

# Tópicos Especiais em Linguagens de Programação

## Shell Script

## Comandos Básicos

Andrei Rimsa Álvares

*andrei@decom.cefetmg.br*



## Sumário

- Tipos de *shell*
- Shell interativa/não interativa
- Usando a *shell*
- Encontrando documentação
- Comandos básicos

## TIPOS DE SHELL

---

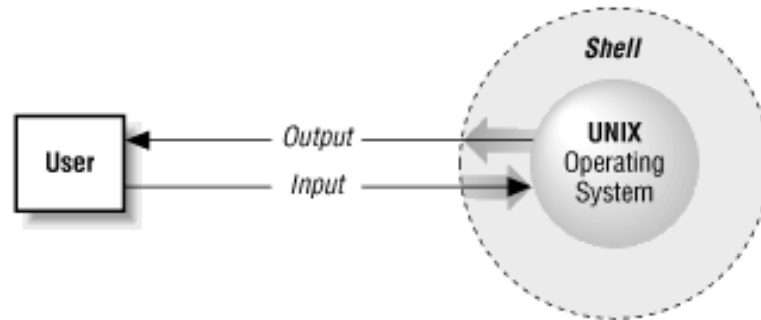


Shell Script

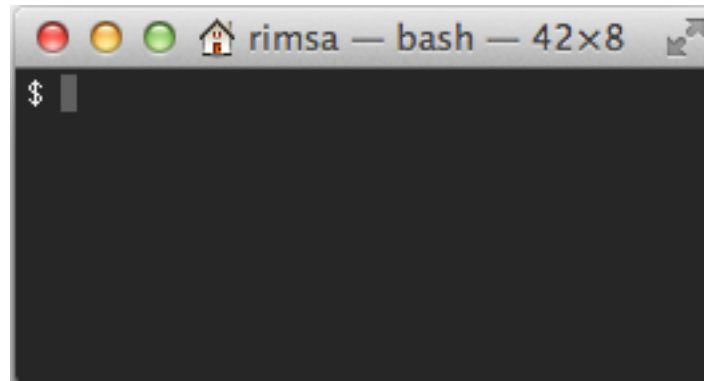


## O Que é Mesmo o Shell?

- O shell é um programa que faz a interface entre o usuário e o sistema operacional (*kernel*)



- A interface é dada através de entradas e saídas em formato textual



Existe mais de um tipo de shell?



## Tipos de Shell

- Existem diversos tipos de shell, sendo os mais famosos aqueles derivados do Bourne Shell distribuídos nos primeiros UNIX's
  - **Bourne Shell (old sh):** O shell que começou tudo. Foi a shell padrão no começo da disseminação do UNIX, tendo sido base de inspiração para as shells modernas encontradas atualmente
  - **POSIX shell (sh):** A shell base. Trouxe muitas inovações com relação a antiga. A padronização trouxe uma shell estável com todas as funcionalidades da padronização suportadas.
  - **Almquist shell (ash):** reimplementação razoavelmente compatível com as shell POSIX distribuídas principalmente em variantes BSD. A shell é bem pequena, mas é ainda POSIX-compatível. Também conhecida como a shell do Busybox



## Tipos de Shell

- Existem diversos tipos de shell, sendo os mais famosos aqueles derivados do Bourne Shell distribuídos nos primeiros UNIX's
  - **Bourne-again shell (bash):** desenvolvida pela GNU, é a maior e indiscutivelmente a mais completa shell. Têm um histórico agressivo de adoção de funcionalidades. É distribuída em sistemas Linux e Mac OS X
  - **Debian Almquist shell (dash):** é uma shell derivada da Almquist utilizado principalmente em sistemas Debian e derivados como Ubuntu. É uma shell pequena que possui uma implementação eficiente da shell portátil básica
  - **Korn shell (ksh):** desenvolvida na AT&T por David Korn. Foi uma das primeiras shells derivadas de Bourne a adicionar funcionalidades atualmente adotadas pela maioria das outras



