Comandos Introdutórios

José João Almeida
José Bernardo Barros
Pedro Rangel Henriques

Fevereiro de 1991

Universidade do Minho
Departamento de Informática
Grupo Disciplinar de Ciências de Computação

INESC – Braga

Indice

1 Introdução

O UNIX é um sistema operativo com as seguintes características genéricas:

- multi-utilizador;
- núcleo (*kernel*) reduzido;
- comandos versáteis;
- possibilidade de construir novos comandos à custa dos existentes;
- vocacionado para utilizadores experientes.

O interpretador de comandos não faz parte do núcleo do sistema operativo sendo por isso considerado como um *utilitário*. As versões mais divulgadas são a **Bourne Shell** e a **C-Shell**.

Para *entrar* no sistema, o utilizador tem de fornecer o seu nome de acesso e a sua senha (esta última não é ecoada).

Para terminar a sessão pode-se fazer **ˆD** (eof) ou invocar os comandos **logout** ou **login**.

Para cada utilizador do sistema existe uma entrada (linha) no ficheiro "*/etc/passwd*" que contém (separados pelo caracter **:**) os seguintes campos:

1. nome de acesso;
2. palavra de acesso (password) - codificada;
3. número do utilizador;
4. número do grupo a que o utilizador pertence;
5. nome do utilizador;
6. directoria base (*home directory*) do utilizador;
7. programa que é executado quando o utilizador entra no sistema (*shell*).

2 Estrutura de Ficheiros

Tal como em *MS-DOS* os ficheiros estão organizados numa árvore de directorias em que a raiz é a directoria /.

Existem três tipos de ficheiros

ordinários texto, executáveis, dados, ...

directorias

especiais associados a dispositivos de I/O (disco, periféricos, ...)

O nome de um ficheiro pode ter até 14 caracteres quaisquer excepto o caracter /.

A especificação do nome de um ficheiro (*pathname*) é uma sequência de nomes de directorias separados por / e terminada pelo nome do ficheiro e pode ser

absoluta se começar pelo caracter /;

relativa à directoria actual no caso contrário.

Em cada directoria (excepto na raiz - /) existem definidas as directorias

.. directoria *mãe*

. própria directoria

A cada ficheiro estão associados

- identificação do dono do ficheiro (normalmente o utilizador que criou o ficheiro)
- 9 *bits* de controlo de acessos ($b_0...b_8$) divididos em 3 grupos de 3 *bits* cada com o seguinte significado

  $b_0...b_2$ permissões de acesso para o dono do ficheiro;

  $b_3...b_5$ permissões de acesso para os utilizadores do mesmo grupo do do dono do ficheiro;

  $b_6...b_8$ permissões de acesso para os restantes utilizadores.

  Dentro de cada grupo de 3 *bits* o seu significado é o seguinte:

  1º acesso para leitura (**r**)

  2º acesso para escrita (**w**)

  3º acesso para execução (**x**)

  Em qualquer dos casos, quando o bit tem o valor "1" o acesso respectivo é permitido, quando é "0" o acesso é negado.

- 3 bits de controlo

Os ficheiros especiais encontram-se na directoria **/dev**.
Do ponto de vista do utilizador, o acesso a um destes ficheiros é semelhante ao acesso a um ficheiro ordinário.

3 Redireccionamento

Em UNIX, existem pre-definidos três ficheiros especiais:

stdin Canal standard de entrada de dados.
É deste ficheiro que se lê quando se faz um *scanf* em "C" ou um *read* em "Pascal" sem especificar o nome do ficheiro.

stdout canal standard de saida de dados.
É neste ficheiro que se escreve quando se faz um *fprintf* em "C" ou um *write* em "Pascal" sem especificar o nome do ficheiro.

stderr canal standard de diagnóstico.
É neste ficheiro que, por exemplo, se escrevem os erros de uma compilação .

Por defeito tanto *stdout* como *stderr* encontram-se associados ao ecran, enquanto que *stdin* se encontra associado ao teclado.

No entanto estas associações podem ser alteradas se na linha de comando aparecer:

< **fich** Abre o ficheiro *fich* e este passa a ser o *standard input*

<< **palavra** Lê o *standard input* até que apareça uma linha igual a *palavra* terminando aí a leitura.

> **fich** O ficheiro *fich* é apagado (caso exista) e depois usado como *standard output*.

>> **fich** Idêntico ao anterior só que o ficheiro não é apagado e o *output* é acrescentado a *fich*.

>& **fich** Tanto o *standard output* como o canal standard de diagnóstico (*standard error*) são redireccionados para *fich*. O conteudo anterior de *fich* é apagado.

>>& **fich** Idêntico ao anterior só que o ficheiro não é apagado e as escritas são feitas no fim do ficheiro (append).

4 Pipe's

Ao juntarmos dois comandos usando o símbolo |, o canal de entrada do segundo comando passa a estar ligado ao canal de saída do primeiro comando.

Existe contudo uma diferença entre fazer **com1 | com2** e
com1 > temp ; com2 < temp ; rm temp.

A diferença reside não no uso de um ficheiro intermédio **temp**, mas na sincronização dos comandos **com1** e **com2**. Assim, enquanto que no 2º caso **com2** só começa a ser executado depois de **com1** ter terminado, no 1º caso os comandos são executados concorrentemente.

Vejamos um caso concreto. Seja **com1** um programa que escreve números inteiros até que seja interrompido (ciclo infinito). Seja **com2** o comando *head* que escreve as 10 primeiras linhas de um ficheiro.

