PROGRAMAÇÃO EM SHELL SCRIPT

**CARACTERISTICAS DO SHELL**

* Programa de usuário
* Interpreta programas
* Oferece linguagem de programação
* Pode ser interativo ou não
* Pode-se ter shell recursivo
* Tipagem **fraca e dinâmica**

**TIPOS PRINCIPAIS DE SHELL**

|  |  |
| --- | --- |
| **sh** | Bourn Shell |
| **ksh** | Korn Shell |
| **csh** | C Shell |
| **bash** | Bourne Again Shell |

Para instalar um Shell usar o padrão da distribuição ex. *apt install  
 yum install*

|  |  |
| --- | --- |
| **PRINCIPAIS PALAVRAS RESERVADAS** | |
| ! | Pepelines |
| [ ] | Conditional Constructs |
| { } | Command Grouping |
| case | Conditional Constructs |
| do | Looping Constructs |
| done | Looping Constructs |
| elif | Conditional Constructs |
| else | Conditional Constructs |
| esac | Conditional Constructs |
| for | Looping Constructs |
| function | Shell Functions |
| if | Conditional Constructs |
| in | Conditional Constructs |
| select | Conditional Constructs |
| then | Conditional Constructs |
| time | Pepelines |
| until | Looping Constructs |
| while | Looping Constructs |

**VARÁVEIS PRÉ-DEFINIDAS**

|  |  |
| --- | --- |
| $? | Armazena o status de saída do último programa executado |
| $# | Armazena a quantidade de parâmetros de linha de comandos |
| $$ | Armazena o valor PID (Process Identifier) do script em shell que estiver em execução |
| $@ | Armazena o valor de todos os parâmetros passados |
| $! | Armazena o PID do último processo em segundo plano |
| $0 … $9 | Armazena os valores de todos os parâmetros de linha de comando separadamente. |

**VARIAVEIS GLOBAIS**

Variáveis globais ou variáveis de ambiente globais, são variáveis criadas/definidas com o comando export e podem ser utilizadas por múltiplos scripts em Shell.

|  |  |
| --- | --- |
| PATH | define diretórios de procura por programas executados no shell |
| USER | informa o nome do usuário do shell |
| HOME | informa o caminho do diretório home do usuário |
| LANG | Idioma/Linguagem, especificada como locale |
| PWD | diretório atual |
| TERM | Tipo de terminal atual em uso |
| UID | UID do usuário atual |
| RANDOM | Gera um número aleatório |

*env* – mostra as variáveis de ambiente  
As variáveis são definidas sem espaços. Ex: *teste=’teste’* e são exibidas com $  
*echo $teste*  
*export teste* para tornar a variável global  
*unset teste* para desalocar uma variável  
Todas as variáveis são globais por definição a não ser que os comandos “*local*” ou *declare*” sejam usados

|  |  |  |
| --- | --- | --- |
| **CARACTERES ESPECIAIS** | | |
| Aspas Duplas | " " | Ignora caracteres especiais exceto **$** e **`** |
| Apóstrofos | ' ' | Ignora todos os casacteres |
| Contrabarras | \ | Ignora o caractere logo a seguir |

O comando *local* só pode ser usado dentro de uma função, sendo assim a variável tem escopo restrito a função.



Um Script pode ser criado e editando usando o vi ou nano por exemplo. Também pode ser criado usando o comando touch.

Ex: touch teste.sh  
 vi teste.sh

No vi: digitando ***i*** entra no modo de inserção  
 ***esc*** sai deste modo  
 ***:wq*** sai do editor de texto

Dentro do script a primeira linha deve definir o shell a ser usado. Ex.  
***#!/bin/bash***

Para que o script seja executado é necessário dar a permissão de execução através do ***chmod*** . Ex.  
***chmod 755 teste.sh***

Para executar basta indicar o local e o script a ser executado. Ex.  
***./teste.sh***

**VARIÁVEIS**

Podem ter até **20 caracteres** e é **case sensitive**

Não se usa espaços entre o nome da variável e sua atribuição

Para exibir é preciso usar o $ antes do nome da variável

*texto=”variavel tipo string”  
numero=23  
echo $texto  
echo $numero*

**ARRAYS**

Variáveis com múltiplos valores  
Um array é declarado dentro de parênteses e os elementos são separados por um espaço em branco, se usa aspas duplas para elementos com espaços dentro do texto.  
Para exibir um array é preciso inseri-lo dentro de chaves e indicar o índice do elemento a ser exibido dentro de colchetes ou usar o caractere asterisco dentro dos colchetes.