## Tipos de Shell

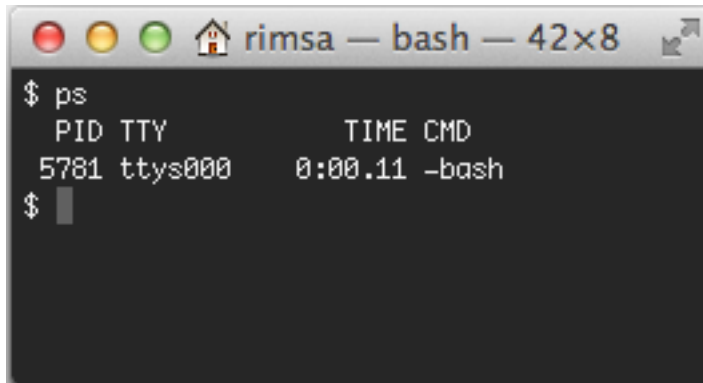
- Existem diversos tipos de shell, sendo os mais famosos aqueles derivados do Bourne Shell distribuídos nos primeiros UNIX's
  - **Public-domain Korn shell (pdksh):** antes da korn shell se tornar free software, essa foi um clone de domínio público. Embora possam haver algumas diferenças, são implementações no geral compatíveis
  - **Z shell (zsh):** é provavelmente a shell mais diferente de todas as outras listadas. Contudo, ela pode ser configurada como uma sólida shell POSIX. Em alguns sistemas, pode ser a única shell disponível capaz de executar comandos POSIX.

Qual dessas shells  
você está utilizando?



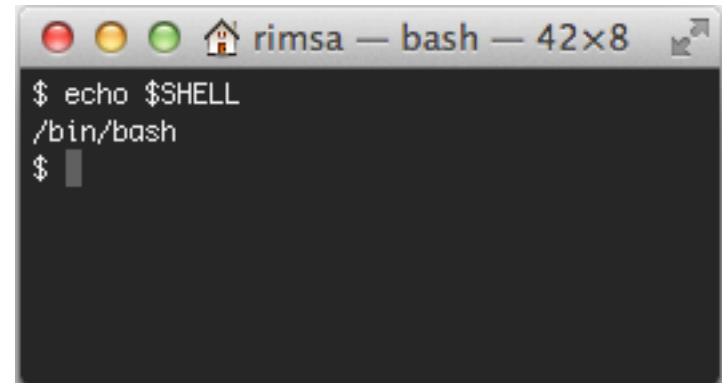
## Verificando sua Shell

- Como **bash** é distribuída em Linux e em Mac OS X, a probabilidade de você estar executando uma shell desse tipo é muito grande
- Mas você pode consultar de duas formas



```
rimsa — bash — 42x8
$ ps
  PID TTY          TIME CMD
  5781 ttys000    0:00.11 -bash
$
```

Pelo nome do processo



```
rimsa — bash — 42x8
$ echo $SHELL
/bin/bash
$
```

Consultando a variável \$SHELL



# SHELL INTERATIVA/NÃO INTERATIVA



Shell Script



## Shell Interativa

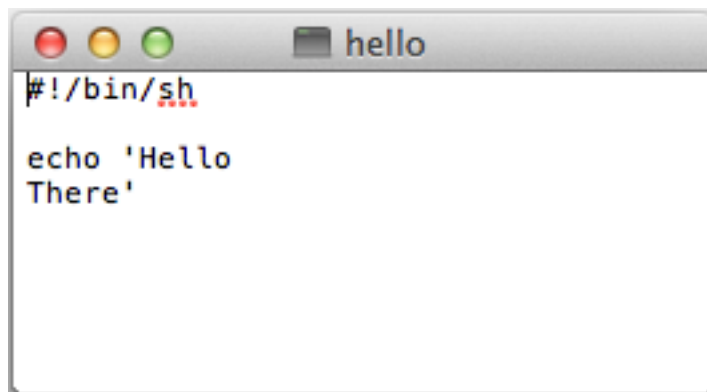
- No modo interativo, a shell indica que está pronta para receber comandos de entrada mostrando um texto chamado de **prompt**, normalmente indicado pelo símbolo \$ (dolar)
- Se a shell estiver esperando uma continuação do comando anterior, o prompt é modificado para um símbolo como > (maior)
- No modo interativo, normalmente a shell exibe a saída de um comando antes de mostrar o próximo prompt

```
$ echo 'Hello'
> There '
Hello
There
$
```

