04

Chamando a função dentro dela mesma

Transcrição

Montamos um script que procura por um arquivo e faz a comparação para saber se esse arquivo é um diretório ou não. Caso ele seja um diretório, temos que acessá-lo, fazer a varredura novamente para poder verificar se o arquivo do próximo diretório é outro diretório ou um arquivo de imagem.

Perceba que, conforme acessamos os diretórios, temos que fazer esse mesmo processo várias e várias vezes!

A parte interna do bloco `if`, é exatamente igual a parte externa. E como já vimos, ficar repetindo código não é uma boa prática e nem a solução mais elegante que podemos adotar.

Vamos alterar a parte interna do código para tentar resolver esse problema. Abriremos o mesmo script, porém, utilizando o ***Gedit*** para ficar mais fácil de editar as informações.

```
$ gedit conversao-novos-livros.sh
```

Apagaremos toda a parte interna no bloco do `if` que está se repetindo.

```
#!/bin/bash

cd ~/Downloads/imagens-novos-livros
for arquivo in *
do
    if [ -d $arquivo ]
    then

    else
        #Conversao jpg para png
    fi
done
```

Se nessa parte do código, conseguíssemos de alguma forma **chamar** o bloco externo, nosso problema estaria resolvido!

Como faremos? Colocaremos o código dentro de uma função e chamaremos ela no `then` do if. Ou seja, estaremos chamando a função **dentro dela mesma** e, assim, criando uma **estrutura de recursão**!

Envolveremos esse bloco de código em uma função chamada `varrer_diretorio()`:

```
#!/bin/bash
varrer_diretorio(){
    cd ~/Downloads/imagens-novos-livros
    for arquivo in *
    do
        if [ -d $arquivo ]
        then

        else
```

```
            #Conversao jpg para png
        fi
    done
}
```

Dentro do `if`, como os passos se repetem, chamaremos a função `varrer_diretorio`. Entretanto, precisamos prestar atenção em um ponto: quando chegar a hora de chamar a função `varrer_diretorio()`, temos que passar para ela o **conteúdo da variável** `arquivo`. Para passar um parâmetro para a função é só colocá-lo após a chamada do método.

```
do
    if [ -d $arquivo ]
    then
        varrer_diretorio $arquivo
    else
        #Conversao jpg para png
    fi
done
```

Uma vez que passarmos esse parâmetro para a função `varrer_diretorio()`, faremos a varredura. Como fazemos para pegar o parâmetro assim como fizemos anteriormente? Utilizamos `$` + [numero que se refere à posição do parâmetro]!

```
#!/bin/bash
varrer_diretorio(){
    $1
    cd ~/Downloads/imagens-novos-livros
    for arquivo in *
    do
        if [ -d $arquivo ]
        then
            varrer_diretorio $arquivo
        else
            #Conversao jpg para png
        fi
    done
}
```

Então, quando a linha `varrer_diretorio $arquivo` for executada a função `varrer_diretorio()` será chamada e passará o conteúdo da variável `arquivo` como **parâmetro** para essa função.

Pegamos o conteúdo dessa variável pelo **$1**. Mas, queremos entrar nesse diretório e fazer a varredura. Na linha `cd ~/Downloads/imagens-novos-livros` já estamos alterando o diretório para `/imagens-novos-livros` e, assim, estaremos sempre entrando no diretório que vai ser passado como parâmetro no bloco do `if`.

Ao invés de colocar um caminho estático na linha do comando `cd`, deixaremos claro para sempre estar entrando no diretório que será passado pelo bloco dentro do `if`. Com essa ideia, já conseguimos resolver um grande problema.

Vamos dar uma olhada de como está a estrutura do diretório `/imagens-novos-livros`.

Logo que entramos na pasta, vemos alguns diretórios e algumas imagens. E quando acessamos alguns desses diretórios, encontramos outros diretórios, um dentro do outro.

