Programação Shell Script

REDES DE COMPUTADORES

Prof. Dr. João Paulo Aramuni

UNIVERSIDADE FUMEC

Execução Condicional

* Execução Condicional
* Os operadores de controle

Os operadores de controle

* Os operadores de controle são:

&& (AND) e || (OR)

Os operadores de controle

* ***AND***
* Ex: comando1 && comando2
  * O comando 2 é executado se, e somente se, o comando 1 retorna sucesso (status de saida = 0)

* ***OR***
* Ex: comando1 || comando2
  * O comando 2 é executado se, e somente se, o comando 1 retorna falha (status de saida != 0)

Os operadores de controle

* Também podemos utilizar ambos em conjunto:

Os operadores de controle simultâneos

* Sintaxe:
* comando1 && comando2 (se o status de saida=0) || comando3 (se o status de saida!=0)

  * Se o comando1 for executado com sucesso, então o shell irá executar o comando2
  * … mas se o comando1 falhar, ou seja, não for executado com sucesso, então o shell irá executar o comando3

Os operadores de controle simultâneos

* Exemplo:

* $ rm arq && echo "Arquivo removido com sucesso" || echo "O arquivo nao foi removido"

Redirecionamento de E/S (I/O)

* Como vocês já sabem, os redirecionadores de E/S são utilizados para mandar a saída de um comando para um arquivo

* ... ou para ler sua entrada.

Redirecionamento de E/S (I/O)

* Exemplo:

$ cat > arq

Este eh meu arquivo aluno

^D (pressione CTRL + D para salvar o arquivo)

* O comando acima envia a saída do comando cat para o arquivo "arq"

Redirecionamento de E/S (I/O)

* Exemplo 2:

  $ cal > meucalendario

* O comando cal envia seu resultado (saída) para o arquivo "meucalendario"

* Isto é o que chamamos de redirecionamento de saída!

Redirecionamento de E/S (I/O)

* Exemplo 3:

```
$ sort
5
7
2
3
^D
2
3
5
7
```

* O comando sort lê a "entrada" do teclado e ordena os números imprimindo o resultado, ou seja, enviando a saída para a tela.

Redirecionamento de E/S (I/O)

* Vamos supor que queremos ler as entradas de um arquivo (para o comando sort) e não do teclado.

* O que fazer?

Redirecionamento de E/S (I/O)

* Exemplo 4:

```
$ cat > nums
5
7
2
3
^D
```

* Digite:

```
$ sort < nums
```

Redirecionamento de E/S (I/O)

* Primeiro criamos o arquivo "nums" utilizando o comando "cat".
* Em seguida, o arquivo nums foi utilizado como entrada para o comando sort
* … o qual imprimiu na tela os números deste arquivo, ordenados!

* Isto é o que chamamos de redirecionamento de entrada!

Descritores de Arquivos

* No Linux (e na Linguagem de programação C) seu teclado, monitor e etc... são tratados como se fossem arquivos comuns.

Descritores de Arquivos

Nome do Arquivo	Num. do Descritor de Arquivo	Utilização	Exemplo
stdin	0	Como entrada padrão	Teclado
stdout	1	Como saída padrão	Monitor
stderr	2	Como erro padrão	Monitor

Descritores de Arquivos

* No Linux, por padrão, todo programa possui três arquivos associados a ele.
* Quando rodamos um programa esses três arquivos são automaticamente abertos pelo nosso shell.
* Já vimos como utilizar entrada/saída padrão (stdin/stdout).

Descritores de Arquivos

* O terceiro arquivo, stderr (descritor de arquivo: 2) é utilizado pelos programas para imprimir erros na tela.
* Podemos redirecionar a saída de qualquer descritor de arquivo diretamente para um arquivo.
* Sintaxe:

  numero-descritor-arquivo>nomedoarquivo

Descritores de Arquivos - Exemplo

* Vamos assumir que o arquivo "arqdefeituoso" não existe

  $ rm arqdefeituoso

  * rm: cannot remove `arqdefeituoso': No such file or directory

* O comando acima retorna um erro como saída para a tela e não para um arquivo, pois não o direcionamos para nenhum arquivo.

Descritores de Arquivos

* Se quisermos direcionar o erro para um arquivo?
* Se tentássemos:
* $ rm arqdefeituoso > erro (Funciona?)

Descritores de Erros

* $ rm arqdefeituoso > erro (Funciona?)

* **Não.** O erro continua sendo mandado para a tela ao invés do arquivo.

* Tente visualizar o conteúdo de "erro", você verá que continua vazio, Por que?

Descritores de Erros

* Porque o erro esta sendo enviado para o dispositivo de erros, stderr, que utiliza o monitor como saída

* ... e você não pode redirecioná-lo direto para o arquivo.

Descritores de Erros

* Para resolver este problema, informe o descritor de arquivo referente aos erros antes de redirecionar, veja abaixo

$ rm arqdefeituoso 2>erro

  * Não existe espaços entre o descritor 2 e o sinal de >.
  * O 2>erro direciona a saída do erro padrão (stderr) para o arquivo "erro".
  * O numero 2 é o numero do descritor de arquivo referente ao stderr.