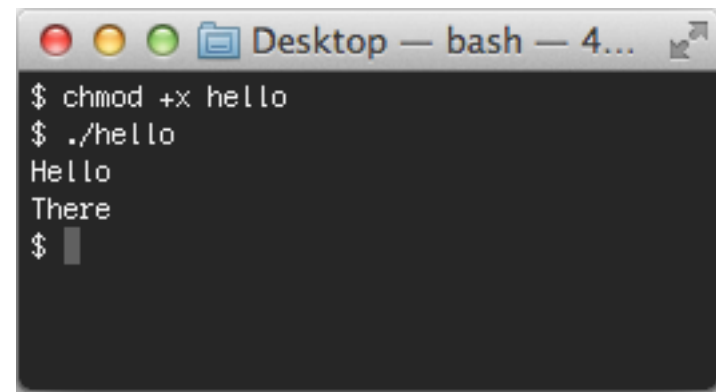


## Shell Não Interativa

- Quando a shell executa um comando dado de uma fonte não interativa, nenhum prompt é exibido
- Para isso, basta colocar os comandos em um arquivo (como hello) e
  - executá-lo em sua shell preferida (sh hello); ou
  - criar um shell script executável com o shebang (`#!/bin/sh`)



```
#!/bin/sh  
  
echo 'Hello  
There'
```



```
$ chmod +x hello  
$ ./hello  
Hello  
There  
$
```

## ENCONTRANDO DOCUMENTAÇÃO

---



Shell Script



## Documentação

- Normalmente distribuições Linux não vêm com cópias físicas com a documentação de referência
  - Em contrapartida, sua documentação online é uma de seus maiores pontos fortes
- As páginas de manuais e informações estão facilmente acessíveis através dos comandos `man` e `info` desde os primórdios da criação do sistema operacional
- Documentações podem ser encontradas
  - pela opção `--help` de comandos utilitários
  - pelo comando `man` (mostrar o manual do sistema)
  - pelo comando `apropos` (procurar por palavras chave)
  - pelo comando `info` (mostrar informações sobre utilitários)



## Opção `--help`

- A maioria das ferramentas GNU possuem a opção `--help` que disponibiliza informações sobre a ferramenta
  - Algumas ferramentas não-GNU podem usar as opções `-h` ou `-help` para mostrar informações de ajuda

- Exemplo:

**Dica:** Se a informação passar da tela, pode-se usar o paginador `less` com um pipe (`|`):

```
$ ls --help | pager
```

```
$ cat --help
```

```
Usage: cat [OPTION]... [FILE]...
```

```
Concatenate FILE(s), or standard input, to standard output.
```

```
-A, --show-all
```

```
equivalent to -vET
```

```
-b, --number-nonblank
```

```
number nonempty output lines
```

```
-e
```

```
equivalent to -vE
```

```
-E, --show-ends
```

```
display $ at end of each line
```

```
...
```



## man

- O utilitário man mostra páginas (man) da documentação do sistema em um ambiente textual
- Essa documentação é útil quando se sabe qual ferramenta usar, mas esqueceu exatamente como a usa
- Pode-se usar os manuais para obter mais informações sobre um determinado tópico ou funcionalidade disponível
- Como as descrições na documentação são normalmente concisas, elas são mais úteis se você já é familiarizado com as funcionalidades básica do utilitário

**Dica:** pode-se procurar por tópicos na documentação usando a ferramenta **apropos**.



# man

- Exemplo

```
$ man ls
```

O manual é exibido por um paginador (normalmente less) para auxiliar na navegação

```
rimsa — logytech@logytech: /usr/share/man — ssh — 80x24
LS(1)                                User Commands                                LS(1)
NAME
  ls - list directory contents

SYNOPSIS
  ls [OPTION]... [FILE]...

DESCRIPTION
  List information about the FILES (the current directory by default).
  Sort entries alphabetically if none of -cftuvSUX nor --sort.

  Mandatory arguments to long options are mandatory for short options
  too.

-a, --all
  do not ignore entries starting with .

-A, --almost-all
  do not list implied . and ..

--author
  with -l, print the author of each file

Manual page ls(1) line 1 (press h for help or q to quit)
```

O que é esse número no manual?





