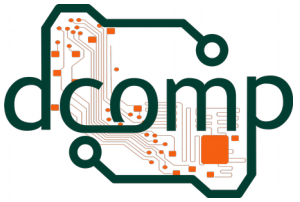
Universidade Federal do Espírito Santo
Centro de Ciências Agrárias – CCENS UFES
Departamento de Computação



Shell Script

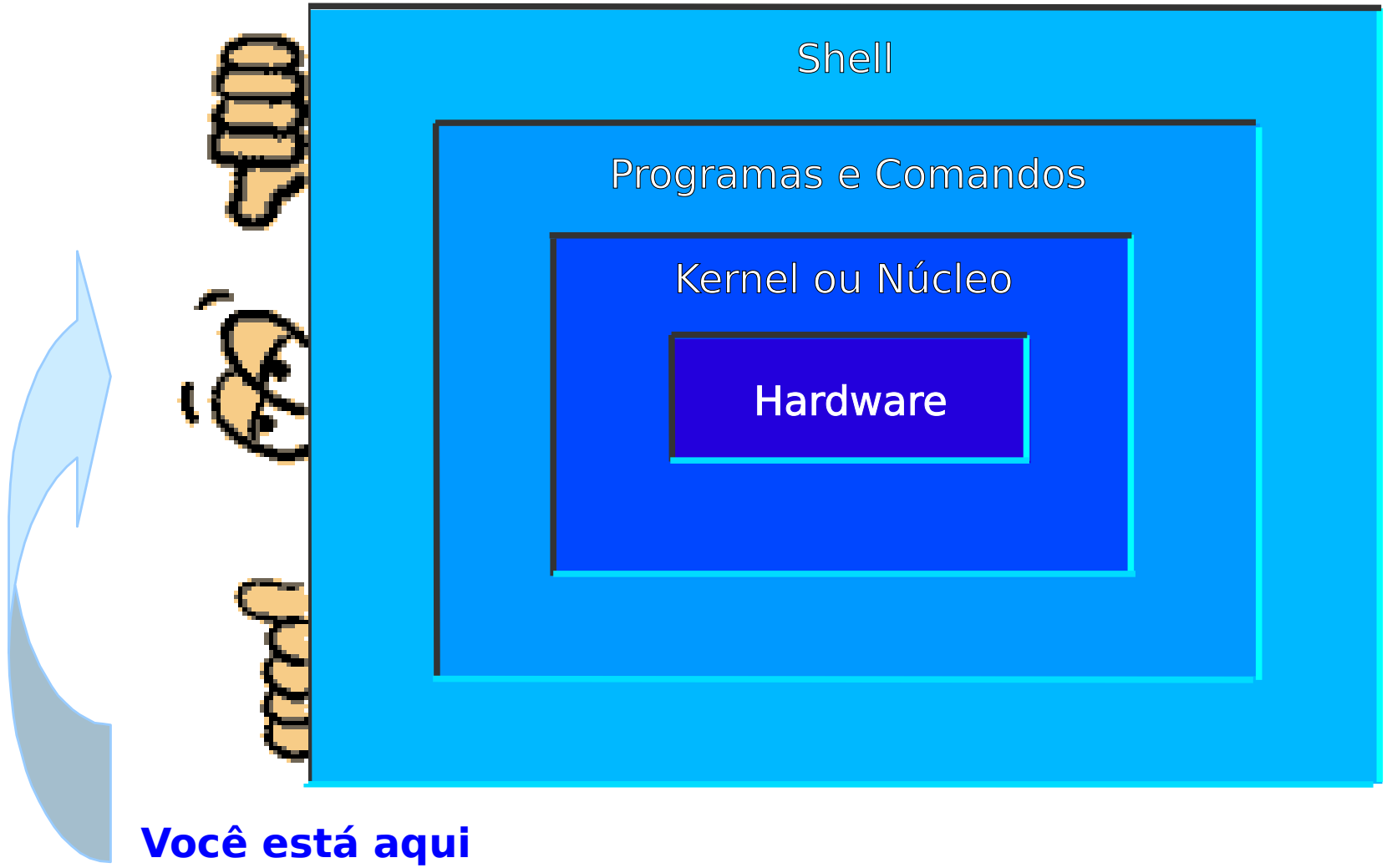
Sistemas de Software Livre

Site: http://jeiks.net
E-mail: jacsonrcsilva@gmail.com

Conteúdo

- O que é o Shell
- Linha de Comandos
- Redirecionamentos
- Manipulação de arquivos
- Tratamento de cadeias
- Condicionais
- Case
- Laços

Onde está o Shell?



