tool
```

Achamos um bom servidor FTP, ele nos mostra o nome e uma curta descrição. Para instalar este programa fazemos:

```
sudo apt-get install vsftpd
```

[COPIAR CÓDIGO](#)

E após alguns segundos o programa está instalado! Será mesmo? Para testarmos, podemos tentar conectar ao servidor da própria máquina utilizando o comando:

```
ftp localhost
```

[COPIAR CÓDIGO](#)

Será pedido um usuário e senha para podermos acessar a máquina.

```
guilherme@ubuntu:~$ ftp localhost
Connected to localhost.
220 (vsFTPD 3.0.2)
Name (localhost:guilherme): guilherme
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

Depois disso podemos manipular os arquivos via FTP executando quaisquer comandos que já aprendemos até agora, como `ls` e outros.

Para encerrar a conexão, usamos o comando `exit`.

Experimente conectar com outros usuários disponíveis na máquina também.

Para que possamos remover programas, utilizamos o comando `apt-get remove` seguido do nome do programa. Como exemplo, vamos desinstalar o servidor que instalamos.

```
sudo apt-get remove vsftpd
```

[COPIAR CÓDIGO](#)

Será pedida uma confirmação e depois a desinstalação seguirá normalmente. Garanta que a desinstalação foi feita com sucesso tentando fazer login via FTP novamente. Caso o programa realmente tenha sido desinstalado, uma mensagem de erro semelhante a seguinte deve ser impressa no Terminal.

```
ftp: connect: Connection refused
```

[COPIAR CÓDIGO](#)

Aprendemos a instalar e desinstalar programas de forma fácil e prática com auxílio do gerenciador de pacotes `apt`. Os comandos são:

- `apt-get install` : instala um programa dado o nome

- `apt-get remove` : desinstala um programa dado o nome
- `apt-get update` : busca uma lista das versões atualizadas dos programas
- `apt-cache search` : procura os programas disponíveis para instalação



04

Instalando um servidor ftp



75%

Vamos instalar no nosso sistema um servidor de ftp.

ATIVIDADES
4 DE 4

Atualize a lista de pacotes disponíveis para garantir que iremos instalar as versões mais atualizadas dos programas.

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



Instale o pacote `vsftpd` no sistema.

Aproveite para realizar um teste com o programa conectando-se na sua máquina, verificando se o programa foi instalado corretamente.



Por fim, desinstale o pacote do seu sistema.



Opinião do instrutor



42.7k xp



Para atualizar a lista de pacotes disponíveis, utilizamos o comando `$ apt-get update .`

Para instalar um pacote, utilizamos o comando `$ apt-get install nome_do_pacote .` O `apt-get` é uma



operação que necessita ser executada com poderes de administrador, por isso utilizamos o comando `sudo` antes do comando `apt-get`:



75%

ATIVIDADES
4 DE 4

[COPIAR CÓDIGO](#)

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



Para realizar um teste, podemos nos conectar na nossa máquina:

```
$ ftp localhost
Connected to localhost.
220 (vsFTPd 3.0.2)
Name (localhost:lucaS):
```

[COPIAR CÓDIGO](#)



Podemos ver que o programa de `ftp` foi instalado corretamente. Caso deseje, você pode fornecer usuário e senha, caso não, basta utilizar as teclas `Ctrl + c`.

Para desinstalar o programa, fazemos:

```
$ sudo apt-get remove vsftpd
```

[COPIAR CÓDIGO](#)



42.7k xp

a



75%

ATIVIDADES
4 DE 4

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



42.7k xp

a



Transcrição

Podemos instalar programas que não estão disponibilizados na central de programas do Ubuntu, ou seja, sem o uso do `apt`. Para isso nós baixamos um pacote desse programa em um site e depois o instalamos. O formato desse pacote é `dpkg`, que é um arquivo com a extensão `.deb`.

