Introdução ao Unix

José Orlando Pereira
Departamento de Informática
Universidade do Minho

"Introdução ao Unix"
Edição de 18 de setembro de 2025.

Conteúdo

Capítulo 1

Introdução

1.1 Interface de linha de comando

Além das interfaces gráficas atualmente muito utilizadas, é tradicional nos sistemas UNIX a existência de uma interface de linha de comando, aliás, tal como acontece com o sistema WINDOWS. No entanto, ao contrário do que acontece com a o sistema WINDOWS, esta interface de linha de comando continua a ser frequentemente preferida. As razões para que isto aconteça são fundamentalmente:

- a interface de linha de comando dos sistemas UNIX é mais flexível e poderosa que a do WINDOWS;
- os utilizadores de UNIX são normalmente mais exigentes e como tal não podem restringir-se à simplicidade da interface gráfica;
- os sistemas UNIX são frequentemente usados em servidores, dispositivos e outros equipamentos dedicados em que não faz sentido instalar interfaces gráficas.

O poder da interface de linha de comando do UNIX resulta da conjugação de dois fatores:

- a existência de uma grande variedade de comandos, cada um deles simples, que faz apenas uma tarefa, embora a faça bem;
- a existência de mecanismos para combinar esses comandos simples de modo a desempenhar tarefas complexas.

Contraste-se esta filosofia com o que é vulgar em termos de interfaces gráficas, que facilitam o desempenho de algumas tarefas pré-definidas, mas

que dificilmente podem ser adaptadas a tarefas mais complicadas ou fora do comum. É também raro que diversas ferramentas possam ser combinadas para automaticamente desempenhar uma tarefa complexa.

1.2 Convenções tipográficas

Neste texto apresentam-se assim os resumos das páginas de manual dos comandos UNIX discutidos:

---

date

`date` – mostra a data/hora do sistema

`-s` *`data`* modifica a data/hora do sistema de acordo com o parâmetro

---

As componentes do comando que são fixas são apresentadas em `espaçamento fixo`. As componentes que são dados escolhidos pelo utilizador são apresentadas em *`itálico de espaçamento fixo`*.

Os exemplos de utilização dos mesmos comandos são apresentados assim:

Exemplo 1 *Este é um exemplo da utilização de um comando, neste caso de* `date`*:*

```
$ date
Mon May 17 15:15:26 WET DST 1999
$ date -s 15:30
$ date
Mon May 17 15:30:05 WET DST 1999

$ _
```

Nestes exemplos, é simulado o aspeto de um terminal interativo. No entanto, para facilitar a compreensão, o texto escrito pelo utilizador é apresentado em *`itálico de espaçamento fixo`* sendo os resultados devolvidos pelo sistema apresentados em `espaçamento fixo`.

O carácter `$` é *prompt* normal no sistema UNIX para utilizadores normais, ou seja, todos menos o administrador do sistema.

Capítulo 2

Utilização básica

2.1 Interface básica

Uma interface de linha de comando num sistema UNIX (*shell*), indica ao utilizador que está disponível para receber um comando afixando uma solicitação (*prompt*) que normalmente termina com o caráter `$`, por exemplo:

```
[jop@jetpac:aulas]$ _
```

A informação contida nesta solicitação normalmente inclui o nome do utilizador e o nome da máquina em que estamos a trabalhar, neste caso, respetivamente `jop` e `jetpac`. Esta informação é útil porque é comum ter ao mesmo tempo múltiplas sessões em diferentes servidores e com diferentes utilizadores. Um caso especial é o utilizador `root`, que é o superutilizador ou administrador, e tem autorização para fazer operações de manutenção da máquina, por exemplo, a instalação de *software*.

A informação apresentada inclui também o nome da diretoria de trabalho atual, neste caso, `aulas`. Esta informação desempenha o mesmo papel que a indicação da pasta atual no explorador gráfico: identifica os ficheiros que são visualizados e sobre os quais podemos operar diretamente.

2.2 Comandos e argumentos

Os comandos mais comuns permitem-nos observar e manipular os ficheiros na diretoria de trabalho atual. Em primeiro lugar, para observar o conteúdo dessa diretoria usamos:

```
$ ls
fich1.txt fich2.pdf info.pdf
$ _
```

É normal que o comportamento de cada comando seja modificado através de argumentos: palavras adicionais que são escritas a seguir ao comando. Por exemplo, no caso de precisarmos mais detalhe sobre os ficheiros, usamos o argumento `-l` que inclui dados como o nome do dono do ficheiro, data de modificação e tamanho:

```
$ ls -l
total 12
-rw-r-r-.  1 jop jop 31 set 20 15:06 fich1.txt
-rw-r-r-.  1 jop jop 83 set 20 15:06 fich2.pdf
-rw-r-r-.  1 jop jop 177 set 20 15:14 info.pdf
$ _
```

Muitos comandos aceitam como argumento o nome das diretorias ou dos ficheiros sobre os quais vão operar. Por exemplo, para listar o conteudo da diretoria `/tmp` em vez da diretoria atual podemos usar:

```
$ ls /tmp
tmp.1M8DB2 tmp.XH45A2
$ _
```

Sempre que um comando aceita nomes de ficheiros como argumentos, podemos usar o caráter especial $*$, que corresponde a qualquer sequência de carateres, para filtrar vários ficheiros que nos interessem. Por exemplo:

```
$ ls *.pdf
fich2.pdf info.pdf
$ ls *.txt
fich1.txt
$ ls fich*
fich1.txt fich2.pdf
$ _
```

Finalmente podemos renomear, copiar ou remover ficheiros com os seguintes comandos:

```
$ mv fich1.txt ficheiro1.txt
$ ls
fich2.pdf ficheiro1.txt info.pdf
$ cp ficheiro1.txt novo.txt
$ ls
fich2.pdf ficheiro1.txt info.pdf novo.txt
$ rm fich2.pdf
$ ls
```

```
ficheiro1.txt info.pdf novo.txt
$ _
```