Tipos de Shell

- Os principais Shells são:

  sh, bash, ksh, dash

- Efetivamente, o Shell é o programa associado ao último campo de /etc/passwd

```
$ grep jneves /etc/passwd
 jacson:x:54002:101::/home/jacson:/bin/bash
$ grep desliga /etc/passwd
 desliga:x:0:0::/dsv/desliga:/usr/local/bin/desliga
```

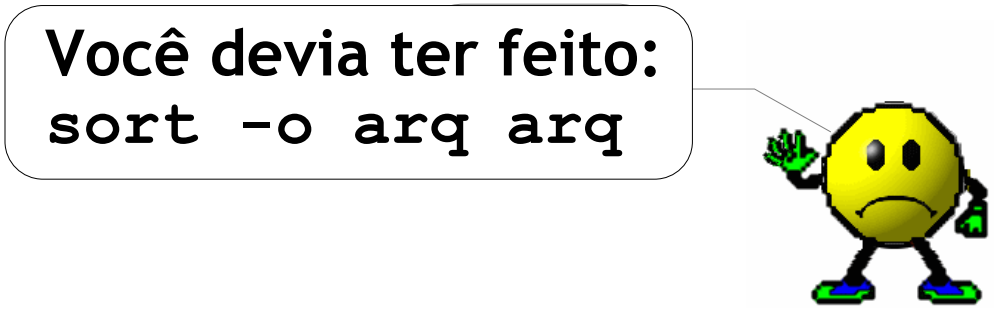
Linha de Comandos

- Atribuição ou comando?
  - Atribuição: `VAR=5`
  - Comando sem opção e argumento: `who`
  - Comando só com opção: `who -H`
  - Comando só com argumento: `who am i`
  - Comando com opção e argumento: `who am i -H`

Linha de Comandos

- Comando `grep -i $CADEIA * > progs`
  - Indentifica o nome do programa `grep`
  - Pesquisa sua existência (PATH) e permissões ✔ `OK`
  - Identifica opções/parâmetros `-i, $CADEIA, *`
  - Identifica redirecionamentos `>`
  - Identifica variáveis `$CADEIA`

Linha de Comandos



- Comando:
  - Resolução de redirecionamentos: `sort arq > arq`
  - Substituição de variáveis: `echo $LOGNAME`
  - Substituição de meta caracteres: `echo *`
  - Passa a linha interpretada para execução

Redirecionamentos de saída

> Redireciona a saida de um comando para um arquivo, destruindo seu conteúdo.

>> Redireciona a saída de um comando para um arquivo, mantedo intacto o conteúdo.

2> Redireciona a saída de erros para um arquivo, destruindo seu conteúdo (anexa ao fim do arquivo).

```
$ ls JaEra
ls: JaEra: No such file or directory
$ ls JaEra 2> ArqErr
$ cat ArqErr
ls: JaEra: No such file or directory
```

Redirecionamento de entrada

< Avisa ao Shell que a entrada não será feita pelo teclado, mas sim por um arquivo

```
$ mail usuario < ArqMala
```

<< Indica ao Shell que o escopo do comando começa na linha seguinte e termina quando encontrar um rótulo (label) definido.

```
$ mail jeiks << FimMail
>   SP, `date`
>   Hoje o conteudo do diretorio era `ls -l`
> FimMail
```

Prompts de continuação (PS2)

Redirecionamentos especiais

| Passa a saída de um comando para a entrada de outro. Conhecido como “pipe”.

tee Passa a saída de um comando para a saída padrão (tela) e também para um arquivo.

```
$ ls c* | tee arq.tee
calculadora.sh
cores.sh
cripta.sed
$ cat arq.tee
calculadora.sh
cores.sh
cripta.sed
```

Manipulação de Arquivos

- O Shell e os aplicativos da GNU fazem de forma Simples, Imediata e Eficiente.
- Principais ferramentas:

  redirecionamentos

  `sed`

  `awk`

  `fgrep; grep; egrep (ou grep -e)`

  `find`

Inserindo conteúdo em arquivos

```
$ cat arq
linha1
linha3
$ echo linha2 > arq
$ cat arq
linha2
$ echo linha1 >> arq
$ sort arq
linha1
linha2
```

```
$ cat arq
linha2
linha1
```

Apagando linhas

```
$ cat arq
linha1
linha2
linha3
$ grep -v linha3 arq > arq.novo
$ mv -f arq.novo arq
$ cat arq
linha1
linha2
$ sed 2d arq > arq.novo
$ mv -f arq.novo arq
$ cat arq
linha1
```

Alterando valores

```
$ cat arq
linha1
$ sed s/1/Única/ arq > arq.novo
$ mv -vf arq.novo arq
$ cat arq
linhaÚnica
$ echo linha1 > arq
$ sed 's/1/Única/;s/l/L/' arq > arq.novo
$ mv -vf arq.novo arq
$ cat arq
LinhaÚnica
```

Tratamento de cadeias

`cut`

- Corta um pedaço ou um campo de uma cadeia

`paste`

- O oposto do cut. Cola cadeias de caracteres

`tr`

- Transforma <dos-caracteres> <para-os-caracteres>

`awk`

- Linguagem própria.
- Permite trabalhar com as colunas e com o texto.