Arrays são dinâmicos, ou seja, basta adicionar um posição para aumentar o array

*testearray=(A 1 abc “um espaço”)  
 echo ${testearray[\*]}  
 testearray[2]=C*

Para desalocar o espaço de memória de uma variável usa-se o ***unset***

*unset=testearray*

O comando *exit* especifica o status de saída quando o script termina

**Editores** de texto são diferentes de **Processadores** de texto.

O comando *set* pode ser usado para exibir variáveis locais e seu valor:

Ex: set | grep $testeVarLocal

**STATUS DE SAÍDA**

|  |  |
| --- | --- |
| Vai de 0 a 255 | |
| 0 | Sucesso |
| 1 | Erro desconhecido |
| 126 | Não executavel ou sem permissão |
| 127 | Não encontrado |
| 128+n | Sinal fatal de número n |
| 130 | Finalização forçada |

As **crases ` ` (backticks)** são usadas para o shell identificar que é um comando a ser executado e prioriza-lo

**Alias** é um ‘apelido’ para abreviar um comando ou uma sequência de comandos:

*alias clr=clear*

no exemplo é criado o **alias *clr*** que será equivalente ao comando ***clear***

**Unalias** remove um alias criado e se o ***alias*** for digitado sozinho exibirá os já configurados

*alias lshome=’cd /home; ls’*

neste, ***lshome*** é a sequência que se desloca até o diretório ***home*** e lista o seu conteúdo.

|  |  |  |
| --- | --- | --- |
| **Redirecionamento de Saída** | | |
|  | Como Funciona | Sintaxe |
| > | Saida de um comando cria um arquivo novou ou sobrescreve | *ls>log.txt* |
| >> | cria ou escreve a saida do comando no fim do arquivo | *ls>>log.txt* |
| 2> | cria ou sobrescreve os erros de um comando | *grep 2> log.txt* |
| 2>> | cria ou escreve na ultima linha o erro de um comando | grep 2>> log.txt |

|  |  |  |
| --- | --- | --- |
| **Redirecionamento de Entrada** | | |
| < | Sinaliza entrada de um arquivo especificado | *wc < log.txt* |
| << | here document' sinaliza que o escopo de um comando começa na linha seguinte e termina no **label** que segue o << | *wc <<* ***EoF*** |
| **Redirecionamentos Especiais** | | |
| | | pipe' direciona a saida de um comando para a entrada de outro | *ls -l | less* |
| tee | Envia a saida de um comando para arquivo e para tela ao mesmo tempo | *ls | tee log.txt* |

|  |  |
| --- | --- |
| **Operadores de comparação numerica** | |
| -eq | equal |
| -ge | greater than of equal |
| -gt | greater than |
| -le | less than or equal |
| -lt | less than |
| -ne | not equal |
| **Operadores de comparação de strings** | |
| = | identicas |
| != | diferentes |
| > | Maior ou menor em relação a ordem alfabetica |
| < |
| -n | Verifica se o comprimento da string é maior que zero |
| -z | Verifica se a string é vazia |
| **Operadores Lógicos** | |
| && | AND |
| || | OR |

|  |  |
| --- | --- |
| **Operadores Aritméticos** | |
| + | Adição |
| - | Subtração |
| \* | Multiplicação |
| / | Divisão |
| % | Módulo da divisão inteira |
| \*\* | Exponenciação |
| **Operadores de Atribuição** | |
| = | Atribui um valor a variavel |
| += | incremente por uma constante |
| -= | decrementa por uma constrante |
| \*= | multiplica por uma constante |
| /= | divide por uma constante |
| %= | resto da divisao por uma constante |
| ++ -- | Incremento ou decremento de 1 |
| **Operadores de deslocamento de BIT** | |
| << | Desloca o bit para esquerda preenchendo com zero |
| >> | Desloca o bit para a direita preenchendo com zero |

|  |  |
| --- | --- |
| **Operadores Relacionais** | |
| == | igual a |
| != | diferente de |
| > | maior que |
| < | menor que |
| >= | maior ou igual a |
| <= | menor ou igual a |
| **Operadores de atribuição de BIT** | |
| <<= | Deslocamento à esquerda |
| >>= | Deslocamento à direita |
| &= | AND de BIT |
| |= | OR de BIT |
| ^= | XOR de BIT |
| **Operadores lógicos de BIT** | |
| & | AND de BIT |
| | | OR de BIT |
| ^ | XOR de BIT |
| ~ | negação de BIT |
| ! | NOT de BIT |

