Programação para Redes de Computadores 31/03/2014

Aula 7: Manipulando Arquivos e Argumentos em Bash

Professor: Jorge H. B. Casagrande casagrande@ifsc.edu.br
Notas de aula adaptada da original do prof. Emerson R. de Mello

1 Argumentos de linha de comando

Ao invocar um *shell script* é possível fornecer valores para este através da linha de comando. Por exemplo, o comando `mkdir` recebe como argumento de linha de comando o nome do diretório que desejamos criar (`mkdir novo-dir`). É possível passar mais de um argumento de linha de comando para um *shell script*, abaixo é apresentado a listagem com o nome das "variáveis" em um *shell script* necessárias para operar com argumentos de linha de comando. A figura 1 ilustra o uso das variáveis apresentadas na tabela 1.

Variável	Descrição
$0	Para obter o nome do arquivo (do *shell script*) que está sendo executado
$1	Para obter o valor fornecido pelo primeiro argumento de linha de comando
$2	Para obter o valor fornecido pelo segundo argumento de linha de comando. E assim sucessivamente
$#	Para obter o número total de argumentos fornecidos via linha de comando
$*	Para obter a lista com todos os argumentos fornecidos

Tabela 1: Variável usadas com argumentos de linha de comando

```
1  #!/bin/bash
2
3  if [ $# -eq 0 ];
4     then
5        echo "Voce nao forneceu argumentos de linha de comando"
6     else
7        echo "Voce forneceu $# argumentos de linha de comando"
8        echo "O valor do primeiro argumento e': $1"
9        echo "Segue a listagem com todos os argumentos"
10       i=1
11       for arg in $*; do
12          echo "$i o. argumento: $arg"
13          i=$((i+1))
14       done
15 fi
```

Figura 1: Usando argumentos de linha de comando

2 Operadores para teste com arquivos

Retorna **true** (verdade) se:

-e	arquivo existe	-f	é um arquivo regular (não é diretório nem dispositivo)
-s	o arquivo não possui tamanho zero	-d	é um diretório
-b	é um dispositivo de bloco (cdrom, disquete, etc.)	-c	é um dispositivo de caracter (teclado, modem, placa de som, etc.)
-h	é um *link* simbólico	-r	o arquivo possui permissão de leitura (para o usuário que está executando o teste)
-w	o arquivo possui permissão de escrita (para o usuário que está executando o teste)	-x	o arquivo possui permissão de execução (para o usuário que está executando o teste)
-nt	(f1 -nt f2) f1 é mais novo que f2	-ot	(f1 -ot f2) f1 é mais velho que f2
-ef	(f1 -ef f2) f1 e f2 são *hard links* para o mesmo arquivo		

A figura 2 ilustra um exemplo para listar somente os diretórios presentes no diretório corrente.

```
1  #!/bin/bash
2
3
4  # percorrendo a listagem de arquivos do diretorio corrente
5  for arq in *; do
6     if [ -d $arq ]
7        then
8           echo "$arq e' um diretorio"
9     fi
10 done
```

Figura 2: Exibindo a lista de diretórios

A figura 3 ilustra um exemplo para listar o conteúdo de um arquivo. São feitos testes para garantir que o arquivo seja informado via argumentos de linha de comando e que de fato seja um arquivo regular e não um diretório, por exemplo.

3 Trabalhando com cores no shell

No *shell* é possível alterar a cor do texto, a cor do fundo além de outros modificadores como o negrito, sublinhado e até mesmo para tornar o texto intermitente. A tabela 2 ilustra o código das cores e dos modificadores.

A modificação das propriedades do *shell* é feita através de três campos separados por ponto e vírgula. O primeiro campo é para modificar a cor do texto; o segundo campo é para modificar a cor do fundo; e o terceiro campo é para modificar as propriedades como negrito, sublinhado, etc. É possível omitir os campos os quais não se deseja modificar. Por exemplo, é possível somente informar somente um valor no terceiro campo e omitir valores para os dois primeiros campos. A figura 4 indica como redefinir a cor no *shell*, ilustrando os casos de omissão dos campos.

```bash
#!/bin/bash

#O arquivo a ser lido e' passado com argumento de linha de comando
if [ $# -ne 1 ];
   then
      echo "Sintaxe errada!"
      echo "Exemplo: $0 nome-do-arquivo"
      exit 1
fi

# verifica se e' realmente um arquivo
if [ -f $1 ];
   then
      # lista o conteudo do arquivo
      while read linha; do
         echo $linha
      done < $1
   else
      echo "Argumento informado nao e' um arquivo regular"
      exit 1
fi
```

