

## Introdução à bash

### Utilização

José Pedro Oliveira  
(jpo@di.uminho.pt)

Grupo de Sistemas Distribuídos  
Departamento de Informática  
Escola de Engenharia  
Universidade do Minho

Sistemas Operativos  
2006-2007

José Pedro Oliveira

Interpretador de comandos  
Execução de comandos  
Redirecção  
Histórico da linha de comando

Introdução à bash

Interpretores de comandos  
Ficheiros de configuração  
Caracteres especiais  
Variáveis de ambiente

José Pedro Oliveira

Interpretador de comandos  
Execução de comandos  
Redirecção  
Histórico da linha de comando

Introdução à bash

Interpretores de comandos  
Ficheiros de configuração  
Caracteres especiais  
Variáveis de ambiente

## Conteúdo

- 1 Interpretador de comandos
  - Interpretadores de comandos
  - Ficheiros de configuração
  - Caracteres especiais
  - Variáveis de ambiente
- 2 Execução de comandos
  - Variável de ambiente PATH
  - Código de saída
  - Combinação de comandos
  - Execução de tarefas em background e jobs
- 3 Redirecção
- 4 Histórico da linha de comando

## Interpretores de comandos

### Interpretador de comandos

Programa que aceita comandos do teclado e os executa. A **bash** é um exemplo de um interpretador de comandos UNIX.

### Funções de um interpretador de comandos

- fornecer um interface de linha de comando
- realizar redirecção de Entrada/Saída (I/O - Input/Output)
- realizar substituição de nome de ficheiros
- realizar substituição de variáveis
- fornecer uma linguagem de programação interpretada

José Pedro Oliveira

Introdução à bash

## Interpretores de comandos de *login* válidos

### Interpretador de comandos de *login*

É o especificado no ficheiro `/etc/passwd`.

### Ficheiro `/etc/shells`

Ficheiro de texto que contem os caminhos absolutos (*full pathnames*) de interpretadores de comandos de *login* válidos.

### `$ cat /etc/shells`

```
/bin/sh  
/bin/bash  
/sbin/nologin  
/bin/bash2  
/bin/ash  
/bin/bsh  
/bin/tcsh  
/bin/csh
```

José Pedro Oliveira

Introdução à bash

## Arrancar e terminar uma shell

### Arrancar outra shell

Basta invocar o binário da shell pretendida. Exemplos:

- sh
- csh

Nota: O prompt apresentado varia conforme o tipo de shell.

### Terminar

- logout (*se shell de login*)
- exit
- CTRL+D (*marca de fim de ficheiro*)

## Caracteres especiais

### Caracteres especiais

white space	caracteres (espaços e tabs) usados para separar argumentos
newline	indica o fim de uma linha de comando
' " \	caracteres de citação; permitem alterar a maneira de como a shell interpreta caracteres especiais
&	no fim de um comando indica à shell para correr esse comando em <i>background</i>
< > >>	caracteres de redirecção
* ? [ ] [^	substituição de caracteres em nomes de ficheiros
\$	indica a presença de uma variável
;	usado para separar comandos numa mesma linha

## Ficheiros de configuração da bash

### Ficheiros de configuração globais

- /etc/profile
- /etc/profile.d/\*.sh

### Ficheiros de configuração pessoais

- ~/.bash\_profile
- ~/.bashrc
- ~/.bash\_logout

O carácter '~' representa a *homedir* do utilizador.

## Caracteres de citação (quoting)

### Caracteres de citação

- \ - retira o significado especial ao carácter seguinte
- '...' - retira o significado especial de todos os caracteres especiais delimitados pelo carácter "'"
- "..." - retira o significado especial de todos os caracteres especiais delimitados pelo carácter "" com a excepção do carácter '\$', ou seja, permite que variáveis sejam interpoladas.

## Exemplos

```
$ echo $USER
```

```
jpo
```

```
$ echo "$USER"
```

```
jpo
```

```
$ echo \ $USER
```

```
$USER
```

```
$ echo '$USER'
```

```
$USER
```

## Variáveis de ambiente: comando printenv

### Comando externo printenv

Permite listar variáveis de ambiente. Quando invocado sem opções lista o nome e valor de cada variável de ambiente.

```
$ printenv EDITOR
```

```
/usr/bin/vim
```

## Variáveis de ambiente: comando set

### Comando interno set

Permite manipular variáveis de ambiente. Quando invocado sem opções lista o nome e valor de cada variável de ambiente.

```
$ set
```

```
BASH=/bin/bash
```

```
BASH_COMPLETION=/etc/bash_completion
```

```
BASH_COMPLETION_DIR=/etc/bash_completion.d
```

```
BASH_VERSION='3.00.14(1)-release'
```

```
COLORS=/etc/DIR_COLORS.xterm
```

```
...
```

## Variáveis de ambiente: comando echo

### Comando interno echo

Permite enviar para o STDOUT linhas de texto. Por omissão força uma mudança de linha e não interpola sequências de escape.

