Tópicos Especiais em Fundamentos da Computação
Shell Scripting

Processamento de Texto

Andrei Rimsa Álvares
andrei@cefetmg.br

CEFET-MG

DECOM

Comandos

- Comandos para processamento de arquivos texto

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

Comandos

- Comandos para processamento de arquivos texto

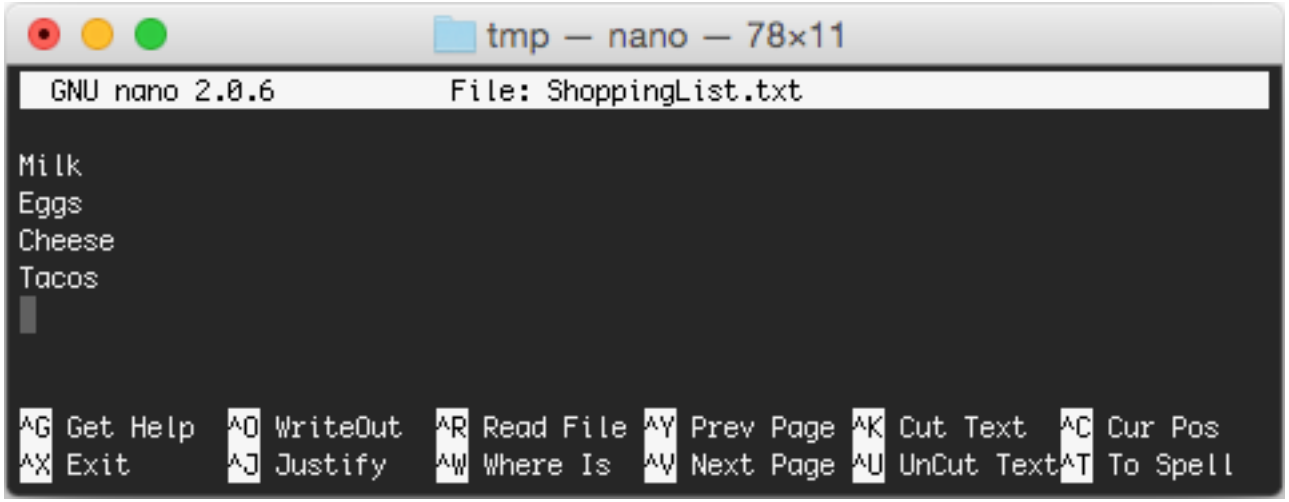
Comando	Propósito
`head`	Mostrar as primeiras partes de um arquivo
`tail`	Mostrar as últimas partes de um arquivo
`tee`	Mostrar a saída de um comando e escrever a saída em um arquivo
`grep`	Casar padrões e filtrar dados
`sort`	Ordenar o conteúdo de um fluxo de dados de entrada ou arquivo
`zcat`	Ler o conteúdo de um arquivo compactado
`diff`	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 texto simples encontrado na maioria dos sistemas Linux e BSD; é o editor mais recomendado para novos usuários