Figura 3: Exibindo o conteúdo de arquivo

Código	Cor	Código	Modificador
0	preto	0	reinício das configurações
1	vermelho	1	negrito
2	verde	4	sublinhado
3	amarelo	5	piscando
4	azul	7	reverso
5	rosa		
6	ciano		
7	branco		

Tabela 2: Código das cores e modificadores

```bash
#!/bin/bash

# \e[3xm - indica a cor do texto, pondendo X ser de 0 ate' 7.
echo -e "\e[32m Mudando somente a cor do texto para verde \e[m"

# \e[4xm - indica a cor do fundo, pondendo X ser de 0 ate' 7.
echo -e "\e[42m Mudando somente a cor do fundo para verde \e[m"

echo -e "\e[32;40m Mudando a cor do texto para verde e o fundo para preto \e[m"

# \e[33;44;xm - x e' o modificador, podendo ser 0, 1, 4, 5 ou 7
echo -e "\e[33;44;1m Deixando o texto amarelo em negrito sobre o fundo azul \e[m"

# o \e[m no final da linha serve para retornar as propriedades padroes do terminal
echo -e "\e[1m Deixando o texto somente em negrito \e[m"
```

Figura 4: Usando cores

4 Variável IFS – *Internal Field Separator*

A variável `IFS` é usada pelo *shell* para delimitar palavras em uma lista, a qual é usada por diversos comandos, por exemplo, `read`, `for`, `select`, etc. O valor padrão da variável `IFS` consiste nos caracteres de espaço em branco, tabulação e de nova linha, mas é possível redefiní-la para uma melhor adequação aos nossos *scripts*. A figura 5 ilustra um exemplo sem redefinir a variável `IFS` e um segundo exemplo redefinindo-a. Na figura 6 é apresentado um exemplo para listar todo o conteúdo do arquivo `/etc/passwd` e apresentar a saída formatada.

```
#!/bin/bash

# criando uma lista de palavras
lista="Aula PRC; Tecnico em Redes"

# percorrendo a lista de palavras com um for
for i in $lista; do
   echo "palavra: $i"
done

# O resultado sera'
# palavra: Aula
# palavra: PRC;
# palavra: Tecnico
# palavra: em
# palavra: Redes

# Redefinindo a variavel IFS, colocando o caracter ; como delimitador
IFS=";"

# percorrendo a lista de palavras com um for
for i in $lista; do
   echo "palavra: $i"
done

# O resultado sera'
# palavra: Aula PRC
# palavra: Tecnico em Redes

# limpando a variavel IFS (boa pratica) apos seu uso
unset IFS
```

Figura 5: Redefinindo variável `IFS`

```
#!/bin/bash
# Listando o conteudo do arquivo /etc/passwd e formatando a saida
IFS=":"

while read usuario senha uid gid nome home shell; do
echo -e "Usuario: $usuario \nSenha:\t $senha \nNome:\t $nome"
echo -e "UID:\t $uid \nGID:\t $gid"
echo -e "Home:\t $home \nShell:\t $shell \n"
echo -e "------------------"
done < /etc/passwd
# o conteudo arquivo /etc/passwd esta' sendo enviado para o while
# o arquivo sera' lido linha a linha ate' chegar-se ao seu fim

unset IFS
```

Figura 6: Saída formatada do arquivo `/etc/passwd`

5 Criando menus através do comando `select`

O `bash` possui um comando chamado `select` que permite a criação de menus de opções. A figura 7 ilustra um exemplo usando o comando `select`.

```
#!/bin/bash

# definindo a mensagem que sera' exibida
PS3="Entre com uma opcao: "

# definindo as opcoes disponiveis no menu
menu="Soma Subtracao Divisao Multiplicacao Sair"

select opcao in $menu; do

   echo "Voce escolheu: $opcao"
   # A variavel especial REPLY contem o valor escolhido

   case $REPLY in
      1)
         echo "Soma "
         ;;
      2)
         echo "Subtracao"
         ;;
      3)
         echo "Divisao"
         ;;
      4)
         echo "Multiplicacao"
         ;;
      5)
         echo "Saindo"
         break
         ;;
      *)
         echo "Entrou com uma opcao diferente de 1, 2, 3, 4 ou 5"
         ;;
   esac
done
```

Figura 7: Comando `select`

6 Exercícios

Desenvolva um *shell script* que:

1. Faça a listagem das cores disponíveis no *shell*. O *script* deverá exibir uma frase variando somente a cor do texto e depois exibir a frase variando somente a cor de fundo.

