03

Criando e removendo diretórios

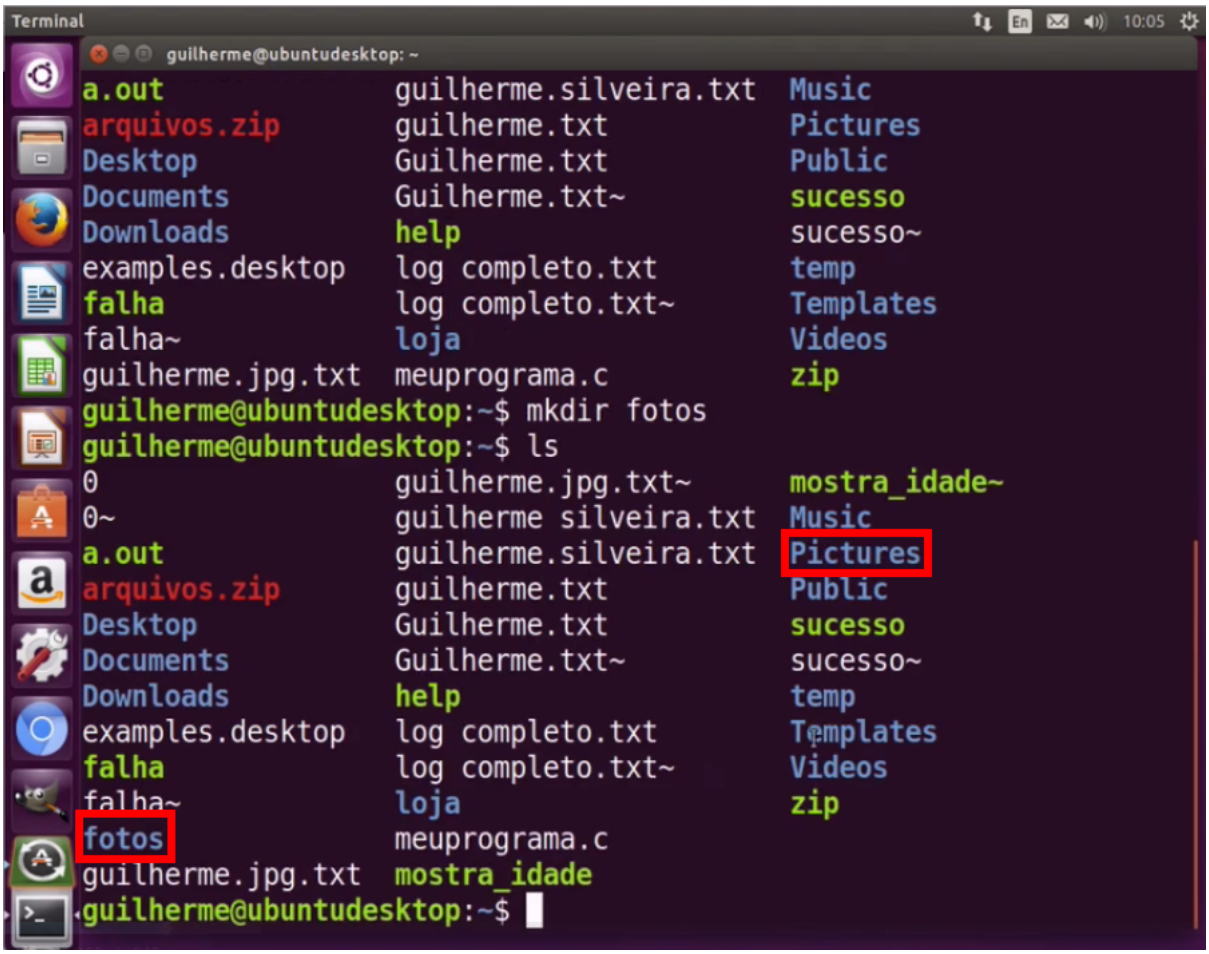
Criando e removendo diretórios

Vamos falar agora sobre como trabalhar para criar e remover diretórios. Em breve falaremos também sobre como criar, copiar, mover e remover arquivos.

Para criar um diretório o comando utilizado é o `mkdir`. Considerando que o diretório atual é o diretório `home` do usuário, se eu quiser criar um diretório para armazenar minhas fotos, posso fazer:

```
$ mkdir fotos
```

Ao executar o comando `ls` podemos ver o diretório `fotos` criado.



