Programação Shell Script

REDES DE COMPUTADORES

Prof. Dr. João Paulo Aramuni

UNIVERSIDADE FUMEC

Revisão

* **Bourne Shell**
  * Nosso foco é no shell padrão do UNIX chamado Bourne shell.
  * Durante o desenvolvimento do UNIX, Dennis Ritche e Ken Thompson desenvolveram um shell muito simples.
    * O primeiro shell, que continha as principais funcionalidades, foi desenvolvido nos anos 70 por Stephen Bourne (Apenas – sh).

Revisão

* **Scripts**
  * É a funcionalidade mais interessante do shell.
  * São as **estruturas sofisticadas** de programação citadas anteriormente.
  * Scripts **são arquivos** que possuem dentro uma lista de comandos que você deseja executar.

Revisão

* **Scripts**
  * Cada script está escrito dentro de um arquivo, e cada arquivo possui um nome diferente...
  * Isso permite que exista uma **combinação** entre eles para que novos programas sejam gerados.
  * ...tudo a fim de se **resolver um problema**.

Primeiros Passos

* Como ***escrever*** Shell Scripts?

Escrevendo Shell Scripts

* Para escrevermos um shell script corretamente devemos seguir as seguintes regras abaixo:

  * 1) Utilizar um **editor de texto** para escrever o shell script.
  * 2) Configurar **as permissões** de modo que o shell possa executá-las.
  * 3) Colocar o script onde o shell possa **encontrá-lo**.

Escrevendo Shell Scripts

* Utilizando um **Editor de Textos** Linux

Editores de Texto:

Nome	Descrição	Interface
vi, vim	É o grande pai dos editores de texto do UNIX. Conhecido pela sua complexidade no aprendizado, ele é o mais poderoso, leve e rápido de todos os editores	Linha de Comandos – Modo Texto
emacs	O Emacs é considerado por muitos o editor de texto mais poderoso que existe. Sua base em Lisp, especificamente num dialeto de Lisp chamado Emacs Lisp, permite que ele se torne configurável ao ponto de se transformar em uma ferramenta de trabalho completa, uma espécie de "canivete suíço" para escritores, analistas e programadores. Existe uma briga eterna e saudável entre os editores vi e emacs	Linha de Comandos – Modo Texto
nano	Nano's ANOther editor ou Not ANOther editor é um editor de texto simples para o console, desenvolvido para substituir ou, ao menos, oferecer uma alternativa ao popular Pico, o editor padrão do cliente de correio Pine. O nano possui fácil utilização, mas ele é muito limitado em recursos.	Linha de Comandos – Modo Texto
gedit	Editor de textos distribuído em conjunto com o GNOME	Gráfica
kwrite	Editor de textos distribuído em conjunto com o KDE	Gráfica

Escrevendo Shell Scripts

* Qual editor de textos iremos utilizar?
  * ***vi, vim***

Meu primeiro Shell Scripts

* primeiro_script.sh
  * Abra o editor de textos vi ou vim e escreva o seguinte texto abaixo:

```
$ vi primeiro_script.sh
#!/bin/bash
#Meu primeiro Shell Script
clear
echo "Conhecimento eh Poder!"
```

* Tente executar este shell script utilizando o comando "**./**"

Escrevendo Shell Scripts

* Como configurar as Permissões para executar um shell script?

primeiro_script.sh

Escrevendo Shell Scripts

* Configurando as permissões:
  * Sintaxe:
    * **chmod permissão <nome_script>**
  * Exemplos:
    * **- $ chmod +x <nome_script>**
    * **- $ chmod 755 <nome_script>**

Escrevendo Shell Scripts

* Configurando as permissões:
  * O comando chmod reconhece letras e números:

    * **1 = execução**
    * **2 = escrita**
    * **4 = leitura**

    * **x = execução**
    * **r = leitura**
    * **w = escrita**

Escrevendo Shell Scripts

* Configurando as permissões:
  * Somando-se temos as permissões:
  * Exemplo: **755**
    * **7** – permissão total para o dono do arquivo
    * **5** – leitura e execução para o grupo do dono do arquivo
    * **5** – leitura e execução para outros

Executando meu Primeiro Shell Script

* Neste ponto já podemos executar nosso primeiro programa:
  * **1º Forma** (Somente após as permissões)
    * $ ./primeiro_script .sh
  * **2º Forma** (Funciona mesmo sem **setar** as permissões)
    * $ sh primeiro_script.sh

Executando meu Primeiro Shell Script

* Como informar ao shell o local onde ele pode encontrar o(s) nosso(s) script(s)?

Como o shell localiza os arquivos executáveis?

* O shell mantém uma *lista de diretórios* onde os arquivos *executáveis* são armazenados.
* Se ele precisar executar qualquer programa ou comando, ele apenas procura se o mesmo existe nessa lista.
* Se ele não o encontrar, uma mensagem de erro será exibida (“command not found”)

Como o shell localiza os arquivos executáveis?

* Essa lista de diretórios é chamada de *path*.
  * Você poderá visualizar a lista de diretórios contidos no *path* com o seguinte comando:
    * **$echo $PATH**

Como o shell localiza os arquivos executáveis?

* Ok, mas...
  * Como fazer para o shell encontrar a *minha lista de scripts*?

Como o shell localiza os arquivos executáveis?

* Precisamos apenas incluir o nosso diretório de scripts no:

$PATH

Configurando o $PATH

* Exercício:
  * Crie um diretório chamado **meus_scripts** na sua pasta de usuário.
  * Mova o arquivo **primeiro_script** para essa pasta.
  * Inclua este diretório no **path**.

$ export PATH=$PATH:diretório

Exercício 2

* Digite o seguinte shell script, rode e veja o resultado.
* Descreva para que serve este script.
* Pesquisa o que faz cada um dos comandos.

```
#!/bin/bash
clear
echo "Hello $USER"
echo "Today is"; date
echo "Number of user login:"; who | wc -l
echo "Calendar"
cal
exit 0
```

Obrigado.

Contato: joaopauloaramuni@gmail.com

UNIVERSIDADE FUMEC

50 Anos