2.3 Diretorias

Os sistemas operativos atuais organizam os ficheiros em diretorias, que nas interfaces gráficas são visualizadas como pastas. Existe pois um conjunto de comandos para manipular estas diretorias mas também para navegar entre elas, selecionando qual é a diretoria de trabalho atual. Esta navegação é feita com o comando `cd` que recebe como argumento único o nome da diretoria desejada. Considere a seguinte diretoria:

```
$ ls -l
total 4
-rw-r-r-.  1 jop jop 177 set 20 15:14 info.pdf
drwxr-xr-x.  1 jop jop 12 set 20 15:48 trabalho
$ _
```

Neste caso observamos que na diretoria atual temos um ficheiro (`info.pdf`) e uma subdiretoria (`trabalho`), assinalada pela letra `d` no início da linha. Podemos então navegar para essa diretoria e observar o seu conteúdo com os comandos:

```
$ cd trabalho
$ ls
main.c
$ _
```

Além de um nome relativo à diretoria de trabalho atual, podemos especificar um nome absoluto começando a partir da raiz e indicado pelo caráter `/` no início, por exemplo, `/home/jop/Desktop`. Existem ainda outros atalhos úteis:

- `cd ..` muda para a diretoria acima da atual, ou seja, tem o efeito equivalente à seta para cima no navegador da interface gráfica.
- `cd -` muda para a diretoria anterior, ou seja, tem o efeito equivalente à seta para trás no navegador da interface gráfica.
- `cd ~` ou simplesmente `cd` muda para a diretoria do utilizador, ou seja, tem o efeito equivalente à casa no navegador da interface gráfica.

Finalmente, é possivel criar novas diretorias com o comando `mkdir` e remover diretorias vazias com o comando `rmdir`.

Capítulo 3

Manipulação de texto

3.1 Introdução

Este capítulo é dedicado a um conjunto de comandos UNIX cujo objetivo é manipular texto. Por texto compreende-se texto ASCII simples, sem caracteres especiais de formatação exceto a separação em linhas, tal como é produzido pelos editores como o `vi`, `emacs` ou `pico` e não por processadores de texto.

A importância destes comandos decorre do facto do formato texto ASCII ser utilizado para interligar os diversos comandos UNIX. Ou seja, tal como os circuitos integrados respeitam um norma comum para os níveis elétricos de modo a que possam ser interligados e funcionar em conjunto, os utilitários UNIX utilizam texto ASCII para trocar informação.

Os comandos apresentados neste capítulo manipulam texto de um forma abstracta, ou seja, independentemente do modo como foi produzido ou qual é o seu significado. São úteis em duas situações:

- quando a operação abstrata sobre o texto corresponde diretamente a um operação que se quer realizar, por exemplo, a seleção do nome completo de um utilizador a partir do ficheiro que contém a descrição das contas;
- como "cola" para composição de vários comandos, quando o formato de texto produzido por um deles não é exatamente o formato esperado pelo seguinte.

Convém sublinhar que grande parte da informação guardada em disco em sistemas UNIX é guardada em formato de texto, nomeadamente a informação respeitante à configuração do sistema, o que torna as ferramentas de manipulação de texto em úteis e versáteis ferramentas de admi-

nistração do sistema. Por exemplo, os utilizadores autorizados de uma máquina estão enumerados no ficheiro `/etc/passwd`, que pode ser listado utilizando o comando `cat`[1]:

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
mail:*:8:12:mail:/var/spool/mail:
operator:*:11:0:operator:/root:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
jop:x:652:200:Jose Orlando Pereira:/home/linux/jop:/bin/bash
outro:x:653:200:Outro Utilizador:/home/linux/outro:/bin/bash
maisum:x:654:200:Mais um:/home/linux/maisum:/bin/bash
$ _
```

Neste ficheiro cada linha contém a identificação do utilizador, a *password*, o número do utilizador, o número do grupo, o nome completo, a área de trabalho e o nome do interpretador de comandos (*shell*). Serve também como base para grande parte dos exemplos apresentados neste capítulo.

3.2 Seleção de linhas

A operação possivelmente mais simples que se pode fazer sobre um texto é a seleção de algumas das suas linhas. Esta seleção pode ser feita segundo vários critérios, mas fundamentalmente pode distinguir-se:

- a seleção em função da posição;
- a seleção em função do conteúdo.

Os comandos mais comuns de seleção de linhas pela sua posição são os comandos `head` e `tail`.

head

`head` – seleciona as primeiras linhas de um ficheiro

`-n` *n* é o número de linhas selecionadas

Exemplo 2 *Seleção das primeiras 10 linhas do ficheiro* `/etc/passwd`*:*

[1]Este comando é semelhante ao `type` do WINDOWS, na medida em que se limita a apresentar o conteúdo do ficheiro no terminal.

```
$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
mail:*:8:12:mail:/var/spool/mail:
operator:*:11:0:operator:/root:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
jop:x:652:200:Jose Orlando Pereira:/home/linux/jop:/bin/bash
$ _
```

Exemplo 3 *Seleção das primeiras 5 linhas do ficheiro* `/etc/passwd`*:*

```
$ head -5 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:

$ _
```

Uma utilização possível para o comando `head` é como alternativa ao `cat`, de modo a apresentar as primeiras linhas de um ficheiro extenso de modo a evitar que rolem para cima.

tail

`tail` – seleciona as ultimas linhas de um ficheiro

- `-n` *n* é o número de linhas a contar do fim do ficheiro, que são selecionadas
- `+n` *n* é o número de linhas a contar do início do ficheiro, que não são selecionadas

Exemplo 4 *Seleção das últimas linha do ficheiro* `/etc/passwd`*:*

