



Trabalhando na linha de comando

Sumário

Capítulo 1

Trabalhando na linha de comando	3
1.1. Mãos a obra.....	4

Capítulo 2

Gerenciando	5
2.1. Objetivos.....	5
2.1. Troubleshooting.....	5

Índice de tabelas

Índice de Figuras

Capítulo 1

Trabalhando na linha de comando

- Usar comandos de shell;
- Sequencia de linha de comando para executar tarefas básicas;
- Usar e editar o histórico de comandos;
- Os comandos invoke dentro e fora do caminho definido.

1.1. Mãos a obra

O ambiente em modo texto em distribuições GNU/Linux é conhecido com Shell, e é o local onde os comandos digitados pelos usuários são interpretados. É possível configurar esse ambiente automatizando pequenas tarefas do dia a dia, e ainda fazer uso de comandos internos que o próprio ambiente nos trás.



Mas afinal o que é um shell?

O shell é um programa que permite ao usuário interagir com o sistema operacional através de comandos digitados no teclado. No MS-DOS o shell era o command.com, que permitia executar alguns comandos como cd, dir, etc.

O shell mais conhecido no mundo GNU/Linux é o bash, o padrão para novos usuários quando são criados no sistema. É possível verificar qual o seu shell atual, através do comando finger ou da variável SHELL. Vamos a pratica:



```
$ finger aluno | grep Shell
```

```
Directory: /home/aluno
```

```
Shell: /bin/bash
```



```
$ echo $SHELL
```

```
/bin/bash
```

Em nosso exemplo o usuário aluno esta utilizando o shell bash, mas é possível ver a lista de outros shells exibindo o conteúdo do arquivo /etc/shells



```
$ cat /etc/shells
```

```
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
/bin/bash
/bin/rbash
```

O bash traz muitas funcionalidades como comandos internos, histórico de comandos, autocompletar, variáveis de ambiente, etc. Para você exibir quais comandos são internos use help como no exemplo abaixo:



```
$ help
```

```
GNU bash, version 3.2.39(1)-release (i486-pc-linux-gnu)
These shell commands are defined internally.  Type 'help' to see this list.
Type 'help name' to find out more about the function 'name'.
Use 'info bash' to find out more about the shell in general.
Use 'man -k' or 'info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.
```

```
JOB_SPEC [&]                (( expression ))
. filename [arguments]      :
[ arg... ]                  [[ expression ]]
alias [-p] [name[=value] ... ]  bg [job_spec ...]
bind [-lpsPVS] [-m keymap] [-f fi break [n]
builtin [shell-builtin [arg ...]] caller [EXPR]
case WORD in [PATTERN [| PATTERN]) cd [-LI-P] [dir]
command [-pVv] command [arg ...] compgen [-abdefgjkuv] [-o option
complete [-abdefgjkuv] [-pr] [-o continue [n]
declare [-afFrtx] [-p] [name[=val dirs [-clpv] [+N] [-N]
disown [-h] [-ar] [jobspec ...] echo [-neE] [arg ...]
enable [-pnds] [-a] [-f filename] eval [arg ...]
exec [-cl] [-a name] file [redirec exit [n]
export [-nf] [name[=value] ...] or false
fc [-e ename] [-nrl] [first] [last fg [job_spec]
for NAME [in WORDS ... ;] do COMMA for (( exp1; exp2; exp3 )); do COM
```

Histórico de comandos

Uma das funções mais uteis no dia a dia é trabalhar com comandos do histórico. Você pode acessar esses comandos usando as teclas de navegação para cima e para baixo, ou através do comando history. Vamos a prática:



```
$ history
```

```
1 ls
2 date
3 cal
4 clear
5 history
```

A lista é uma ordem de comandos que estão guardados no histórico. Para executar um comando da lista use exclamação + numero do comando. Exemplo:



```
$ !4
```

Para limpar a lista de comando use o comando history com a flag -c:



```
$ history -c
```

Variáveis

No GNU/Linux as variáveis são muito usadas na criação de shell scripts, mas também podem ser declarada diretamente no terminal, e assim gerenciadas pelo shell.