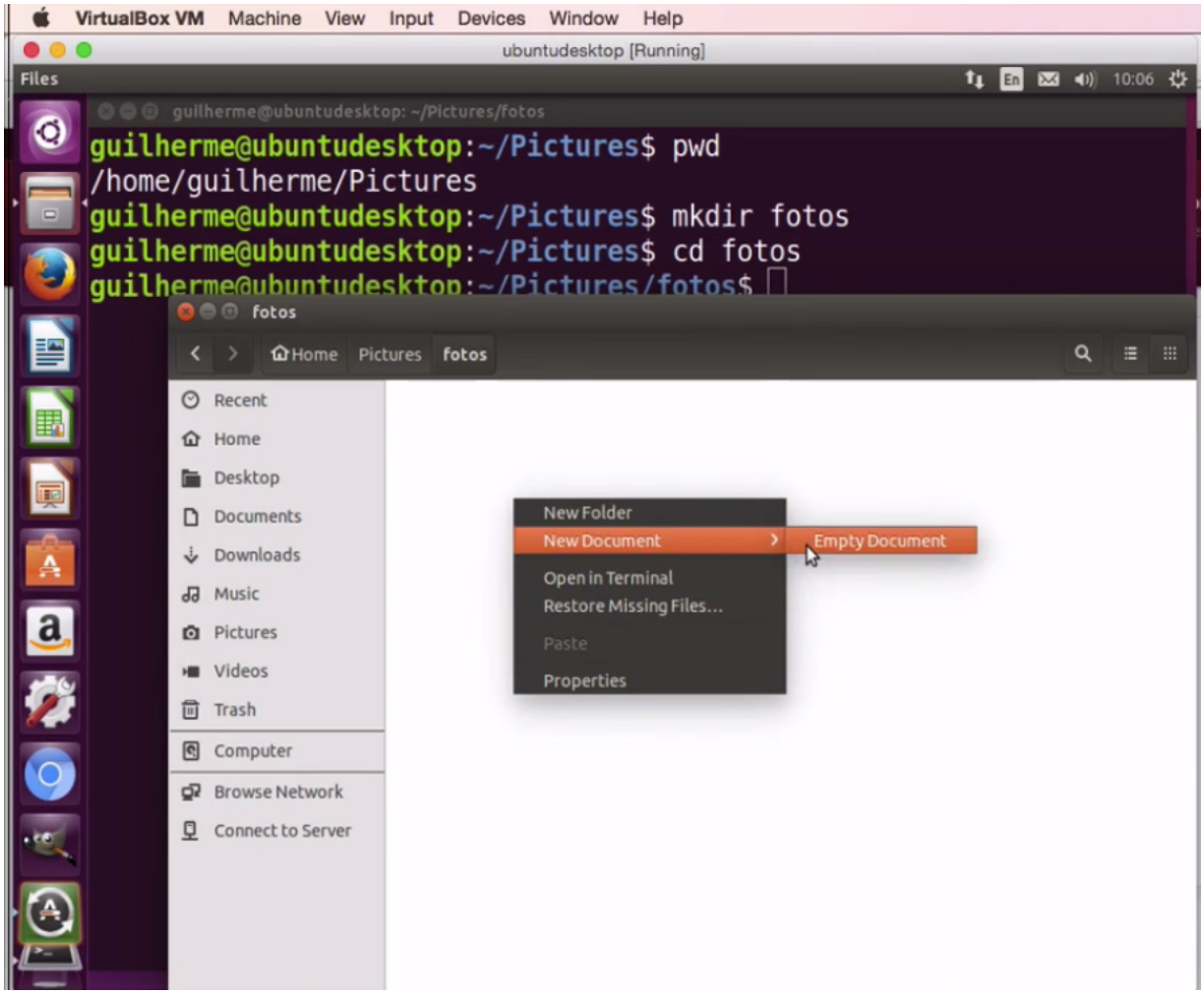
Se olharmos vamos ver que já existe um diretório onde podemos armazenar nossas fotos: o `Pictures`. Então não fez muito sentido criarmos o diretório `fotos`. Portanto vamos remover o diretório. Para remover um **diretório** utilizamos o comando `rmdir`:

