Programação Shell Script

Prof. Dr. Marcelo Barros de Almeida

Centro Universitário Barão de Mauá

1. Copie o shell script a seguir e rode-o. Tente entender o que aconteceu com cada variável contida nele e fique atento aos pequenos detalhes no tratamento de variáveis. As opções (`-x`) e (`-v`) na execução do bash pode ajudar (`bash -x script`).

```
#!/bin/bash
# Retirado de: www.di.ubi.pt/~operativos Unix Documents
# variaveis de ambiente
echo $PATH
echo $USER $HOME

# pode-se ver todas as variáveis do ambiente com o comando env | less
# variaveis locais
ola="bom dia"
echo "$ola Paulo"
echo "$olaPaulo"   #Texto colado na variavel ...
echo "${ola}Paulo" #Protegendo a variável com as chaves
mesg="$ola $USER"
echo $mesg

# input
echo "Introduza qualquer Coisa" ; read var
echo "Introduziu $var"

# execução
data=`date`
echo $data
info=`echo $HOME ; echo " estamos em "; pwd`
echo $info

# contas
x=1
let x=x*2+3
echo "x=$x"
x=$((x+3))
echo "x=$x"
let x--
echo "x=$x"

#variaveis especiais
echo "Numero de Arguments para este script $#"
echo "Todos os argumentos para este script $*"
echo "O primeiro $1 e segundo $2 argumentos"
echo "O nome deste ficheiro $0"
echo "O Processo ID deste script $$"
echo "Exit status do comando anterior $?"
```

2. Faça um script com a linha de comando dada a seguir e tente explicar o resultado apresentado na tela.

```
banner $(date | cut -c13-20)
```

3. Escreva um script que mostre a data de hoje e depois todos os arquivos dentro da sua conta.

```
#!/bin/bash
date
ls -laR ~
```

4. Escreva um pequeno script que irá testar se determinado arquivo existe ou não. O nome do arquivo será passado via linha de comando.

```
#!/bin/bash
if [ $# -eq 0 ]
then
        echo "Sintaxe: $0 nome_do_arquivo"
        exit 1
fi

if [ -f $1 ]
then
        echo "$1 existe e é um arquivo"
else
        echo "$1 não é um arquivo"
fi
```

ou ...

```
#!/bin/bash
if [ $# -eq 0 ]
then
        echo "Sintaxe: $0 nome_do_arquivo"
        exit 1
fi
[ -f $1 ] && echo "$1 existe e é um arquivo" || echo "$1 não é um arquivo"
```

5. Faça um script que permita dizer se o arquivo passado pela linha de comando, caso exista, é maior do que 100 bytes. O comando `stat` pode ajudar na determinação do tamanho.

```
#!/bin/bash
if [ $# -eq 0 ]
then
        echo "Sintaxe: $0 nome_do_arquivo"
        exit 1
fi

if [ ! -f $1 ]
then
        echo "$1 nao existe."
        exit 1
fi

sz=$(stat -c%s $1)

if [ $sz -gt 100 ]
then
        echo "O arquivo é maior que 100 bytes ($sz bytes)"
else
        echo "O arquivo é menor que 100 bytes ($sz bytes)"
fi
```

6. Faça um script que imprima quantos processo estão atualmente em execução na sua máquina. Use os comandos wc e ps para isso.

```
#!/bin/bash
tot=$(( $(ps aux | wc -l) -1 ))
echo "Existem $tot processos em execução"
```

7. Crie um script que peça um mês e um ano do usuário e depois apresente um calendário do mês/ano pedido. Use o comando cal para lhe ajudar.

```
#!/bin/bash
read -p "Entre o mês desejado (1-12): " mes
read -p "Entre o ano desejado (aaaa): " ano
cal $mes $ano
```

8. Faça um scritp que renomeia todos os arquivos `".mpeg3"` para `".mp3"` em um determinado diretório.

```
#!/bin/bash

dir="/tmp"
old="mpeg3"
new="mp3"

IFSOLD=$IFS
IFS=$'\n'
for fo in $(ls $dir/*.$old)
do
        fn=${fo/%$old/$new}
        mv $fo $fn
        echo "Movendo $fo -> $fn ..."
done
IFS=$IFSOLD
```

9. Repita o exercício anterior mas de forma interativa, perguntando na linha de comando o diretório desejado e as extensões.

```
#!/bin/bash
read -p "Entre o diretorio: " dir
read -p "Extensao antiga: " old
read -p "Extensao nova: " new

IFSOLD=$IFS
IFS=$'\n'
for fo in $(ls $dir/*.$old)
do
        fn=${fo/%$old/$new}
        mv $fo $fn
        echo "Movendo $fo -> $fn ..."
done
IFS=$IFSOLD
```