Para não perder a referência dos locais desses arquivos e diretórios é importante que tenhamos o caminho completo deles!

Voltemos ao terminal para realizar alguns testes.

Então, queremos *encontrar o caminho completo* de um diretório ou de um arquivo que está dentro do diretório `/imagens-novos-livros`. Usaremos o comando `find` passando uma dica de onde queremos que ele procure o arquivo ou diretório.

Sabemos que, as imagens dos livros estão espalhadas em vários diretórios, mas todos eles tem `/imagens-novos-livros` como **diretório pai**!

Então, vamos falar para o comando `find`, para que ele encontre essa informação dentro da home (`~/Downloads/imagens-novos-livros`).

Dessa forma, estamos falando pra ele encontrar esse diretório ou esse aquivo de imagem que vai estar dentro de `/imagens-novos-livros`. E qual é o nome desse arquivo? Usamos o comando `-name` para passar o nome do arquivo ou diretório. Pegaremos como exemplo o diretório `/java_basico`, que está dentro de `/java`, que por sua vez está dentro de `/backend`, que está dentro de `/imagens-novos-livros`, que está dentro de `/Downloads`, que está, por fim, dentro de `/home`.

Veremos se esse comando mostrará o caminho completo da pasta `java_basico`

```
$ find ~/Downloads/imagens-novos-livros -name java_basico
```

O que temos é justamente a impressão desse caminho completo! Veja o que foi exibido:

```
/home/rafael/Downloads/imagens-novos-livros/backend/java/java_basico
```

Com isso, não perderemos mais a referência de onde o diretório está localizado. Vamos copiar esse comando, para colar em nosso script.

Quando o laço `for` do script estiver fazendo a varredura, podemos pegar o caminho completo da variável `arquivo`.

```
#!/bin/bash

varrer_diretorio(){
    cd $1
    for arquivo in *
    do
        find ~/Downloads/imagens-novos-livros -name $arquivo
        if [ -d $arquivo ]
        then
            varrer_diretorio $arquivo
        else
            #Conversao jpg para png
        fi
    done
}
```

Trocamos a variável estática `java_basico` pelo conteúdo de `arquivo`, assim podemos pegar o caminho **completo** do `arquivo`. Armazenaremos o **resultado** dentro da variável `caminho_arquivo`. Não podemos nos esquecer de colocar todo esse comando entre `$()`, assim estamos dizendo que este é um comando que deve ser executado e que o seu resultado será armazenado na variável `caminho_diretorio`.

```
#!/bin/bash

varrer_diretorio(){
    cd $1
    for arquivo in *
    do
        caminho_arquivo=$(find ~/Downloads/imagens-novos-livros -name $arquivo)
        if [ -d $arquivo ]
        then
            varrer_diretorio $arquivo
        else
            #Conversao jpg para png
        fi
    done
}
```

Como a variável está dentro da função `varrer_diretorio()` dizemos que ela é uma **variável local**, assim, evitamos o vazamento de escopo, como já visto. Agora, vamos fazer toda referência dos diretórios e dos arquivos pelo caminho completo deles, já que temos a variável local `caminho_arquivo`.

```
if [ -d $caminho_arquivo ]
then
    varrer_diretorio $caminho_arquivo
else
    #Conversao jpg para png
fi
```

Dessa forma, estamos passando o caminho completo do diretório ou arquivo, *sem perder a referência*.

O que nos resta agora é tratar da parte de converter as imagens de `.jpg` para `.png`. Criamos uma função que será responsável somente por converter as imagens. Vamos chamá-la de `converte_imagem`.

