Instituto Federal de Santa Catarina – IFSC
Campus São José

Programação para Redes de Computadores

Introdução ao Shell

Prof. Francisco de Assis S. Santos, Dr.

São José, 2014.

O Interpretador de comandos

O *shell* consiste em um interpretador de comandos presente em todos os sistemas operacionais variantes dos *Unix*, que inclui *Linux*, BSD e ate o MacOS. No *linux* existem diversos tipos de *shell*, sendo estes: csh, bash, ksh e zsh.

Como dito, o *shell* e um interpretador de comandos e temos a opção de entrar com uma sequência de comandos sempre que desejarmos realizar uma tarefa ou podemos colocar tal sequência dentro um arquivo e chamar este arquivo sempre que necessário. E assim temos o *shell* script ilustrado abaixo.

```
1 #!/bin/bash
2
3 echo "Ola mundo!"
```

Principais comando para *Shell Script*

• **echo** - tem por objetivo imprimir mensagens no dispositivo de saída padrão, no caso o monitor.

•Abaixo algumas opções:

•-e Ativa a interpretação de caracteres de escape (\)

- \n -- nova linha; - n -- sem pular linha\t -- tab; \a alerta (beep)

• **read** - Permite que o usuário forneça informações via teclado (e necessário pressionar ENTER para finalizar a leitura).

• **expr** - para fazer cálculos, porém só faz operações com inteiros. Exemplo de uso:

```
soma=`expr 2 + 2` # expr entre crases
```

bc - trata-se de uma calculadora, ideal para quando necessitamos efetuar cálculos com números reais. Exemplo de uso:

```
1 # executando o bc em um terminal, combinado com o echo
2 echo "scale=2; 1/2" | bc
3
4 # armazendo o resultado da saída do bc na variavel 'resultado'
5 resultado=`echo "scale=2; 1/2" | bc`
```

Shell Script: Variáveis

Nas linguagens de programação as variáveis possuem uma função semelhante com as variáveis da matemática, ou seja, armazenam valores para que possam ser recuperados posteriormente. A seguir são listadas algumas formas para atribuir e obter valores em variáveis.

```
#!/bin/bash

echo -\n "Entre com o seu nome: "
read nome
echo "Ola $nome!"
```

Shell Script: Variáveis

```
#!/bin/bash
# Isto é um comentário. Todo texto apos o caracter # será ignorado
echo "Trabalhando com variáveis"
a=1
b=2
c=`expr $a + $b` # a expressão está entre crases
d=$((c+a))
echo "O valor de a é $a, o valor de b é $b, o valor de c é' $c e o valor de d é $d"
curso="Aula de PRC"
echo "O conteúdo de curso e' $curso"

# outro exemplo
versao=$(uname -r)

echo "A versao do kernel e' $versao"
```

Shell Script: Operadores Relacionais e Lógicos

Numéricos		**Cadeia de caracteres**		**Operadores lógicos**	
-eq	Igual	=	igual	-a	E lógico (AND)
-ne	Diferente	!=	diferente	&&	E lógico (AND)
-ge	Maior ou igual	-n	Não é nula	-o	OU lógico (OR)
-le	Menor ou igual	-z	É nula	||	OU lógico (OR)
-gt	Maior			!	negação
-lt	menor				

Shell Script: Estruturas de Decisão

Se...então...senão

```
#!/bin/bash

nota=5

if [ $nota -ge 5 ];
 then

 echo "nota maior ou igual a 5"

 else

echo "nota menor que 5"

 fi
```

Shell Script: Estruturas de Decisão

Se...então...senão

```
#!/bin/bash

nota=5

if [ $nota -ge 5 ];
then

echo "nota maior ou igual a 5"

else

echo "nota menor que 5"

fi
```

Shell Script: Estruturas de Decisão

Se...então...senão com o operador AND

```
#!/bin/bash

a=3
b=2
c=1

# usando o operador E
if [ $a -gt $b ] && [ $a -gt $c ];
 then
 echo "A é o maior"
 else
 echo "A não é o maior"
 fi
```