Vamos instalar novamente o servidor de FTP, porém, usando o pacote `.deb`. Procuramos na internet e baixamos via navegador o `vsftpd` no endereço: http://ftp.br.debian.org/debian/pool/main/v/vsftpd/vsftpd_3.0.2-17+deb8u1_i386.deb (http://ftp.br.debian.org/debian/pool/main/v/vsftpd/vsftpd_3.0.2-17+deb8u1_i386.deb). Ele estará salvo no diretório de *Downloads*. Para instalar entramos nesse diretório com o comando `cd` e utilizando o comando `dpkg` fazemos:

```
sudo dpkg -i vsftpd_3.0.2-17+deb8u1_i386.deb
```

[COPIAR CÓDIGO](#)

O `-i` indica que estamos instalando o programa. Lembre-se de que o nome do arquivo precisa ser fornecido e que o comando deve ser executado como *root* por causa de suas permissões. E para desinstalar um programa pelo `dpkg`? Muito simples, fazemos:

```
sudo dpkg -r [nome do pacote]
```

[COPIAR CÓDIGO](#)

Perceba que para a desinstalação não usamos o nome do arquivo baixado, mas sim o nome do pacote do programa, que neste caso é `vsftpd`.

```
sudo dpkg -r vsftpd
```

[COPIAR CÓDIGO](#)

Lembre-se de testar a instalação e desinstalação da mesma forma que fizemos na aula passada, efetue login, encerre a conexão, faça login com outros usuários, execute comandos via ftp.

Então vimos duas formas de instalar e desinstalar programas no Linux:

1. Via *apt*: quando o programa já está disponibilizado na central do Sistema Operacional Linux.
2. Via *dpkg*: quando baixamos pelo navegador da internet um pacote *.deb* do programa.



04

Instalando e removendo o servidor ftp



82%

ATIVIDADES
4 DE 4

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



Baixe o pacote .deb do vsftpd na seguinte url: [\(https://packages.debian.org/bullseye/vsftpd\).](https://packages.debian.org/bullseye/vsftpd)

O arquivo .deb para sistemas de 64 bits pode ser baixado por aqui:

<http://ftp.br.debian.org/debian/pool/main/v/vsftpd/> (<http://ftp.br.debian.org/debian/pool/main/v/vsftpd/>).



Instale o arquivo .deb do vsftpd , realize um teste conectando-se e desinstale o pacote.

Dica: Caso aconteçam erros devido as dependências do pacote vsftpd não estarem instaladas, você pode executar o comando \$ sudo apt-get -f install para tentar resolver esse problema.



43.0k xp



Opinião do instrutor



Vamos considerar que o nosso diretório



atual é o diretório em que se encontra o arquivo .deb . Após baixar o pacote, podemos instalar utilizando o comando `dpkg` :



82%

```
$ sudo dpkg -i vsftpd_3.0.3-2_i386.deb  
[sudo] password for lucas:
```

ATIVIDADES
4 DE 4

[COPIAR CÓDIGO](#)

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



Perceba que ocorreu um erro, pois o pacote do `vsftpd` necessita que outros pacotes estejam instalados. Até poderíamos instalar um por um manualmente, mas para tentar resolver isso de forma automática, fazemos:

```
$ sudo apt-get -f install
```

[COPIAR CÓDIGO](#)



Para testar a instalação do `vsftpd`, você pode pressionar `Ctrl + c`:

```
$ ftp localhost  
Connected to localhost.  
220 (vsFTPd 3.0.2)  
Name (localhost:lucas):
```

[COPIAR CÓDIGO](#)



43.0k xp



Para remover podemos passar o nome do

pacote:



\$ sudo dpkg -r vsftpd

[COPIAR CÓDIGO](#)



82%

ATIVIDADES
4 DE 4

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



MODO
NOTURNO



ABRIR
CADERNO



43.0k xp

a



Transcrição

Vamos instalar novamente o servidor FTP, para verificarmos o funcionamento de serviços no Linux. Usaremos o *apt* para isso:

```
sudo apt-get install vsftpd
```

[COPIAR CÓDIGO](#)

Para testarmos se o *vsftpd* foi instalado corretamente, podemos tentar realizar uma conexão, ou, verificar se o processo deste programa está em execução com ajuda dos comandos *ps* e *grep* da seguinte forma:

```
ps -ef | grep vsftpd
```

[COPIAR CÓDIGO](#)

```
guilherme@ubuntu:~$ ps -ef | grep vsftpd
root      21606     1  0 12:56 ?        00:00:00 /usr/sbin/vsftpd
1000      21638 19679  0 12:56 pts/2    00:00:00 grep --color=auto vsftpd
```

O programa está em execução, podemos para-lo, mas ao invés de usarmos o comando *kill*, usaremos o comando *service* seguido do nome do programa, e um indicador de ação, neste caso *stop*, para que o programa pare sua execução. Usamos o comando *service* pois o *vsftpd* é um **serviço** que roda toda vez que ligamos e desligamos a máquina.

```
sudo service vsftpd stop
```

[COPIAR CÓDIGO](#)

Experimente verificar se o serviço realmente parou de executar como já fizemos com o comando `ps`.

