

## Tópicos Especiais em Linguagens de Programação Shell Script

# Processamento de Texto

Andrei Rimsa Álvares  
*andrei@decom.cefetmg.br*



# Comandos

- Comandos para processamento de arquivos texto

Comando	Propósito
<b>nano</b>	Editor de texto simples
<b>vi/vim</b>	Editor de texto completo
<b>emacs</b>	Editor de texto robusto e extensível
<b>sed</b>	Editor de fluxo complexo
<b>awk</b>	Processador de texto com casamento de padrões
<b>strings</b>	Extrair caracteres legível de arquivos binários
<b>cat</b>	Concatenar arquivos e mostrar seu conteúdo
<b>tac</b>	Concatenar arquivos em ordem reversa
<b>wc</b>	Contar o número de linhas, palavras e caracteres de um arquivo
<b>more</b>	Mostrar a saída de um comando ou arquivo de texto uma página por vez
<b>less</b>	Mostrar a saída de um comando ou arquivo de texto uma página (ou linha) por vez



# Comandos

- Comandos para processamento de arquivos texto

Comando	Propósito
<b>head</b>	Mostrar as primeiras partes de um arquivo
<b>tail</b>	Mostrar as últimas partes de um arquivo
<b>tee</b>	Mostrar a saída de um comando e escrever a saída em um arquivo
<b>grep</b>	Casar padrões e filtrar dados
<b>sort</b>	Ordenar o conteúdo de um fluxo de dados de entrada ou arquivo
<b>zcat</b>	Ler o conteúdo de um arquivo compactado
<b>diff</b>	Comparar arquivos



## nano

- **Propósito:** editor de texto simples
- **Sintaxe:** nano [options] [file]

**Dica:** o editor de texto pico funciona de forma semelhante

- nano é um editor de simples encontrando na maioria dos sistemas Linux e BSD; é o editor mais recomendado para novos usuários

```
tmp - nano - 78x11
GNU nano 2.0.6      File: ShoppingList.txt

Milk
Eggs
Cheese
Tacos
█

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```



## nano

- As funcionalidades como procurar, salvar e fechar arquivos são controladas por teclas de funções, listadas embaixo da tela; algumas funções são listadas a seguir

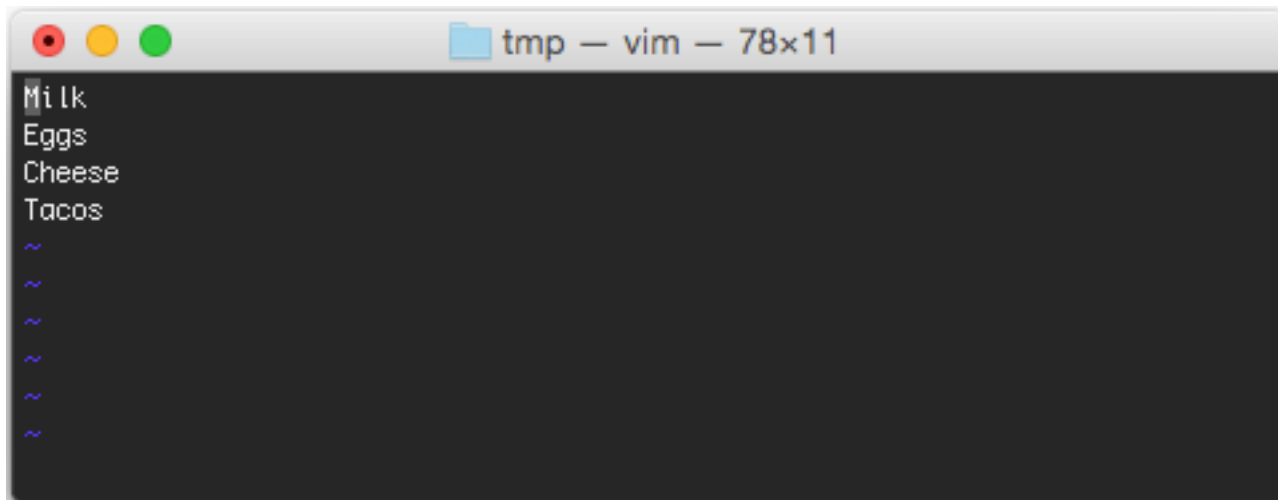
Teclas	Função
CTRL + O	Salvar
CTRL + G	Ajuda
Page Up	Página anterior
ESC + D	Quantidade de palavras
CTRL + K	Cortar a linha atual
CTRL + U	Colar
CTRL + W	Procurar
CTRL + A	Ir para o começo da linha
ESC + \	Ir para a primeira linha do arquivo

Teclas	Função
CTRL + X	Sair
ESC + X	Mostrar o menu
Page Down	Próxima página
CTRL + C	Posição atual do cursor
ESC + 6	Copiar a linha atual
CTRL + \	Encontrar e substituir
CTRL + _	Ir para a linha
CTRL + E	Ir para o final da linha
ESC + /	Ir para a última linha do arquivo



## vi/vim

- **Propósito:** editor de texto completo
- **Sintaxe:** `vi [options] [file]`
  - O editor vi (ou sua versão aprimorada vim) é um editor complexo e completo para sistemas Unix, onde normalmente é o editor padrão



