treinadev

Introdução ao Terminal

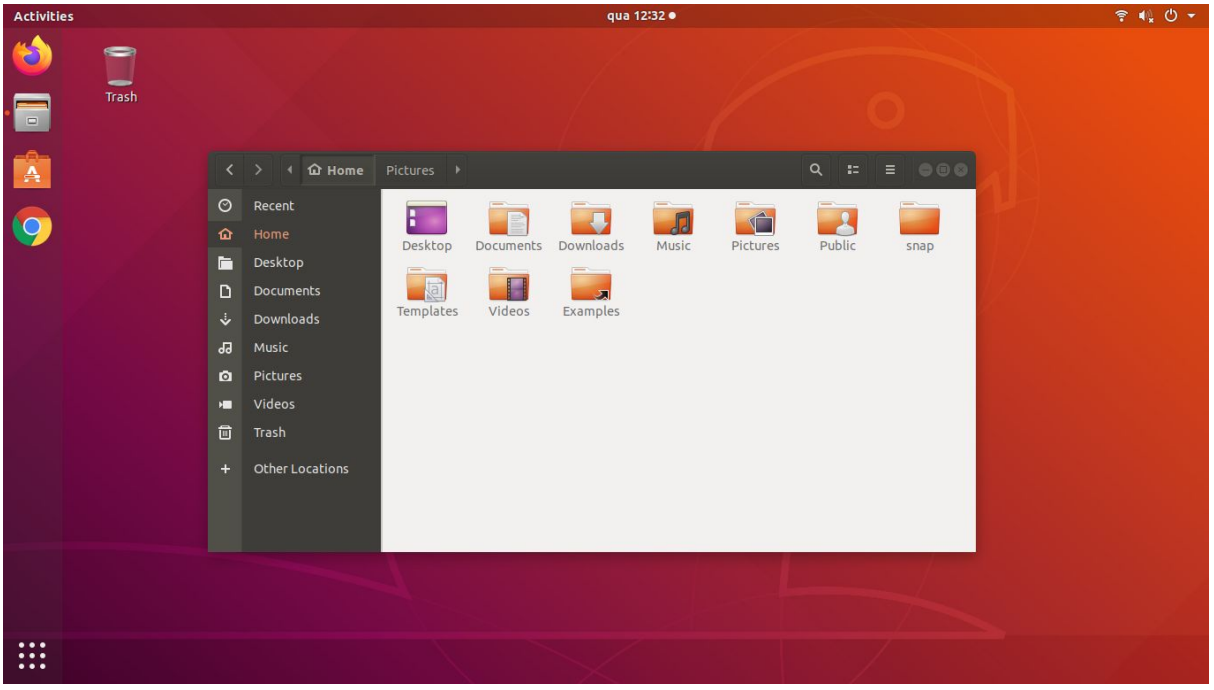
CAMPUS CODE

Introdução ao Terminal

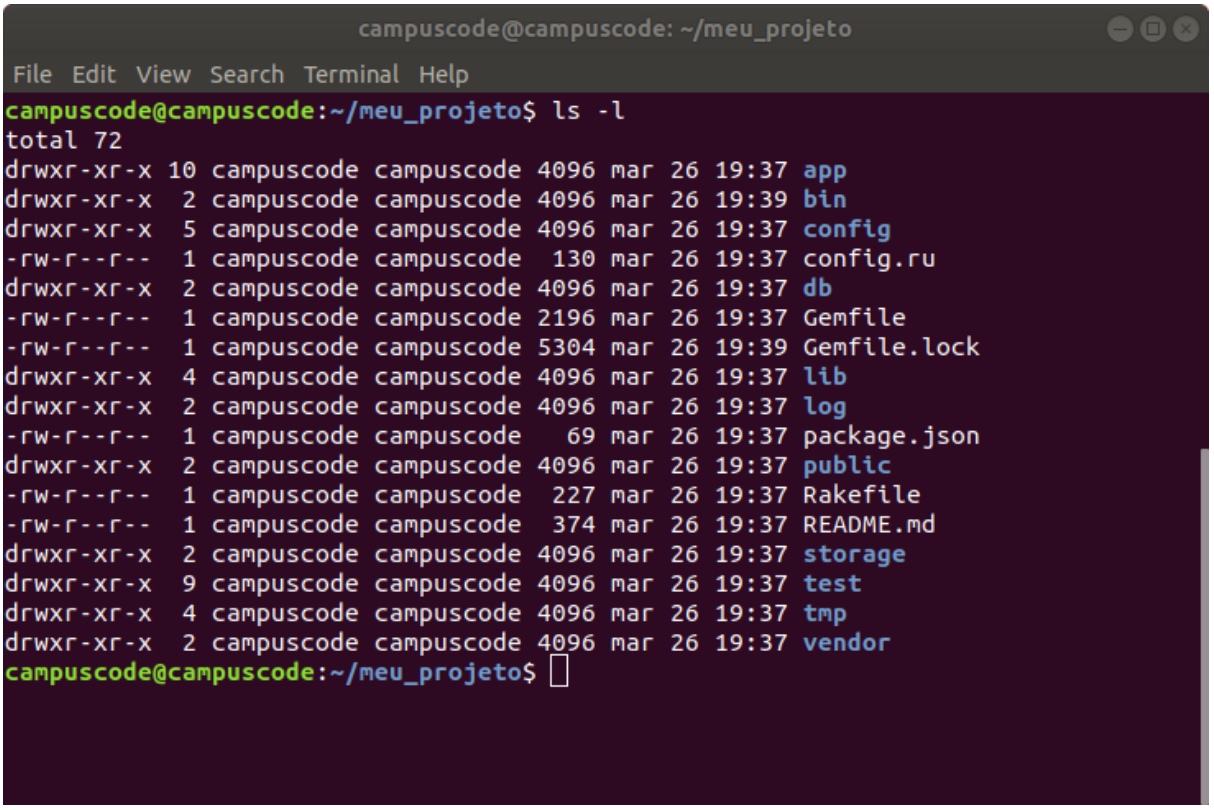
Sem medo da tela preta

Tratado com muito receio pelos iniciantes, o Terminal – ou Linha de Comando (*Command Line*) – é um programa que permite a execução de comandos diretamente no Sistema Operacional (SO). Instruções como criar, apagar, mover ou executar arquivos podem ser feitas com linhas de texto puro, sem a necessidade de uma interface gráfica.

Interfaces gráficas são o meio pelo qual o usuário interage com um sistema, mais comumente sistemas operacionais de aparelhos eletrônicos. Essas interfaces costumam possuir textos, ícones e outros indicadores visuais, por meio dos quais o usuário executa certas ações. Elas se opõem às interfaces de linhas de comando, em que todas as ações são inseridas por comandos digitados com o teclado.



Interface gráfica do sistema operacional Ubuntu



Interface de linha de comandos

Alguns dos motivos que tornam o Terminal um recurso tão importante durante o desenvolvimento de software são:

- A maior parte das linguagens de programação são instaladas, configuradas e estão 100% disponíveis no Terminal;