nano

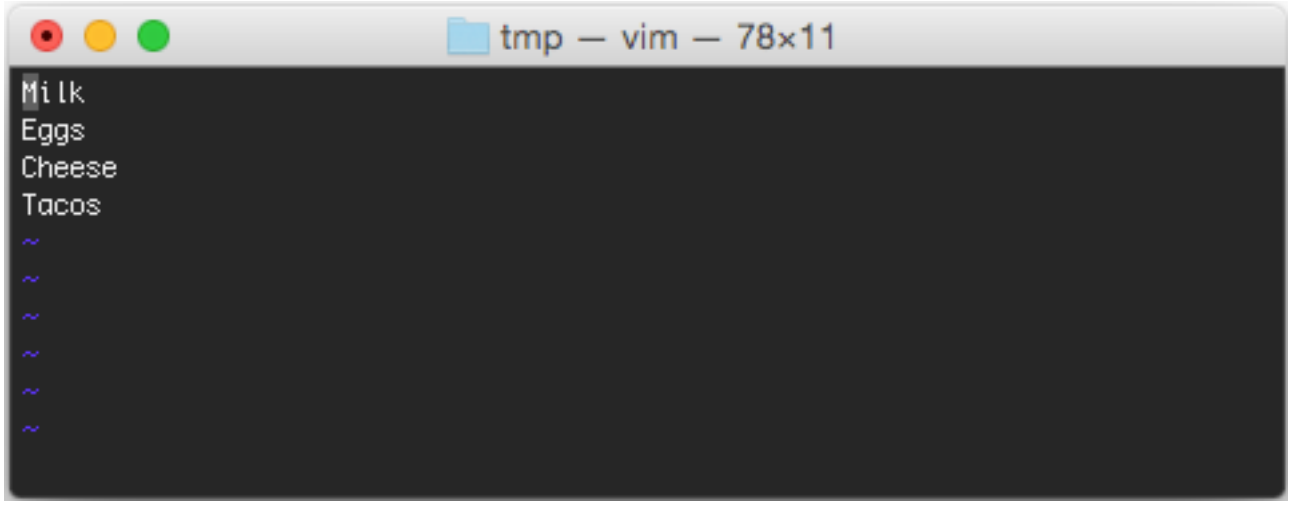
- As funcionalidades como procurar, salvar e fechar arquivos são controladas por teclas de funções, listadas embaixo da tela; algumas funções são listadas a seguir

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, em que normalmente é o editor padrão



vi/vim

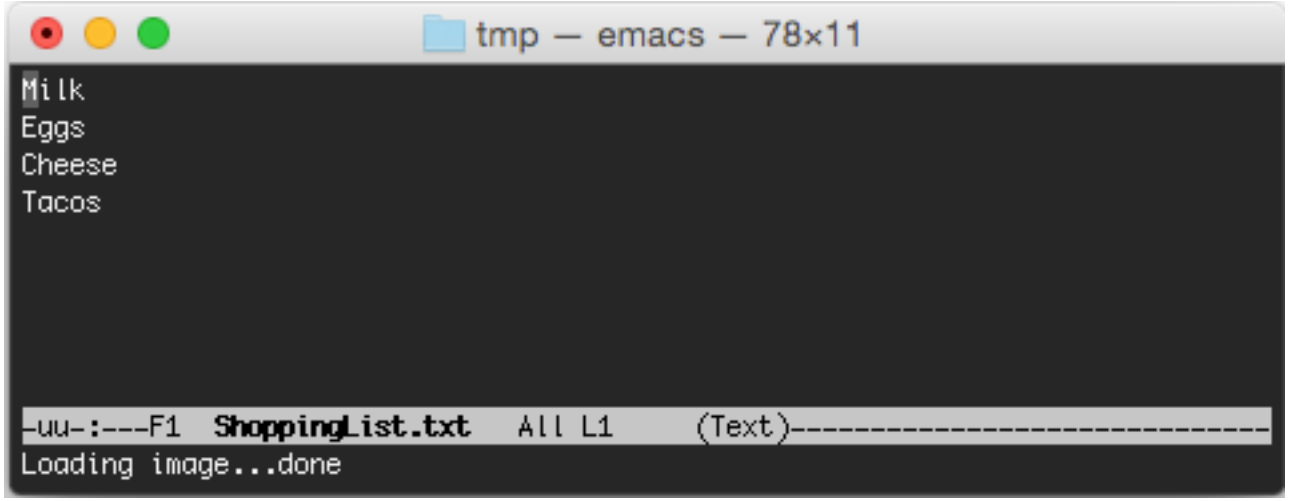
- O editor vi/vim possui dois modos de operação:
  modo de comando e **modo de edição**
  - Pressionar ESC ativa o **modo de comando**, em que as teclas de comando podem ser usadas para ativar edições específicas
  - No modo de comando, algumas teclas podem ser usadas para ativar o **modo de edição**

