Shell Script do zero

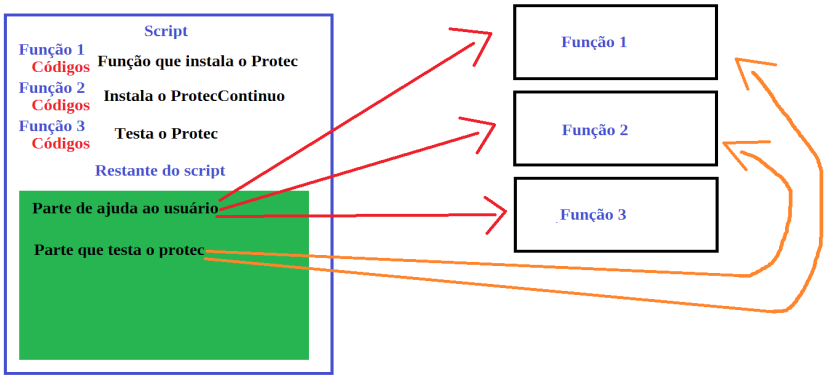
Aula 11 – Função e Parâmetros

Função

O papel da função é segmentar vários códigos no scripts tornando-os acessíveis a qualquer momento e em qualquer parte, usar a função é como colocar vários scripts num só script. Um exemplo prático disto é o protec onde eu tinha muitas opções a serem executadas de várias partes do script (o case não resolveria).

Para quem não conhece o protec (não deixa o protetor de tela ficar entrando nos nossos vídeos), é só baixá-lo no link → http://www.mediafire.com/download/rrdrwl0ceicz8y9/protec.zip

No desenho ao lado podemos reparar que temos 3 funções e elas são acessadas dentro dos códigos de ajuda e teste.



Não seria viável fazê-lo com o case porque eu teria que repetir os códigos em várias partes do script e o que eram quase 1000 linhas se tornariam 1500 ou mais.

Na parte de ajuda em determinado momento eu perguntarei ao usuário se ele deseja instalar o Protec, ProtecContinuo ou testar, usando o valor da variável vou comparar no if ou case executando assim a função que desejamos. Então no desenho apresentado podemos visualizar com as setas para onde o shell pode ir e da mesma forma poderíamos criar acessos de vários pontos para vários pontos.

Entenda que podemos fazer uma “salada mista” com os comandos que vimos até aqui, while dentro de while, if dentro de if e dentro deste if um while etc. Não fique com receio de fazer estes tipos de códigos porque muitas das vezes é esta “bagunça” que deixa o nosso script mais fácil de montar

Sintaxe

A sintaxe da função será sempre a mesma, como mostrado abaixo, primeiro o nome dela (no meu exemplo “instala”) depois dois parênteses sem espaço dentro e uma chave apontando para esquerda. Nas linhas abaixo você coloca os códigos a serem executados na função e quando terminar coloque outra chave apontando para direita fechando a função.

```
instala(){

# Aqui dentro os codigos a serem executados na função

}
```

Depois de declarar a função ao shell vamos chamá-la, para fazer isto basta escrever o nome da função na linha desejada, no exemplo abaixo o usuário já instalou o Protec e se desejar instalá-lo para outro player deve digitar a letra “i”, assim cai no if que tem uma linha chamando a função de nome “instala” o que faz ele “pular” de onde está indo para outra parte do script.

```
echo "Digite 'i' para instalar em outro player ou 's' para sair"
read prox

if [ "$prox" = "i" ];then
   instala
fi
```

A função deve ser mostrada ao shell antes de ser chamada, vamos supor, eu tenho uma função no meio do script e resolvo chamá-la no começo, não vai funcionar porque o shell não leu ainda, eu sempre coloco as funções no começo do script assim dá menos trabalho

Parâmetros

Os parâmetros são valores armazenados em variáveis pré-determinadas, estes valores são fornecidos quando o usuário digita o comando que chama o programa/script pelo terminal, depois de digitar o “chamador” damos espaço e digitamos o primeiro parâmetro e assim sucessivamente conforme mostrado abaixo.

```
luiz@linuxmint ~ $ ./script parametro1 parametro2
```

Ou na prática → abaixo eu tenho o script “back” que em seu código dá um cp (copiar) no primeiro parâmetro e colando no segundo parâmetro.

```
luiz@linuxmint ~ $ ./back /home/luiz /Dropbox
```

Sem saber usamos os parâmetros no terminal, o comando apt-get install firefox, apt-get é o comando do programa, install e firefox são os parâmetros, internamente o apt compara o valor do primeiro parâmetro para saber se instala, remove etc. E depois ele pega o segundo parâmetro que será o objeto a ser tratado.

É claro que o apt é muito mais do que isso e temos comandos gigantescos com vários parâmetros que eu não faço ideia da explicação, porém se você pegou a noção de parâmetros já é o suficiente.

Se o usuário não digitar os parâmetros o script vai rodar, o que pode acontecer é o mesmo dar pau por falta de dados

Sintaxe

Nós temos os parâmetros $1 $2 $3 $4 $5 $6 $7 $8 $9, então colocamos ele no script não para receber valor, mas já usando os valores que serão digitados no terminal.

```
#!/bin/bash

echo $1 $2 $3 $4 $5 $6 $7 $8 $9
```

Mais do que 9 o shell começa a fazer leituras erradas

O script anterior demonstra o funcionamento dos parâmetros, se o usuário digitar o comando do script e do lado colocar → Linux, então o script executará o comando echo mostrando a palavra Linux, se ele digitar “Linux e livre” serão 3 parâmetros e o shell executará o echo até o terceiro parâmetros, e assim sucessivamente.

As aplicações são infinitas, eu poderia usar o valor de um parâmetro para determinar a quantidade de loops, para colher um nome de usuário, para receber valores a serem calculados etc. O script abaixo por exemplo, dorme pelo tempo determinado pelo usuário e depois mata o programa indicado também pelo usuário.

```
#!/bin/bash

sleep $1

killall $2
```

O usuário daria um comando parecido com este → ./script 100 firefox

Tratamos os parâmetros como as variáveis, só tendo em mente que o seu valor será dado pelo usuário antes de iniciar o script

Exercício 7 – Função e parâmetro no mesmo script

- O usuário entra no programa utilizando o seu nome como parâmetro
- Aparecerá a tela de boas vindas constando o nome deste usuário
- Ele faz um cadastro (poucos dados só para simular, cpf é obrigatório)
- Ele é jogado para a segunda parte (de compra), nesta parte é pedido o cpf novamente e se o mesmo estiver errado (de acordo com o que foi cadastrado), o script volta para cadastrar o cpf novamente

Resolução → http://www.mediafire.com/download/r58wm59p4fnlzb0/cpf

Bons estudos e procure entender a lógica de programação mais do que decorar comandos