O comando ***test***  testa o conteúdo de uma string e retorna um valor lógico escrito na variável **$?** Sendo **0** para retorno **verdadeiro** ou **1** para retorno **falso.** Pode ser usado como   
*teste 1 = 1; echo $? Ou [ 1 = 1 ]; echo $?*

|  |  |
| --- | --- |
| **Parâmetros aplicados ao *test*** | |
| -b arquivo - | Verdadeiro se o arquivo é um block device. |
| -c arquivo - | Verdadeiro se o arquivo é um character device. |
| -d arquivo - | Verdadeiro se o arquivo é um diretório. |
| -e arquivo - | Verdadeiro se o arquivo existe. |
| f1 -ef f2 - | (equivalent file) Verdadeiro se o arquivo f1 é um hard-link para f2. |
| n1 -eq n2 - | (equal) Verdadeiro se n1 for igual a n2. |
| -f arquivo - | Verdadeiro se o arquivo existe e é um arquivo regular. |
| n1 -ge n2 - | (greater than or equal) Verdadeiro se n1 maior ou igual a n2. |
| n1 -gt n2 - | (greater than) Verdadeiro se n1 for maior que n2. |
| -g arquivo - | Verdadeiro se o SGID estiver ativado. |
| -G file - | Verdadeiro se o arquivo (efetivamente) pertencer ao grupo sob qual o script estiver rodando. |
| -h arquivo - | Verdadeiro de o arquivo for um link simbólico. |
| -k arquivo - | Verdadeiro se o arquivo tiver a permissão sticky bit ativada. |
| n1 -le n2 - | (less than or equal) Verdadeiro se n1 menor ou igual a n2. |
| n1 -lt n2 - | (less than) Verdadeiro se n1 menor que n2. |
| -n s (or just s) | (not null) Verdadeiro se a string não for vazia ou nula. |
| -N arquivo - | Verdadeiro se o arquivo tiver novo conteúdo (desde a última vez que foi lido). |
| n1 -ne n2 - | (not equal) Verdadeiro se n1 não for igual a n2. |
| -O file - | Verdadeiro se o arquivo pertencer ao usuário sob o qual o script estiver rodando. |
| f1 -nt f2 | (newer than) Verdadeiro se o arquivo f1 for mais recente que f1. |
| f1 -ot f2 | (older than) Verdadeiro se o arquivo f1 for mais antigo que f1. |
| -p file - | Verdadeiro se o arquivo for um pipe. |
| -r file - | Verdadeiro se um arquivo for lido (pelo seu script). |
| -s arquivo - | Verdadeiro se o arquivo existe e não estiver vazio. |
| -S arquivo - | Verdadeiro se o arquivo for um socket. |
| -t fd - | Verdadeiro se o descritor do arquivo estiver aberto em um terminal. |
| -u arquivo - | Verdadeiro se o arquivo tiver set-user-id (suid) ativada. |
| -w arquivo - | Verdadeiro se o arquivo tiver permissão de escrita (pelo seu script). |
| -x arquivo - | Verdadeiro se o arquivo for executável. |
| -z s - | (zero length) Verdadeiro se a string for igual a zero ou não possuir nenhum valor atribuído. Ex. variável nula ou arquivo vazio. |

**USANDO O VIM**

O editor vi é um dos mais utilizados para programar em Shell Script, duas das 3 formas de operar o vi são, **modo de comando** e **modo de inserção**

Quando se executa o vi com um arquivo ele entra por padrão no modo de comando e para entrar no modo de inserção pode-se usar o **a** ou **i**.

Para sair do modo de inserção usa-se o **esc**. Em teclados que não possuem essa tecla, pode ser substituída por: **^[** , **tecla <meta>** **<alt> <diamond> <start> <menu>** , **^3** , **F11**.

Alguns comando são:

*:w* grava e continua a edição

*ZZ* grava o texto e sai do vi

*:wq* grava e sai do vi

*:wq!* grava mesmo sendo read only

*:q* sai sem salvar mas alerta

*:q!* força saída sem salvar

*h* desloca cursor para esquerda

*j* desloca cursor para baixo

*k* desloca cursor para cima

*l* desloca cursor para direita

**EXPRESSÕES REGULARES**

Método formal para especificar um padrão de texto

Metacaracteres são combinações de caracteres com funções específicas

Um exemplo é:

[rgp]ato pode casar com ‘rato’, ‘gato’ e ‘pato’