

Passando parâmetros para o script

Transcrição

Na etapa anterior começamos a montar o script que era capaz **somente** de realizar a conversão do arquivo `amazon_aws.jpg` para o arquivo `amazon_aws.png`. Entretanto, ainda não conseguimos oferecer uma grande interação com o usuário final.

A ideia é, quando o usuário final executar o script, que ele seja capaz de passar parâmetros para serem interpretados.

Por exemplo, dentro da pasta "imagens-livros", temos o arquivo `arduino_pratico.jpg`, então o usuário poderia passar o nome desse arquivo como parâmetro para ser interpretado, assim o script se encarrega de fazer a conversão desse livro:

```
$ bash conversao-jpg-png.sh arduino_pratico
```

Perceba que dessa forma, o script converterá apenas a imagem de um único livro, pois o arquivo está estático e isso não é interessante para nós. Precisamos fazer com que o script pegue o **conteúdo** do primeiro parâmetro.

Para isso, acessaremos novamente o script com `nano conversao-jpg-png.sh` e implementaremos a tarefa de pegar o conteúdo do **primeiro** parâmetro passado pelo usuário. Para pegar o conteúdo, utilizamos o símbolo `$`.

E já que estamos falando do *primeiro* parâmetro, podemos referenciá-lo pelo número 1.

```
#!/bin/bash
```

```
convert ~/Downloads/imagens-livros/$1.jpg ~/Downloads/imagens-livros/$1.png
```

Após essas alterações, vamos fechar o arquivo com "Ctrl + x" e "y" para salvar. Vamos rodar esse script passando o nome do livro:

```
$ bash conversao-jpg-png.sh arduino_pratico
```

Aparentemente, foi feita a conversão. Vamos dar uma olhada na pasta.

Dentro da pasta "imagens-livros" encontramos o livro `arduino_pratico.png` ! Ou seja, o script funcionou corretamente! Só que, temos **vários** arquivos dentro dessa pasta prontos para serem convertidos. Será que conseguimos passar mais de 1 parâmetro ao mesmo tempo? Vamos tentar!

Voltando ao script com o comando `nano conversao-jpg-png.sh`, queremos colocar o **segundo** parâmetro a ser interpretado. Pela lógica, assim como o parâmetro 1 é referenciado pelo primeiro, o segundo parâmetro será referenciado pelo número 2!

```
#!/bin/bash
```

```
convert ~/Downloads/imagens-livros/$1.jpg ~/Downloads/imagens-livros/$1.png  
convert ~/Downloads/imagens-livros/$2.jpg ~/Downloads/imagens-livros/$2.png
```

Perceba que esse é um código repetitivo e estamos sujeitos a cometer erros mais facilmente. A partir do momento que o código apresenta sinais de repetição de um determinado comando, podemos **isolá-los** em uma constante e fazer a **referência** ao conteúdo da mesma.

```
#!/bin/bash
```

```
CAMINHO_IMAGENS=~Downloads/imagens-livros
```

```
convert $CAMINHO_IMAGENS/$1.jpg $CAMINHO_IMAGENS/$1.png
```

```
convert $CAMINHO_IMAGENS/$2.jpg $CAMINHO_IMAGENS/$2.png
```

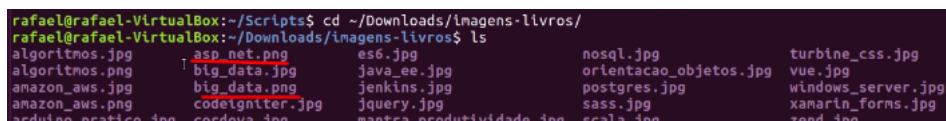
Legal! Vamos rodar esse script novamente para ver o resultado. Se tudo estiver certo, o usuário será capaz de passar 2 parâmetros simultaneamente.

```
$ bash conversao-jpg-png.sh asp_net big_data
```

Após a execução, veremos se o arquivo convertido realmente está lá.

```
$ cd ~/Downloads/imagens-livros/
```

```
$ ls
```



```
rafael@rafael-VirtualBox:~/Scripts$ cd ~/Downloads/imagens-livros/
rafael@rafael-VirtualBox:~/Downloads/imagens-livros$ ls
algoritmos.jpg      asp_net.png        es6.jpg            nosql.jpg          turbine_css.jpg
algoritmos.png      big_data.jpg       java_ee.jpg        orientacao_objetos.jpg  vue.jpg
amazon_aws.jpg      big_data.png       jenkins.jpg        postgres.jpg       windows_server.jpg
amazon_aws.png      codeigniter.jpg    jquery.jpg          sass.jpg           xamarin_forms.jpg
arduino_pratico.jpg  cordova.jpg        mantra_produtividade.jpg  scala.jpg          zood.jpg
```

Agora, o script consegue suportar 2 parâmetros que são passados simultaneamente pelo usuário!

Voltemos ao script:

```
$ cd ~/Scripts/
```

```
$ nano conversao-jpg-png.sh
```

Após observar atentamente o script, chegamos a conclusão que ele não está muito legal, pois temos duas linhas de comando que estão fazendo a mesma coisa! E como podemos melhorar o nosso script?

Veremos a seguir.