# man

- Baseado no FHS (*Filesystem Hierarchy Standard*), o manual de sistema do Linux é dividido em 10 seções
  - 1) **Comandos de usuários**
  - 2) Interfaces de programação para **chamadas de sistema** de kernel
  - 3) Interfaces de programação para a **biblioteca de C**
  - 4) **Arquivos especiais** tais como dispositivos e drivers
  - 5) **Formatos de arquivos**
  - 6) **Jogos e divertimentos**
  - 7) Variados (*miscellaneous*)
  - 8) **Comandos de administração do sistema**
  - 9) **Kernel**
  - 10) **New**

E como fazer quando existem vários manuais para um mesmo nome?

Esses dois são menos comuns



## man

- Para especificar um número de seção, usa-se o man da seguinte forma:

`$ man [seção] [termo]`

- Exemplo

`$ man 5 passwd`

```
rimsa — logytech@logytech: /usr/share/man/man4 — ssh — 80x24
PASSWD(5)                                File Formats and Conversions                                PASSWD(5)

NAME
  passwd - the password file

DESCRIPTION
  /etc/passwd contains one line for each user account, with seven fields
  delimited by colons ("::"). These fields are:

  • login name
  • optional encrypted password
  • numerical user ID
  • numerical group ID
  • user name or comment field
  • user home directory
  • optional user command interpreter

Manual page passwd(5) line 1 (press h for help or q to quit)
```

Se não especificar um número de seção, será obtida sempre a primeira instância do casamento, tipicamente na primeira seção



## apropos

- Quando não se sabe o nome de um comando que se deseja usar, pode-se usar o comando `apropos` com uma palavra-chave para buscar
  - Ele procura pela palavra na linha da descrição curta (linha de cima) de todas as páginas de manual

- Exemplo

O comando `man` quando chamado com a opção `-k` tem o mesmo efeito de `apropos`

```
$ apropos who
```

```
at.allow (5)
```

```
at.deny (5)
```

```
bsd-from (1)
```

```
from (1)
```

```
w (1)
```

```
w.procps (1)
```

```
who (1)
```

```
who-uploads (1)
```

```
whoami (1)
```

```
whodepends (1)
```

```
- determine who can submit jobs via at or batch
```

```
- determine who can submit jobs via at or batch
```

```
- print names of those who have sent mail
```

```
- print names of those who have sent mail
```

```
- Show who is logged on and what they are doing.
```

```
- Show who is logged on and what they are doing.
```

```
- show who is logged on
```

```
- identify the uploaders of Debian source packages
```

```
- print effective userid
```

```
- check which maintainers' packages depend on a package
```



## whatis

- O utilitário `whatis` é similar a ferramenta `apropos`, mas encontra apenas casamentos para palavras completas para o nome do utilitário pesquisado
- Exemplo

```
$ whatis who  
who (1)
```

- show who is logged on



# info

- A ferramenta de texto info é um sistema hipertexto baseado em menu desenvolvido pelo projeto GNU
  - A ferramenta é capaz de mostrar documentação de várias shells, utilitários e programas desenvolvidos pela GNU

\$ info coreutils

```
rimsa — logytech@logytech: /usr/share/man/man4 — ssh — 80x24
File: coreutils.info, Node: Top, Next: Introduction, Up: (dir)

GNU Coreutils
*****

This manual documents version 8.5 of the GNU core utilities, including
the standard programs for text and file manipulation.

Copyright (C) 1994-1996, 2000-2010 Free Software Foundation, Inc.

Permission is granted to copy, distribute and/or modify this
document under the terms of the GNU Free Documentation License,
Version 1.3 or any later version published by the Free Software
Foundation; with no Invariant Sections, with no Front-Cover Texts,
and with no Back-Cover Texts. A copy of the license is included
in the section entitled "GNU Free Documentation License".

* Menu:

* Introduction::          Caveats, overview, and authors
* Common options::       Common options
* Output of entire files:: cat tac nl od base64
--zz-Info: (coreutils.info.gz)Top, 333 lines --Top-----
Welcome to Info version 4.13. Type h for help, m for menu item.
```

O utilitário **info** mostra informações mais completas e atualizadas para ferramentas GNU do que páginas man