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 [options] [expression]`
  - `awk` trata cada linha da entrada como uma série de campos, em que 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 binários contêm dados que são ilegíveis por ferramentas padrões de processamento de texto, como arquivos `mp3`, `jpeg` e `mpeg`; o exemplo a seguir extrai 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 isto pode ser alterado pela opção `-n`

cat

- **Propósito:** concatenar arquivos e mostrar seu conteúdo
- **Sintaxe:** `cat [options] [files]`
  - O comando `cat` pode ser usado 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 adjacentes

tac

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

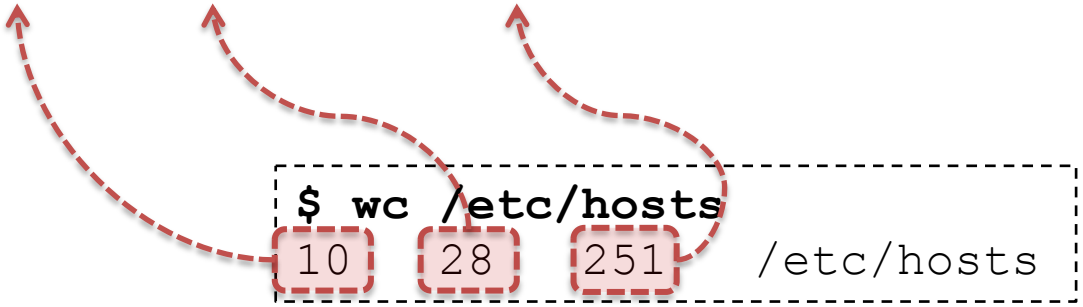
```
$ cat ShoppingList.txt
Milk
Eggs
Cheese
Tacos
$ tac ShoppingList.txt
Tacos
Cheese
Eggs
Milk
```

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

O que aconteceria se `tac` fosse usado com mais de um arquivo?

wc

- **Propósito:** contar o número de linhas, palavras e caracteres de um arquivo
- **Sintaxe:** `wc [options] [files]`
  - O comando `wc` (***W**ord **C**ount*) mostra o número total de **linhas**, **palavras** e **caracteres** no arquivo especificado



wc

- Usos comuns

Comando	Propósito
`wc [file]`	Mostrar o número de linhas, palavras e caracteres em um arquivo
`wc -l [file]`	Mostrar o número de linhas em um arquivo
`wc -w [file]`	Mostrar o número de palavras em um arquivo
`wc -c [file]`	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] [files]`
  - O comando `more` é útil para mostrar arquivos longos ou comandos que geram muitas linhas de saída; o exemplo a seguir mostra 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
`more [file]`	Mostrar o arquivo especificado uma página por vez
`more +[num] [file]`	Começar a leitura do arquivo a partir da linha especificada
`[command] | more`	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] [files]`
  - 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] [files]`
  - O comando `head` mostra 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 *num* primeiras linhas do arquivo

tail

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

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

tail

- Usos comuns

Comando	Propósito
`tail [file]`	Mostrar as 10 últimas linhas do arquivo
`tail -n [num] [file]`	Mostrar as *num* últimas linhas do arquivo
`tail -f [file]`	Seguir um arquivo enquanto ele cresce

tee

- **Propósito:** mostrar a saída de um comando e escrever a saída em um arquivo
- **Sintaxe:** `tee [options] [files]`
  - 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
`[command] | tee [file]`	Mostrar e salvar a saída de um comando para o arquivo
`[command] | tee -a [file]`	Anexar a saída ao arquivo especificado

grep

- **Propósito:** casar padrões e filtrar dados
- **Sintaxe:** `grep [options] [files]`
  - 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 [string] [file]`	Ignorar a caixa nos casamentos
`grep –v [string] [file]`	Inverte o casamento mostrando as linhas que não possuem o padrão
`[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] [files]`
  - 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] [files]`
  - 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] [file1] [file2]`
  - 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
`diff [file1] [file2]`	Comparar dois arquivos e mostrar sua diferença
`diff -y [file1] [file2]`	Comparar dois arquivos lado a lado
`diff -u [file1] [file2]`	Gerar a diferença entre dois arquivos como um *patch*

ISSO É TUDO, PESSOAL!