Laboratório de Sistemas Operacionais II – Aula prática 02 - 16/08/2016

Trabalhando com Comandos Condicionais (IF)

1. **Introdução** a Comandos Condicionais. Ler o item 6.0 na página 7 do Manual_Unix_3 (programação Shell).

2. **Criar** uma pasta para a aula de hoje. Chame de Aula2Shell (***mkdir Aula2Shell***).

3. **Conhecer** o funcionamento do **Código de Retorno** de uma instrução – associada à variável ***$?***
   Se a instrução é executada com sucesso ***$?*** é igual a Zero (***Sucesso***), caso contrário o valor retornado será diferente de Zero (***Insucesso***).
   Exemplos: (Estes testes você faz no prompt do linux mesmo).

ls –l Aula2Shell	- mostra os detalhes da pasta Aula2Shell
echo $?	- Lista o conteúdo da variável de retorno
0	- Retornou 0 porque Aula2Shell existe
ls –l Lampada	- mostra os detalhes da pasta Lampada
echo $?	- Lista o conteúdo da variável de retorno
1	- Retornou 1 porque Lampada não existe

4. **Conhecer** o comando ***PIPESTATUS*** - Resultado do **pipe**:

Quando se executa diversos comandos encadeados em um ***pipe*** (|), o código de retorno dado por echo ***$?*** Reflete apenas o resultado de saída do último comando executado no pipe.

O array ***PIPESTATUS*** armazena em cada elemento o resultado respectivo de cada um dos comandos do pipe. ***PIPESTATUS[0]*** tem o retorno do primeiro comando, ***PIPESTATUS[1]*** do segundo, e assim por diante.

Exemplos: O comando ***PIPESTATUS*** é escrito em maiúscula.

date | grep Wed | wc –l
echo ${PIPESTATUS[*]}
0 1 0

Obs.:

(1) O comando ***date | grep Wed | wc –l*** faz o seguinte: O ***date*** pega a data do Sistema. Ao encontrar o ***pipe***, ele transfere a data para o comando ***grep***. O comando ***grep*** procura na data recebida pela palavra ***Wed*** (Quarta). Se achar ou não achar, ele manda o resultado para o comando ***wc*** porque ele encontrou mais um ***pipe***. O comando ***wc*** conta quantas linhas ele recebeu do comando grep. Será 0 se não recebeu nenhum ou será 1 se ele recebeu um, ou seja, se hoje for uma quarta-feira.

(2) O comando ***echo ${PIPESTATUS[*]}*** mostrará os status de todos os comandos linux, neste caso, ***date, grep e wc***.

(3) O resultado ***0 1 0*** indica que: O comando ***date*** deu sucesso (0), porque sempre mostra uma data. O comando ***grep*** deu insucesso (1) porque o conjunto de comandos certamente foi dado em qualquer dia, menos uma quarta-feira. O comando ***wc*** deu sucesso (0), porque ele sempre conta algo, indiferente se recebeu uma informação ou não.

5. **Executar** o conjunto:
   date | grep Wed | wc –l
   echo ${PIPESTATUS[*]}

   No lugar do ***** (asterisco), coloque ***0*** para ver o resultado do ***date***. Depois coloque ***1*** para ver o resultado do ***grep***, e finalmente coloque ***2*** para ver o resultado do ***wc***.

6. Seguir para a pasta criada Aula2Shell (***cd Aula2Shell***).

7. **Explorar** o comando ***IF*** – Controle de decisão:
   O comando IF tem a seguinte sintaxe:

```
if      <condição>          - Aqui deve haver uma condição
then
        <comando1>          - Se condição for verdadeira, serão
        <comando2>            executados comandos 1, 2, etc.
        < ..........  >
else
        <comando3>          - Se condição for falsa, serão
        <comando4>            executados comandos 3, 4, etc.
        < ..........  >
fi
```

   Obs.:
   - As palavras reservadas ***then*** e ***fi*** serão sempre obrigatórias.
   - A palavra reservada ***else*** existirá se houver uma segunda ação para a mesma condição.

8. **Criar** script usando ***cat > PS11.ksh*** ou ***vi PS11.ksh***.

```
#!/bin/bash
#
# Verifica se determinado Usuário está logado
#
if who | grep $1
then
        echo $1 está logado
else
        echo $1 não está logado
fi
```

Para executar o programa, coloque permissão de acesso (***u+x***) e execute-o da seguinte forma:

./PS11.ksh fsa33333

Ao executar, coloque o nome da pessoa que você deseja saber se está logada logo após o nome do programa.

(1) O comando ***if who | grep $1*** fará o seguinte: O ***who*** obtém a lista de todos que estão logados no sistema neste momento. Ao encontrar o ***pipe***, ele repassa a lista para o comando ***grep***. O comando ***grep*** com o parâmetro $***1*** vai procurar pelo nome que você digitou junto com o programa na lista repassada pelo comando ***who***. O resultado será listado na sua tela.

9. **Criar** script usando ***cat > PS12.ksh*** ou ***vi PS12.ksh***.

```
#!/bin/bash
#
# Verifica se determinado telefone pesquisado existe ou não
# Caso não exista, solta mensagem de aviso
#
grep  $1  ../telefones.txt
var=`echo $?`
if  (($var=="1"))
then
        printf "Telefone do $1 não Encontrado"
else
        printf "Telefone do $1 Encontrado"
fi
```

Para executar o programa, coloque permissão de acesso (***u+x***) e execute-o da seguinte forma:

./PS12.ksh Pedro

Se não encontrar, use uma dos nomes da tabela e execute novamente.