```
$ tail -1 /etc/passwd
maisum:x:654:200:Mais um:/home/linux/maisum:/bin/bash
$ _
```

Exemplo 5 *Seleção das últimas linhas do ficheiro* `/etc/passwd` *a partir da décima linha:*

```
$ tail +10 /etc/passwd
jop:x:652:200:Jose Orlando Pereira:/home/linux/jop:/bin/bash
```

```
outro:x:653:200:Outro Utilizador:/home/linux/outro:/bin/bash
maisum:x:654:200:Mais um:/home/linux/maisum:/bin/bash
$ _
```

O comando `tail` é bastante útil para inspecionar os ficheiros de registo de mensagens de erro[2] em que as últimas linhas de cada ficheiro correspondem às últimas mensagens de erro registadas.

A seleção com base no conteúdo da linha é feita usando o comando `grep`. Este comando imprime as linhas que contenham algures a expressão desejada. Esta expressão pode ser uma expressão fixa ou uma expressão regular[3]

grep

`grep` – seleciona todas as linhas que contêm uma expressão

- `-v` inverte o funcionamento normal, selecionando linhas que não contêm a expressão
- `-n` apresenta também o número de cada linha
- `-l` seleciona o nomes do ficheiros que contenham a expressão

Exemplo 6 *Seleção da linha contendo a informação relativa ao utilizador `jop`:*

```
$ grep jop /etc/passwd
jop:x:652:200:Jose Orlando Pereira:/home/linux/jop:/bin/bash
$ _
```

Exemplo 7 *Seleção da linha contendo a informação relativa ao utilizador `jop` indicando qual a sua posição no ficheiro:*

```
$ grep -n jop /etc/passwd
10:jop:x:652:200:Jose Orlando Pereira:/home/linux/jop:/bin/bash
$ _
```

Exemplo 8 *Seleção de todos os verdadeiros utilizadores da máquina, ou seja, aqueles cujo grupo é 200, que neste caso corresponde ao grupo `users`:*

```
$ grep :200:  /etc/passwd
jop:x:652:200:Jose Orlando Pereira:/home/linux/jop:/bin/bash
```

[2]Normalmente conhecidos como ficheiros de *log* e armazenados na diretoria `/var/log/` nos sistemas LINUX.

[3]Uma expressão regular é um modo de especificar expressões a procurar de um modo semelhante ao uso dos carácteres `*` e `?` usados na linha de comando para selecionar múltiplos ficheiros. As expressões regulares são também vulgarmente usadas em outros sítios, como no editor de texto `vi`.

```
outro:x:653:200:Outro Utilizador:/home/linux/outro:/bin/bash
maisum:x:654:200:Mais um:/home/linux/maisum:/bin/bash
$ _
```

Exemplo 9 *Seleção das contas usadas para administração da máquina, ou seja, aqueles cujo grupo não é o 200:*

```
$ grep -v :200:  /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
mail:*:8:12:mail:/var/spool/mail:
operator:*:11:0:operator:/root:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
$ _
```

O comando `grep` pode também ser utilizado para selecionar os ficheiros que contêm uma determinada expressão, por oposição ao seu uso mais vulgar, em que seleciona linhas.

Exemplo 10 *Seleção dos nomes de todos os ficheiros na diretoria `cartas/` que contêm a expressão cumprimentos:*

```
$ grep -l "Cumprimentos"cartas/*
carta1.tex
carta5.tex
$ _
```

3.3 Seleção de colunas

Além da seleção de linhas de um texto, quando a informação está organizada em matriz, como por exemplo, no ficheiro `/etc/passwd` que faz a correspondência entre cada conta e a sua descrição ou no ficheiro `/etc/fstab` que faz a correspondência entre cada disco e o seu nome lógico, é frequentemente útil selecionar apenas algumas colunas de modo a recolher apenas alguma atributos.

Estes dois ficheiros são também exemplos de diferentes maneiras de organizar a informação por colunas. No primeiro caso, a separação é feita com base na posição relativa ao princípio da linha. Por exemplo, a coluna relativa ao tipo de sistema de ficheiros encontra-se em cada linha entre

os carateres 49 e 56. No segundo caso, a informação encontra-se separado pelo carácter `:`, por exemplo, o nome completo de cada utilizador é a quinta coluna, ou seja, a informação entre o quarto e o quinto carácter `:` de cada linha.

Exemplo 11 *Ficheiro de texto em que a separação em colunas é feita pela posição da informação:*

```
$ cat /etc/fstab
/dev/hda1            /                    ext2    defaults        1 1
/dev/hda3            /home/ftp/pub        ext2    defaults        1 2
/dev/hda2            swap                 swap    defaults        0 0
/dev/fd0             /mnt/floppy          msdos   noauto,user     0 0
/dev/cdrom           /mnt/cdrom           iso9660 noauto,ro       0 0
none                 /proc                proc    defaults        0 0
none                 /dev/pts             devpts  mode=0622       0 0
$ _
```

Exemplo 12 *Ficheiro de texto em que a separação em colunas é feita utilizando um carácter separador, neste caso `:` (dois pontos):*

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
mail:*:8:12:mail:/var/spool/mail:
operator:*:11:0:operator:/root:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
jop:x:652:200:Jose Orlando Pereira:/home/linux/jop:/bin/bash
outro:x:653:200:Outro Utilizador:/home/linux/outro:/bin/bash
maisum:x:654:200:Mais um:/home/linux/maisum:/bin/bash
$ _
```

cut

`cut` – seleciona colunas

`-c` *`colunas`*	escolhe quais as colunas com base na sua posição
`-f` *`colunas`*	escolhe quais as colunas com base num separador
`-d` *`separador`*	indica qual o carácter separador, caso seja diferente do carácter especial TAB que é usado por omissão

Em ambos os casos, a indicação das colunas a selecionar é feita tendo em conta que:

- várias colunas individuais podem ser selecionadas com uma lista separado por vírgulas;

- várias colunas consecutivas podem ser selecionadas indicando os extremos separados por um traço;
- quando na especificação de colunas consecutivas se omite o inicio ou fim da seleção, é usada respetivamente a primeira ou última coluna.