**CEFET-MG**

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

## COMANDOS BÁSICOS

---



**Shell Script**



# Comandos Básicos

Comando	Propósito
<b>ls</b>	Lista o conteúdo de um diretório
<b>pwd</b>	Mostra o diretório de trabalho atual
<b>cd</b>	Muda de diretório (navegação)
<b>tree</b>	Mostra o conteúdo de um diretório em um formato de árvore hierárquica
<b>find</b>	Procurar por arquivos e diretórios
<b>locate</b>	Procurar por arquivos e diretórios armazenados em um banco de dados
<b>whereis</b>	Mostra a localização de comandos, páginas de manual e código-fonte
<b>file</b>	Mostra o tipo do arquivo especificado
<b>stat</b>	Mostra informações detalhadas sobre arquivos e diretórios
<b>date</b>	Mostra ou define o relógio do sistema
<b>cal</b>	Mostra um calendário em linha de comando
<b>history</b>	Mostra os comandos executados recentemente
<b>clear</b>	Limpa o conteúdo da tela atual
<b>logout</b>	Logout do sistema
<b>exit</b>	Sair do terminal



# ls

- **Propósito:** listar o conteúdo de diretórios
- **Sintaxe:** `ls [options] [directories/files]`
  - Executar o comando `ls` faz com que sejam exibidos uma lista de arquivos no diretório atual

```
$ ls
```

```
Notes.txt ShoppingList.txt ToDoList.txt
```

- Executar o comando com a opção `-l` traz mais informações sobre os arquivos

```
$ ls -l
```

```
-rw-r--r-- 1 nick sales 35068 2009-05-19 08:41 Notes.txt  
-rw-r--r-- 1 nick sales 23 2009-05-19 08:43 ShoppingList.txt  
-rw-r--r-- 1 nick sales 37 2009-05-19 08:43 ToDoList.txt
```

Permissões

Links

Dono

Tamanho

Nome

Grupo

Data última  
modificação





# ls

- A maioria dos programas em linha de comando possuem numerosa quantidade de opções disponíveis, `ls` não é exceção
  - Por exemplo, pode-se combinar as opções `-a` para produzir saída detalhada incluindo arquivos ocultos (que começam com ponto)

```
$ ls -l -a
drwxr-xr-x  2 nick sales  4096 2009-05-19 21:14 .
drwxr-xr-x 92 nick sales  4096 2009-05-19 20:46 ..
-rw-r--r--  1 nick sales   168 2009-05-19 21:14 .MyHiddenFile
-rw-r--r--  1 nick sales 35068 2009-05-19 08:41 Notes.txt
-rw-r--r--  1 nick sales    23 2009-05-19 08:43 ShoppingList.txt
-rw-r--r--  1 nick sales    37 2009-05-19 08:43 ToDoList.txt
```

**Dica:** Muitas opções em linha de comando podem ser combinados usando uma notação abreviada, como por exemplo `ls -la`



# ls

- Usos comuns

Comando	Propósito
ls	Mostrar uma lista básica de arquivos no diretório atual
ls [directory]	Mostrar uma lista básica de arquivos no diretório especificado
ls -l	Listar arquivos com detalhes
ls -la	Listar arquivos incluindo arquivos ocultos
ls -lh	Listar arquivos em formato legível por humanos (KB, MB, ...)
ls -R	Recursivamente listar todos os subdiretórios
ls -d [directory]	Listar apenas o diretório especificado (não seu conteúdo)



## pwd

- **Propósito:** mostrar o diretório atual/de trabalho
- **Sintaxe:** pwd
  - O comando pwd (***P**rint **W**orking **D**irectory*) mostra sua atual localização no sistema de arquivos (nesse caso, o diretório de trabalho atual é `/home/nick`)

```
$ pwd  
/home/nick
```

Sistemas baseados em Unix usam **barra** (ex.: `/home/nick`) para separar diretórios ao invés de barra invertida usada por sistemas Windows (ex.: `c:\Windows\system32`)



## cd

- **Propósito:** alterar (navegar em) diretórios
- **Sintaxe:** cd [directory]
  - O comando cd (***C**hange **D**irectory*) muda a sua localização no sistema de arquivo para o caminho especificado (/etc nesse caso)

```
$ cd /etc
$ pwd
/etc
```

- Nesse outro caso, o diretório de partida é /home/nick e entrar no caminho Documents faz /home/nick/Documents o novo diretório

```
$ pwd
/home/nick
$ cd Documents
$ pwd
/home/nick/Documents
```