- Não ficamos limitados a opções que o SO oferece por meio da interface gráfica;
- A performance ao executar comandos pelo Terminal geralmente é maior;
- Servidores de aplicações Web são acessados, na maioria das vezes, única e exclusivamente via Terminais;
- Podemos criar sequências de comandos e automatizar tarefas através de scripts.

Além desses motivos, se você pretende trabalhar com desenvolvimento é muito provável que em algum momento seja necessário executar comandos em um servidor Unix, por exemplo via comando SSH (*Secure Shell*), e nesse ponto torna-se essencial compreender como funciona o Terminal.

Dica

Unix? Vamos citar com frequência o termo ‘sistemas operacionais Unix’. Unix foi um Sistema Operacional que surgiu nos anos 1960 e influenciou a criação de diversas vertentes como Linux e Mac OS. Por terem as mesmas raízes, costumamos chamá-los de 'sistemas baseados em Unix’ ou ‘similares ao Unix’. Sistemas operacionais baseados em Unix são usados em mais de 65% dos servidores no mundo.

Vamos utilizar o Sistema Operacional Ubuntu para todos os exemplos descritos aqui. O Ubuntu é uma distribuição Linux bastante popular entre desenvolvedores de software e sua instalação é gratuita. Para conhecer mais você pode acessar o site oficial.