Como boa prática, sempre temos que colocar a função **antes** dela ser invocada.

```
#!/bin/bash

converte_imagem(){



}

varrer_diretorio(){
    cd $1
    for arquivo in *
    do
        caminho_arquivo=$(find ~/Downloads/imagens-novos-livros -name $arquivo)
```

```
        if [ -d $caminho_arquivo ]
        then
            varrer_diretorio $caminho_arquivo
        else
            converte_imagem
        fi
    done
}
```

Quando o bloco do `else` for executado é porque o conteúdo da variável `$caminho_arquivo` **não é um diretório**! Se ele não for um diretório, então ela só pode ser uma imagem! Precisamos passar essa imagem para a função `converte_imagem()`.

```
if [ -d $caminho_arquivo ]
then
    varrer_diretorio $caminho_arquivo
else
    converte_imagem $caminho_arquivo
fi
```

Uma vez que o parâmetro é passado para o `converte_imagem()`, temos que pegá-lo, da mesma forma que fizemos com `varrer_diretorio()`.

Para pegar o parâmetro, utilizamos `$1`, mas, antes, colocaremos o `$` em uma variável local, prezando pela legibilidade do código:

```
#!/bin/bash

converte_imagem(){
    local caminho_imagem=$1
}

varrer_diretorio(){
    // laço de repetição
}
```

Agora que temos o caminho completo da imagem, precisamos **remover** a extensão `.jpg` do caminho da imagem. Faremos essa tarefa com a ajuda do **`awk`**! Adicionaremos outra variável local chamada `imagem_sem_extensao`, para que seja armazenado nessa variável o resultado do comando a seguir:

```
converte_imagem(){
    local caminho_imagem=$1
    local imagem_sem_extensao=$(ls $caminho_imagem | awk -F. '{ print $1 }')
}
```

Nos resta realizar a conversão das imagens.

Queremos converter o conteúdo da variável `$imagem_sem_extensao` utilizando o comando `convert`. Vamos converter para o formato `.png`.

```
converte_imagem(){
    local caminho_imagem=$1
    local imagem_sem_extensao=$(ls $caminho_imagem | awk -F. '{ print $1 }')
    convert $imagem_sem_extensao.jpg $imagem_sem_extensao.png
}
```

O script está quase completo. Só nos resta chamar a função `varrer_diretorio()` passando o diretório pai, para iniciar todo o processo de varredura.

```
#!/bin/bash

converte_imagem(){
    local caminho_imagem=$1
    local imagem_sem_extensao=$(ls $caminho_imagem | awk -F. '{ print $1 }')
    convert $imagem_sem_extensao.jpg $imagem_sem_extensao.png
}

varrer_diretorio(){
    cd $1
    for arquivo in *
    do
        local caminho_arquivo=$(find ~/Downloads/imagens-novos-livros -name $arquivo)
        if [ -d $caminho_arquivo ]
        then
            varrer_diretorio $caminho_arquivo
        else
            converte_imagem $caminho_arquivo
        fi
    done
}

varrer_diretorio ~/Downloads/imagens-novos-livros
```

É necessário conferir se o script realmente conseguiu fazer a conversão ou se teve algum problema. Verificaremos o **status de saída** da execução! Se ele for **zero**, é porque tudo foi realizado com sucesso e se for **diferente de zero**, houve algum problema, Vamos colocar essa mensagem no terminal:

```
varrer_diretorio ~/Downloads/imagens-novos-livros
if [ $? -eq 0 ]
then
    echo "Conversão realizada com sucesso"
else
    echo "Houve um problema na conversão"
fi
```

Vamos testar esse script, esperamos que ele funcione como esperado. Utilizamos o "Ctrl + x" para sair e "y" para salvar as alterações e executamos o script!

```
$ bash conversao-novos-livros.sh
```

Após instantes, obtemos o seguinte resultado:

```
Conversao realizada com sucesso
```

Quando abrimos o diretório, nos deparamos com os arquivos `.jpg` e seus respectivos arquivos `.png`. Em todas as pastas, vamos encontrar esses arquivos.

A tarefa de montar um script que realizasse a conversão das imagens espalhadas em vários diretórios foi concluída com sucesso!