# cd

- Usos comuns

Comando	Propósito
cd [directory]	Navegar para o diretório especificado
cd	Navegar para o diretório pessoal ( <i>home</i> ) do usuário
cd -	Volta para o diretório de trabalho anterior
cd ..	Navega um nível acima na hierarquia de diretórios



```
$ cd /
$ tree -d -L 2

.
|-- bin
|-- boot
| `-- grub
|-- cdrom -> media/cdrom
|-- dev
| |-- block
| |-- bus
| |-- char
| |-- disk
|   |-- fd -> /proc/self/fd
|   |-- input
...
|-- etc
|   |-- ConsoleKit
|   |-- NetworkManager
|   |-- PolicyKit
| |-- X11
| |-- acpi
| |-- alsa
|   |-- alternatives
...
```

## tree

- **Propósito:** mostrar o conteúdo de um diretório em formato hierárquico de árvore
- **Sintaxe:** `tree [options] [directory]`
  - Esse comando é útil para visualizar o layout de uma estrutura de diretórios; no exemplo à esquerda o comando exibe dois níveis de diretórios a partir da localização atual



# tree

- Usos comuns

Comando	Propósito
<code>tree</code>	Mostrar o conteúdo do diretório atual em formato de árvore
<code>tree [directory]</code>	Mostrar o conteúdo do diretório especificado em formato de árvore
<code>tree -a</code>	Incluir arquivos ocultos na listagem em árvore
<code>tree -d</code>	Listar apenas diretórios
<code>tree -L [num]</code>	Listar um número específico de níveis de profundidade



# find

- **Propósito:** procurar por arquivos e diretórios
- **Sintaxe:** `find [path] [options] [criterias]`
  - O comando `find` faz uma busca bruta no sistema de arquivos para localizar os itens especificados, onde pode-se procurar por várias características como nome, dono, tamanho ou data de modificação; por exemplo procurar na raiz por arquivos cujo nome seja `hosts` ou procurar no diretório `/var` arquivos cujo dono seja `nick`

```
$ find / -name hosts  
/etc/avahi/hosts  
/etc/hosts  
/usr/share/hosts
```

```
$ find /var -user nick  
/var/mail/nick
```

Como `find` faz uma pesquisa bruta no sistema de arquivos, obter os resultados pode ser lento





# find

- Usos comuns

Comando	Propósito
<code>find [path] -name [name]</code>	Encontrar arquivos com o nome especificado
<code>find [path] -user [username]</code>	Encontrar arquivos cujo dono é o usuário especificado
<code>find [path] -size [filesize]</code>	Encontrar arquivos maiores que o tamanho especificado
<code>find [path] -mtime 0</code>	Encontrar arquivos modificados nas últimas 24 horas



## locate

- **Propósito:** procurar no banco de dados de localidades por arquivos
- **Sintaxe:** `locate [options] [directory/file]`
  - O comando **locate** mostra a localização de arquivos de nome especificado significativamente mais rápido que **find**, já que busca em um banco de dados indexado. Contudo, não é tão poderoso quanto **find**, já que não possui a habilidade de procurar por características avançadas como dono, tamanho, etc.

```
$ locate hosts  
/etc/avahi/hosts  
/etc/hosts  
/usr/share/hosts
```



## locate

- O banco de dados **locate** normalmente é atualizado diariamente via uma tarefa **cron** automaticamente agendada que indexa todos os arquivos do sistema de arquivos
  - Por padrão, isso acontece apenas uma vez por dia, o que significa que resultados não são atualizados imediatamente; novos arquivos e arquivos removidos podem não refletir nos resultados até o próximo agendamento de atualização
- É possível atualizar o banco de dados **locate** imediatamente com o comando **updatedb**, mas essa pode ser uma tarefa lenta

```
# updatedb  
#
```



# locate

- Usos comuns

Comando	Propósito
<code>locate [file]</code>	Localizar o arquivo especificado
<code>locate -i [file]</code>	Ignorar caixa ao procurar o arquivo