Exemplo 13 *Nomes completos dos utilizadores, obtidos cortando a quinta coluna do ficheiro* `/etc/passwd`*:*

```
$ cut -d:  -f5 /etc/passwd
root
bin
daemon
adm
lp
mail
operator
FTP User
Nobody
Jose Orlando Pereira
Outro Utilizador
Mais um
$ _
```

Exemplo 14 *Identificação, nome completo e* shell *preferida de cada utilizador, obtidos cortando as colunas 1, 5 e 7 do ficheiro* `/etc/passwd`*:*

```
$ cut -d:  -f1,5,7 /etc/passwd
root:root:/bin/bash
bin:bin:
daemon:daemon:
adm:adm:
lp:lp:
mail:mail:
operator:operator:
ftp:FTP User:
nobody:Nobody:
jop:Jose Orlando Pereira:/bin/bash
outro:Outro Utilizador:/bin/bash
maisum:Mais um:/bin/bash
$ _
```

Exemplo 15 *Toda a informação de cada utilizador exceto a* password*, obtidos cortando todas as colunas exceto a segundo do ficheiro* `/etc/passwd`*:*

```
$ cut -d:  -f1,3- /etc/passwd
root:0:0:root:/root:/bin/bash
bin:1:1:bin:/bin:
```

```
daemon:2:2:daemon:/sbin:
adm:3:4:adm:/var/adm:
lp:4:7:lp:/var/spool/lpd:
mail:8:12:mail:/var/spool/mail:
operator:11:0:operator:/root:
ftp:14:50:FTP User:/home/ftp:
nobody:99:99:Nobody:/:
jop:652:200:Jose Orlando Pereira:/home/linux/jop:/bin/bash
outro:653:200:Outro Utilizador:/home/linux/outro:/bin/bash
maisum:654:200:Mais um:/home/linux/maisum:/bin/bash
$ _
```

Exemplo 16 *Listagem dos nomes lógicos dos discos montados e qual o modo usado, recortando as segunda e quarta colunas do ficheiro* `/etc/fstab`*:*

```
$ cut -c25-48,57-72 /etc/fstab
/                       defaults
/home/ftp/pub           defaults
swap                    defaults
/mnt/floppy             noauto,user
/mnt/cdrom              noauto,ro
/proc                   defaults
/dev/pts                mode=0622
$ _
```

Exemplo 17 *A utilização do comando* `cut -c` *quando as colunas não estão perfeitamente alinhadas é normalmente inútil. Por exemplo, qualquer tentativa de retirar informação relevante do ficheiro* `/etc/passwd` *com este comando não é bem sucedida:*

```
$ cut -c10-20 /etc/passwd
0:root:/roo
:bin:/bin:
2:2:daemon:
:adm:/var/a
lp:/var/spo
12:mail:/va
:11:0:oper
50:FTP User
99:99:Nobod
:200:Jose O
53:200:Outr
654:200:Mai
$ _
```

Exemplo 18 *A utilização do comando* `cut -f` *quando as colunas não estão delimitadas por um carácter único e bem definido é também é normalmente inútil. Por exemplo, qualquer tentativa de retirar informação relevante do ficheiro* `/etc/fstab` *com este comando não é bem sucedida:*

```
$ cut -d/ -f3 /etc/fstab
hda1
hda3
hda2               swap                    swap    defaults        0 0
fd0
cdrom

pts                devpts  mode=0622       0 0
$ _
```

3.4 Colagem de linhas

Ao contrário dos comandos vistos até agora, que apenas permitem selecionar partes de um texto, às vezes é necessário o inverso, ou seja, colar vários textos para formar um texto maior. Para o efeito pode ser utilizado o já conhecido comando `cat`, que concatena todos os ficheiros que lhe são indicados na linha de comando e os apresenta no terminal. Obviamente, quando apenas se indica um único ficheiro, o `cat` limita-se a apresentar esse ficheiro.

Exemplo 19 *Apresentação do ficheiro `/etc/fstab` com uma linha de legenda retirada de um ficheiro chamado `legenda`:*

```
$ cat legenda
# device             mountpoint              type    options         order
# ------------------ ----------------------- ------- --------------- -----
$ cat legenda /etc/fstab
# device             mountpoint              type    options         order
# ------------------ ----------------------- ------- --------------- -----
/dev/hda1            /                       ext2    defaults        1 1
/dev/hda3            /home/ftp/pub           ext2    defaults        1 2
/dev/hda2            swap                    swap    defaults        0 0
/dev/fd0             /mnt/floppy             msdos   noauto,user     0 0
/dev/cdrom           /mnt/cdrom              iso9660 noauto,ro       0 0
none                 /proc                   proc    defaults        0 0
none                 /dev/pts                devpts  mode=0622       0 0
$ _
```

3.5 Colagem de colunas

Também é possível fazer a colagem de colunas, mais uma vez:

- através da sua posição, ou seja, cada linha de um ficheiro é colada com a mesma linha de outro ficheiro;
- através do seu conteúdo, ou seja, cada linha de um ficheiro é colada com uma linha de outro ficheiro que contenha a mesma expressão.

