08

Status de saída

Transcrição

Conseguimos montar um script, capaz de remover a extensão `.jpg` que está presente nos arquivos originais, além de verificar se o diretório `png` existia dentro da pasta `imagens-livros`.

Dentro do laço `for`, estamos passando todo o conteúdo das imagens `.png` para o diretório `png`. Entretanto, há outra etapa a ser feita. Foi pedido para passar uma mensagem de sucesso ou de falha quando ocorresse o processo de conversão das imagens.

Analisaremos o código a seguir:

```
#!/bin/bash

cd ~/Downloads/imagens-livros
if [ ! -d png ]
then
    mkdir png
fi

for imagem in *.jpg
do
    imagem_sem_extensao=$(ls $imagem | awk -F. '{ print $1 }')
    convert $imagem_sem_extensao.jpg png/$imagem_sem_extensao.png
done
```

Basicamente, esse código está fazendo a conversão de imagens. Assim como foi visto no curso de Lógica de Programação (https://cursos.alura.com.br/course/logica-programacao-javascript-html?preRequirementFrom=shellscripting), conseguimos isolar o código em uma **função**!

No começo do script, criaremos uma função que terá o nome de ***converte_imagem()*** e englobará todo o código. Portanto, é necessário "abrir e fechar chaves".

```
#!/bin/bash

converte_imagem(){
cd ~/Downloads/imagens-livros
if [ ! -d png ]
then
    mkdir png
fi

for imagem in *.jpg
do
    imagem_sem_extensao=$(ls $imagem | awk -F. '{ print $1 }')
    convert $imagem_sem_extensao.jpg png/$imagem_sem_extensao.png
done
}
```

Uma vez criada essa função, precisamos **invocá-la** apenas colocando o nome dela. É muito importante colocar a invocação da criação depois que ela for criada. Devido à execução dos comandos que são executados em ordem sequencial.

Repare que dentro da função `converte_imagem()` há variáveis que foram criadas, por exemplo, a variável `imagem_sem_extensão` . Uma vez que essa variável foi criada dentro da função `converte_imagem()` o conteúdo dela só deveria ser acessado dentro do escopo da função em que ela está.

Faremos um teste. Vamos ver o que acontece se pedirmos para imprimir o conteúdo da variável, estamos fora da função.

```
#!/bin/bash

converte_imagem(){
cd ~/Downloads/imagens-livros
if [ ! -d png ]
then
    mkdir png
fi

for imagem in *.jpg
do
    imagem_sem_extensao=$(ls $imagem | awk -F. '{ print $1 }')
    convert $imagem_sem_extensao.jpg png/$imagem_sem_extensao.png
done
}

converte_imagem

echo $imagem_sem_extensao
```

Salvamos e executamos esse arquivo.

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

Podemos ver, que foi impresso o nome "**zend**". **Zend** é o último arquivo que temos em nosso diretório `imagens-livros` . E quando pedimos para imprimir a variável `imagem_sem_extensao` , o último valor que ela tinha era o "zend".

Isso significa que conseguimos acessar o conteúdo da variável `imagem_sem_extensao` estando fora da função em que ela foi criada. Obviamente, não queremos que isso aconteça. E sim, que essa variável **somente** exista dentro da função `converte_imagem()` , afim de evitar o vazamento de escopo.

Então, para dizer que essa variável só pode ser acessada dentro dessa função convertemos a imagem e colocamos o termo **local**. **Local** indica que o conteúdo dessa variável, só pode ser acessado dentro do `converte_imagem()` .

```
for imagem in *.jpg
do
    local imagem_sem_extensao=$(ls $imagem | awk -F. '{ print $1 }')
    convert $imagem_sem_extensao.jpg png/$imagem_sem_extensao.png
done
```

Vamos salvar e rodar novamente esse script, para ver se conseguimos acessar o conteúdo dessa variável fora do escopo do método.

Observando o resultado vemos que nenhum dado foi impresso! Isto ocorreu devido ao uso da palavra **local** antes da variável, e assim o conteúdo só é acessado dentro da função.

Voltemos ao script. Vamos focar na tarefa que os diretores nos pediram: Se o script for executado com sucesso ou se ocorre algum problema na conversão das imagens.

Quando executamos um comando no Linux, ele emite um **status de saída**. Quando o comando é executado *com sucesso*, o status de saída é zero (`0`). Já quando ocorre falhas, esse status de saída pode variar de 1 a 255.

Nossa estratégia é verificar o status de saída da função `converte_imagem()` . Se ele for zero, ela obteve sucesso na conversão. Se for diferente de zero, ocorre alguma falha. Para isso, usaremos um `if` :

```
converte_imagem
if [ $? -eq 0 ]
then
    echo "Conversão realizada com sucesso"
else
    echo "Houve uma falha no processo"
fi
```

*** Vamos entender:**

Usamos `$?` para pegar o **status de saída** da função `converte_imagem()` . Com o `-eq` , dizemos que alguma coisa é igual a outra, no caso, comparamos o valor do status de saída com o número zero. Se for igual a zero, imprime-se "Conversão realizada com sucesso". Se *não* for zero, é porque houve algum problema, então imprime "Houve uma falha no processo". Não podemos nos esquecer que, sempre que abrimos um `if` , temos que fechá-lo também com `fi` .