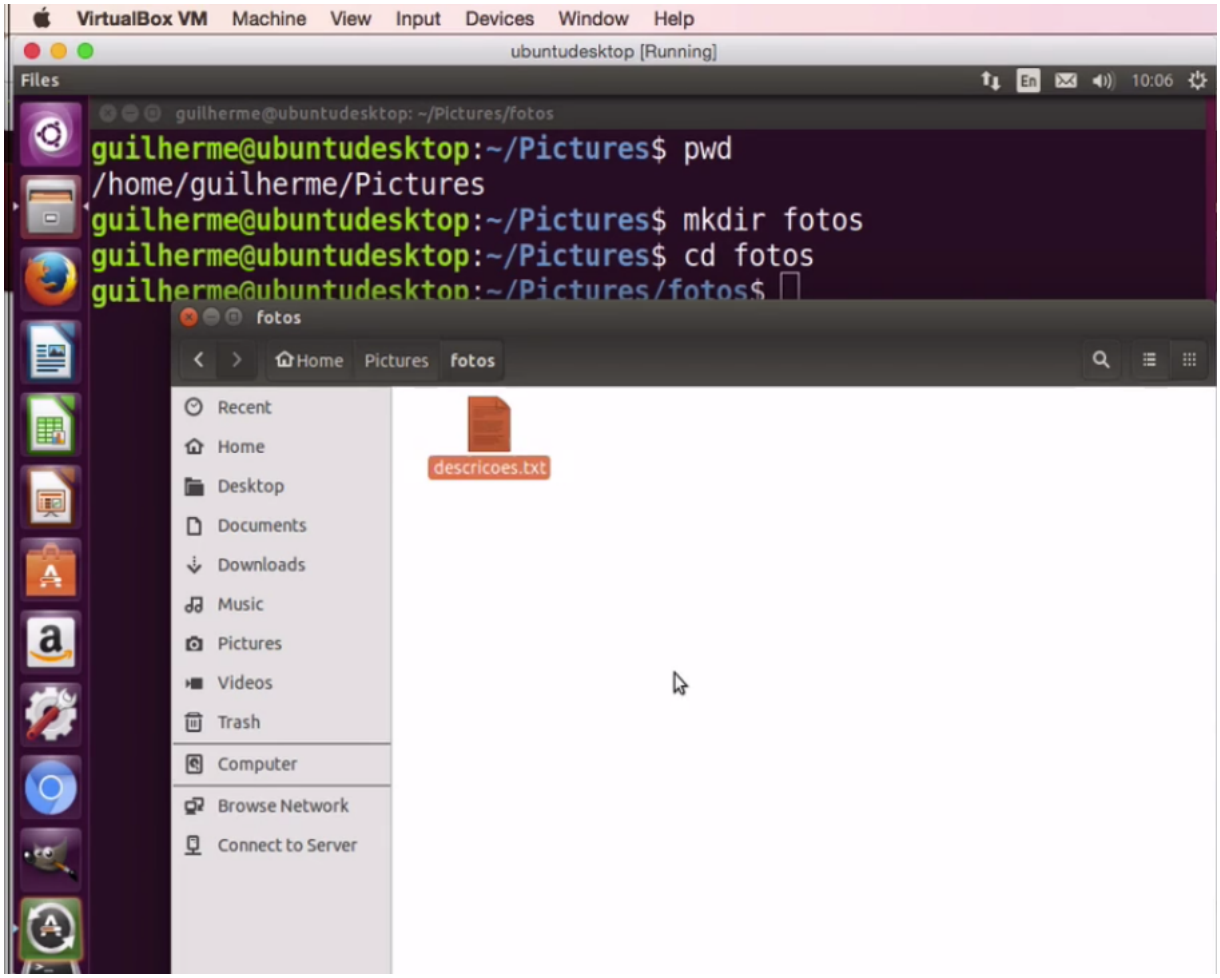
```
$ rmdir fotos
```

Se executarmos o comando `ls` o diretório `fotos` não existe mais. Vamos limpar a tela do terminal utilizando o atalho "Ctrl + L" ou o comando `clear`. Agora vamos entrar no diretório `Pictures`. Dentro dele vamos criar o diretório `fotos`.

```
$ cd Pictures
$ mkdir foto
```

Dentro do diretório `fotos` podemos fazer o que quiser. Por exemplo criar um arquivo com a descrição das nossas fotos. Acessando o diretório através do explorador de arquivos, basta clicar com o botão direito e escolher "New Document > Empty Document".