### Algumas opções

-e - interpola as sequências de escape (exemplo: \t)

-n - suprime a mudança de linha

```
$ echo "Utilizador: $USER"
```

```
Utilizador: jpo
```

```
$ echo -e "a\tb"
```

```
a      b
```

## Conteúdo

- 1 Interpretador de comandos
  - Interpretadores de comandos
  - Ficheiros de configuração
  - Caracteres especiais
  - Variáveis de ambiente
- 2 Execução de comandos
  - Variável de ambiente PATH
  - Código de saída
  - Combinação de comandos
  - Execução de tarefas em background e jobs
- 3 Redirecção
- 4 Histórico da linha de comando

José Pedro Oliveira

Introdução à bash

## Variável de ambiente PATH

### Variável de ambiente PATH

Contem lista de directórios que o interpretador pesquisa para encontrar comandos externos.

```
$ echo $PATH
```

```
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/jpo/bin
```

### Comando which

Mostra o caminho completo (*pathname*) de comandos.

```
$ which perl
```

```
/usr/bin/perl
```

José Pedro Oliveira

Introdução à bash

## Execução de um comando

### Execução de um comando

- 1 esperar que o utilizador introduza um comando
- 2 realizar certas tarefas se o comando contiver caracteres especiais (reservados)
- 3 se for um comando interno, executá-lo. Saltar para o ponto 1.
- 4 encontrar o executável do comando (externo). Se o ficheiro não for encontrado gerar uma mensagem de erro.
- 5 criar um processo filho que irá executar o comando
- 6 esperar que o processo termine e retornar ao início da lista

José Pedro Oliveira

Introdução à bash

## Código de saída

### Código de saída (exit code)

Praticamente todas as invocações de comandos geram um número inteiro *código de saída* que pode ser utilizado para modificar como um outro comando é executado. Para a grande maioria de comandos um código de saída zero indica sucesso. Problemas são indicados através de valores diferentes de zero.

### Variável \$?

**\$? = 0** - sucesso

**\$? != 0** - problemas

José Pedro Oliveira

Introdução à bash

## Código de saída POSIX

### Códigos de saída POSIX

0	comando terminou com sucesso
> 0	erro
1 - 125	comando não terminou com sucesso; o significado dos códigos de saída são definidos por cada comando
126	comando encontrado, mas o ficheiro não era executável
127	comando não encontrado
> 128	comando terminou devido à recepção de um sinal

## Combinação de comandos

### Combinação de comandos

`cmd` - executa o comando **cmd**

`cmd &` - executa o comando **cmd** em *background*

`cmd1 ; cmd2` - execução sequencial de comandos

`(cmd1 ; cmd2) | cmd3` - o *output* dos comandos **cmd1** e **cmd2** é enviado para o *standard input* do comando **cmd3**

`cmd1 && cmd2` - o comando **cmd2** só é executado se e só se o comando **cmd1** tiver terminado correctamente (código de saída igual a zero)

`cmd1 || cmd2` - o comando **cmd2** só é executado se e só se o comando **cmd1** tiver terminado incorrectamente (código de saída diferente de zero)

## Código de saída: exemplos

```
$ date; echo $?
```

```
Mon Apr 5 21:36:13 WEST 2004
0
```

```
$ rm ficheiro_inexistente; echo $?
```

```
rm: cannot lstat 'ficheiro_inexistente': No such
file or directory
1
```

## Combinação de comandos: exemplos

```
$ ( echo "Bom dia"; echo "Boa tarde" ) | grep dia
```

```
Bom dia
```

```
$ make && make test && make install
```

```
...
(output omitido)
...
```

```
$ rm ficheiro_inexistente 2> /dev/null || echo "Falhou"
```

```
Falhou
```

## Execução de tarefas em *background* e *jobs*

### Executar tarefas em *background*

terminar a linha de comando com o caracter '&'