Exemplo 20 *Colagem lado a lado de dois ficheiros,* `fich1` *e* `fich2`*, usando o carácter* `:` *como separador:*

```
$ cat fich1
aaa
bbb
ccc
$ cat fich2
111
222
333
$ paste -d:  fich1 fich2
aaa:111
bbb:222
ccc:333
$ _
```

Exemplo 21 *Colagem lado a lado de dois ficheiros,* `fich1` *e* `fich2`*, usando o carácter* `:` *como separador e fazendo a correspondência entre a segunda coluna do primeiro ficheiro com a primeira coluna do segundo ficheiro:*

```
$ cat fich1
a:primeira
c:terceira
$ cat fich2
primeira:1
segunda:2
terceira:3
$ join -t:  -j1 2 -j2 1 fich1 fich2
primeira:a:1
terceira:c:3
$ _
```

Capítulo 4

Composição de comandos

4.1 Redirecionamento para ficheiros

A mais simples das várias possibilidades para combinar comandos em UNIX é o redirecionamento de entrada e saída para ficheiros. Para o efeito, o sistema UNIX permite enviar todo o texto que um programa tenta escrever no ecrã para um ficheiro selecionado. Para o efeito, utiliza-se o símbolo > seguido do nome do ficheiro a criar.

Exemplo 22 *Demonstraçao de como o resultado do comando `ls` é enviado para um ficheiro chamado `lixo`, que é por sua vez apresentado no ecrã utilizando o comando `cat`:*

```
$ ls -la
total 5674
drwxr-sr-x   2 root     ftp          1024 Sep  8  1998 ./
drwxrwsr-x  16 root     ftp          1024 Dec 19 01:19 ../
-r--r--r--   1 root     ftp         50206 Sep  8  1998 faq.htm
-r--r--r--   1 root     ftp         14596 Sep  8  1998 prguide.htm
-r--r--r--   1 root     ftp       1082355 Sep  8  1998 sd22lux.tar.Z
-r--r--r--   1 root     ftp          6869 Sep  8  1998 sf22lux.htm
-r--r--r--   1 root     ftp       1942573 Sep  8  1998 sf22lux.tar.Z
-r--r--r--   1 root     ftp       1077457 Sep  8  1998 sm22htm.tar.Z
-r--r--r--   1 root     ftp       1584240 Sep  8  1998 st22lux.tar.Z
-r--r--r--   1 root     ftp          7235 Sep  8  1998 td000001.htm
$ ls -la > lixo
$ cat lixo
total 5674
drwxr-sr-x   2 root     ftp          1024 Sep  8  1998 ./
drwxrwsr-x  16 root     ftp          1024 Dec 19 01:19 ../
-r--r--r--   1 root     ftp         50206 Sep  8  1998 faq.htm
-r--r--r--   1 root     ftp         14596 Sep  8  1998 prguide.htm
-r--r--r--   1 root     ftp       1082355 Sep  8  1998 sd22lux.tar.Z
```

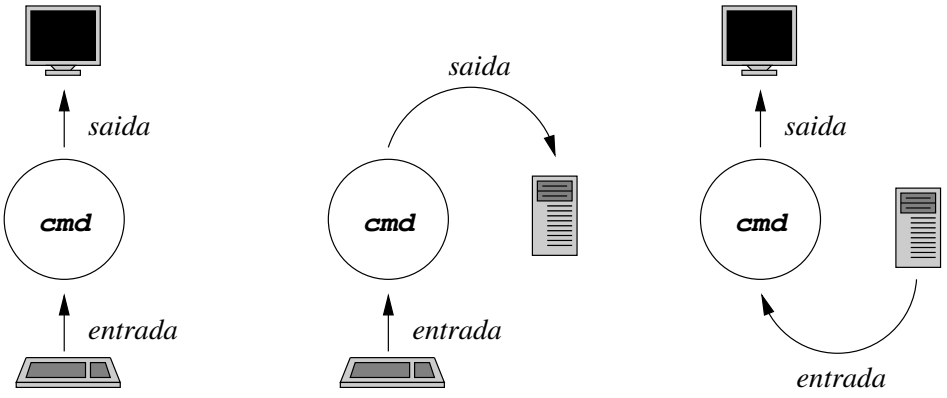

Figura 4.1: redirecionamentos de entrada e de saída. (a) Situação normal em que o programa lê do teclado e escreve no ecrã (`cmd`); (b) reddireccionamento da saída para um ficheiro (`cmd > ficheiro`) ; (c) redirecionamento da entrada a partir de um ficheiro (`cmd < ficheiro`).

```
-r--r--r--   1 root     ftp          6869 Sep  8  1998 sf22lux.htm
-r--r--r--   1 root     ftp       1942573 Sep  8  1998 sf22lux.tar.Z
-r--r--r--   1 root     ftp       1077457 Sep  8  1998 sm22htm.tar.Z
-r--r--r--   1 root     ftp       1584240 Sep  8  1998 st22lux.tar.Z
-r--r--r--   1 root     ftp          7235 Sep  8  1998 td000001.htm
-r--r--r--   1 root    root             0 May 21  1999 lixo
$ _
```

Note-se que esta possibilidade é independente da forma como o programa foi escrito. Ou seja, qualquer programa que imprima texto para o ecrã pode ser utilizado para escrever para um ficheiro, uma vez que o redirecionamento é feito ao nível do sistema operativo (ver Figura 4.1 (b)).

Da mesmo forma, é possível redirecionar a entrada de um comando. Ou seja, sempre que o programa tentar ler dados a partir do teclado o sistema operativo fornece-lhe dados a partir de um ficheiro utilizando o símbolo < (ver Figura 4.1 (c)).

Exemplo 23 *O comando `bc` é uma calculadora, aceitando as expressões a calcular a partir do teclado. Pode no entanto redirecionar-se a entrada para um ficheiro de modo a efetuar cálculos previamente guardados num ficheiro:*

```
$ bc -q
2+2
4
```

```
quit
$ cat calculos
2+2
quit
$ bc -q < calculos
4
$ _
```

É então possível combinar diversos comandos utilizando ficheiros para guardar os resultados intermédios. Pode-se assim:

- combinar vários comandos simples de manipulação de texto para obter resultados que não eram possíveis com nenhum deles em separado;
- utilizar os comandos de manipulação de texto para modificar a saída (ou entrada) de outros comandos.