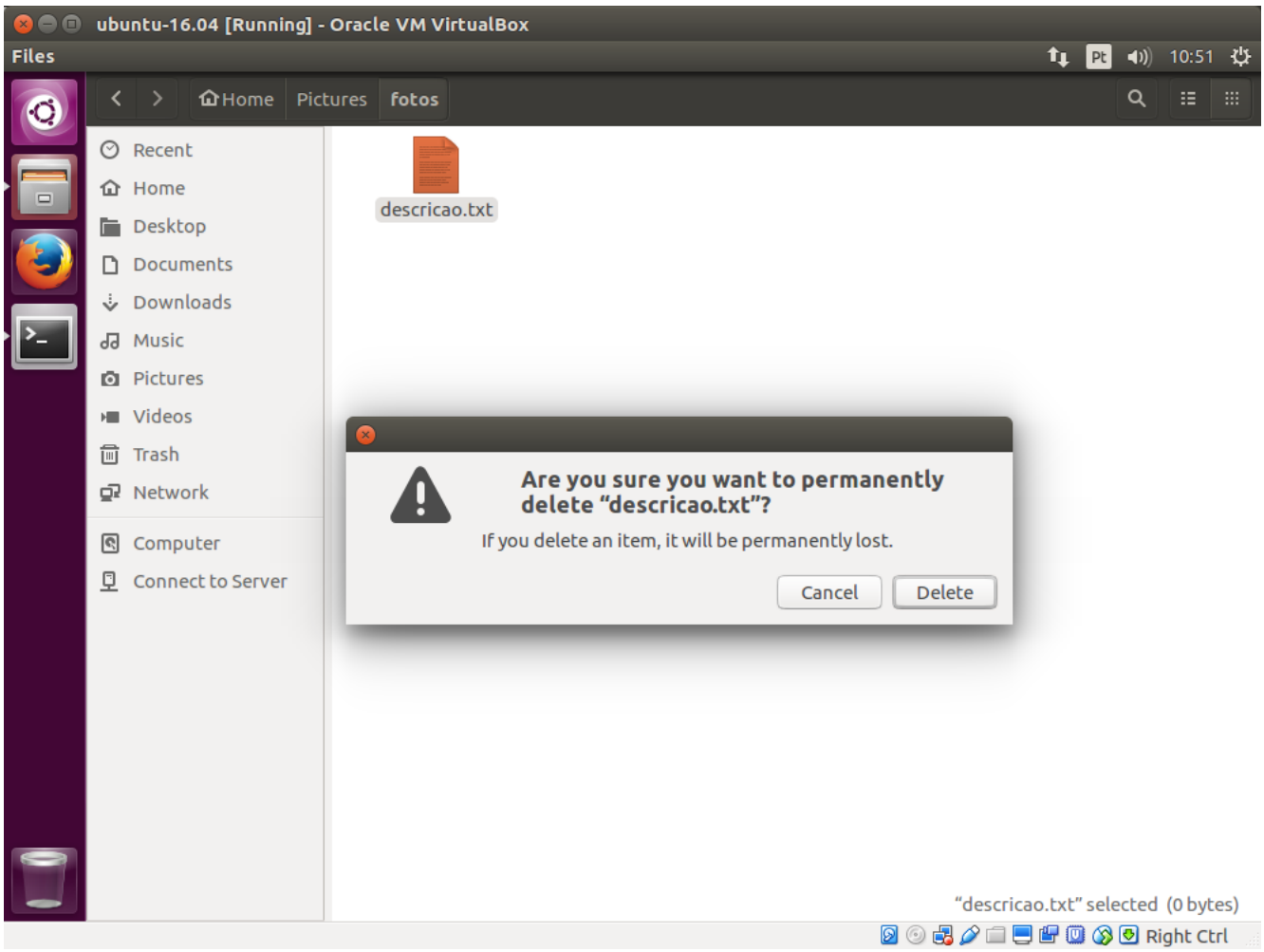




Vamos retornar para o diretório `Pictures`. E tentar remover o diretório `fotos`:

```
$ cd ..
$ rmdir fotos
rmdir: failed to remove 'fotos/': Directory not empty
```

Perceba que recebemos um erro. O `rmdir` remove um diretório apenas se ele estiver vazio.Se quisermos remover o diretório `fotos` utilizando o `rmdir`, primeiro será necessário remover o arquivo `descricao.txt`. Por enquanto, vamos remover o arquivo utilizando a interface gráfica. Basta selecionar o arquivo e pressionar "Shift + Del" para excluir de forma permanente.



Agora podemos remover o diretório `fotos`:

```
$ rmdir fotos
```

Vamos retornar à home do usuário. E se quisermos criar o diretório `Pictures/fotos`? É possível:

```
$ cd ..
$ mkdir Pictures/fotos
```

Nesse caso, o diretório `Pictures` já existe. O `mkdir` sempre cria o último diretório do caminho que passamos como argumento. Agora que o diretório `fotos` foi criado, podemos criar o diretórios, 2016, 2017 e 2018.

```
$ mkdir Pictures/fotos/2016
$ mkdir Pictures/fotos/2017
$ mkdir Pictures/fotos/2018
```

Dentro do diretório 2016, vamos criar o primeiro trimestre:

```
$ mkdir Pictures/fotos/2016/trimestre1
```

Mas o que acontece se quisermos criar, dentro de `Pictures/fotos` que já existe, o diretório `2019` e dentro de `2019`, o diretório `trimestre1`, em um só comando?

```
$ mkdir Pictures/fotos/2019/trimestre1
mkdir: cannot create directory ‘Pictures/fotos/2019/trimestre1’: No such file or directory
```

Como falamos, o `mkdir` por padrão cria apenas o último diretório do nome. Como `2019` não existe, recebemos um erro. Uma opção seria criar o diretório `2019` e em seguida o diretório `trimestre1` :

```
$ mkdir Pictures/fotos/2019 Pictures/fotos/2019/trimestre1
```

Como isso pode ser uma operação comum, o `mkdir` nos fornece uma opção: o `-p` .

```
$ mkdir -p Pictures/fotos/2020/trimestre1
```

Isso indica para o `mkdir` que sabemos que nem todos os diretórios que estamos passando existem, e que queremos criá-los.