Mas o que é variável?

Variável pode ser definida como um objeto, ou uma posição localizada na memória, que guarda um valor ou expressão. Vamos ver na prática como declarar uma variável.



```
$ a=10
```

Veja que em nosso exemplo a variável “a” foi declarada com o valor 10. Para exibir o conteúdo da variável use o comando echo, cifrão e o nome da variável.



```
$ echo $a
```

É possível somar o conteúdo de 2 ou mais variáveis usando alguns comandos, vamos a prática:

Declare um conteúdo para 2 variáveis:



```
$ b=20
```



```
$ c=30
```

Para somar use o comando expr ou o próprio echo. Exemplo:



```
$ expr $b + $c
```



```
$ echo $((b+c))
```

Variáveis locais x globais

Quando você declara uma variável ela pode ser considerada pelo sistema como local ou global. A diferença está na maneira de declarar a variável. Exemplo:

Variável local



```
$ curso=4linux
```

Variável global



```
$ export curso=4linux
```

Veja que o comando `export` foi usado antes de declarar a variável. Para listar os tipos de variáveis use os comandos `env` e `set`:

Para exibir variáveis locais use o comando `set`



```
$ set
```

Para exibir variáveis globais use o comando `env`



```
$ env
```

Para deletar uma variável da memória use o comando `unset`



```
$ unset curso
```


Variáveis de ambiente

O shell guarda informações do ambiente dentro de algumas variáveis, chamadas de variáveis de ambiente. Veja a descrição abaixo:

HOME - Exibe o diretório do usuário logado;

SHELL - Exibe qual shell está sendo usado;

TERM - Exibe o tipo de terminal que está sendo usado;

USER - Exibe o nome do usuário logado;

PATH - Exibe quais diretórios pesquisar e a ordem na qual eles são pesquisados para encontrar um determinado comando;

MAIL - Exibe o local onde ficam armazenados os emails do usuário logado;

OSTYPE - Exibe o tipo de sistema operacional em uso;

PWD - Exibe a localização do diretório atual;

OLDPWD - Exibe a localização do diretório anterior;

PS1 - Exibe a aparência do prompt, como o nome de usuário, máquina e diretório atual;



Não esqueça que os nome das variáveis de ambiente são apresentadas em maiúsculas!



```
$ echo $PS1
```

Outros comandos interessantes usado no shell

uname

O comando `uname` exibe informações do sistema como a versão do kernel, processador, sistema operacional, entre outros. Veja a descrição das opções usadas com o comando `uname`:

- i - Tipo de processador.
- m - Arquitetura da maquina.
- n - Nome da maquina na rede.
- p - Processador.
- o - Sistema operacional.
- r - Versão do código fonte do kernel.
- s - Nome do kernel.
- v - Versão de compilação do kernel.

`exec`

Alterna de um shell para outro, exemplo do `bash` para `sh`



```
$ exec sh
```

`man`

Exibe o manual de um comando. Exemplo de como trazer o manual do comando `ls`



```
$ man ls
```

Capítulo 2

Gerenciando

2.1. Objetivos

- Troubleshooting: Usar e modificar o ambiente shell.

2.1. Troubleshooting



Como posso personalizar meu ambiente shell?

Cada usuário pode personalizar seu ambiente através do shell, declarando variáveis, apelidos para comandos (alias) e ainda executar comandos ou scripts no login e logout. Veja a lista de arquivos que podem ser personalizados.

/etc/profile - Este arquivo contém comandos que são executados para todos os usuários do sistema no momento do login (somente o usuário pode editar);

~/.bash_profile - Executado por shells que usam autenticação (nome e senha). Para root o arquivo é o **.profile**;

~/.bashrc - Executado por shells que não requerem autenticação (seção de

terminal no X);

~/bash_logout - É lido e executado toda vez que saímos de um shell;

~/bash_history - Lista dos comandos digitados pelos usuários.