2. Apresente um menu com as seguintes opções: Soma, Subtração, Divisão, Multiplicação e Sair. O usuário deverá selecionar uma dessas opções, informar dois operandos e o *script* deverá apresentar o resultado. O *script* ficará em execução até que o usuário escolha a opção `Sair`.

3. Leia dois operandos e um operador, através de argumentos de linha de comando, e realize a operação aritmética correspondente. Exemplo: `./exercicio3.sh 10 + 2`, resultado: `12`.

4. Leia duas datas no formato AAAA-MM-DD, através de argumentos de linha de comando, e informe a diferença em dias entre essas datas.
   Exemplo: `./exercicio4.sh 2009-09-21 2001-10-02`.

5. Leia o nome de uma interface de rede (argumentos de linha de comando) e informe o IP, a máscara de rede e o endereço de broadcast associados a esta.

   Dica: Faça uso do comando `ip` combinado as ferramentas `grep, tail` e `cut`.
   Exemplo: `ip a | grep eth0 | tail -1`

6. Que converta para minúsculo os nomes de todos os arquivos regulares (não pode ser diretório, *link* simbólico) de um diretório fornecido como argumento de linha de comando.
   Exemplo: `./exercicio6.sh documentos`.

   Dica: Faça uso do comando `tr`.
   Exemplo:

```
1  echo "TesTando a FerraMENTA" | tr '[:upper:]' '[:lower:]'
```

   textttjpg do diretório atual para essa resolução. Deve-se preservar o arquivo original e o arquivo criado deverá ter em seu nome a sua resolução.

7. O aplicativo **convert** (pertencente ao pacote `imagemagick`) é famoso por fazer a conversão de formatos de imagens, por exemplo JPG para PNG, bem como o redimensionamento, rotacionamento entre outras tarefas sobre uma imagem. Alguns exemplos de uso são apresentados abaixo:

```
1  # convertendo de JPG para PNG. Basta indicar a extensao desejada para fazer a
       conversao
2  convert arquivo.jpg arquivo.png
3
4  # redimensionando uma imagem JPG
5  convert arquivo.jpg -resize 800x600 -quality 89 arquivo.jpg
6
7  # rotacionando em 90 graus uma imagem
8  convert arquivo.jpg -rotate 90 arquivo.jpg
```

   (a) Desenvolva um *shell script* que receba 2 argumentos, formato de origem e formato de destino e converta todos os arquivos do diretório atual do formato de origem para o formato de destino;

   (b) Desenvolva um *shell script* que redimensione todos arquivos do diretório atual. Para isto deve-se fornecer 2 argumentos, formato do arquivo e tamanho desejado;

   (c) Desenvolva um *shell script* que converta todas imagens que estejam no formato "retrato" para o formato "paisagem". Sabemos que uma imagem está no formato retrato quando sua largura é inferior a sua altura. Uma forma para obter as dimensões de uma imagem é mostrada abaixo:

```
1  # usando o comando cut
2  identify imagem.jpg | cut -d " " -f 3
3
4  # usando a linguagem awk
5  identify imagem.jpg | awk '{print $3}'
```

8. A ferramenta **netcat** é conhecida como um "caniveto suíco TCP/IP" pois pode ser usada das mais diversas formas para ler e escrever dados através de conexões de rede TCP e UDP. Por exemplo, é possível usar o **netcat** para verificar se um determinado serviço está rodando em um servidor. No quadro abaixo é feito uso do **netcat** para verificar se o servidor **HTTP** (porta 80) da máquina `www.sj.ifsc.edu.br` está rodando, ou seja, aceitando conexões. É feito também um teste sobre a porta 81 para a mesma máquina.

```
# Verificando porta 80
netcat -zv -w 10 www.sj.ifsc.edu.br 80
   sj [200.135.37.65] 80 (www) open

# Verificando porta 81
netcat -zv -w 10 www.sj.ifsc.edu.br 81
   sj [200.135.37.65] 81 (?) : Connection refused
```

O resultado do netcat[1] para a porta 80 foi **open** e o resultado para a porta 81 foi **Connection refused**.

(a) Desenvolva um *shell script* para fazer uma varredura de portas (de uma porta inicial até uma porta final, especificadas pelo usuário) em uma dada máquina e que indique quais portas (ex: 80) e serviços (ex: www) estão rodando nesta máquina.
Exemplo: `./varredor.sh 192.168.0.1 10 1024`. Sendo **10** a porta inicial e **1024** a porta final.

[1] Dúvidas sobre os parâmetros do netcat? Veja `man netcat`.