Note-se que convém apagar os ficheiros utilizados para guardar os resultados intermédios, logo que deixem de ser necessários.

Exemplo 24 *Obter a quinta linha do ficheiro* `/etc/passwd`*, escolhendo a última das primeiras cinco linhas:*

```
$ head -5 /etc/passwd > intermedio
$ tail -1 intermedio
lp:*:4:7:lp:/var/spool/lpd:
$ rm intermedio

$ _
```

Exemplo 25 *Obter a quinta linha do ficheiro* `/etc/passwd`*, escolhendo a primeira das últimas linhas a contar da quinta:*

```
$ tail +5 /etc/passwd > intermedio
$ head -1 intermedio
lp:*:4:7:lp:/var/spool/lpd:
$ rm intermedio

$ _
```

Exemplo 26 *Listagem do conteúdo de uma diretoria apresentando apenas os nomes dos ficheiros e respetivos atributos. Para o efeito, selecionam-se as linhas a partir da segunda, de modo a retirar o linha de total, e nestas a primeira e última colunas:*

```
$ ls -la > temporario1
```

```
$ tail +2 temporario1 > temporario2
$ cut -c-11,56- temporario2
drwxr-sr-x ./
drwxrwsr-x ../
-r--r--r-- faq.htm
-r--r--r-- prguide.htm
-r--r--r-- sd22lux.tar.Z
-r--r--r-- sf22lux.htm
-r--r--r-- sf22lux.tar.Z
-r--r--r-- sm22htm.tar.Z
-r--r--r-- st22lux.tar.Z
-r--r--r-- td000001.htm
-r--r--r-- temporario1
$ rm temporario1 temporario2

$ _
```

4.2 Pipes

A composição de comandos utilizando ficheiros intermédios pode ser simplificada. Para o efeito, o sistema operativo UNIX permite redirecionar a saída de um comando diretamente para a entrada do comando seguinte, utilizando *pipes*.

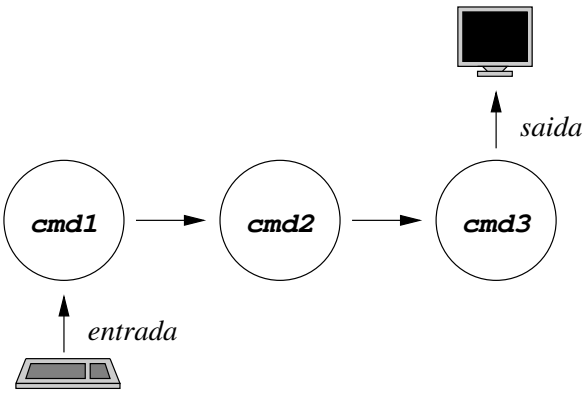

Figura 4.2: Composição de vários comandos com *pipes* (`cmd1 | cmd2 | cmd3`).

A sintaxe utilizada na *shell* para indicar a composição de comandos com *pipes* usa o símbolo | para separar cada os vários comandos. Os dados circulam do comando mais à esquerda para o comando mais à direita.

Além disso, o resultado final pode ser enviado para um ficheiro, da mesma forma descrita na secção anterior.

Exemplo 27 *Reescrita do Exemplo 24 utilizando pipes, ou seja obter a quinta linha do ficheiro* `/etc/passwd`*, escolhendo a última das primeiras cinco linhas:*

```
$ head -5 /etc/passwd | tail -1
lp:*:4:7:lp:/var/spool/lpd:

$ _
```

Exemplo 28 *Reescrita do Exemplo 26 utilizando pipes:*

```
$ ls -la | tail +2 | cut -c-11,56-
drwxr-sr-x ./
drwxrwsr-x ../
-r--r--r-- faq.htm
-r--r--r-- prguide.htm
-r--r--r-- sd22lux.tar.Z
-r--r--r-- sf22lux.htm
-r--r--r-- sf22lux.tar.Z
-r--r--r-- sm22htm.tar.Z
-r--r--r-- st22lux.tar.Z
-r--r--r-- td000001.htm
-r--r--r-- temporario1

$ _
```

sort

`sort` – ordena um alfabeticamente linhas de texto por ordem crescente

- `-r` ordenação decrescente
- `-n` ordenação numérica

Exemplo 29 *Listagem da diretoria ordenada pelo tamanho dos ficheiros:*

```
$ ls -la | tail +2 | cut -c30-42,56- | sort -n
        1024 ../
        1024 ./
        6869 sf22lux.htm
        7235 td000001.htm
       14596 prguide.htm
       50206 faq.htm
     1077457 sm22htm.tar.Z
     1082355 sd22lux.tar.Z
     1584240 st22lux.tar.Z
     1942573 sf22lux.tar.Z
$ _
```

Exemplo 30 *Listagem decrescente dos noms dos cinco maiores ficheiros (i.e. o Top 5):*

```
$ ls -la | tail +2 | cut -c30-42,56- | sort -rn | head -5 | cut -c 14-
sf22lux.tar.Z
st22lux.tar.Z
sd22lux.tar.Z
sm22htm.tar.Z
faq.htm
$ _
```

4.3 Comandos como parâmetros

Como alternativa à utilização de *pipes*, em que o resultado de um comando é utilizado como entrada para um outro comando, às vezes é desejável utilizar o resultado como parte de outro comando.