```
tmp - vim - 78x11
Milk
Eggs
Cheese
Tacos
~
~
~
~
~
```



## vi/vim

- O editor vi e vim possuem dois modos de operação:  
**modo de comando** e **modo de edição**
  - Pressionar ESC ativa o **modo de comando**, onde teclas de comando pode ser usadas para ativar edições específicas
  - No modo de comando, algumas teclas podem ser usadas para ativar o **modo de edição**

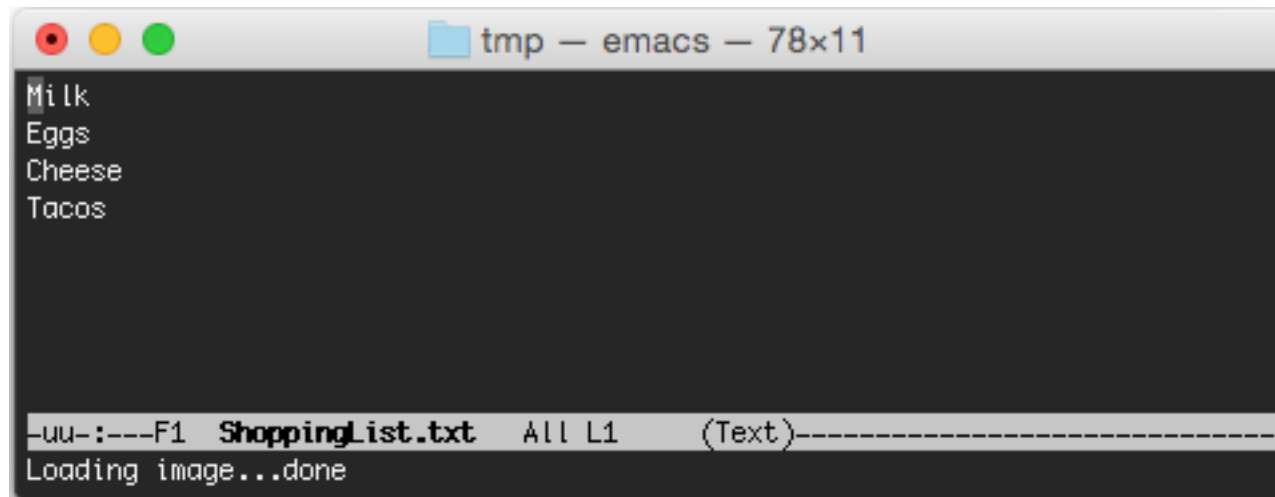
Teclas	Função
:w	Salvar
:x	Salvar e sair
:q	Sair
i	Inserir texto antes
I	Inserir texto após
a	Anexar texto antes

Teclas	Função
A	Anexar texto após
r	Substituir texto antes do cursor
R	Substituir texto após o cursor
yy	Copiar a linha atual
p	Colar texto copiado
/[TEXT]	Procurar por texto especificado



## emacs

- **Propósito:** editor de texto robusto e extensível
- **Sintaxe:** emacs [options] [file]
  - O emacs é um dos mais antigos editores de texto ainda em uso em sistemas Unix; Suas macros programáveis e *code syntax highlight* o tornam uma opção popular para desenvolvedores







# sed

- **Propósito:** editor de fluxo complexo
- **Sintaxe:** `sed [options] [file]`
  - sed é um editor de fluxo complexo que usa expressões regulares para modificar o fluxo de dados; por exemplo para substituir *Tacos* por *Nachos*