Como exemplo pratico vamos personalizar o login de um usuário para exibir um calendário, alem de declarar uma variável e criar um alias para um comando.

Com um usuário comum faça login no sistema e abra o arquivo .bashrc



`$ vim .bashrc`

```

1 # ~/.bashrc: executed by bash(1) for non-login shells.
2 # see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
3 # for examples
4
5 # If not running interactively, don't do anything
6 [ -z "$PS1" ] && return
7
8 # don't put duplicate lines in the history. See bash(1) for more options
9 # don't overwrite GNU Midnight Commander's setting of 'ignorespace'.
10 export HISTCONTROL=$HISTCONTROL${HISTCONTROL+,}ignoredups
11 # ... or force ignoredups and ignorespace
12 export HISTCONTROL=ignoreboth
13
14 # append to the history file, don't overwrite it
15 shopt -s histappend
16
17 # for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
18
19 # check the window size after each command and, if necessary,
20 # update the values of LINES and COLUMNS.
21 shopt -s checkwinsize
22
23 # make less more friendly for non-text input files, see lesspipe(1)
24 #[ -x /usr/bin/lesspipe ] && eval "$(SHELL=/bin/sh lesspipe)"

```

Na linha 4 adicione um comando, linha 5 um alias e na linha 6 uma variável

```

1 # ~/.bashrc: executed by bash(1) for non-login shells.
2 # see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
3 # for examples
4 cal
5 alias ll="ls -lah --color"
6 curso=4linux
7 # If not running interactively, don't do anything
8 [ -z "$PS1" ] && return

```

Faça logout com o usuário e ao se logar teste o alias e a variável.

```
Debian GNU/Linux 5.0 maq2 tty1
```

```
maq2 login: aluno
```

```
Password: _
```



```
$ ll
```

```

drwxr-xr-x 14 aluno aluno 4,0K Set  1 19:01 .
drwxr-xr-x  7 root  root  4,0K Set  1 20:43 ..
-rw-----  1 aluno aluno   374 Set  1 19:23 .bash_history
-rw-r--r--  1 aluno aluno    15 Ago 21 00:19 .bash_logout
-rw-r--r--  1 aluno aluno   3,1K Ago 21 00:20 .bashrc
drwx-----  3 aluno aluno  4,0K Feb 24  2010 .dbus
drwxr-xr-x  2 aluno aluno  4,0K Feb 24  2010 Desktop
-rw-----  1 aluno aluno    28 Feb 24  2010 .dmrc
-rw-r--r--  1 aluno aluno   786 Ago 20 15:42 fstab
drwx-----  4 aluno aluno  4,0K Feb 24  2010 .gconf
drwx-----  2 aluno aluno  4,0K Feb 24  2010 .gconfd
drwxr-xr-x  3 aluno aluno  4,0K Feb 24  2010 .gnome
drwx-----  7 aluno aluno  4,0K Feb 24  2010 .gnome2
drwx-----  2 aluno aluno  4,0K Feb 24  2010 .gnome2_private
drwx-----  2 aluno aluno  4,0K Feb 24  2010 .gnupg
drwxr-xr-x  2 aluno aluno  4,0K Feb 24  2010 .gststreamer-0.10
-rw-----  1 aluno aluno   159 Feb 24  2010 .ICEauthority
-rw-r--r--  1 aluno aluno    12 Ago 20 15:42 .message
drwx-----  3 aluno aluno  4,0K Feb 24  2010 .metacity
drwxr-xr-x  3 aluno aluno  4,0K Feb 24  2010 .nautilus
-rw-r--r--  1 aluno aluno   675 Feb 24  2010 .profile
drwx-----  2 aluno aluno  4,0K Feb 24  2010 .ssh
-rw-----  1 aluno aluno   665 Set  1 19:01 .viminfo
-rw-r--r--  1 aluno aluno   818 Feb 24  2010 .xsession-errors

```



```
$ echo $curso
```