Se executarmos o mesmo comando novamente, não receberemos nenhum erro. Se verificarmos o resultado da sua execução, vamos ver que foi executado com sucesso.

```
$ mkdir -p Pictures/fotos/2020/trimestre1
$ echo $?
0
```

Com a opção `-p` , o `mkdir` tenta criar o caminho. Como já existe ele também não nos retorna algum erro. Se rodarmos sem o `-p` , veremos um erro indicando que o diretório já existe:

```
lucas@lucas-VirtualBox:~$ mkdir  Pictures/fotos/2020/trimestre1
mkdir: cannot create directory ‘Pictures/fotos/2020/trimestre1’: File exists

$ echo $?
1
```

Portanto o `mkdir` com a opção `-p` não irá retornar erro caso o caminho já exista e seja um diretório. Se você tentar criar um diretório com o mesmo nome de um arquivo que já existe, ambos irão retornar um erro:

```
$ mkdir guilherme.txt
mkdir: cannot create directory ‘guilherme.txt’: File exists

~$ mkdir -p guilherme.txt
mkdir: cannot create directory ‘guilherme.txt’: File exists
```

A questão aqui é que `guilherme.txt` existe como um arquivo. Quando usamos o `-p` com um diretório que já existe ele não retorna erro. A lógica aqui é que o `-p` serve para garantirmos que um *path* (caminho) exista, quando já existe ele simplesmente não faz nada. Sem o `-p` , estamos falando apenas para o `mkdir` criar o último diretório do caminho, portanto se o diretório já existe ele não irá criar e recebemos um erro.

Agora vamos ver uma opção do `rmdir` . Perceba que nesse momento, o diretório `Pictures/fotos/2020` contém apenas o diretório `trimestre1` , que por sua vez está vazio. Se rodarmos o `rmdir` nesse caminho, algo como:

```
$ rmdir Pictures/fotos/2020/trimestre1
```

Ele removerá apenas o último diretório (`trimestre1`), desde que ele esteja vazio. O `2020` continuará a existir. O que podemos fazer para remover todos se eles estiverem vazios? Para isso, o `rmdir` também fornece a opção `-p` . Vamos tentar remover os diretórios `trimestre1` e `2019` :

```
~$ rmdir -p Pictures/fotos/2019/trimestre1
rmdir: failed to remove directory 'Pictures/fotos': Directory not empty
```

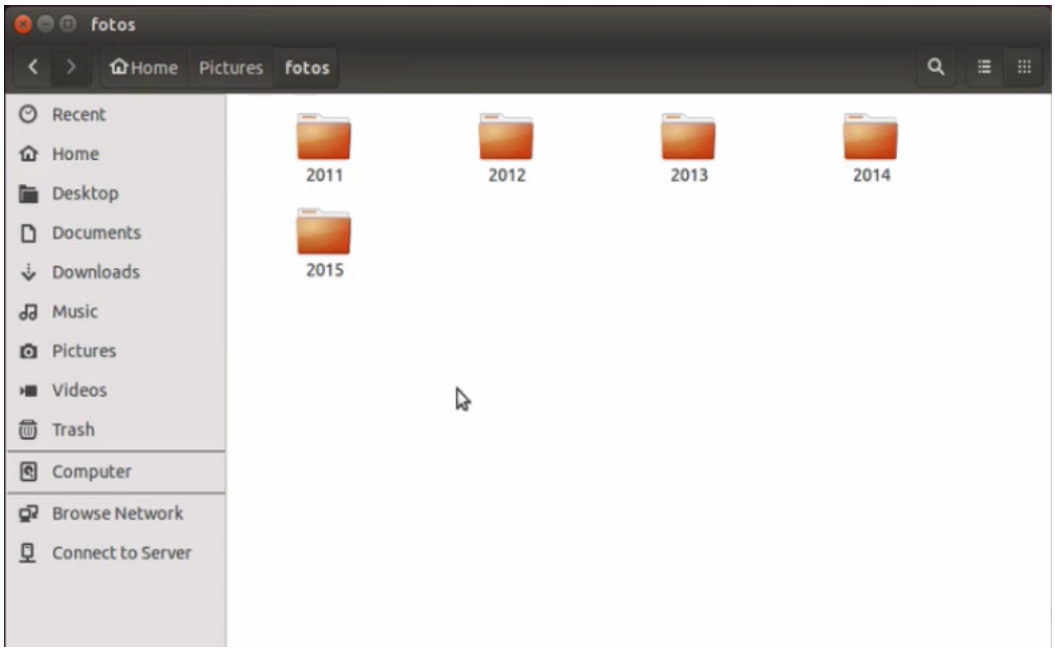
Primeiro o `rmdir` irá remover o diretório `trimestre1` . A partir desse momento o diretório `2019` não está mais vazio e também será removido. Como o diretório fotos possui outros diretório, ele não é removido. Se tivéssemos uma sequência de diretórios vazios, todos seriam removidos.