Tratamento de cadeias

```
$ who   # Este comando lista usuários logados
bolpetti   pts/0         Mar 25 08:06
jneves     pts/2         Mar 25 17:58
bolpetti   pts/1         Mar 25 08:07
rlegaria   pts/4         Mar 25 09:04

$ who | cut -f1 -d" "
bolpetti
jneves
bolpetti
rlegaria

$ who | cut -f1 -d" " | sort | uniq
bolpetti
jneves
rlegaria
```

Tratamento de cadeias

```
$ echo pocoto | tr o u
pucutu
$ cat > arq1
Linha1
Linha2
CTRL+D
$ cat > arq2
Continuacao1
Continuacao2
CTRL+D
$ paste arq1 arq2 > arq3
$ cat arq3
Linha1  Continuacao1
Linha2  Continuacao2
$ awk '{print $1"--"$2}' arq3
Linha1--Continuacao1
Linha2--Continuacao2
```

Instruções condicionais `(if)`

- Os quatro mandamentos do `if`:
  - Não testa condições
  - Testa instruções
  - O comando `test` testa condições
  - Para testar condições use o `if` junto com o `test`

```
$ if  ls -l diret | grep ^d; then
>     cd diret
> else
>     mkdir diret
>     cd diret
> fi
```

if + test

```
$ test -d diret
$ echo $?
0
$ if  test -d diret ; then
>     cd diret
> else
>     mkdir diret
>     cd diret
> fi
```

if + test

```
$ IDADE=27
$ if [ $IDADE -gt 18 ];then
> echo 'Você é maior de idade'
> else
> echo 'Você é menor de idade'
> fi
```

Instruções condicionais (case)

O comando `case` suporta uso de metacaracteres:

```
case var in
    padrão1) comandos ;;
    padrão2) comandos ;;
          *) comandos ;;
esac
```

```
case $(date '+%H') in
  0?|1[01]) echo Bom Dia
                ;;
  1[2-7]) echo Boa Tarde
                ;;
  *) echo Boa Noite
                ;;
esac
```

```
case $CHAR in
    [a-z])
     echo Letra Minuscula    ;;
    [A-Z])
     echo Letra Maiuscula    ;;
    [0-9])
     echo Numero             ;;
        *)
     echo Caracter Especial  ;;
esac
```

Instruções de Laço (`Loop`)

- Comando `for`
  - Recebe cadeias de caractere como parâmetro
- Comando `while`
  - Assim como o if, este comando testa instruções
- Comando `until`
  - Também testa instruções.

Instruções de laço (`for`)

```
$ for VAR in a b c d
> do
> echo $VAR
> done

$ for ((i=0;i<=10;i++))
> do
> echo $i
> done
```

Instruções de laço (`while`)

```
$ NUM = 0
$ while sleep 1;do
> echo $NUM
> let NUM++
> done

$ NUM=0
$ while [ $NUM -lt 10 ];do
> echo $NUM
> let NUM++
> done
```

Instruções de laço (`until`)

```
$ Resp=
$ until [ "$Resp" = S -o "$Resp" = N ];do
> read -p “Deseja Continuar? “ Resp
> done
```

Leitura (`read`)

```
$ read var1 var2 var3
Isto é um erro de pontuação!
$ echo "($var1) , [$var2] . $var3"
Isto,é.um erro de pontuação!
```

E qual é o resultado do comando abaixo?

```
$ echo '($var1) , [$var2] . $var3'
???
```

Leitura de arquivos

```
# 1ª forma:
cat /etc/passwd | \
while read Linha; do
  User=$(echo $Linha | cut -f1 -d:)
  UID=`echo $Linha | cut -f3 -d:`
  Coment=`echo $Linha | cut -f5 -d:`
done
echo $User $UID $Coment
```

Leitura de arquivos

```
# 2ª forma:
while read Linha; do
   User=$(echo $Linha | cut -f1 -d:)
   UID=`echo $Linha | cut -f3 -d:`
   Coment=`echo $Linha | cut -f5 -d:`
Done < /etc/passwd
echo $User $UID $Coment
```

Opções do `read`

Otimizando a captura dos dados

Formas otimizadas do comando `read`:

```
read -p      # Prompt
read -n      # Número de caracteres
read -t      # Tempo
read -s      # Silenciosamente
```

Comando `read`

```
$ read -p "Digite seu nome: " Nom
Botelho
$ echo O nome digitado foi: $Nom
O nome digitado foi: Botelho

$ if  read -t 20 -p 'Opção: ' Opc ; then
>     echo A opção escolhida foi: $Opc
> else
>     echo Fim do prg. 20 segundos sem uso.
>     exit 1
> fi
```

Obtendo informações do kernel

```
$ grep -m1 'model name' /proc/cpuinfo | \
  cut -d: -f2
$ grep -m1 'MHz' /proc/cpuinfo | \
  cut -d: -f2
$ grep 'MemTotal\|MemFree' /proc/meminfo

$ awk '{
if ($1=="cpu" && $2=="MHz")
   print $NF
}' /proc/cpuinfo
```