O comando `service` deve ser executado como *root*, caso não, teremos erros de permissão. Para ele voltar a execução, mudamos apenas a ação do comando `service` de `stop` para `start`:

```
sudo service vsftpd start
```

[COPIAR CÓDIGO](#)

Por trás dos panos o que está realmente acontecendo é que um *script* de inicialização e desligamento (*shutdown*) que está localizado em `/etc/init.d/vsftpd` está sendo executado. Ele é executável! Sendo assim, podemos executá-lo diretamente:

Lembre-se que podemos verificar se algo é executável ou não verificando suas permissões com o comando `ls -l`.

Para encerrar a execução, fazemos:

```
sudo /etc/init.d/vsftpd stop
```

[COPIAR CÓDIGO](#)

Para inicializa-lo novamente, fazemos:

```
sudo /etc/init.d/vsftpd start
```

[COPIAR CÓDIGO](#)

O `/etc/init.d/vsftpd` é o *script* que, "educadamente", notifica o *vsftpd* para parar ou inicializar.

Os *scripts* dentro do diretório `/etc/init.d` são os programas que são executados no *startup* da máquina. Eles podem continuar rodando até desligarmos o computador ou rodar por apenas um instante e depois parar. Caso queiramos que um programa seja executado sempre ao iniciar da máquina, basta que deixemos esse programa nesta pasta.



02

Gerenciando serviços



85%

Neste capítulo vimos como gerenciar serviços em um sistema Linux.

ATIVIDADES
2 DE 4

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD

Caso o servidor `vsftpd` não esteja instalado no seu sistema, instale através do comando:

```
$ sudo apt-get install vsftpd
```

[COPIAR CÓDIGO](#)



Utilize o comando necessário para parar o servidor de `ftp`. Por fim, inicie novamente o serviço.



Opinião do instrutor



43.1k xp

Para parar um serviço, utilizamos o comando `service`, fornecendo o nome do serviço e o que desejamos fazer. Como queremos parar o serviço, utilizamos o `stop`:

```
sudo service vsftpd stop
```



Para iniciar novamente o serviço,
utilizamos o `start` :

[COPIAR CÓDIGO](#)



`sudo service vsftpd start`

85%

[COPIAR CÓDIGO](#)

ATIVIDADES
2 DE 4

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



MODO
NOTURNO



ABRIR
CADERNO



43.1k xp

a



03

Verificando os serviços existentes



87%

ATIVIDADES
3 DE 4

FÓRUM DO
CURSO
[VOLTAR
PARA
DASHBOARD](#)



Opinião do instrutor



Ao executar o comando `ls /etc/init.d/`, podemos verificar os serviços que possuímos no nosso sistema. No meu caso, obtive o seguinte retorno:

```
$ ls /etc/init.d/
acpid          cgmanager
alsa-utils     cgproxy
anacron        checkfs.sh
apparmor       checkroot-bootcleaner
apport         checkroot.sh
avahi-daemon   console-setup
bluetooth      cron
```



43.2k xp





bootmisc.sh

brltty

cups

cups-browsed

[COPIAR CÓDIGO](#)



87%

ATIVIDADES

3 DE 4

FÓRUM DO CURSO

[VOLTAR
PARA
DASHBOARD](#)



MODO
NOTURNO



ABRIR
CADERNO



L

43.2k xp

a



04

Verificando status dos serviços



89%

Utilize o retorno do comando `ls /etc/init.d` para verificar se os serviços estão nesse momento em execução.

ATIVIDADES
4 DE 4

Você pode utilizar o comando `ps -e` junto com o `grep`:

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD

```
$ ps -e | grep vsftpd  
8298 ? 00:00:00 vsftpd
```

[COPIAR CÓDIGO](#)



MODO
NOTURNO



ABRIR
CADERNO

Uma outra forma de verificar se um serviço está em execução é utilizando a opção `status`:

```
$ sudo service vsftpd status  
● vsftpd.service - vsftpd FTP :  
  Loaded: loaded (/lib/systemd/  
  Active: active (running) si  
# algumas linhas foram omitidas
```

[COPIAR CÓDIGO](#)



