09

Conclusão

Transcrição

Parabéns! Você chegou ao final do curso de Shell Scripting parte 1!

Nesse curso, recebemos a tarefa de converter todas as imagens do diretório `/imagens-livros` de `.jpg` para `.png`. Ao decorrer do nosso treinamento, criamos vários cenários. Vimos como passar parâmetros para o script utilizando o **$1** e **$2**. Vimos como pegar todos os parâmetros com o **$@**.

Ao construir o script, o primeiro passo foi acessar o diretório `/imagens-livros` a partir do comando **cd**. Verificamos se nesse diretório, já existia o diretório `png`, onde nós salvamos os arquivos convertidos. Caso ele não existisse, ele seria criado.

Em seguida, também vimos o laço de repetição `for`, onde usamos para fazer a varredura dentro de um diretório. Pegamos todas as imagens com a extensão `.jpg`, e com a ajuda do comando `awk`, retiramos a extensão, assim o que sobrava eram somente os nomes dos arquivos, sem a sua extensão.

Utilizamos o ***imagemagick*** para fazer a conversão da extensão `.jpg` para `.png`, salvando essas imagens no diretório `/png`.

Vimos também, sobre guardar informações de uma função. Para invocar essa função, é importante que a função sempre seja declarada primeiro, e depois podemos invocá-la.

Abordamos a questão dos **descritores de arquivos**, o valor zero para a *entrada padrão*, o valor um para a *saída padrão* e o valor dois para o *erro padrão*. Então, através do parâmetro `2>`, enviamos as mensagens de erro para um arquivo chamado de `erros_conversao.txt`. Também vimos como saber se uma tarefa foi executada com sucesso, através do **status de saída**.

Em uma outra situação, precisávamos realizar a conversão de várias imagens em diferentes diretórios. Para isso, utilizamos a **recursividade**, ou seja, chamar a função dentro dela mesma! Então, nós isolamos, em uma função, a parte do código que faz a varredura, e caso encontrássemos um outro diretório, chamaríamos a função novamente. Quando era encontrado uma imagem, chamávamos a função que realiza a conversão da imagem.

Também abordamos o uso do status de saída, mandando uma mensagem de sucesso ou de fracasso para o usuário.

Na última etapa, vimos os processos com maior quantidade de memória alocada. Organizamos esses processos em 10 posições, utilizando o comando `--sort -size` que era baseado no tamanho. Pegamos 11 linhas em que a primeira era o cabeçalho `HEAD`. E com o `grep`, utilizamos somente as linhas com números.

Utilizamos o comando `bc`, para nos ajudar na tarefa de converter o arquivo em kilobytes para megabytes. Salvamos todo o conteúdo no diretório `/log`. Cada arquivo nesse diretório, contém informações como a data, o horário, e a quantidade de memória alocada em **megabytes**.

Esse foi um resumo do que vimos na primeira parte do curso Shell Scripting. Já na segunda parte veremos como fazer a monitoração no servidor da Apache, a partir de um conteúdo web, onde faremos essa monitoração constante. Caso esse conteúdo web pare de ser acessado pelos usuários, veremos como notificar o administrador, e depois realizaremos um *backup* das informações do banco de dados, e por fim, mandaremos para o serviço de *cloud* da **Amazon**.

Até mais e bons estudos!