Shell Script: Estruturas de Decisão

Escolha ...caso...

```
#!/bin/bash

echo -n "Entre com um numero de 1 a 5: "
read numero

case $numero in
1)
echo "Voce escolheu 1"
;;
2)
echo "Voce escolheu 2"
;;
3)
echo "Voce escolheu 3"
;;
4 | 5)
echo "Voce escolheu 4 ou 5"
;;
*)
echo "Voce escolheu um numero diferente de 1, 2, 3, 4 ou 5"
;;
esac
```

Shell Script: Estruturas de Repetição

Enquanto

```
#!/bin/bash

num=10

while [ $num -gt 0 ]; do
echo "contando $num"
num=$((num-1))
done

#-------------------------#
#usando o operador de negacao '!'
num=10

while ! [ $num -eq 0 ]; do
echo "contando $num"
num=$((num-1))
done
```

Shell Script: Estruturas de Repetição

Para (for)

```
#!/bin/bash

# maneira tradicional do shell
for contador in `seq 1 10`; do
  echo $contador
done



# percorrendo uma lista de palavras separadas por espaco
lista="Cabeamento Gerencia Programacao Equipamentos"
for disciplina in $lista; do
  echo "Disciplina $disciplina"
done
# Dica: que tal listar os arquivos de um diretorio?
# lista=`ls` # o ls esta' entre crases
```

Exercícios

1)Desenvolva um algoritmo que leia um número inteiro positivo e imprima a sequência de 0 até este número. O programa devera tratar caso o usuário forneça um número menor que zero.
2)Desenvolva um algoritmo que leia um numero inteiro e calcule seu fatorial.
3)Desenvolva um algoritmo que simule a autenticação de usuários. O usuário deve fornecer uma senha e se esta senha for igual a palavra secreta devera exibir a mensagem “Acesso autorizado”, caso contrario devera exibir “Acesso negado”. O algoritmo devera solicitar a senha ao usuário ate que este forneça a senha correta ou ate que o número de tentativas permitidas seja alcançado. No caso, o número máximo de tentativas e 3.

Funções

O uso de funções permite o reaproveitamento de código além de tornar mais fácil a leitura do código. Uma função e uma sub-rotina que implementa um conjunto de operações destinada a uma tarefa especifica. Trata-se de uma “caixa preta”, ao invocar uma função espera-se que esta realize algumas operações, porém não e necessário saber como essas operações são executadas.

Funções: Sem Passagem de Argumentos e Sem Retorno de Parâmetro

```
#!/bin/bash

imprimir(){
   echo -e "\n ola mundo!"
}

espera(){
   cont=1
   while [ $cont -le 5 ]; do
      echo -n "$cont,"
      sleep 1
      cont=$((cont+1))
   done
}

echo "Invocando as funcoes..."

imprimir
espera
imprimir
```

Funções: Com passagem de Argumentos e Retorno Parâmetro

```
#!/bin/bash

testandoParametros(){
 if [ -z "$1" ]
   then
      echo "O 1o. parametro e' de tamanho zero"
   else
      echo "O 1o. parametro e': $1"
 fi

 if [ "$2" ]
   then
      echo "O 2o. parametro e': $2"
 fi

 return 150
}

testandoParametros "teste"
retorno=$? # vai conter o valor retornado pela funcao invocada acima

echo "Retorno contem: $retorno"
```

Bibliografia

MORAES. P. S. 2000. UNICAMP – Centro de Computação. Lógica de Programação. Disponível em: <http://www.lab.ufra.edu.br/lasic/images/AULAS/PROF_CHASE/TEC_PROG_I/UFRA_TEC_PROG_P_01.pdf>. Acessado em: 08/2014.

J. L. Güntzel e F. A. Nascimento, Introdução aos Sistemas Digitais, Vol. 1, 2001.

J. H. C. Casagrande. Notas de Aula. Acessado em: Julho de 2014, Disponível em: <http://www.sj.ifsc.edu.br/casagrande/PRC>.