43.3k xp



Perceba que a saída do comando indica que o serviço `vsftpd` está em execução (`active (running)`). Agora vamos parar o serviço e verificar o `status` novamente:



89%

```
$ sudo service vsftpd stop  
$ sudo service vsftpd status  
● vsftpd.service - vsftpd FTP :  
  Loaded: loaded (/lib/systemd/  
  Active: inactive (dead) since  
# algumas linhas foram omitidas
```

[COPIAR CÓDIGO](#)

ATIVIDADES
4 DE 4

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



Agora o serviço está inativo (inactive (dead)). Para finalizar, é só iniciar o vsftpd novamente:

```
sudo service vsftpd start
```

[COPIAR CÓDIGO](#)

Opinião do instrutor



Aproveite para testar com outros serviços que você viu no diretório /etc/init.d e verificar se esses serviços estão em execução.



43.3k xp

a



89%

ATIVIDADES
4 DE 4

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



43.3k xp

a



Transcrição

Vimos até agora duas maneiras de instalar um programa no Linux:

1. Via `apt` : quando o programa já está disponibilizado na central de programas do Ubuntu.
2. Via `dpkg` : quando baixamos pelo navegador um pacote do programa com a extensão `.deb`.

Porém, é possível que um programa não esteja disponível em nenhuma das duas formas. Nesse caso vamos ter que baixar seu código fonte, compilá-lo e instalá-lo. Faremos um teste baixando o código fonte de um projeto, o *Git*, um programa para controle de versão.

Primeiro vamos entrar no site do *git-scm* em [git-scm.com \(https://git-scm.com/\)](https://git-scm.com/). Na aba *Tarballs* encontramos as versões compactadas do código fonte do *Git*. Lá, baixaremos a versão `git-1.8.3.1.tar.gz`. É importante que seja o `.tar.gz` por que o mesmo mantém as permissões de execução dos arquivos, enquanto um `.zip` não mantém.

No Terminal, entraremos no diretório de *Downloads* e descompactaremos o arquivo com o comando `tar` :

```
tar zxf git-1.8.3.1.tar.gz
```

[COPIAR CÓDIGO](#)

Será criado um diretório com nome `git-1.8.3.1` com vários *scripts* que podem ser executados. Entraremos nele para os próximos passos.

O padrão de criação para instalação de um projeto através do código fonte em *C** é *primeiramente testarmos a configuração da nossa máquina, ou seja, verificar se está faltando algum arquivo ou programa em nossa máquina que o *Git precise para funcionar*. Para isso, em geral, é disponibilizado um *script* chamado "*configure*". O *Git* disponibiliza esse *script*, Vamos executá-lo:

`./configure`

[COPIAR CÓDIGO](#)

O *script* *configure* fará uma série de checagens em nossa máquina, e ao fim, caso nenhum problema seja encontrado, poderemos usar o comando `make`, que é o padrão para rodar o *build* do projeto em *C**, *no caso do *Git*. Porém o comando `make` nos retorna a seguinte mensagem:

```
guilherme@ubuntu:~/Downloads/git-1.8.3.1$ make
GIT_VERSION = 1.8.3.1
    * new build flags
    CC credential-store.o
In file included from credential-store.c:1:0:
cache.h:19:18: fatal error: zlib.h: No such file or directory
compilation terminated.
make: *** [credential-store.o] Error 1
```

O erro indica que não foi encontrada a biblioteca `zlib.h`. O *script* *configure* não apontou essa falta. Vamos ter que instalá-la manualmente. Seu nome é `zlib1g-dev`:

`sudo apt-get install zlib1g-dev`

[COPIAR CÓDIGO](#)

Neste ponto, estamos informando diretamente o nome da biblioteca ser instalada, mas lembre-se, podemos buscar por seu nome através do `apt-cache search zlib` e usar o comando `grep` para filtragem: `apt-cache search zlib | grep dev`

Agora sim o `make` irá funcionar e gerar o *Git*. O próximo passo é, enfim, instalá-lo na nossa máquina:

```
sudo make install
```

[COPIAR CÓDIGO](#)

Caso não haja nenhum erro, o comando `git` estará disponível no terminal. Experimente digitar `git .` Uma listagem de ajuda deverá ser exibida.