10. Repita o exercício anterior mas passe o diretório e as extensões por linha de comando.

```
#!/bin/bash
if [ $# -lt 3 ]
then
        echo "Sintaxe: $0 nome_do_diretorio extensao_velha extensao_nova"
        exit 1
fi

dir="$1"
old="$2"
new="$3"

if [ ! -d $dir ]
then
        echo "Diretorio ($dir) invalido" '!'
        exit 1
fi

IFSOLD=$IFS
IFS=$'\n'
for fo in $(ls $dir/*.$old)
do
        fn=${fo/%$old/$new}
        mv $fo $fn
        echo "Movendo $fo -> $fn ..."
done
IFS=$IFSOLD
```

11. Escreva um script para listar todos os arquivos de um determinado diretório (passado via linha de comando) mas no seguinte formato:
    1: nome a
    2: nome b
    ...

```
#!/bin/bash
if [ $# -lt 1 ]
then
        echo "Sintaxe: $0 nome_do_diretorio"
        exit 1
fi

dir="$1"

if [ ! -d $dir ]
then
        echo "Diretório invalido ($dir)"
        exit 1
fi
IFSOLD=$IFS
IFS=$'\n'
for fn in $(ls $dir/*)
do
        if [ -f $fn ]
        then
                echo "--> $fn"
        fi
done
IFS=$IFSOLD
```

12. Escreva um script capaz de dizer o número de linhas num arquivo (passado via linha de comando) e o número de palavras. O comando `wc` pode ajudar.

```
#!/bin/bash

arq="$1"

if [ ! -f $arq ]
then
        echo "Arquivo invalido ($arq)"
        exit 1
fi

lin=$(cat $arq | wc -l)
pal=$(cat $arq | wc -w)

echo "Numero de linhas..: $lin"
echo "Numero de palavras: $pal"
```

13. Desenvolva um script que receba um número como parâmetro e vá imprimindo na tela uma contagem regressiva até chegar a zero, imprimindo a contagem na tela a cada um segundo (use o comando `sleep` para esperar).

```
#!/bin/bash

if [ "$1" = "" ]
then
        echo "Sintaxe: $0 contagem"
        exit 1
fi

num="$1"

while [ $num -ge 0 ]
do
        echo "Contagem: $num"
        num=$((num - 1))
        sleep 1
done
```

14. Mostra na tela todos os parâmetros recebidos na linha de comando (podem ser mais que 9, logo o shift será necessário), imprimindo-os como a seguir:
1: parâmetro 1

2: parâmetro 2
...

```
#!/bin/bash

n=1

while [ -n "$1" ]
do
        echo "$n: $1"
        shift
        n=$((n + 1))
done
```

15. Crie um script para mostrar (`cat`) todos os usuários cadastrados no sistema (`/etc/passwd`) ordenados em ordem alfabética.

```
#!/bin/bash
cat /etc/passwd | cut -f1 -d: | sort
```

16. Um dos parâmetros de cada linha do arquivo (`/etc/passwd`) é o shell usado pelo usuário (o sétimo campo). Escreva um programa capaz de listar todos os shells únicos existente no `passwd`. O programa `uniq` pode ser útil.

```
#!/bin/bash
cat /etc/passwd | cut -f7 -d: | uniq
```

17. Crie um pequeno script de backup capaz de receber um nome de diretório a ser compactado e também um nome de diretório onde o arquivo com o backup será será armazenado. O nome do arquivo de backup deve serguir o formato `"BKP-AAAA-MM-DD.tar.gz"`, onde AAAA é o ano, MM o mês e DD o dia. Use o comando tar para fazer a criação do backup. Inclua testes para determinar se os diretórios de origem e destino realmente existem.

```
#!/bin/bash
if [ $# -lt 2 ]
then
        echo "Sintaxe: $0 dir_src dir_dst"
        exit 1
fi

src="$1"
dst="$2"

if [ ! -d $src ]
then
        echo "Diretório de origem invalido ($src)"
        exit 1
fi

if [ ! -d $dst ]
then
        echo "Diretório de destino invalido ($dst)"
        exit 2
fi

if [ "$dst" = "$src" ]
then
        echo "Diretório de destino e source devem ser diferente"
        exit 3
fi

nome="BKP-"$(date +"%F")".tar.gz"
tar cvzf "$dst/$nome" "$src"
```