Para o efeito utiliza-se o símbolo ‘ (acento grave) como se utilizam os parentsis em expressões matemáticas. Ou seja, o comando que mais interior é executado, sendo substituído na expressão pelo seu resultado. A expressão resultante é então executada.

du

`du` – calcula o espaço ocupado por uma diretoria

`-s` apresenta apenas o total

Exemplo 31 *Cálculo do espaço ocupado pela área de trabalho do utilizador* `jop`*. Sem recorrer à substituição de partes de comandos, ter-se-ia que efectuar em dois passos:*

```
$ grep jop /etc/passwd | cut -d:  -f6
/home/linux/jop
$ du -sk /home/linux/jop
215430  /home/linux/jop/
$ _
```

Com a substituição de comandos, pode fazer-se tudo com apenas uma linha. Note-se que internamente os dois passos mostrados acima são executados automáticamente:

```
$ du -sk ‘grep jop /etc/passwd | cut -d:  -f6‘
215430  /home/linux/jop/
$ _
```

Exemplo 32 *Cálculo pelo espaço ocupado pelas áreas de trabalho de todos os utilizadores normais:*

```
$ du -sk `grep :200:  /etc/passwd/ | cut -d:  -f6`
215430  /home/linux/jop/
10      /home/linux/outro/
13      /home/linux/maisum/
$ _
```

Exemplo 33 *Cálculo pelo espaço ocupado pelas áreas de trabalho de todos os utilizadores normais, ordenado:*

```
$ du -sk `grep :200:  /etc/passwd/ | cut -d:  -f6` | sort -n
10      /home/linux/outro/
13      /home/linux/maisum/
215430  /home/linux/jop/
$ _
```

Exemplo 34 *Escolha do utilizador que mais espaço em disco ocupa:*

```
$ du -sk `grep :200:  /etc/passwd/ | cut -d:  -f6` | sort -n | tail -1
215430  /home/linux/jop/
$ _
```

find

`find` – procura ficheiros numa diretoria e todas as suas subdiretorias

`-name` *nome*	especifica o nome dos ficheiros procurar
`-print`	especifica que cada um dos nomes de ficheiros encontrados deve ser impresso no ecrã
`-exec` *cmd*	especifica um comando a executar com cada um dos ficheiros encontrados

Exemplo 35 *Todos os ficheiros com extensão* `.java` *na diretoria* `pdb/`*:*

```
$ find pdb -name *.java -print
pdb/files2pdb.java
pdb/pdb2files.java
$ _
```

Exemplo 36 *Cópia dos ficheiros com extensão* `.java` *na diretoria* `pdb/` *para a diretoria* `backup/`*:*

```
$ cp `find pdb -name *.java -print` backup/

$ _
```

Note-se que por substituição este comando é equivalente a:

```
$ cp pdb/files2pdb.java pdb/pdb2files.java backup/

$ _
```

wc

`wc` – conta palavras/caracteres/linhas de um ficheiro

- `-c` conta caracteres
- `-w` conta palavras
- `-l` conta linhas

Exemplo 37 *Contagem das linhas dos ficheiros com extensão* `.java` *na diretoria* `pdb/`*:*

```
$ wc -l `find pdb -name *.java -print`
    162 pdb/files2pdb.java
    144 pdb/pdb2files.java
    306 total
$ _
```

4.4 Shell scripts

Apesar da composição de comandos utilizando *pipes* ser bastante conveniente, nem todas as tarefas podem ser expressas apenas como uma linha de comandos. Nestes casos, interessa poder escrever um guião de comandos para o sistema executar sequencialmente. Para o efeito, cria-se com qualquer editor de texto um ficheiro em que cada linha é um comando a executar, indicando qual o interpretador adequado. O resultante, normalmente chamado de *shell script* ou simplesmente *script* pode ser utilizado como qualquer outro comando do sistema. Em resumo, as regras para fazer um uma *script* são:

- a primeira linha deve conter `#!` seguido no nome do interpretador, por exemplo, `#!/bin/sh` no caso de se prentender a *shell* normal do sistema;
- as restantes linhas devem conter comandos válidos para esse interpretador;
- o ficheiro deve ter o atributo `x`, o que pode ser activado com o comando `chmod a+x` *`script`*.

Exemplo 38 *Exemplo de criação de uma script para obter uma versão personalizada do comando* `ls`*. Note-se que o ficheiro* `meu_ls` *pode ser criado com o conteúdo correcto com qualquer editor de texto e não apenas com o redirecionamento do comando* `cat`*.*

```
$ cat > meu_ls
#!/bin/sh
ls -la
^D[1]
$ chmod a+x meu_ls
$ meu_ls
total 5674
drwxr-sr-x   2 root     ftp          1024 Sep  8  1998 ./
drwxrwsr-x  16 root     ftp          1024 Dec 19 01:19 ../
-r--r--r--   1 root     ftp         50206 Sep  8  1998 faq.htm
-r--r--r--   1 root     ftp         14596 Sep  8  1998 prguide.htm
-r--r--r--   1 root     ftp       1082355 Sep  8  1998 sd22lux.tar.Z
-r--r--r--   1 root     ftp          6869 Sep  8  1998 sf22lux.htm
-r--r--r--   1 root     ftp       1942573 Sep  8  1998 sf22lux.tar.Z
-r--r--r--   1 root     ftp       1077457 Sep  8  1998 sm22htm.tar.Z
-r--r--r--   1 root     ftp       1584240 Sep  8  1998 st22lux.tar.Z
-r--r--r--   1 root     ftp          7235 Sep  8  1998 td000001.htm
-r-xr-xr-x   1 root    root            17 May 25  1999 meu_ls
$ _
```

Exemplo 39 *Outra aplicação simples de scripts é como abreviaturas para comandos complexos usados frequentemente:*

```
$ cat > home_jop
#!/bin/sh
grep jop /etc/passwd | cut -d:  -f6
^D
$ chmod a+x home_jop
$ home_jop
/home/linux/jop
$ _
```

Nos Exemplos 38 e 39 surgem algumas dificuldades, respectivamente:

- embora o comando `ls` possa ser usado para lista o conteúdo de qualquer diretoria ou listar um tipo particular de ficheiros, por exemplo, com `ls /etc` ou `ls *.c`, o comando `meu_ls` não é capaz de fazer o mesmo uma vez que não utiliza os parâmetros que lhe são passados;
- o comando `home_jop` seria bastante mais útil se pudesse receber o nome do utilizador como parâmetro em vez de fazer parte de própria *script*.