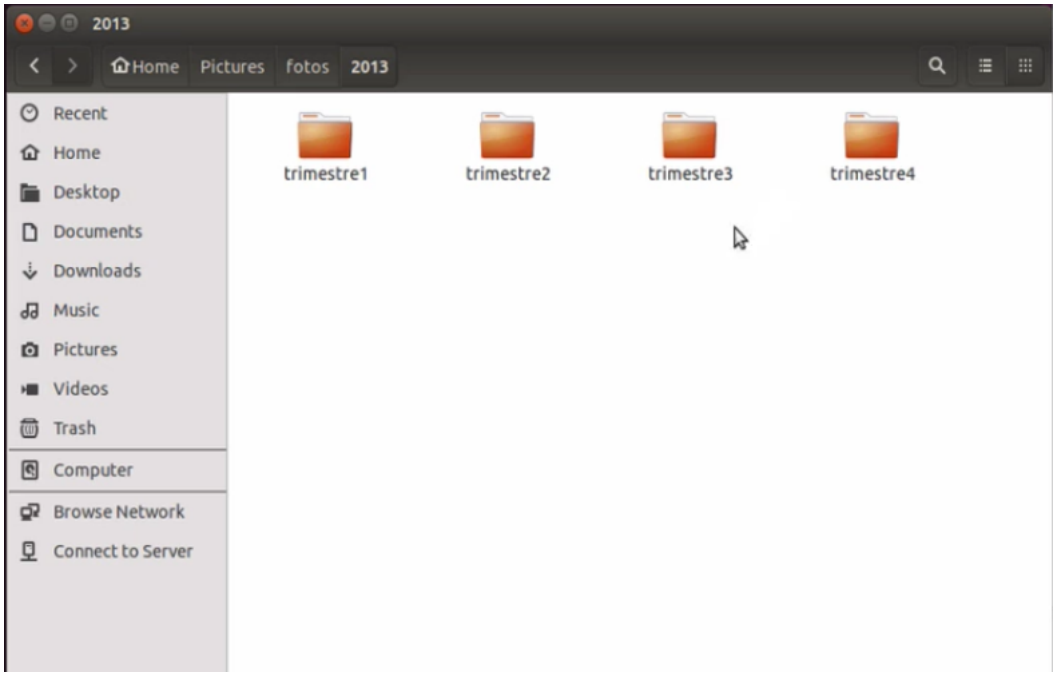
Criando e removendo diretórios com globbing

Juntando o `-p` com o globbing conseguimos fazer várias coisas bem interessantes. Vamos abrir o nosso explorador e vamos na pasta "Pictures", dentro dela temos outra pasta, o "fotos" e dentro desta temos outras pastas nomeadas de 2016, 2017, 2018 e 2020 vamos remover a pasta "fotos" e tudo o que ela possui dentro. Teremos apenas a pasta "Pictures". Vamos retornar ao Terminal e se digitarmos um `ls Pictures/` veremos que não temos nada dentro dele. Agora que não temos mais nada dentro do "Pictures" vamos tentar recriar um "Pictures/fotos" e dentro gostaríamos que tivessem as pastas dos 2011, 2012, 2013, 2014 e 2015. Para isso digitaremos `mkdir Pictures/fotos/201{1,2,3,4,5}` . Escrevendo `201{1,2,3,4,5}` estamos dizendo 2011, 2012, 2013, 2014 e 2015. Estamos utilizando a vírgula para separar as classes e dizer, classe 1,2,3,4 e 5 e podemos acrescentar uma barra e dizer que queremos os `trimestres{1,2,3,4}` . Para fazer isso funcionar vamos passar o `-p` e teremos ao todo o seguinte:

```
mkdir -p Pictures/foto/201{1,2,3,4,5}/trimestre{1,2,3,4}
```

Damos um "Enter" nisso e vamos conferir se tudo isso foi criado? Abrimos nosso *explorer* e entramos em "Pictures > fotos". Em fotos temos tudo o que tínhamos pedido e qualquer uma das pastas dentro da pasta "fotos" possuímos "trimestre1", "trimestre2", "trimestre3" e "trimestre4". Observe:





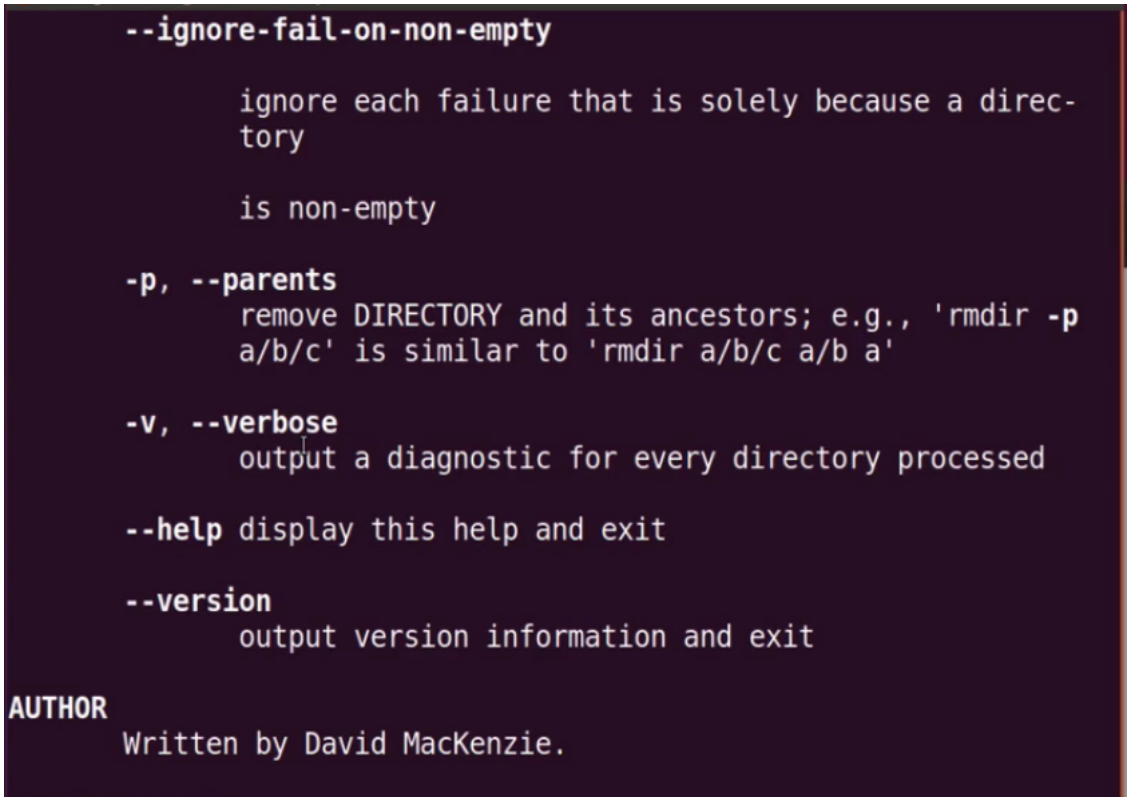
Se você quiser criar uma estrutura de diretórios que possui certo padrão podemos, paratanto, misturar o `mkdir -p` com o *globbing*.

O `rmdir` com o *globbing* é menos comum, não utilizamos muito ele na prática, mas poderíamos tentar usá-lo para remover o `Pictures/foto/201{1,2,3}/trimestre{1,2,3,4}` . Com isso estamos dizendo que queremos remover dentro da pasta "fotos" a pasta "2011, "2012, "2013" e dentro destas pastas queremos remover todas as referentes aos trimestres. De trás para frente, o que tentaremos fazer é apagar as pastas de trimestres e as pastas dos anos 2011, 2012 e 2013, o "fotos" e o "Pictures". Dando um "Enter" nisso teremos o seguinte:

```
guilherme@ubuntudesktop:~$ rmdir -p Pictures/fotos/201{1,2,3}/trime
stre{1,2,3,4}
rmdir: failed to remove directory ‘Pictures/fotos/2011’: Directory
not empty
rmdir: failed to remove directory ‘Pictures/fotos/2011’: Directory
not empty
rmdir: failed to remove directory ‘Pictures/fotos/2011’: Directory
not empty
rmdir: failed to remove directory ‘Pictures/fotos’: Directory not e
mpty
rmdir: failed to remove directory ‘Pictures/fotos/2012’: Directory
not empty
rmdir: failed to remove directory ‘Pictures/fotos/2012’: Directory
not empty
rmdir: failed to remove directory ‘Pictures/fotos/2012’: Directory
not empty
rmdir: failed to remove directory ‘Pictures/fotos’: Directory not e
mpty
rmdir: failed to remove directory ‘Pictures/fotos/2013’: Directory
not empty
```

Ele nos diz que falhou em remover o que pedimos, não conseguiu apagar o 2011, o 2012 e 2013. Vamos conferir no *explorer* se realmente não funcionou? Ao abrir o explorer e clicarmos em nossa pasta "fotos" vemos que ele apagou os anos "2011", "2012 e "2013". Por que ele fala, então, que não foi capaz de apagar as coisas que queríamos?