```
$ cat ShoppingList.txt
```

```
Milk
```

```
Eggs
```

```
Cheese
```

```
Tacos
```

```
$ sed "s/Tacos/Nachos/" ShoppingList.txt
```

```
Milk
```

```
Eggs
```

```
Cheese
```

```
Nachos
```



## awk

- **Propósito:** processador de texto com casamento de padrões
- **Sintaxe:** `awk [expression]`
  - awk trata cada linha da entrada como uma série de campos, onde os campos são agrupados em um arranjo e uma variável relacionada a cada posição; por exemplo para extrair os campos um (\$1) e oito (\$8) da saída do comando `ls`

```
$ ls -l ShoppingList.*
-rw-r--r-- 1 nick nick 24 2010-04-12 23:10 ShoppingList.old
-rw-r--r-- 1 nick nick 23 2010-04-12 19:33 ShoppingList.txt
$ ls -l ShoppingList.* | awk -F" " '{print $1 " " $8}'
-rw-r--r-- ShoppingList.old
-rw-r--r-- ShoppingList.txt
```



## awk

- O comando `awk` pode ser usado para fazer processamentos em lote, como renomear múltiplos arquivos como no exemplo a seguir

```
$ ls
File1 File3 File5 File7 File9
File2 File4 File6 File8
$ ls | awk '{print "mv "$1 "$1".txt"}' | sh
$ ls
File1.txt File3.txt File5.txt File7.txt File9.txt
File2.txt File4.txt File6.txt File8.txt
```



# strings

- **Propósito:** extrair caracteres legíveis de arquivos binários
- **Sintaxe:** strings [options] [file]
  - Arquivos binário contém dados que são ilegíveis por ferramentas padrões de processamento de texto, como arquivos mp3, jpeg e mpeg; o exemplo a seguir extrair informações imprimíveis do arquivo de música

```
$ strings unknown.mp3
...
TAG Girl You Know It's True
Milli Vanilli
Girl You Know It's True
1989
...
```

Por padrão strings somente imprime textos com mais de 4 caracteres, mas pode ser alterado pela opção -n



# cat

- **Propósito:** concatenar arquivos e mostrar seu conteúdo
- **Sintaxe:** `cat [options] [files]`
  - O comando `cat` pode ser usada para mostrar o conteúdo de um ou mais arquivos, conforme exemplo a seguir

```
$ cat ShoppingList.txt NachoIngredients.txt
Milk
Eggs
Cheese
Tacos
2 cloves garlic, crushed
6 green onions, sliced, white parts and tops separated
1 cup salsa
1/2 (12 ounce) package tortilla chips
1 (8 ounce) package shredded Cheddar/Monterey Jack cheese blend
1/2 large tomato, diced
```



# cat

- Usos comuns

Comando	Propósito
cat [file]	Mostrar o conteúdo do arquivo especificado
cat [file1] [...] [fileN]	Concatenar os arquivos especificados
cat -n [file]	Mostrar saída numerada para cada linha
cat -s [file]	Imprimir suprimindo linhas em branco



# tac

- **Propósito:** concatenar arquivos em ordem reversa
- **Sintaxe:** `tac [file]`
  - O comando `tac` é similar ao comando `cat`, exceto que mostra o conteúdo em ordem reversa, como pode ser observado no exemplo a seguir

```
$ cat ShoppingList.txt
Milk
Eggs
Cheese
Tacos
$ tac ShoppingList.txt
Tacos
Cheese
Eggs
Milk
```

**Dica:** `tac` é muito útil para ler arquivos de log, onde as últimas entradas são as mais novas



## WC

- **Propósito:** contar o número de linhas, palavras e caracteres de um arquivo
- **Sintaxe:** `wc [options] [file]`
  - O comando `wc` (*Word Count*) mostra o número total de **linhas, palavras e caracteres** no arquivo especificado

```
$ wc /etc/hosts
10 28 251 /etc/hosts
```





## WC

- Usos comuns

Comando	Propósito
<code>wc [file]</code>	Mostrar o número de linhas, palavras e caracteres em um arquivo
<code>wc -l [file]</code>	Mostrar o número de linhas em um arquivo
<code>wc -w [file]</code>	Mostrar o número de palavras em um arquivo
<code>wc -c [file]</code>	Mostrar o número de caracteres em um arquivo



## more

- **Propósito:** mostrar a saída de um comando ou arquivo de texto uma página por vez
- **Sintaxe:** `more [options] [file]`
  - O comando `more` é útil para mostrar arquivos longos ou comandos que geram muitas linhas de saída; o exemplo a seguir mostrar a leitura de um arquivo uma página por vez