Terminal, bash, dash, fish e command prompt

O Terminal é apenas uma “casca” para rodar o *shell*. O *shell* é a interface por meio da qual acessamos nosso Sistema Operacional, ou seja, o *shell* permite receber comandos amigáveis e enviar ações para o *kernel*, que é o responsável por gerenciar todos os recursos do computador em sistemas operacionais baseados em Unix. Nos sistemas Unix há diversas opções de *shell*, entre elas, o *bash*, o *dash* e o *fish*.

Bash é o *shell* mais comum nos sistemas Linux. Ele é basicamente um programa executável disponível em `/bin/bash` que possui alguns recursos exclusivos, principalmente para scripts. Temos também o sh, que é o antecessor do bash.

CMD ou *Command Prompt* é o utilitário do Windows que aceita linhas de comando. Muitos se referem a ele erroneamente como “*shell* do DOS”, mas ele é um programa Windows que emula alguns comandos do DOS.

Dica

DOS, abreviação de *Disk Operating System*, é basicamente um Sistema Operacional que pode usar discos de armazenamento de dados, como disquetes, discos rígidos ou discos ópticos. Ele foi um dos sistemas operacionais utilizados nos primeiros computadores pessoais entre os anos 1970 e 1990 para navegação de arquivos.

Terminal no Windows, tem como?

No início de 2016, a Microsoft fez algo que era considerado impossível até então: permitiu instalar de forma nativa o *bash* no Windows 10 por meio do “Subsistema do Windows para Linux” (WSL). Estão disponíveis diversas versões de distribuição (chamadas distros), como: Ubuntu, Kali Linux e Debian, entre outras.

Dica

Para saber mais sobre Subsistema do Windows para Linux acesse: Introdução ao Subsistema do Windows para Linux.

Básico de Terminal

No Ubuntu, você pode abrir o Terminal com o atalho `Ctrl + Alt + t` ou clicar no primeiro ícone do lançador de aplicações e digitar Terminal. Você também pode escrever *shell*, *command*, *prompt*, que são todos sinônimos comuns utilizados por desenvolvedores.

Onde estou?

Toda vez que um Terminal é aberto, o *bash* é inicializado mostrando um diretório padrão. Normalmente esse diretório é sua pasta de usuário. A forma como seu Terminal mostra esse local pode variar de um sistema para outro. Para saber exatamente em qual diretório você está, use o comando `pwd` – *print working directory*. Digite `pwd` (com todas as letras minúsculas) e pressione a tecla `Enter`. Você receberá como retorno o caminho até o diretório atual. No Ubuntu você deve ver algo como `/home/nome_do_usuario` por ser uma convenção do sistema operacional criar um diretório com seu usuário. Note que o comando permanece na linha em que foi inserido e a resposta de saída deste comando é impressa na linha logo abaixo.

```
$ pwd
/home/nome_do_usuario/
```

Assim que todas as respostas de saída terminam de imprimir na tela, o Terminal se mostrará pronto para receber novos comandos.

Ver o conteúdo do diretório

No Linux, as pastas são chamadas de diretórios, então vamos usar essa nomenclatura a partir de agora. O sinal `~` é usado como um atalho para o diretório do usuário atual do computador. A seguir, veremos alguns comandos para organizar um diretório com documentos.

Para visualizar o conteúdo de um diretório, vamos usar o comando `ls`, que lista o conteúdo do diretório em que você está.

```
$ ls
Desktop     Downloads     Documents
```

No entanto, você pode precisar de mais detalhes que apenas o nome dos diretórios, para isso, digite `ls -l`. O conteúdo será apresentado em forma de lista e com informações como tamanho, data de criação/modificação, permissões etc.

```
$ ls -l
total 52K
drwxr-xr-x 2 nome_do_usuario 4,0K abr  8 12:50 Desktop
drwxr-xr-x 2 nome_do_usuario 4,0K abr  8 12:50 Documents
drwxr-xr-x 2 nome_do_usuario 4,0K abr 30 17:32 Downloads
drwxr-xr-x 2 nome_do_usuario 4,0K abr  8 12:50 Music
drwxr-xr-x 2 nome_do_usuario 4,0K abr 12 16:53 Pictures
```

Outra forma de listar o conteúdo de um diretório é com `ls -a`. O `-a` é uma opção para mostrar todo o conteúdo, inclusive arquivos que estão ocultos. Em Unix (Linux e Mac OS) existe o conceito de arquivo oculto – como no Windows.