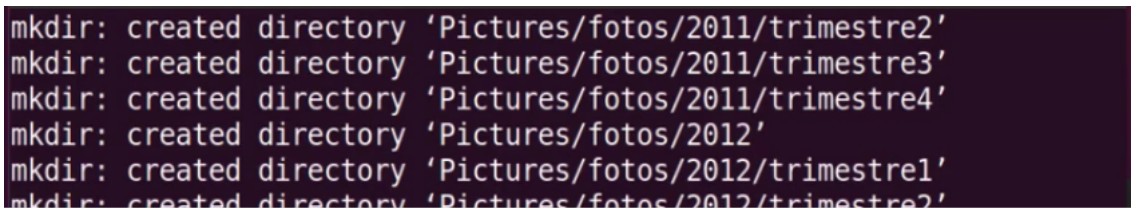
Vamos dar um `man rmdir` e ver o que ele nos mostra. Podemos analisar que existe uma opção chamada `-v`, de `--verbose`, e essa opção é muito interessante de ser utilizada para mostrar mais informações para nós.



Muitos comandos possuem um `-v` na hora de copiar, mover e remover, é bem comum que tenhamos um `-v` para sabermos, exatamente o que aconteceu. Então, é exatamente isso que queremos fazer. Vamos remover o diretório "fotos" mais uma vez e agora vamos recriar os diretórios usando o `mkdir` e vamos utilizar o `-v` para que ele nos diga o que está acontecendo. Digitaremos:

```
mkdir -p Pictures/fotos/201{1,2,3,4,5}/trimestre{1,2,3,4}
```

E dando um "Enter" vemos que ele nos mostra todos os diretórios que foram criados:



Vamos, agora, tentar remover o que acabamos de criar? Para isso podemos digitar

```
rmdir -p -v Pictures/fotos/201{1,2,3}/trimestre{1,2,3,4}
```

ou

```
rmdir -pv Pictures/fotos/201{1,2,3}/trimestre{1,2,3,4}
```

A única diferença entre o primeiro e o segundo é que neste último o `p` se junta ao `v`. Dando um "Enter" nisso serão removidos o que pedimos. Teremos o seguinte:

```
not empty
rmdir: removing directory, 'Pictures/fotos/2012/trimestre4'
rmdir: removing directory, 'Pictures/fotos/2012'
rmdir: removing directory, 'Pictures/fotos'
rmdir: failed to remove directory 'Pictures/fotos': Directory not e
mpty
rmdir: removing directory, 'Pictures/fotos/2013/trimestre1'
rmdir: removing directory, 'Pictures/fotos/2013'
```

O que ele tentou fazer?

Repare que na primeira linha ele fala que tentou remover o diretório `Pictures/fotos/2011/trimestre1` e isso funcionou, na segunda linha ele fala que tentou remover `Pictures/fotos/2011` , mas não conseguiu pois, não removeu, ainda, os diretórios "trimestre2", "trimestre3" e "trimestre4". Vamos continuar lendo, ele fala, na sequência, que consegue remover o `Pictures/fotos/2011/trimestre2` , mas na linha seguinte ele volta a tentar remover o `Pictures/fotos/2011` , mas sem ainda ter apagado os diretórios "trimestre3" e "trimestre4". Avançando mais um pouco ele fala que consegue apagar o "trimestre3" e que, novamente, dá erro ao tentar apagar o diretório 2011 e por fim, ele chega ao "trimestre4" e, finalmente, é capaz de apagar o diretório "2011". Na sequência da leitura veremos a mesma coisa acontecer quando ele tenta apagar o "fotos", assim, a mesma situação que se repetirá com o "2012", "2013" e etc.

Repare que o `-v` mostra diversas informações úteis para entendermos o que o comando executou de verdade.

Quando digitamos o seguinte comando `rmdir -pv Pictures/fotos/201{1,2,3}/trimestre{1,2,3,4}` o *globbing* na verdade faz o seguinte:

```
rmdir -pv Pictures/fotos/2011/trimestre1 Pictures/fotos/2011/trimestre2 Pictures/fotos/2011/trimestre3 //...
```

E por aí vai, nessa linha de cima poderíamos, ainda acrescentar todas as demais combinações. Lembre-se, o *globbing* é o *shell* que pega a nossa expressão e extrapola ela. Ele interpola os dados e os valores possíveis. Então, temos a pasta "2011" e as pastas trimestre 1, 2, 3 e 4, a pasta "2012" e as pastas trimestre 1, 2, 3 e 4 e a pasta "2013" e as pastas trimestre, 1, 2, 3 e 4 e assim por diante. Vamos digitar um "Ctrl C" para que isso não seja executado.

Repare que o *globbing* junto ao `mkdir` possui um poder muito interessante, podemos passar as chaves e as classes que queremos para executar o comando para vários padrões. As vezes, inclusive, podemos fazer o mesmo com outros comandos, junto com o *globbing*, e não apenas com o `mkdir` e o `rmdir` .

O interessante dessa sacada é criar um padrão que queremos executar para várias coisas ao mesmo tempo, coisas que serão executadas umas depois das outras, claro. Usamos o *globbing* e podemos ver o que acontece. Para vermos e tentarmos