Resumo Importante sobre o Curso de LPIC-1

Tópico 103.1 - Trabalhando na Linha de Comando – Shell, Bash, Echo, Type, Path

Douglas Luna - @DipionX02

O que estudamos nesta aula?

SHELL

O shell é a ligação entre o usuário e o sistema. É ele quem interpreta os comandos entrados para outros aplicativos ou diretamente em chamadas de sistema. Além disso, os recursos do shell são indispensáveis para lidar com muitos arquivos ao mesmo tempo, para realizar uma tarefa repetidamente ou para programar uma ação para determinada ocasião, entre outros recursos. Começamos apresentando os tipos de shell mais difundidos e depois definindo alguns conceitos que serão úteis na sua utilização, para então tratarmos de exemplos práticos.

FUNÇÕES DO SHELL

- Analisar dados a partir do prompt (dados de entrada);
- Interpretar comandos;
- Controlar ambiente Unix-like (console);
- Fazer redirecionamento de entrada e saída;
- Execução de programas;
- Linguagem de programação interpretada.

TIPOS DE SHELL (OS PRINCIPAIS)

Bourne Shell

É o shell padrão para Unix, ou seja, a matriz dos outros shells, portanto é um dos mais utilizados. É representado por "**sh**". Foi desenvolvido por Stephen Bourne, por isso Bourne Shell.

Korn Shell

Este shell é o Bourne Shell evoluído, portando todos os comandos que funcionavam no Bourne Shell funcionarão neste com a vantagem de ter mais opções. É representado por "**ksh**".

C Shell

É o shell mais utilizado em BSD, e possui uma sintaxe muito parecida com a linguagem C. Este tipo de shell já se distancia mais do Bourne Shell, portanto quem programa para ele terá problemas quanto a portabilidade em outros tipos. É representado por "**csh**".

Bourne Again Shell (FOCO)

É o shell desenvolvido para o projeto GNU usado pelo GNU/Linux, é muito usado pois o sistema portador dele evolui e é adotado rapidamente. Possui uma boa portabilidade, pois possui características do Korn Shell e C Shell. É representado por "**bash**". **O FOCO DE ESTUDO É ESTE SHELL – BASH.**

VERIFICAR QUAL É O SHELL

Para verificar qual SHELL estamos usando, basta dar o comando abaixo, ele irá imprimir na tela o caminho do shell bash "/bin/bash":

- Comando: echo $SHELL
- Resultado: /bin/bash
- Ou seja, o Shell que estamos usando é o BASH.

O comando echo

$ echo $SHELL

```
kali@kali:~$ echo $SHELL
/bin/bash
```

echo – O comando serve para imprimir informações na tela. Em conjunto com o símbolo de redirecionamento de saída **>>**, estudado mais à frente, também é utilizado para **concatenar** (ou seja, adiciona a informação no arquivo, sem retirar as já existentes) informações para dentro de um arquivo. Exemplo:

$ echo "Este é o curso de lpi-1" >> /home/kali/Documents/teste.txt

```
kali@kali:~$ echo "Este é o curso de lpi-1" >> /home/kali/Documents/teste.txt
kali@kali:~$ cat /home/kali/Documents/teste.txt
"Este é o curso de lpi-1"
kali@kali:~$
```

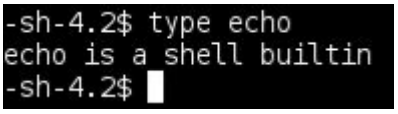
No exemplo acima, introduzimos a frase no arquivo teste.txt, mesmo o arquivo não existindo no diretório, ele é criado automaticamente ao executarmos o comando.

O comando type

No Linux existem comandos que são internos do shell, que já são embutidos no programa SHELL e também comandos que são externos ou migrados de outras aplicações que estão em nosso sistema. Para sabermos se um comando é de origem SHELL, usamos o comando **type**, exemplo:

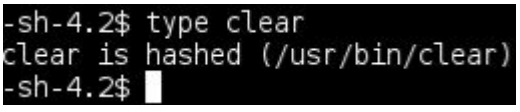
```
$ type echo
```

Após esse comando ele nos retornará uma mensagem nos dizendo se é ou não um comando interno do shell, ou seja, já está em sua compilação de origem.



Quando fazemos o type apontando para outro comando que é externo, ele nos mostra a localização do comando:

Um comando mesmo que externo após digitado muitas vezes o Linux nos cria um cache interno como é o caso do comando “clear”. O comando “clear” é um comando externo.
Após algumas vezes digitado ele passa a ser hashed. “cacheado” pelo sistema Linux:



O comando PATH

O PATH serve para indicar para o sistema o caminho dos comandos externos dentro do Linux. Quando damos o comando abaixo ele nos mostra os diretórios em que os programas e comandos estão localizados:

```
$ echo $PATH
```

Quando um comando é acionado ele procura em cada diretório onde está o programa correspondente. Os diretórios estão separados por "**:**".

```
kali@kali:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
kali@kali:~$
```

1º - /usr/local/bin
2º - /usr/bin
3º - /bin
4º - /usr/local/games
5º - /usr/games

Há também formas de chamar um comando quando ele não é interno e nem está presente dentro dos diretórios dos programas externos acima. Para fazer isso podemos fazer das seguintes formas:

Usaremos o arquivo do exercício disponibilizado no curso como é mostrado na aula:

Caminho absoluto

Dessa forma ele executará normalmente, pois indicamos o path exato de onde está o programa que queremos executar.

```
$ /home/kali/Exercicios/Script_Exemplo.sh
```

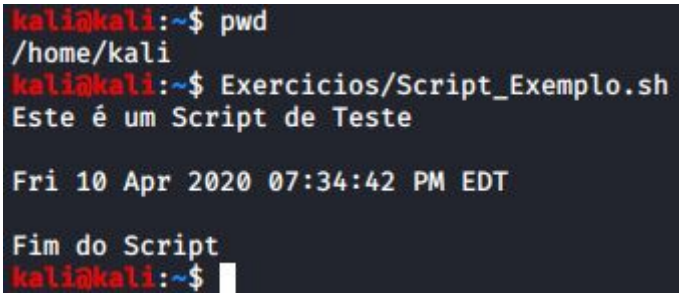
```
kali@kali:~$ /home/kali/Exercicios/Script_Exemplo.sh
Este é um Script de Teste

Fri 10 Apr 2020 07:31:18 PM EDT

Fim do Script
kali@kali:~$
```

Caminho Relativo ou Parcial

Neste caso, se você já estiver em uma pasta que está no mesmo caminho que o script que deseja executar, basta continuar o comando até o caminho do script. Perceba que, estamos dentro da pasta do usuario **/home/kali**, podemos conferir com o comando "**pwd**", o script se encontra dentro de **Exercicios/Script_Exemplo.sh** que já é parte do caminho do script:

```
$ pwd
```

```
$ Exercicios/Script_Exemplo.sh
```



Dentro do diretório

Neste caso, usaremos "**./** " **que quer dizer: neste diretório**. Para executar o comando dessa forma temos que estar dentro do diretório em que o script se encontra, e lá, fazemos da seguinte forma para termos sucesso no comando:

```
$ ./Script_Exemplo.sh
```