Como você pode perceber, os comandos podem ser incrementados por opções. Acima, falamos do `-l` e agora do `-a`, mas eles podem ser usados ao mesmo tempo, como em `ls -la`.

Criar um novo diretório

O comando `mkdir` – *make directory* – vai criar um novo diretório dentro do diretório que você está. Ele depende de um argumento: o nome do diretório. Tente criar, por exemplo, um diretório chamado `meus_documentos`:

```
$ mkdir meus_documentos
```

Agora, se você pedir para listar o conteúdo com `ls`, seu novo diretório vai aparecer na lista.

```
$ ls
Desktop     Downloads     Documents      meus_documentos
```

Dica

Quando estamos no Terminal é comum evitarmos o uso de espaços (podemos utilizar `_` ou `-` no lugar de espaço) ou caracteres especiais como acentos e cedilhas (ê, ç, /, á) nos nomes dos nossos diretórios e arquivos para facilitar o acesso e uso dos comandos.

Navegar pelos diretórios

Ao criar um diretório, você não entra automaticamente nele, apenas o cria. Para isso, usamos o comando `cd` – *change directory* – seguido do nome do diretório em que queremos entrar. Então o comando ficaria mais ou menos assim `cd nome_do_diretorio`. No nosso exemplo: `cd meus_documentos`.

```
$ cd meus_documentos

$ pwd

/home/nome_do_seu_usuario/meus_documentos
```

Dica

Se você digitar `cd D` e pressionar a tecla `tab`, o Terminal vai procurar algum diretório que comece com `D` e vai completar a palavra. Caso não a encontre, nada vai acontecer. Se encontrar mais de um diretório – Desktop, Downloads, Documentos, por exemplo –, vai mostrar todas as opções e, ao apertar `tab` novamente, vai alternar entre elas.

Você pode usar essa estratégia com mais de uma letra, não precisa ser apenas a primeira, além disso, o *shell* diferencia entre maiúsculas e minúsculas.

Para organizar nossos documentos no mundo físico costumamos separá-los em pastas, por exemplo: pessoal, casa, educação, carro, faculdade. Aqui, faremos o mesmo, mas em diretórios.

```
$ pwd

/home/nome_do_seu_usuario/meus_documentos

$ mkdir pessoal

$ mkdir casa

$ mkdir educacao

$ mkdir carro

$ mkdir faculdade

$ ls

carro     casa     educacao     faculdade     pessoal
```

Em teoria, um diretório recém-criado deve estar vazio, certo? Se pedirmos para listar os arquivos dentro de algum dos diretórios recém criados com `ls`, de fato ele vai parecer vazio, mas ao usar `ls -a`, veremos o `.` e o `..` .

```
$ cd carro
$ ls -a
.  ..
```

Eles não são arquivos, mas **referências**:

- o `.` é uma referência ao diretório atual. Tente o comando `cd .` Você verá que nada acontece, mas também não há retorno de erro;
- o `..` é uma referência ao nível acima. Tente o comando `cd ..` e o diretório atual mudará. Teste onde você está agora com o `pwd`!

Existem diversas formas de transitar entre os diretórios. Para voltar diretamente para `home` do usuário, não importando onde está agora, use `cd ~`.

Se você souber qual o caminho de um diretório, poderá entrar diretamente nele sem entrar sucessivamente em cada diretório. Para praticar esse comando, precisamos primeiro de uma hierarquia de pastas. Dentro do diretório `educacao`, crie o diretório `cursos` e, dentro dele, o `campus_code`. Em seguida, vá para a home com `cd ~`.

Agora, lembrando de usar o `tab` para completar automaticamente, na `home`, digite o comando:

```
$ cd meus_documentos/educacao/cursos/campus_code
```

O *shell* vai direto para o diretório `campus_code`. Confira a localização atual com `pwd`.