[1] Sinal de fim de ficheiro, obtido com a combinação de teclas CTRL+D.

Ambos estes problemas podem ser resolvidos utilizando o parâmetros que são passados à *script* e que podem ser acedidos através dos símbolos `$1`, `$2`, etc. O número indica qual o parâmetro desejado. Durante a execução da *script* o símbolo é substituído pelo valor associado ao parâmetro correspondente antes de executar cada uma das linhas. O símbolo `$*` refere-se a todos os parâmetros, independentemente do seu número e o símbolo `$#` refere-se ao número de parâmetros.

echo

`echo` – imprime para o ecrã

`-n` não muda de linha depois de imprimir

Exemplo 40 *Exemplo de utilização dos parâmetros de uma script:*

```
$ cat > teste
#!/bin/sh
echo Recebi $# parâmetros.
echo O primeiro é $1.
echo O terceiro é $3.
echo Ou seja, são:  $*.
^D
$ chmod a+x teste
$ teste "um parametro"outro "mais um"
Recebi 3 parâmetros.
O primeiro é um parametro.
O terceiro é mais um.
Ou seja, são:  um parametro outro mais um.
$ _
```

Exemplo 41 *Melhoramento da script do Exemplo 38 de modo a aceitar parâmetros:*

```
$ cat > meu_ls
#!/bin/sh
ls -la $*
^D
$ chmod a+x meu_ls
$ meu_ls *.htm
-r--r--r--   1 root     ftp         50206 Sep  8  1998 faq.htm
-r--r--r--   1 root     ftp         14596 Sep  8  1998 prguide.htm
-r--r--r--   1 root     ftp          6869 Sep  8  1998 sf22lux.htm
-r--r--r--   1 root     ftp          7235 Sep  8  1998 td000001.htm
$ _
```

Exemplo 42 *Melhoramento da script do Exemplo 39 de modo a aceitar parâmetros:*

```
$ cat > home
#!/bin/sh
grep $1 /etc/passwd | cut -d:  -f6
^D
$ chmod a+x home
$ home jop
/home/linux/jop
$ home outro
/home/linux/outro
$ _
```

4.5 Programação em scripts

Além de listas simples de comandos as *shell scripts* podem servir para fazer programas simples, tendo para o efeito estruturas de controlo e variáveis.

A utilização de variáveis é extremamente simples quando comparada com linguagens de programação modernas, uma vez que todas as variáveis são do mesmo tipo e como tal não existe necessidade de as declarar. As acções sobre variáveis reduzem-se pois a:

- guardar um valor numa variável utilizando o símbolo =;
- mostrar o valor de uma varável utilizando o símbolo $.

Exemplo 43 *Utilização de variáveis em shell scripts:*

```
$ cat > exemplovar
#!/bin/sh
var=123
echo a variavel var tem o valor $var var=‘home jop‘
echo a variavel var agora tem o valor $var ^D
$ chmod a+x exemplovar
$ exemplovar
a variavel var tem o valor 123
a variavel var agora tem o valor /home/linux/jop
$ _
```

Exemplo 44 *Utilização de variáveis em shell scripts para substituir ficheiros intermédios:*

```
$ cat > homesize
#!/bin/sh
h=‘home jop‘
t=‘du -sk $h‘
echo $t | cut -d" -f1
```

```
^D
$ chmod a+x exemplovar
$ homesize jop
215442
$ _
```

O valor das variáveis também pode ser lido diretamente do teclado, ou seja, pedido interativamente ao utilizador através do comando `read`. Refira-se que, como qualquer comando UNIX, as *scripts* que utilizam este mecanismo podem ver a sua entrada redirecionada de um ficheiro ou de outro comando.

Exemplo 45 *Modificação da script do Exemplo 42 de modo a ler o nome do utilizador pretendido a partir do teclado, podendo como tal, ser utilizada em pipes com a entrada redirecionada:*

```
$ cat > home
#!/bin/sh
read user
grep $user /etc/passwd | cut -d:  -f6
^D
$ chmod a+x home
$ home
jop
/home/linux/jop
$ echo jop | home
/home/linux/jop
$ _
```

Em termos de estruturas de controlo, as mais úteis são o `if`, cuja utilização é semelhante às linguagens de programação, e a iteração sobre listas com o comando `foreach` que se adapta bastante bem ao tipo de problemas que se tentam resolver com *shell scripts*.

Exemplo 46 *Utilização do `if` de modo escrever uma única script que posso ser utilizada como as dos Exemplos 42 e 45, tendo em conta o número de parâmetros recebidos:*

```
$ cat > home
#!/bin/sh
if [ $# = 0 ]
then read user
else user=$1
fi
grep $user /etc/passwd | cut -d:  -f6
^D
$ chmod a+x home
```

```
$ home
jop
/home/linux/jop
$ home jop
/home/linux/jop
$ _
```

Exemplo 47 *Utilização simples do comando* `foreach`*::*

```
$ cat > teste
#!/bin/sh
foreach num in 1 2 3
do
echo $num
done
^D
$ chmod a+x teste
$ teste
1
2
3
$ _
```

Exemplo 48 *Utilização do comando* `foreach` *para melhorar a script do Exemplo 42 de modo poder receber mais do que um nome de utilizador na linha de comando:*

```
$ cat > homes
#!/bin/sh
foreach user in $*
do
grep $user /etc/passwd | cut -d:  -f6
done
^D
$ chmod a+x homes
$ homes jop root maisum
/home/linux/jop
/root
/home/linux/maisum
$ home outro
/home/linux/outro
$ _
```