Obs.:

(1) O comando ***grep $1 ../telefones.txt*** faz o seguinte: procura pelo nome que você digitou na lista de telefones. Veja que a lista de telefones deve estar no diretório raiz, como feito na aula anterior. Ele pode tanto achar (sucesso = 0) ou não achar (insucesso = 1).

(2) O resultado, que está na variável do sistema ***$?*** será mandada para a variável ***var*** do programa.

(3) Leia na página 8 do ***Manual_Unix_3*** sobre ***Operadores Relacionais***. Esses operadores serão utilizados nas condições dos comandos ***if***.

(4) O comando ***var =`echo $?`*** não possui espaço nem antes nem após o sinal de **=**. Quando var for referenciado no ***if***, ele recebe um $ (***$var***).

10. **Verificando** o conteúdo de uma variável.

Criar script usando ***cat > PS13.ksh*** ou ***vi PS13.ksh***

```
#!/bin/bash
#
# Verifica o conteúdo de uma variável digitada
#
if expr $1 + 1 > /dev/null 2> /dev/null
then
        echo $1 é um numero
else
        echo $1 não é um número
fi
```

Executar: ***./PS13.ksh Carlos*** e depois testar ***./PS13.ksh 234***

Obs.:

(1) O comando ***expr*** auxilia o programa nas operações aritméticas. Neste caso, a parte do comando ***expr $1 +1*** está somando uma constante 1 à variável recebida quando você digitou algo junto com o nome do programa.

(2) A segunda parte do comando ***> /dev/null 2> /dev/null*** manda o resultado para esta pasta temporária.

(3) Veja que se você executar uma variável não numérica o programa avisará.

11. **Conhecendo** o comando ***test***

A sintaxe do comando ***test*** é:

test <expressão>

Obs. Sendo <expressão> a condição que se deseja testar.

Exemplo:

Criar script usando ***cat > PS14.ksh*** ou ***vi PS14.ksh***

```
#!/bin/bash
#
# Testa a resposta a um pedido. Deve ser (S)im ou (N)ão
#
printf "Você me empresta o livro?"
read resp
if test $resp = N
then
        printf "Ela não me empresta....."
else
        if test $resp = S
        then
```

```
        printf "Oba! Ela me empresta."
    else
        printf "Ela está na dúvida!!!!!"
    fi
fi
```

Executar: ***./PS14.ksh***

12. **Mostrar** as formas de comparações lógicas e condicionais.

Leia na página 8 do ***Manual_Unix_3*** sobre ***Operadores Lógicos***. Esses operadores serão utilizados nas condições dos comandos ***if***.

Vamos fazer um exemplo:

Criar script usando ***cat > PS15.ksh*** ou ***vi PS15.ksh***

```
#!/bin/bash
#
# Testando com operadores lógicos
#
typeset –i num=0
printf "informe um número entre 10 e 100"
read num
if (($num >=10 && $num<=100))
then
    printf "Você digitou correto. Obrigado."
else
    printf "Você não prestou atenção no que foi pedido!!!!."
fi
```

Obs.:

(1) Execute ***./PS15.ksh*** e veja o comportamento do programa.

(2) Entenda como foi feito o comando ***if***, porque você deverá fazer vários comandos como este nos próximos programas.

13. **Criar** programa ***PS16.ksh - Peso***.

Faça o programa que atenda o seguinte:

Pedir para usuário digitar um peso.

Se peso estiver entre 60 e 80 soltar mensagem "Você está na média! Parabéns!"

Senão, soltar mensagem "Procure um médico Urgente!"

Dica:

Faça uma cópia do Programa ***PS15.ksh*** e altere o seu nome para ***PS16.ksh***. Depois, abra o programa ***PS16.ksh*** e faça as alterações que atendam as condições solicitadas acima.

Execute ***./PS16.ksh*** e veja o comportamento do programa.

Sempre teste todas as condições do programa.

14. **Criar** programa ***PS17.ksh - Notas***.

- Pedir para usuário digitar notas P1 e AT1
- Se as notas de P1 ou AT1 forem maiores que 10 soltar mensagem "Nota Inválida" e encerrar o programa.
- Senão, calcular Nota Final sendo a soma de P1 com peso 6 e AT1 com peso 4.
- Mostrar a Nota Final da seguinte forma:
- Se Nota Final >= 8 soltar "Parabéns, continue assim!"
- Se Nota Final >= 6 soltar "Está bom, mas você deve estudar mais!"
- Se Nota Final >= 4 soltar "Você está com problemas. Procure ajuda!"

- Se Nota Final >= 2 soltar "A coisa está feia. Candidato a Segunda época!"
- Senão, soltar "Vá pescar que fé melhor! "

Obs.:

(1) Para fazer cálculos você vai usar: (*) para multiplicação, (/) para divisão, (+) para soma ou (-) para subtração.

(2) Não faça cálculos compostos, somente simples. Primeiro calcule a nota da Prova, depois calcule a nota de atividade, e finalmente, obtenha a nota final.

(3) Não deixe nenhum espaço entre os componentes de uma expressão. Por exemplo:

Salário=SalarioAnterior*índice

Pagina=Pagina+1

(4) Para atender ao pedido de encerrar o programa não existe, por enquanto, um comando especial. Você deve fazer seu algoritmo de tal forma que, se soltar a mensagem de erro, ele sai após o último if.

"Eficiência e fazer o trabalho corretamente. Eficácia é fazer o trabalho que deve ser feito."

Zig Ziglar.