Outro comando bastante usado é o `cd -`. Ele vai levar para o diretório anterior – não confunda com o diretório acima. No nosso exemplo, voltará para a `home` (diretório no qual estávamos trabalhando antes) e não para o `cursos`, que é o diretório acima do atual.

Criar um arquivo vazio

Até agora trabalhamos somente com diretórios. A seguir, veremos comandos para tratar arquivos. Usamos o comando `touch` para criar um novo arquivo. O `touch` é responsável por “tocar” um arquivo e atualizar a data de modificação dele. Caso não seja encontrado o arquivo dentro do diretório corrente, será criado um novo. Esperto, não?

O comando `touch` depende de um argumento: o nome do arquivo . Vamos criar um arquivo `despesas.txt` dentro do `meus_documentos`.

```
$ touch despesas.txt
```

Dica

Você não precisa obrigatoriamente colocar a extensão nos arquivos criados em sistemas Unix, no entanto, vai te ajudar a reconhecer o conteúdo. Por exemplo, alguém criou uma arquivo com o nome `campus_code`. Não temos nem mesmo uma pista se é um arquivo de texto, imagem etc.

Sem uma extensão do arquivo, precisaremos abrí-lo para descobrir do que se trata. Se o arquivo possuísse uma extensão, como `campus_code.png`, teríamos uma ideia do que se trata: uma imagem. Economia de tempo!

Criar um arquivo oculto

Agora que já vimos como criar arquivos, temos um tipo de arquivo peculiar nos sistemas Unix, os arquivos ocultos. Eles são comumente usados para guardar preferências, estados de um programa e, é claro, esconder arquivos!

Para criar arquivos ocultos, coloque o `.` antes do nome do arquivo, desta forma: `touch .salario.txt`.

Liste o conteúdo do seu diretório. Apenas o `despesas.txt` aparece. Você se lembra que para mostrar os arquivos ocultos usamos `ls -a`?

Editar arquivos

Com nossos arquivos criados podemos inserir conteúdo dentro deles. Para editar um arquivo, você pode abrí-lo em um programa. No caso do Ubuntu, vamos usar o Gedit, que já vem instalado com o Sistema Operacional e pode ser acionado via Terminal. Para isso, digite `gedit despesas.txt`. Insira um texto qualquer, salve e feche o editor.

Dica

Cada Sistema Operacional possui editores pré-instalados e um editor padrão configurado, mas você pode instalar novos. Aqui tem uma lista de editores Unix que você pode testar.

Visualizar conteúdo de um arquivo

Arquivos que possuem conteúdo podem ser visualizados no próprio *shell*. Use o comando `cat` com o nome do arquivo como argumento: `cat despesas.txt`.

Copiar arquivos

Quer agilizar o trabalho e usar um arquivo que já está criado? O comando `cp` – *copy* – serve para copiar os arquivos. No nosso exemplo, vamos usar o arquivo `despesas.txt`, que criamos anteriormente, no entanto, há duas variações:

- fazer uma cópia do arquivo dentro do mesmo diretório, modificando o nome: `cp despesas.txt despesas_gerais.txt`;
- fazer uma cópia do arquivo para outro diretório sem modificar o nome: `cp despesas.txt casa`.

Podemos copiar diversos arquivos de uma só vez para o mesmo destino. Um exemplo de código poderia ser:

```
$ cp despesas.txt despesas_gerais.txt carro
```

Copiar diretórios

Se você já tem um diretório e quer fazer uma cópia dele em outro local, use o comando `cp -r`, sendo: `cp -r diretorio_original diretorio_destino`. A opção `-r` é de "recursivo", necessária para copiar de forma repetitiva os arquivos de dentro junto com os diretórios.

Crie no diretório `meus_documentos` um diretório chamado `comprovantes` e outro chamado `imagens_doc`, em seguida, faça cópias deles em outros diretórios utilizando o comando `cp -r`.

```
$ cp -r comprovantes carro

$ cp -r imagens_doc carro
```

Trabalhoso? Um pouco, mas podemos copiar diversos diretórios de uma só vez para o mesmo destino. O exemplo acima poderia ser feito assim:

```
$ cp -r comprovantes imagens_doc carro
```

Mover arquivos

Pode acontecer de você criar um arquivo e depois perceber que ele não deveria estar ali, mas em outro local. O comando para mover um ou mais arquivos é o `mv`, sendo que o último argumento será o diretório de destino.