## whereis

- **Propósito:** Mostrar a localização de arquivos binários (programas), páginas de manual e código-fonte do comando especificado
- **Sintaxe:** `whereis [options] [command/file]`
  - O comando **whereis** mostra a localização do arquivo para o comando especificado; nesse exemplo, é mostrada a localização do arquivo binário e da página do manual para o comando `ls`

```
$ whereis ls  
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

**Dica:** o comando **which** é similar ao **whereis**, exceto que apenas mostra resultados para comandos binários



# whereis

- Usos comuns

Comando	Propósito
<code>whereis [command]</code>	Mostra a localização do comando especificado
<code>whereis -b [command]</code>	Mostra apenas programas binários
<code>whereis -m [command]</code>	Mostra apenas páginas de manual
<code>whereis -s [command]</code>	Mostra apenas código-fonte (se disponível)



# file

- **Propósito:** mostra o tipo de arquivo do arquivo especificado
- **Sintaxe:** `file [options] [file]`
  - Sistemas Windows frequentemente usam extensões para identificar o tipo de arquivos (.txt, .exe, ...), porém esse não é caso de sistemas Unix que raramente usam extensões; o comando `file` é fornecido para resolver esse problema

```
$ file /bin/bash
/bin/bash: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.15, stripped
$ file /etc/hosts
/etc/hosts: ASCII English text
$ file /home/nick/backup.tgz
backup.tgz: gzip compressed data, from Unix, last modified:
Tue May 19 22:29:29 2009
$ file /dev/cdrom
/dev/cdrom: symbolic link to 'sr0'
$ file /dev/sr0
/dev/sr0: block special
```



## stat

- **Propósito:** mostrar informações extras sobre um sistema de arquivos, arquivo ou diretório
- **Sintaxe:** `stat [options] [file/directory]`
  - O comando `stat` inclui informações úteis não disponível pelo comando `ls`, como último acesso ao arquivo e informações técnicas sobre a localização no sistema de arquivos; por exemplo, executando para o arquivo `/etc/hosts`

```
$ stat /etc/hosts
```

```
File: `/etc/hosts'
Size: 247          Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d Inode: 915725      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/   root)   Gid: (    0/   root)
Access: 2015-03-13 11:35:45.402391404 -0300
Modify: 2011-10-23 18:02:59.053788437 -0200
Change: 2011-10-23 18:02:59.053788437 -0200
```





## stat

- outro exemplo, executando para o diretório **/etc**

```
$ stat /etc
  File: `/etc'
  Size: 4096      Blocks: 8          IO Block: 4096   directory
Device: 801h/2049d Inode: 915713     Links: 97
Access: (0755/drwxr-xr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2014-08-08 22:12:53.430471299 -0300
Modify: 2015-03-14 09:18:48.244153374 -0300
Change: 2015-03-14 09:18:48.244153374 -0300
```

- Mais um exemplo executando para o sistema de arquivos na raiz (/)

```
$ stat -f /
  File: "/"
    ID: 1b1b2b74fffb1921b Namelen: 255      Type: ext2/ext3
Block size: 4096      Fundamental block size: 4096
Blocks: Total: 8650691    Free: 7881912    Available: 7442476
Inodes: Total: 2199344    Free: 2095302
```



# stat

- Usos comuns

Comando	Propósito
stat [file/dir]	Mostrar informações sobre o arquivo/diretório especificado
stat -f [filesystem]	Mostrar informações sobre o sistema de arquivo especificado



## date

- **Propósito:** mostrar ou ajustar o relógio do sistema
- **Sintaxe:** `date [options] [time/date]`
  - O comando **date** mostra a hora e data atual para o sistema local

```
$ date  
Sat Mar 14 11:07:55 BRT 2015
```

- A opção **-s** pode ser usada para definir a data/hora do sistema; usando o formato *MM/DD/YYYY HH:MM* para definir a data e hora, e *HH:MM* para definir apenas a hora

```
# date -s "07/10/2009 11:30"  
Fri Jul 10 11:30:00 CDT 2009
```

Para ajudar o relógio, é preciso executar o comando como superusuário (**root**)



# date

- Usos comuns

Comando	Propósito
date	Mostrar a data e hora do sistema
date -s [HH:MM]	Definir a hora do sistema
date -s [MM/DD/YYYY HH:MM]	Definir a data e hora do sistema



# cal

- **Propósito:** mostrar o calendário na linha de comando
- **Sintaxe:** `cal [options] [month] [year]`
  - O comando `cal` quando executado sem parâmetros mostra um calendário do mês atual; se adicionar um mês e/ou ano mostra o calendário para aquele mês e/ou ano especificado