Portanto, existem basicamente três passos para instalar um programa a partir de seu código fonte:

1. `./configure` para verificar as dependências e configurações da máquina.
2. `make` para gerar o programa, ou seja, compilar. Lembrando que, neste passo, pode haver outras dependências necessárias para a tarefa e por isso talvez seja preciso realizar instalações de outras bibliotecas.
3. `sudo make install` para que o programa seja instalado em nossa máquina. Lembrando que o `sudo` é necessário por causa de questões de permissão.

Estes mesmos passos podem se repetir para determinados programas onde as dependências não estejam disponíveis através de arquivos `.deb` ou na central de programas.



02

Instalando a partir do código fonte



92%

ATIVIDADES
2 DE 2

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



ABRIR
CADERNO



Obs: Tenha certeza que seu sistema tenha alguma versão de Python instalada usando o comando `python --version`. Caso **não** tenho nenhuma versão de Python, use o comando `sudo apt-get update && sudo apt-get install python3`.

Agora vamos instalar o `git` a partir do código fonte. Você pode baixar o arquivo `.tar.gz` na seguinte url:

<https://www.kernel.org/pub/software/scm/git/git-2.7.1.tar.gz>
[\(https://www.kernel.org/pub/software/scm/git/git-2.7.1.tar.gz\).](https://www.kernel.org/pub/software/scm/git/git-2.7.1.tar.gz)

Após baixar o arquivo, descompacte em um diretório e realize os passos necessários para instalar o `git`.

No fim, execute o comando `git` para ver se a instalação ocorreu da forma esperada.



43.4k xp



Opinião do instrutor



Após baixar o pacote `.tar.gz`, vamos



acessar a pasta em que se encontra o arquivo e descompactá-lo.



92%

Em seguida, vamos acessar a pasta criada após a descompactação.

```
$ tar zxf git-2.7.1.tar.gz
```

ATIVIDADES
2 DE 2

```
$ ls  
git-2.7.1  git-2.7.1.tar.gz
```

FÓRUM DO
CURSO

```
$ cd git-2.7.1/
```

[COPIAR CÓDIGO](#)

VOLTAR
PARA
DASHBOARD



Lembre-se que o formato `.tar` não compacta, apenas empacota. Esse formato é utilizado pois mantém as permissões que os arquivos possuíam antes do empacotamento. Após empacotar com o `.tar` podemos compactar utilizando outra ferramenta.



Primeiro vamos executar o `script configure`, para verificar se está faltando algo para que o `git` possa funcionar corretamente.

```
$ ./configure
```

[COPIAR CÓDIGO](#)



43.4k xp



Vamos executar o comando `make` para realizar o `build` do nosso projeto. Mas



nesse momento receberemos um erro, pois precisamos instalar uma dependência antes. Após instalar a dependência, podemos rodar o `make`.



92%

ATIVIDADES
2 DE 2

Por fim, para instalar o `git`, executamos um `make install`. Para instalar, é necessário permissões de administrador:

```
$ sudo apt-get install zlib1g-dev  
$ make  
$ sudo make install
```

[COPIAR CÓDIGO](#)

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



MODO
NOTURNO



ABRIR
CADERNO



43.4k xp

a



Transcrição

Vamos aprender nesta aula como acessar um servidor remoto no Linux. Para isso, teremos que fazer uma comunicação com o outro servidor. Já vimos o FTP, mas o FTP é para troca de arquivos. O que queremos é nos *logar* como um usuário. Para isso iremos usar o SSH. O primeiro passo é instalá-lo:

```
sudo apt-get install ssh
```

[COPIAR CÓDIGO](#)

Desta forma instala-se tanto o cliente SSH (`ssh-client`), quanto o servidor (`ssh-server`). Para testarmos se o programa instalou corretamente, logaremos na nossa própria máquina utilizando o comando `ssh`, fornecendo o nome de um usuário já criado anteriormente e o *ip* da máquina.

```
ssh jose@localhost
```

[COPIAR CÓDIGO](#)

Será pedida a senha do usuário `jose` e logo após, estaremos logados no servidor remoto. Poderemos executar uma série de comandos, porém não temos acesso às ferramentas e programas gráficos. Não podemos, por exemplo, abrir um navegador.

Para termos essa permissão, precisamos nos conectar usando um

modificador que permita o uso de ferramentas gráficas. O `-X` é esse modificador:

```
ssh -X jose@localhost
```

[COPIAR CÓDIGO](#)

Para encerrar a conexão, usamos o comando `exit` da mesma forma que fizemos quando usamos o `ftp`.