Salvamos e executamos o script para verificar os resultados com as novas alterações.

Observe, a mensagem "Conversão realizada com sucesso" apareceu para nós! Então, além de conseguir a conversão, colocamos uma mensagem para o usuário dizendo que a conversão foi realizada com sucesso.

Agora, forçaremos um erro em nosso script, para ver se de fato ele está funcionando. Para simular um erro, colocamos um diretório que **não existe**.

```
#!/bin/bash

converte_imagem(){
cd ~/Downloads/IMAGENS-LIVROS
if [ ! -d png ]
```

Sabemos que não existe esse diretório, então esperamos que ele acuse um erro, esperamos que apareça a mensagem "Houve uma falha no processo", assim saberemos que o script está funcionando corretamente. Vamos salvar e rodar o script e o resultado é exatamente como esperávamos: "Houve uma falha no processo". Mas, também há uma mensagem no log na qual está sendo descrito o erro. Essa informação não é relevante para o usuário, basta ele saber que o processo deu certo ou não.

Então, vamos redirecionar essa mensagem de erro para um aquivo, assim, um administrador de sistemas poderá entender porque a conversão não foi feita. Será necessário utilizar **descritores de arquivos** que são indicadores de acesso de arquivos e recursos de entrada e saída.

Os fluxos padrões existentes em nosso programa são os de entrada, o de saída e os das mensagens de erro.

Um exemplo para o fluxo de entrada é quando digitamos no teclado. Um exemplo para o fluxo de saída é o resultado que temos. Já o fluxo das mensagens de erro é como o erro apresentado acima.

Esses fluxos padrões são **referenciados por números**.

- 0 é referenciado para entrada padrão
- 1 é referenciado para saída padrão
- 2 é referenciado para mensagens de erro

Queremos redirecionar as mensagens de erro padrão para um arquivo, para que o administrador de sistemas possa analisar com mais detalhes o que exatamente aconteceu.

Voltemos ao script utilizando o comando `nano conversao-jpg-png.sh` . Queremos redirecionar as mensagens de erro, caso ocorra alguma durante o processo, para outro arquivo.

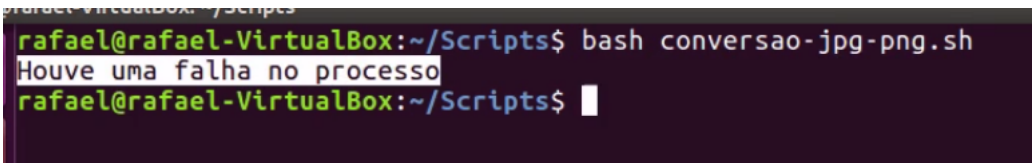
```
#!/bin/bash

converte_imagem(){
cd ~/Downloads/imagens-livros
if [ ! -d png ]
then
    mkdir png
fi

for imagem in *.jpg
do
    imagem_sem_extensao=$(ls $imagem | awk -F. '{ print $1 }')
    convert $imagem_sem_extensao.jpg png/$imagem_sem_extensao.png
done
}

converte_imagem 2>erros_conversao.txt
if [ $? -eq 0 ]
then
    echo "Conversão realizada com sucesso"
else
    echo "Houve uma falha no processo"
fi
```

Esperamos que todas as mensagens de erros, passem para o arquivo `erros_conversao.txt` e que só seja exibido a mensagem "Houve uma falha no processo". Vamos salvar e ver se de fato está tudo funcionando.



Nos é retornada a mensagem "Houve uma falha no processo". Será que temos o arquivo de texto contendo as mensagens de erro?

Estamos executando o script dentro do diretório `/Scripts` e não tínhamos referenciado nenhum outro local para salvar o arquivo `erros_conversao.txt`. Logo, ele **deve** estar presente dentro do diretório `/Scripts`. Usando o comando `ls`, conseguimos, de fato, encontrá-lo lá.

E o que encontramos é...

...O conteúdo da mensagem de erro!

Conseguimos, dessa forma, fazer as duas requisições dos diretores:

1º - fazer a conversão automática de todos os arquivos dentro do diretório `imagens-livros`, colocando todos os arquivos `.png` dentro do diretório `png`;

2º - por meio do status de saída, verificar se a função `converte_imagem()` está sendo executada com sucesso, tendo o status de saída igual a zero ou caso contrário, terá o status diferente de zero e com isso conseguimos redirecionar a mensagem de erro para o arquivo `erros_conversao.txt`.

Vamos seguir em frente para mais uma aula.