Miscelânea

João Marcelo Uchôa de Alencar

Universidade Federal do Ceará - Quixadá

3 de Outubro de 2019

O comando *eval*

$ eval [linha-de-comando]

```
$ cano='|'
$ ls $cano wc -l
ls: cannot access '|': No such file or directory
ls: cannot access 'wc': No such file or directory
$ eval ls $cano wc -l
5
```

O comando *wait*

$ wait [PID]

```
$ sleep 60 &
[1] 3223
$ wait 3223
```

O comando *trap*

$ trap [comando1];[comando2];[comando3]; ... ;[comandoN] sinais

0 EXIT	Saída normal
1 SIGHUP	kill -HUP
2 SIGINT	CTRL + C
3 SIGQUIT	CTRL + \
15 SIGTERM	kill PID

Funções

```
funcao ()
{
    comando1
    comando2
    comando3
    ...
    comandoN
    return $var # O valor de var deve ser número
}
```

Invocação:

`funcao [parametro1] [parametro2] ... [parametroN]`

Cada parâmetro dentro da função é tratado por $1 $2 $3 ...

Funções

```
function command_not_found_handle
{
    echo "Erro na linha ${BASH_LINENO[0]}"
    exit 1
}
```

Exercícios em Sala

O objetivo é criar um *script* chamado *sistema.sh* para permitir monitorar o desempenho de um servidor Linux.

- Ele deve exibir um menu com opções para o usuário.
- Ao digitar uma das opções, a tela deve ser limpa, um comando executado, o resultado exibido.
- Após o usuário apertar *enter* retornar para o menu inicial.

As opções devem ser, de acordo com `http://techblog.netflix.com/2015/11/linux-performance-analysis-in-60s.html`:

1. Tempo ligado (uptime)
2. Últimas Mensagens do Kernel (dmesg | tail -n 10)
3. Memória Virtual (vmstat 1 10)
4. Uso da CPU por núcleo (mpstat -P ALL 1 5)
5. Uso da CPU por processos (pidstat 1 5)
6. Uso da Memória Física (free -m)
7. Sair

Use funções para criar uma função chamada *menu* para exibir as opções.

O comando *mkfifo*

```
# No terminal 1
$ mkfifo cano
$ ls > cano
# No terminal 2
$ cat < cano
cano  sistema.sh  trap.sh
```

Permite comunicação e sincronização entre processos.

O comando *getopts*

```
$ getopts [cadeia de opções] variável
```

- Se o *script* aceita as opções -a -b -c, então a cadeia de opções deve ser *abc*.
- Se uma das opções tem argumento, : deve ser colocado depois da letra, a:bc
- A variável irá armazenar qual opção está sendo tratada, enquanto OPTARG contém o valor passado.

O comando *getopts*

```
$ cat getopts_exemplo.sh
#!/bin/bash
while getopts "a:bc" OPTVAR
do
    echo $OPTVAR $OPTARG
done
$ ./getopts_exemplo.sh -a 1 -b -c
a 1
b
c
$ ./getopts_exemplo.sh -b -c -a 1
b
c
a 1
```

Execução Passo a Passo

Muitas vezes não é fácil encontrar o erro em *Shell Scripts*.

- Uma opção é ativar a execução passo a passo.
  - Coloque *set -x* no início do trecho que deseja analisar.
  - No fim do trecho desabilite com *set +x*.
- Se você quiser listar as linhas com mais detalhes.
  - Ativar com *set -xv*.
  - Desabilitar com *set +xv*.

Exemplo de Execução Passo a Passso

```
#!/bin/bash
echo "Iniciando execução passo a passo."
set -x
echo "Flag ativado."
read -p "Informe um valor:" entrada
entrada=`expr $entrada + 1`
echo "Valor atualizado: $entrada"
set +x
echo "Flag desativado."
```

Exemplo de Execução Passo a Passso

```
#!/bin/bash
echo "Iniciando execução passo a passo."
set -xv
echo "Flag ativado."
read -p "Informe um valor:" entrada

soma=0
for i in `seq $entrada`
do
   echo "Atualizando entrada..."
   soma=`expr $soma + $i`
done

echo "Valor atualizado: $soma"
set +xv
echo "Flag desativado."
```

Exercícios em Sala

Vamos criar um *script* chamado *hosts* que nos ajude a relacionar nomes de máquinas à IPs.

- O *script* deve guardar em um arquivo chamado *hosts.db* um par (nomedamaquina,IP) para cada entrada.
- Você deve criar as seguintes funções para manipular o arquivo que são invocadas com os parâmetros indicados:
  - **adicionar** (parâmetros -a *hostname* -i *IP*)
  - **remover** (parâmetro -d *hostname*)
  - **procurar** (parâmetro *hostname*)
  - **listar** (parâmetro -l)

Exercícios em Sala

```
$ ./hosts -a routerlab -i 192.168.0.1
$ ./hosts -a lab01 -i 192.168.0.100
$ ./hosts -a lab02 -i 192.168.0.101
$ ./hosts -l
routerlab 192.168.0.1
lab01     192.168.0.100
lab02     192.168.0.101
lab03     192.168.0.102
$ ./hosts -d routerlab
$ ./hosts -d lab01
$ ./hosts -l
lab02     192.168.0.101
lab03     192.168.0.102
$ ./hosts lab02
192.168.0.101
$ ./hosts -r 192.168.0.101
lab02
```

Atividades

Refaça o exercício de sala anterior usando o nome *hosts.sh* na pasta *atividades/atividade09*, porém caso o usuário não forneça nenhuma opção por parâmetros, o *script* deve exibir um menu com as opções disponíveis (adicionar, remover, procurar e listar). Se a opção envolver leitura de valores, deve-se requisitar a entrada adequada do usuário.