Lembrando que tudo o que estamos fazendo está sendo executado lá no servidor e não em nossa máquina. Somente o gráfico é mostrado em nossa máquina, as ações são todas remotas.

Agora vamos ver como copiar um arquivo da nossa máquina local para a máquina remota. Fazemos por meio do comando `scp`, indicando para ele qual é o arquivo e qual é o destino do arquivo:

```
scp work.zip jose@localhost:/home/jose
```

[COPIAR CÓDIGO](#)

`/home/jose` é a `home` do usuário `jose` e pode ser substituído por `"~"`:

```
scp work.zip jose@localhost:~/
```

[COPIAR CÓDIGO](#)

Com isso jogamos o arquivo `work.zip` no nosso servidor remoto. Se o buscarmos dentro da outra máquina, nos conectando novamente com o `ssh` e listarmos os arquivos com o comando `ls` iremos perceber que realmente ele foi copiado.

Caso seja necessária a copia de arquivos de forma recursiva, pode-se utilizar a opção `-r` assim como era feito com os comandos `mv` e `cp`.



02

Realizando conexão via ssh



96%

ATIVIDADES
2 DE 4

Instale o pacote `ssh`, que instalará tanto um cliente, para que consigamos nos conectar, quanto um servidor, para que possamos receber conexões.

```
sudo apt-get install ssh
```

[COPIAR CÓDIGO](#)FÓRUM DO
CURSOVOLTAR
PARA
DASHBOARDMODO
NOTURNOABRIR
CADERNO

Caso ainda não tenha criado um usuário, você precisará fazer isso agora. Crie um usuário chamado `jose`.

Se conecte no usuário `jose` através do `ssh`. Após realizar a conexão, utilize o comando `whoami` para garantir que você está logado com outro usuário.



Opinião do instrutor



43.5k xp



Para realizar uma conexão `ssh`, basta indicar para o comando o nome do usuário e o *ip* da máquina que desejamos nos conectar. No nosso caso, utilizaremos



localhost , pois a conexão será na nossa
própria máquina:



96%

```
$ ssh jose@localhost  
jose@localhost's password:
```

[COPIAR CÓDIGO](#)

ATIVIDADES
2 DE 4

Ao executar o comando whoami , podemos
perceber que estamos logados com o
usuário jose :

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD

```
$ whoami  
jose
```

[COPIAR CÓDIGO](#)



Para desconectar, basta usar o comando
exit :



```
$ exit  
logout  
Connection to localhost closed
```

[COPIAR CÓDIGO](#)



43.5k xp

a



03

Transferindo arquivos com scp



98%

Agora nós iremos transferir um arquivo para uma máquina remota utilizando o comando `scp`.

ATIVIDADES
3 DE 4

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



Você pode escolher um arquivo de sua preferência para transferir. Caso queira, pode transferir o diretório `workspace`, criado anteriormente, ou o diretório `scripts` (não se esqueça de utilizar a opção `-r` caso escolha transferir um diretório). Para transferir um arquivo, vamos compactar um dos diretórios (lembre-se de alterar o nome do diretório caso seja necessário):



`$ zip -r work.zip workspace/`

[COPIAR CÓDIGO](#)

Utilize o comando `scp` para copiar o arquivo para a pasta do usuário `jose`, que é um usuário de sua máquina. Se logue com o usuário `jose` e verifique se o arquivo foi copiado.



43.6k xp

a



Opinião do instrutor



98%

ATIVIDADES
3 DE 4

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD

MODO
NOTURNO

ABRIR
CADERNO

Para realizar a cópia, informamos o nome do arquivo para o comando `scp` junto com o nome do usuário, *ip* e local onde copiaremos o arquivo na máquina remota:

```
$ scp work.zip jose@localhost:~
```

[COPIAR CÓDIGO](#)

O `~` representa o diretório do usuário, que nesse caso é `/home/jose/`.

Vamos nos conectar no usuário `jose` via `ssh` e verificar se o arquivo `work.zip` se encontra no diretório do usuário:

```
$ ssh jose@localhost  
jose@localhost's password:
```

```
$ whoami  
jose
```

```
$ ls  
examples.desktop work.zip
```

[COPIAR CÓDIGO](#)

43.6k xp

a

Como podemos ver, o arquivo foi copiado.



Lembre-se que o comando `scp` suporta a opção `-r` para realizar cópia de diretórios.



98%

ATIVIDADES
3 DE 4

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



43.6k xp

a