### Jobs

CTRL+Z - suspender execução do processo em *foreground*

jobs - ver tarefas

fg - enviar tarefa para *foreground*

bg - enviar tarefa para *background*

## Conteúdo

- 1 Interpretador de comandos
  - Interpretadores de comandos
  - Ficheiros de configuração
  - Caracteres especiais
  - Variáveis de ambiente
- 2 Execução de comandos
  - Variável de ambiente PATH
  - Código de saída
  - Combinação de comandos
  - Execução de tarefas em background e jobs
- 3 Redirecção
- 4 Histórico da linha de comando

## Exemplos

### Executar as seguintes operações

- 1 correr o **acroread**
- 2 suspender o **acroread** (CTRL+Z)
- 3 listar as tarefas (*jobs*)
- 4 colocar o **acroread** a executar em *background*

### Executar as seguintes operações

- 1 correr o **vi**
- 2 suspender o **vi** (CTRL+Z)
- 3 listar as tarefas (*jobs*)
- 4 retomar a execução do **vi** em *foreground*

## Redirecção

### Ficheiros abertos por omissão

**stdin** - descritor número 0  
(o teclado)

**stdout** - descritor número 1  
(o ecrã)

**stderr** - descritor número 2  
(as mensagens de erro também são enviadas para o ecrã)

## Redirecção

### Redirecção

- > - redirecção do *standard output*
- >> - redirecção do *standard output* em modo de *append*
- < - redirecção do *standard input*
- n> - redirecção do n-ésimo descritor
- &> e >& - redirecção do *standard output* e do *standard error*
- n>&m - redirecção do descritor n para o descritor m (uso típico: 2>&1)
- | - pipe: o *standard output* é redireccionado para o *standard input* de um segundo processo
- `...` - backticks: a sequência é substituída pelo resultado do comando (delimitado pelos caracteres ``)

## Exemplo de utilização de *backticks*

Determinar que RPM contem o interpretador de Perl

```
$ which perl
/usr/bin/perl
```

```
$ rpm -qf /usr/bin/perl
perl-5.8.8-10
```

A sequência de comandos acima pode ser reduzida a apenas um único

```
$ rpm -qf `which perl`
perl-5.8.8-10
```

## Exemplos

```
$ ls -lR > ls-lR.txt
```

Redirecciona a saída do programa **ls** para o ficheiro **ls-lR.txt**.

```
$ cat /etc/inittab | less
```

Redirecciona a saída do programa **cat** para a entrada do programa **less** (paginador).

```
$ gcc -Wall -pedantic ola.c 2> erros.txt
```

Redirecciona a saída de erros (*standard error*) do programa **gcc** para o ficheiro **erros.txt**.

## Conteúdo

- 1 Interpretador de comandos
  - Interpretadores de comandos
  - Ficheiros de configuração
  - Caracteres especiais
  - Variáveis de ambiente
- 2 Execução de comandos
  - Variável de ambiente PATH
  - Código de saída
  - Combinação de comandos
  - Execução de tarefas em background e jobs
- 3 Redirecção
- 4 Histórico da linha de comando

## Histórico da linha de comando

### Comando `history`

Lista as últimas entradas do histórico de comandos

### Algumas opções

- c - limpa o histórico
- d *n* - limpa a entrada na posição *n*

### Ficheiro de log do histórico

- `~/bash_history`

## Histórico da linha de comando

```
$ history
```

```
...  
1007 pdflatex bash.tex  
1008 man bash  
1009 gvim bash.tex  
1010 acroread bash.pdf  
1011 history
```

```
$ !1009
```

(re-executa o comando 1009 do histórico)

```
$ !pd
```

(re-executa o último comando começado pelos caracteres "pd")

## Utilização do histórico

### Utilização do histórico

teclas navegação - cursores, Page Up/Down, ...

!! - re-executa o último comando

!*n* - re-executa o comando número *n*

!*n* - re-executa o comando *n* linhas atrás

!*string* - re-executa o último comando começado por *string*

!*string*? - re-executa o último comando que contem *string*

^*str1*^*str2*^ - re-executa o último comando substituindo previamente *str1* por *str2*

CTRL+R - pesquisa de comandos no histórico

## Conteúdo

5 Referências



## Referências

## Documentação

• **GNU Bash**

<http://www.gnu.org/software/bash/manual/bash.html>

## Bibliografia

• **Learning the bash Shell (terceira edição)**

<http://www.oreilly.com/catalog/bash3/>

• **Classic Shell Scripting**

<http://www.oreilly.com/catalog/shellsrptg/>

• **Advanced Bash-Scripting Guide**

<http://www.tldp.org/LDP/abs/html/>