```
$ more /var/log/syslog
May 20 11:49:15 e6400 NetworkManager: SCPlugin-Ifupdown: (19599648)
... get_connections (managed=false): return empty list.
May 20 11:49:15 e6400 modem-manager: Loaded plugin Longcheer
May 20 11:49:15 e6400 modem-manager: Loaded plugin Option
May 20 11:49:15 e6400 modem-manager: Loaded plugin MotoC
May 20 11:49:15 e6400 modem-manager: Loaded plugin Option High-Speed
May 20 11:49:15 e6400 modem-manager: Loaded plugin Generic
May 20 11:49:15 e6400 modem-manager: Loaded plugin Gobi
May 20 11:49:15 e6400 modem-manager: Loaded plugin Nokia
May 20 11:49:15 e6400 modem-manager: Loaded plugin Novatel
May 20 11:49:15 e6400 modem-manager: Loaded plugin AnyData
May 20 11:49:15 e6400 modem-manager: Loaded plugin Huawei
May 20 11:49:15 e6400 modem-manager: Loaded plugin ZTE
--More--(14%)
```

**Dica:** espaço para navegar  
por página e *q* para sair



# more

- Usos comuns

Comando	Propósito
<code>more [file]</code>	Mostrar o arquivo especificado uma página por vez
<code>more +[num] [file]</code>	Começar a leitura do arquivo a partir da linha especificada
<code>[command]   more</code>	Mostrar a saída do comando uma página por vez



# less

- **Propósito:** mostrar a saída de um comando ou arquivo de texto uma página (ou linha) por vez
- **Sintaxe:** `less [options] [file]`
  - O comando `less` é similar ao comando `more`, exceto que suporta a navegação em ambas as direções (*up/down*)

```
$ less /var/log/syslog
May 20 11:49:15 e6400 NetworkManager: SCPlugin-Ifupdown: (19599648)
... get_connections (managed=false): return empty list.
May 20 11:49:15 e6400 modem-manager: Loaded plugin Longcheer
May 20 11:49:15 e6400 modem-manager: Loaded plugin Option
May 20 11:49:15 e6400 modem-manager: Loaded plugin MotoC
May 20 11:49:15 e6400 modem-manager: Loaded plugin Option High-Speed
May 20 11:49:15 e6400 modem-manager: Loaded plugin Generic
May 20 11:49:15 e6400 modem-manager: Loaded plugin Gobi
May 20 11:49:15 e6400 modem-manager: Loaded plugin Nokia
May 20 11:49:15 e6400 modem-manager: Loaded plugin Novatel
May 20 11:49:15 e6400 modem-manager: Loaded plugin AnyData
May 20 11:49:15 e6400 modem-manager: Loaded plugin Huawei
May 20 11:49:15 e6400 modem-manager: Loaded plugin ZTE
:
```

**Dica:** *page up/down* para navegar por páginas e *up/down* para navegar por linhas



# less

- Usos comuns

Comando	Propósito
Less [file]	Mostrar o arquivo especificado uma página por vez
less +[num] [file]	Começar a leitura do arquivo a partir da linha especificada
[command]   less	Mostrar a saída do comando uma página por vez



# head

- **Propósito:** mostrar as primeiras partes de um arquivo
- **Sintaxe:** head [options] [file]
  - O comando head mostras as primeiras linhas (cabeça) do arquivo especificado; a opção do exemplo a seguir é usada para mostrar apenas as duas primeiras linhas (por padrão são 10)

```
$ head -n 2 ShoppingList.txt  
Milk  
Eggs
```



# head

- Usos comuns

Comando	Propósito
head [file]	Mostrar as 10 primeiras linhas do arquivo
head -n [num] [file]	Mostrar as <i>num</i> primeiras linhas do arquivo



# tail

- **Propósito:** mostrar as últimas partes de um arquivo
- **Sintaxe:** `tail [options] [file]`
  - O comando `tail` mostra as últimas linhas (cauda) do arquivo especificado; a opção `-n` mostra apenas as últimas 2 linhas (por padrão são 10)

```
$ tail -n 2 ShoppingList.txt  
Cheese  
Tacos
```





# tail

- Usos comuns

Comando	Propósito
<code>tail [file]</code>	Mostrar as 10 últimas linhas do arquivo
<code>tail -n [num] [file]</code>	Mostrar as <i>num</i> últimas linhas do arquivo
<code>tail -f [file]</code>	Seguir um arquivo enquanto ele cresce



## tee

- **Propósito:** mostrar a saída de um comando e escrever a saída em um arquivo
- **Sintaxe:** `tee [options] [file]`
  - O comando `tee` mostra a saída de um comando e também salva a saída em um arquivo; por exemplo no comando a seguir que exibe a saída do comando `ls -l /etc` enquanto simultaneamente o salva no arquivo `etc.txt`