18. Crie um script que utilize um laço para criar 10 diretório com o nome padrão `dir<n>`, onde `n` é o número a ser adicionado ao nome do diretório. Dentro de cada diretório crie 5 arquivos com o padrão `file<m>`, onde `m` é o número a ser adicionado ao nome do arquivo.

```
#!/bin/bash
DIR=./dirteste
FILE=arquivo

for i in $(seq 1 10)
do
        d="$DIR$i"
        mkdir "$d"
        for j in $(seq 1 5)
        do
                f="$FILE$j"
                touch "$d/$f"
        done
done
```

19. Escreva um script capaz de ir adicionando idéias em um "repositório de idéias"situado em `/ideias.txt`. O programa deve ser executado com a idéia como parâmetro da linha de comando e deve fazer entradas que começem com a data/hora do dia (comando date pode audar), por exemplo:
`30/08/2008 10:21 Fazer script para remover usuários`

```
#!/bin/bash
REPO=~/ideias.txt

if [ "$1" = "" ]
then
        echo "Sintaxe: $0 \"nova ideia\""
        exit 1
fi

ideia=$(date)" - $1"

echo $ideia >> $REPO
```

20. Criar um programa que mostre o espaço utilizado pelos arquivos dentro de cada diretório da sua conta no sistema, colocando em ordem numérica o resultado. Use os comandos `du` e `sort`.

```
#!/bin/bash
cd ~
du * -s | sort -n
```

21. Crie um script capaz de listar todos os arquivos da sua conta que possuem a palavra "bash"dentro dele.

```
#!/bin/bash
cd ~
grep bash * -oHalr
```

22. Utilizando o comando `find`, buscar os arquivos que pertencem ao usuário `www-data`, redirecionando a saída para o arquivo `arq.txt`. Depois, busque todos os diretórios que pertencem ao usuário `www-data`, redirecionando a saída para o arquivo `dir.txt`. Finalmente, junte os arquivos `arq.txt` e `dir.txt` em um único arquivo `lista.txt`

```
#!/bin/bash
DIR="/var/www"
DST="/tmp"
find "$DIR" -user www-data -type f > "$DST/arq.txt"
find "$DIR" -user www-data -type d > "$DST/dir.txt"
cat "$DST/arq.txt" "$DST/arq.txt" > "$DST/lista.txt"
```

23. Faça um script que possa ser utilizado dentro do `/etc/init.d/`, isto é, um script capaz de responder às opções `"start"`, `"stop"` e `"restart"` via linha de comando. Estruture adequadamente o seu script empregando um case e chamadas de função para cada ação (stop, restart e start).

```
#!/bin/bash

function start()
{
        echo "starting..."
```

```
}

function stop()
{
        echo "stopping..."
}

function restart()
{
        echo "restarting..."
}

case $1 in
        start)
                start;;
        stop)
                stop;;
        restart)
                restart;;
        *)
                echo "Sintaxe: $0 [start|stop|restart]";;
esac
```

24. Faça um script que deixe o arquivo syslog (`/var/log/syslog`) com apenas as últimas 10 linhas. O comando `tail` pode ajudar.

```
#!/bin/bash
LOG=/var/log/syslog
TMP=/tmp/$$.tmp

tail -n 10 "$LOG" > "$TMP"
mv "$TMP" "$LOG"
```

25. Faça um script capaz de lista todos os usuários da máquina que possuam UID maior que 100 (o comando `cut` pode ser útil). O resultado deve ter o seguinte formato:
user1 (uid1)
user2 (uid2)
...

```
#!/bin/bash
IFSOLD=$IFS
IFS=$'\n'
for line in $(cat /etc/passwd)
do
        uid=$(echo "$line" | cut -d: -f3)
        usr=$(echo "$line" | cut -d: -f1)
        if [ $uid -gt 100 ]
        then
                echo -e "$usr \t ($uid)"
        fi
done
```

26. Escreva um script que liste todos os usuários com a cota estourada. O comando `awk` pode ser útil.

```
#!/bin/bash

OLDIFS=$IFS
IFS=$'\n'
for line in $(repquota -au)
do
        usr=$( echo "$line" | awk '{ print $1 }' )
        cot=$( echo "$line" | awk '{ print $2 }' | grep "+" )
        if [ ! "$cot" = "" ]
        then
                echo "O usuario $usr esta com cota estourada"
        fi
done
IFS=$OLDIFS
```