Por exemplo, acabamos de criar dentro de `meus_documentos` um arquivo de nome `despesas_gerais.txt` e queremos que ele seja movido do diretório em que estamos para o diretório `pessoal`.

```
$ mv despesas_gerais.txt pessoal
```

O comando `mv` também pode ser usado em arquivos para renomeá-los:

```
$ mv nome_original.txt nome_novo.txt
```

Mover diretórios

O comando *shell* para mover diretórios é igual ao utilizado para mover arquivos:

```
$ mv diretorio_a_ser_movido diretorio_destino
```

Por exemplo, você criou os seguintes diretórios: `pessoal`, `educacao`, `casa`, `faculdade` e `carro`, mas percebeu que o faculdade deveria estar dentro de educacao, use:

```
$ mv faculdade educacao
```

É possível mover mais de um diretório de uma só vez com esse comando, sendo que os primeiros argumentos são os diretórios a serem movidos e o último argumento sempre é o destino. Por exemplo, crie os diretórios `cozinha` e `banheiro` em `meus_documentos`. Em seguida, mova-os para `casa`, com o comando `mv cozinha banheiro casa`. Use o comando `ls` para se certificar de que os diretórios foram movidos corretamente.

Da mesma forma, **múltiplos arquivos** podem ser movidos simultaneamente.

Dica

Cansou de repetir os mesmos comandos? À medida que a tecla `up` (seta para cima) é pressionada, o *shell* vai mostrando as últimas linhas digitadas, poupando seu tempo.

Remover um arquivo

Criou um arquivo, mas descobriu que não precisava? Livre-se dele com o comando `rm`. No diretório `meus_documentos` remova o arquivo `despesas.txt` com: `rm despesas.txt`.

Caso você precise remover todos os arquivos do diretório, use `rm *` . Por precaução, é preciso confirmar se quer mesmo deletar os arquivos. E não estranhe, as respostas podem variar entre `y` ou `n`, `yes` ou `no`, `s` ou `n` etc.

```
$ ls
despesas.txt   lista_de_compras.txt
$ rm *
zsh: sure you want to delete all 2 files in
/home/nome_do_seu_usuario/meus_documentos [yn]? n
```

Remover um diretório

Criou o diretório errado e agora precisa se livrar dele? Para essa ação, o comando é semelhante ao anterior, mas específico para diretórios, `rmdir` – *remove directory* – com o nome do diretório como argumento. Por exemplo: `rmdir nome_do_diretorio`. Remova o diretório `carro`.

```
$ rmdir carro
```

Outra maneira de remover diretórios é por meio do comando `rm -r nome_do_diretorio`. Mas tome cuidado com esse comando. O `-r` indica que será executada uma ação recursiva em todas as pastas e arquivos dentro deste diretório, ou seja, tudo será apagado e essa ação não pode ser desfeita! Por isso, cuidado ao usá-lo.

Para checar se deu tudo certo: `ls`. E lembre-se, para remover um diretório você não pode estar dentro dele! Verifique onde está com o `pwd` ;) .

Limpar a tela

Seu Terminal está repleto de comandos e isso está tirando sua atenção? Acabe com a bagunça digitando `clear`!

Para onde vamos daqui

Concluímos nossa rápida introdução ao Terminal. Vimos como navegar pelos diretórios, além de criar e remover arquivos e diretórios. Mas esse é apenas o início. O Terminal é uma ferramenta muito poderosa e versátil.

Esse material está em constante evolução e sua opinião é muito importante. Se tiver sugestões ou dúvidas que gostaria de nos enviar, entre em contato pelo nosso e-mail: treinadev@campuscode.com.br. Você pode encontrar a versão mais atualizada na área do participante do TreinaDev.

Versão	Data de atualização
1.0.0	26/03/2020

treinadev

é um programa gratuito de
formação de devs da Campus Code

Patrocínio:

VINDI

REBASE

PORTAL solar

smart fit

CAMPUS CODE

Al. Santos, 1293, conj. 73
Cerqueira César, São Paulo, SP
Telefone/Whatsapp: [11] 2369-3476