```
$ ls -l /etc/ | tee etc.txt
```

```
-rw-r--r-- 1 root root 2975 2009-02-04 11:12 adduser.conf
-rw-r--r-- 1 root root  46 2010-06-27 16:53 adjtime
-rw-r--r-- 1 root root 532 2009-11-30 14:33 aliases
-rw-r--r-- 1 root root 12288 2010-07-01 12:23 aliases.db
...
```



# tee

- Usos comuns

Comando	Propósito
<code>[command]   tee [file]</code>	Mostrar e salvar a saída de um comando para o arquivo
<code>[command]   tee -a [file]</code>	Anexar a saída ao arquivo especificado



# grep

- **Propósito:** casar padrões e filtrar dados
- **Sintaxe:** `grep [options] [file]`
  - O comando `grep` no exemplo a seguir filtra o arquivo `/var/log/syslog` e mostra apenas as linhas que possuem (casam) com a palavra *failed*

```
$ grep -i failed /var/log/syslog
Apr 4 07:52:44 kernel: [0.000000] Fast TSC calibration failed
Apr 4 07:52:44 kernel: [1.587770] PM: Resume from disk failed.
Apr 4 07:52:44 kernel: [6.252517] PM: Resume from disk failed.
Apr 11 12:11:28 init: Unable to connect to the system bus: Failed
to connect to socket /var/run/dbus/system_bus_socket: Connection refused
...
```

A opção `-i` faz casamento insensível a caixa (*case-insensitive*)



# grep

- Usos comuns

Comando	Propósito
grep [string] [file]	Mostrar linhas casadas de um arquivo
grep -c [string] [file]	Contar o número de linhas casadas em um arquivo
grep -i [strings] [file]	Ignorar a caixa nos casamentos
[command]   grep [string]	Filtrar a saída de um comando para casar com uma string



## sort

- **Propósito:** ordenar o conteúdo de um fluxo de dados de entrada ou arquivo
- **Sintaxe:** `sort [options] [file]`
  - O comando `sort` é usado para ordenar alfabeticamente um arquivo ou saída de um programa

```
$ cat ShoppingList.txt
Milk
Eggs
Cheese
Tacos
$ sort ShoppingList.txt
Cheese
Eggs
Milk
Tacos
```

**Dica:** o comando `uniq` pode ser combinado com `sort` para remover linhas repetidas de um arquivo



# sort

- Usos comuns

Comando	Propósito
sort [file]	Ordenar e mostrar o arquivo especificado
sort -r [file]	Ordenar em ordem reversa e mostrar o arquivo especificado
sort -n [file]	Ordenar usando ordenação numérica
[command]   sort	Ordenar a saída do comando



## zcat

- **Propósito:** ler o conteúdo de um arquivo compactado
- **Sintaxe:** `zcat [options] [file]`
  - O comando `zcat` permite ler o conteúdo de um arquivo compactado sem ter que descompactá-lo manualmente previamente; por exemplo para ler o arquivo compactado `ShoppingList.txt.gz` (compactado com `gzip`)

```
$ file ShoppingList.txt.gz
ShoppingList.txt.gz: gzip compressed data, was "ShoppingList.txt",
from Unix, last modified: Mon Apr 12 19:33:38 2010
$ zcat ShoppingList.txt.gz
Milk
Eggs
Cheese
Tacos
```

O que aconteceria se fosse usado o comando `cat` nesse arquivo?





# diff

- **Propósito:** Comparar arquivos
- **Sintaxe:** `diff [options] [file]`
  - O comando `diff` permite comparar dois arquivos de texto linha por linha e mostrar as diferenças entre eles; usa indicadores para marcar as diferenças
    - `<` indica o texto da linha no primeiro arquivo
    - `>` indica o texto da linha no segundo arquivo

```
$ diff ShoppingList.old ShoppingList.new
4c4
< Tacos
---
> Nachos
```

**Dica:** usar a opção `-u` para gerar um patch



# diff

- Usos comuns

Comando	Propósito
<code>diff [file1] [file2]</code>	Comparar dois arquivos e mostrar sua diferença
<code>diff -y [file1] [file2]</code>	Comparar dois arquivos lado a lado
<code>diff -u [file1] [file2]</code>	Gerar a diferença entre dois arquivos como um <i>patch</i>

# ISSO É TUDO PESSOAL!

---



Shell Script