Descritores de Erros

* Vamos entender um exemplo um pouco mais elaborado.

```
#!/bin/bash
#arq: demonstracao
if [ $# -ne 2 ]
then
   echo "Erro: Numero de argumentos insuficiente"
   echo "Utilize: $./<script> numero1 numero2"
   exit 1
fi
resp=`expr $1 + $2`
echo "A Soma eh: $resp"
//-----------------------------------------
```

* Para rodar o exemplo acima utilize:

```
$ chmod 755 demonstracao
```

Exemplos de Execução

* Exemplo1

  $ ./demonstracao
  Erro: Numero de argumentos insuficiente
  Utilize: $./<script> numero1 numero2

* Exemplo2

  $ ./demonstracao > erro1 ???

* Exemplo3

  $ ./demonstracao 7 9
  A Soma eh: 16

Entendendo os Exemplos

* Exemplo1:
  * O script imprimiu a mensagem de erro na tela indicando que os 2 números não foram passados como argumento
* Exemplo2: Nada foi impresso na tela.
  * A mensagem de erro que supostamente deveria ser mostrada ao usuário no terminal foi direcionada para o arquivo erro1.
  * Isto quer dizer que o script esta mandando as mensagens para a saída padrão (**stdout**) e não para o erro padrão (**stderr**) como deveria ser
  * **...** pois **não deveria ser possível o redirecionamento para um arquivo** (Lembrar da remoção do arqdefeituoso, visto anteriormente).

Direcionamento de erros

* Para resolver este problema vamos substituir as linhas das mensagens de erro do script conforme abaixo

  * **echo** "Erro: Numero de argumentos insuficiente" **1>&2**
  * **echo** "Utilize: $./<script> numero1 numero2" **1>&2**

Direcionamento de erros

* Se rodarmos agora:

  $ ./demonstracao > erro1

  Erro: Numero de argumentos insuficiente

  Utilize: $./<script> numero1 numero2

* ou seja...
  * Agora nosso script esta enviando as mensagens de erro para o erro padrão (**stderr**) como deveria.
  * O 1>&2 no final do comando echo direciona a saída padrão (**stdout** (1)) para o erro padrão (**stderr** (2))

Direcionamento de erros

* Sintaxe:

de>&para

* ou

origem>&destino

Funções

* Quando um programa começa a ficar complexo a solução é apelar para a técnica "Dividir para conquistar".

* Isto significa dividir o programa em pequenos blocos de código conhecidos como Funções.

Funções

* As funções são um conjunto de instruções/comandos.

* Toda função deve ser implementada para realizar uma determinada atividade ou tarefa.

* Para definir uma função:
* Sintaxe:

```
nome-da-funcao ( )
{
    comando1
    comando2
        .....
         ...
    comandoN
    return
}
```

* Obs.: O comando return é utilizado para terminar a função.

Exemplo:

* Digite no prompt de comando a função abaixo

```
$ DigaAlo()
{
   echo "Alo $LOGNAME, Tenha um bom aprendizado"
   return
}
```

* Para executar a função DigaAlo(), digite apenas DigaAlo no prompt de comando conforme abaixo:

```
$ DigaAlo
Alo Aluno, Tenha um bom aprendizado
```

Obs.: Lembre-se que a função apenas será valida para essa sessão. Para manter as funções em todas as suas sessões você deve inserir o código da função no arquivo /etc/bashrc

Funções

* Passagem de parâmetro para as Funções.

Passagem de Parâmetros para as Funções

* Podemos passar parâmetros para funções através da linha de comandos da mesma maneira que passamos argumentos para os nossos scripts.

Passagem de Parâmetros para as Funções

* Como já sabemos, as funções em shell script devem ser declaradas da seguinte maneira:

```
function nome-da-funcao( )
{
    comando1
    comando2
    comandoN
}
```

Passagem de Parâmetros para as Funções

* Podemos chamar a função com:

  nome-da-funcao arg1 arg2 arg3 argN

* ou sem parâmetros:

  nome-da-funcao

Passagem de Parâmetros para as Funções

* Exemplo:

```
#!/bin/bash
#arq: testefuncao
#função para testar a passagem de parâmetros

function funcao1()
{
   echo "Todos os argumentos da funcao1(): $*"
   echo "Primeiro Argumento: $1"
   echo "Segundo Argumento: $2"
   echo "Terceiro Argumento: $3"
   return
}

#
# CHAMADA DA FUNCAO
#
funcao1 -f teste aluno
```

Passagem de Parâmetros para as Funções

* Exemplo 2:

```
#!/bin/bash
#arq: testefuncao2
#função para testar a passagem de parâmetros
function funcao2()
{
    n1=$1 #PRIMEIRO OPERANDO
    op=$2 #OPERADOR
    n2=$3 #SEGUNDO OPERANDO
    resp=0 #RESULTADO
if [ $# -eq 3 ]; then
    resp=$(( $n1 $op $n2 ))  #CONVERSAO DA OPERACAO PARA INTEIRO
    echo "$n1 $op $n2 = $resp"
    return $resp
else
    echo "A funcao2() necessita de pelo menos 3 argumentos"
fi
    return
}
funcao2 $1 $2 $3
funcao2 5 + 10
funcao2 10 – 2
echo $?
```

Obrigado.

Contato: joaopauloaramuni@gmail.com

UNIVERSIDADE FUMEC

50 Anos