```
$ cal
```

```
    March 2015
```

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

```
$ cal 8 2009
```

```
    August 2009
```

Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					



# cal

- Usos comuns

Comando	Propósito
cal	Mostrar o calendário para o mês atual
cal -m	Mostrar segunda-feira como o primeiro dia da semana
cal [month] [year]	Mostrar o calendário para o mês e ano especificado
cal [year]	Mostrar o calendário para o ano especificado
cal -y	Mostrar o calendário para o ano atual



# history

- **Propósito:** mostrar os comandos executados recentemente
- **Sintaxe:** `history [options]`
  - Executar o comando `history` sem argumentos mostra o histórico completo da linha de comando para o usuário atual; para uma lista mais compacta, pode-se especificar um número como argumento

```
$ history 10
686  man uptime
687  cat /etc/hosts
688  ls -l
689  uptime
690  dmesg
691  iostat
692  vmstat
693  ping google.com
694  tracepath google.com
695  history 10
```

**Dica:** pode-se executar um comando anterior usando o comando `![num]`, onde *num* é o número da linha do histórico que se deseja reexecutar

O histórico fica normalmente no arquivo `~/.bash_history`



# history

- Usos comuns

Comando	Propósito
<code>history</code>	Mostrar o histórico completo da linha de comando
<code>history [num]</code>	Mostrar o número de itens especificados do histórico
<code>history   grep [pattern]</code>	Procurar no histórico por um determinado padrão





# clear

- **Propósito:** limpar o conteúdo da tela
- **Sintaxe:** `clear`

**Dica:** pode-se usar o atalho CTRL+L no *bash* para limpar a tela

- Esse comando é útil para limpar a tela depois da execução de vários comandos e preparando para mover para a próxima tarefa

```
rimsa — logytech@logytech: ~ — bash — 6...
$ ls -l
total 8
drwx----- 2 rimsa staff 102 Oct 25 12:44 Applications
drwx-----+ 6 rimsa staff 2584 Mar 14 11:47 Desktop
drwx-----+ 6 rimsa staff 272 Mar 2 22:30 Documents
drwx-----+ 7 rimsa staff 9758 Mar 13 20:55 Downloads
drwx-----@ 7 rimsa staff 374 Mar 13 10:52 Dropbox
drwx-----@ 3 rimsa staff 850 Mar 11 14:24 Google Drive
drwx-----@ 55 rimsa staff 1904 Jan 8 13:08 Library
drwx-----+ 10 rimsa staff 442 Mar 14 08:26 Movies
drwx-----+ 5 rimsa staff 204 Oct 25 01:00 Music
drwxr-xr-x 2 rimsa staff 68 Oct 24 22:50 OneDrive
drwx-----+ 10 rimsa staff 408 Feb 8 20:15 Pictures
drwxr-xr-x 9 rimsa staff 374 Jan 29 19:58 Public
drwxr-xr-x 63 rimsa staff 2176 Mar 12 19:07 Series
drwxr-xr-x 2 rimsa staff 68 Jan 5 20:07 dwhelper
drwxr-xr-x 8 rimsa staff 272 Feb 23 17:06 git
drwxr-xr-x 31 rimsa staff 3502 Mar 9 14:40 tmp
$ clear
```

Antes

```
rimsa — logytech@logytech: ~ — bash — 6...
$
```

Depois



# logout

- **Propósito:** *logout* do sistema
- **Sintaxe:** `logout`
  - O comando `logout` irá terminar sua sessão do terminal e tornar para a tela de *login*

```
$ logout
```

```
Ubuntu 9.04
```

```
login:
```

Alguns sistemas podem ter o arquivo `.logout` ou `.bash_logout` em cada diretório pessoal de usuários com comandos que serão executados no *logout*; normalmente usado para limpar o ambiente

## Dicas:

- 1) o comando *logout* é a forma recomendada para sair da *shell*
- 2) Por razões de segurança, usuários com acesso administrativo devem ser *deslogar* de qualquer sessão aberta de terminal quando fora de sua mesa



## exit

- **Propósito:** sair da shell atual
- **Sintaxe:** `exit [code]`
  - O comando `exit` é similar ao `logout`, mas não executa o script de *logout*; *shell scripts* tipicamente usam `exit` para terminar enquanto usuário usam `logout` para sair do sistema

```
$ exit
```

```
login:
```

Dentro de um terminal virtual em um ambiente gráfico deve-se usar o comando `exit`, ao invés de `logout`.

**Dica:** pode-se usar o atalho `CTRL+D` para sair da *shell*

# ISSO É TUDO PESSOAL!

---



Shell Script