Introdução à Computação - 2023-2 27/11/2023

Laboratório 6: Shell Script - Parte 1

Professor: Diego da Silva de Medeiros
Baseado no material do Prof. Eraldo. diegomedeiros@ifsc.edu.br

1 Objetivos

1. introdução ao shell script;
2. criar e executar um script;
3. trabalhando com variáveis;

2 Criar e executar um shell script

Nota: Um shell script é uma sequência de comandos que você poderia executar na linha de comando mas que serão repetidos eventualmente. Neste sentido, coloca-se os comandos em arquivo e executamos o script como se fosse um comando único.

1. Chame o um editor de texto qualquer e crie um arquivo para ser o seu primeiro script:

```
gedit meu_script.sh
```

2. Entre com o seguinte texto, salve e saia do editor:

```
#!/bin/bash

echo Olá - Vou criar um diretório
mkdir meus_scripts
echo Pronto! Diretório criado. Vou colocar um arquivo lá dentro
touch ./meus_scripts/alfa.txt
ls -l ./meus_scripts
```

3. Atribua direito de execução:

```
chmod u+x ./meu_script.sh
```

4. Teste o script:

```
./meu_script.sh
```

5. Verifique se tudo foi criado devidamente.
6. DESAFIO: Faça um script para criar a seguinte árvore de diretórios e arquivos:

```
docs/relatorios/alfa.txt
    |           /aalfaa.txt
    |           /gama.alfa.txt
    |           /adendos/lixo1.txt
    |                   /.lixo2.txt
    |
    |fichas/gama.txt
```

```
|        /epson.txt
|        /.mu.txt
|
|imagens/foto1.jpg
        /foto2.jpg
```

3 Variáveis e Parâmetros

Nota: Muitas vezes será necessário ter áreas de memória para armazenar informações que possam ser manipuladas pelo script. Podemos então criar variáveis.

- Crie e execute o script abaixo:

```
#!/bin/bash

DIR_PROJETO=/home/aluno/proj1
mkdir $DIR_PROJETO
echo Olá - Vou criar um arquivo dentro do diretorio de projeto
touch $DIR_PROJETO/beta.txt

echo Ok $USER ! Tarefa completa
echo Voce está no diretório $PWD
echo O seu home é $HOME
```

Nota: Observe que existem variáveis que são criadas pelo shell e outras pelo próprio usuário. É o caso da svariáveis PWD, USER e HOME.

- A variável PATH é muito importante pois o seu conteúdo indica onde serão procurados os comandos a serem executados:

  `echo $PATH`

- Adicione o seu diretório de entrada a variável PATH:

  `PATH=$PATH:/home/aluno`

- Agora vamos ver um exemplo de passagem de parâmetro em linha de comando. Construa o seguinte shell script:

```
#!/bin/bash

echo Olá! Seus parâmetros são:
echo Primeiro: $1
echo Segundo: $2
echo Terceiro: $3
```

- Agora execute-o da forma:

  `./meu_script alfa beta gama`

4 Execução condicional - comando IF

Nota: Por vezes é necessário testar uma condição e caso ela seja verdadeira executar um ou mais comandos.

- Antes de mais nada vamos identificar que todo comando possui um valor de retorno que indica se o mesmo se executou corretamente ou não. Este retorno fica armazenado em uma variável chamada $?. Note que de acordo como o comando se executa a saída é zero ou um.

```
cat /etcccc/passwd
echo $?
cat /etc/passwd
echo $?
```

- Vamos agora usar este retorno em comandos condicionais com IF. Implemente e teste o script abaixo:

```
#!/bin/bash

#testa se o usuário passado como parâmetro existe. Se não existe ...

grep $1 /etc/passwd
if [ $? -eq 0 ]; then
    echo usuário existe
else
    echo usuário não existe
    echo temos que criá-lo
fi
```

- Para executar o script anterior, deve ser passado um parâmetro. Se ele não for passado pode-se obter resultados inesperados. Para contornar podemos testar a quantidade de parâmetros (variável $#). Note que o operador ! funcionando como negação da operação.

```
#!/bin/bash

if [ ! $# -eq 1 ]; then
   echo opa! Algo errado...
   echo uso: script parametro
   exit
fi

grep $1 /etc/passwd
if [ $? -eq 0 ]; then
    echo usuário existe
else
    echo usuário não existe
    echo temos que criá-lo
fi
```

5 Aninhamento de comandos IF e Múltiplas cláusulas

1. Observe o script a seguir. Note que o if é tratado como um comando qualquer podendo estar aninhado em outros comandos ifs.

```
#!/bin/bash

if [ -d $1 ]; then
     echo diretório "$1" existe
     if [ -d $1/alfa ]; then
```

```
        echo diretório  "$1/alfa" existe
    else
        echo o diretório "$1/alfa" não existe, vou criá-lo...
        mkdir $1/alfa
    fi
else
    echo diretório "$1" NÃO existe - vou desistir...
fi
```

2. Observe que múltiplas cláusulas podem ser admitidas em um if:

```
#!/bin/bash

FILE="$1"

if ! [ -a $FILE ]; then
    echo o arquivo $FILE não existe
elif [ -d $FILE ]; then
    echo o arquivo é diretório
elif [ -f $FILE ]; then
    echo o arquivo é regular
else
    echo qualquer outra opção ...
fi
```

6 Expressões com conectores lógicos

1. Usando conectores OU lógico (||) e E lógico (&&)

```
#!/bin/bash

FILE="$1"

if [[ -f $FILE || -d $FILE ]]; then
    echo o arquivo é diretório ou regular
else
    echo qualquer outra opção ...
fi
```

7 DESAFIOS

Nota: Consulte a url `http://www.tldp.org/LDP/Bash-Beginners-Guide/html/sect_07_01.html` para os exercícios a seguir.

(a) Fazer um shell script para detectar se um arquivo passado como parâmetro é um link simbólico. Testar.

(b) Fazer um shell script para detectar se um arquivo tem permissão para escrita (para usuário owner). Caso não tenha então acrescentar o direito de escrita para este arquivo.

(c) Fazer um shell script para verificar se um dado arquivo é regular e se possui direito de leitura. Caso tenha estes atributos renomear para um nome de arquivo também passado como parâmetro.