Ao fazermos **com1 | com2** o que acontece é que depois de **com1** produzir 10 linhas de texto, **com2** termina a sua execução e envia um sinal de interrupção a **com1**. Se usássemos a outra forma, o comando não terminaria.

Uma outra variante de *pipe* é o uso do conectivo |& que faz com que o canal de entrada do 2º comando seja ligado tanto ao canal de saida do 1º comando como ao seu canal de diagnóstico.

5 Comandos

man *comando* Dá uma descrição do *comando*. Essa descrição inclui as opções válidas e exemplos de utilização . Ver **man man** .

pwd Dá o nome da directoria actual.

ls [-ltasdrucifg] [*nome ...*] lista os nomes dos ficheiros das directorias especificadas (a actual por defeito). Algumas das opções mais usadas são:

l dá uma linha para cada ficheiro contendo informação acerca dos modos de acesso, espaço ocupado, etc

a lista também os ficheiros cujo nome começa por "**.**"

chmod ***modo nome ...*** Muda a palavra de acesso dos ficheiros especificados de acordo com *modo*.

modo pode ser especificado da seguinte forma:

absoluta da forma **ooo** (em que **o** é um digito octal) faz com que a nova palavra de acesso seja o correspondente binário de *ooo*.

Assim, por exemplo, **chmod 752 fich1**, faz com que a palavra de acesso do ficheiro *fich1* seja **111011010**.

relativa da forma **([ugoa] +– rwx)*** em que:

u – significa o dono

g – grupo

o – outros utilizadores

a – todos os anteriores

– – retirar o previlégio

+ – dar permissão

r – leitura

w – escrita

x – execução

Assim, por exemplo, o comando **chmod a+x fich** faz com que todos os utilizadores tenham acesso de execução do ficheiro *fich* mantendo as restantes permissões (escrita e leitura) inalteradas.

chown ***dono nome ...*** Muda o dono dos ficheiros para **dono**.

chgrp ***grupo nome ...*** Muda o grupo dos ficheiros para **grupo**.

cat ***nome ...*** con**cat**ena os ficheiros e escreve-os em *stdout*.

cd [***nome***] muda a directoria corrente para *nome*. Se não for especificada essa directoria então muda para a directoria por defeito (*home directory*).

pwd Dá o *pathname* da directoria actual.

mkdir ***nome*** cria uma directoria *nome*.

rm ***nome*** Apaga os ficheiros especificados. Pode ser chamado com as opções :

-i faz com que seja pedida confirmação

-r a operação é feita recursivamente por todas as subdirectorias

-f apaga os ficheiros mesmo que o utilizador não tenha acesso de escrita

rmdir ***nome*** Apaga a directoria (vazia) *nome*.

mv ***origem destino*** Muda o nome do ficheiro *origem* para *destino*.

cp ***origem destino*** Copia o ficheiro *origem* para o ficheiro *destino*.

ln ***origem destino*** Cria uma ligação (link) entre o ficheiro *origem* e *destino*.

Em Unix é possível que um mesmo ficheiro (físico) possa ser visto com nomes (*pathnames*) diferentes.

Ao estabelecer uma ligação (link) entre dois ficheiros não estamos a copiar um ficheiro para o outro, uma vez que qualquer alteração a um deles se reflecte no outro.

who [am i] Mostra quais os utilizadores que estão no sistema (logged in). Se invocado com a opção **am i**, diz algumas informações sobre o utilizador (nome de acesso, terminal, hora de entrada).

id Dá o numero e grupo do utilizador.

logname Dá o nome de acesso ao sistema do utilizador.

passwd Permite modificar a palavra de acesso ao sistema (password).

write [***utilizador ...***] Permite escrever mensagens no terminal de outro utilizador que esteja no sistema.

mesg [-yn] Se invocado com **-y** inibe a possibilidade de outros utilizadores escreverem mensagens no terminal.

at [***hora***] Permite executar comandos a uma determinada hora.

nice [***comando***] Faz com que *comando* seja executado com uma prioridade menor.

nohup [***comando***] Faz com que mesmo depois de o processo que invocou *comando* terminar, o processo que executa *comando* não seja terminado.

Este comando é útil quando usado em conjunção com **&** (execução em *background*).

ps [***opções***] lista informação sobre os processos existentes. Se invocado sem opções , lista apenas os processos que foram invocados pelo *pai*.

Algumas opções são:

- **-u** ***user*** Dá informação acerca dos processos criados pelo utilizador *user*.
- **-a** Dá informação acerca de todos os processos existentes.
- **-l** Para cada processo dá uma linha com mais informação .

kill [-***sinal***] ***pid ...*** Envia aos processos *pid ...* o sinal *sinal*.

wait Espera que todos os processos invocados por este terminem.

date Dá a data e hora actuais. Serve ainda para as modificar (superuser).

cal [***mês***] [***ano***] Mostra o calendário desse mês/ano. Se o mês for omitido, mostra o calendário de todo o ano; se o ano for omitido, assume o actual

calendar Consulta o ficheiro *calendar* e imprime todas as linhas onde existam referências ao dia de hoje ou amanhã.

6 Utilitários

mail [*utilizador ...*] Correio electónico.

Se chamado sem argumentos permite visualizar as mensagens recebidas.

Se for invocado com argumentos (nomes de utilizadores) permite enviar mensagens a esses utilizadores.

bc (**b**inary **c**alculator) é um utilitário que permite fazer operações aritméticas com precisão ilimitada.

Pode ainda ser invocado com um ficheiro e neste caso envia para *stdout* o resultado das operações especificadas em tal ficheiro.

od [-bcdosx] [*ficheiro*] (**o**ctal **d**ump) Permite visualizar o conteudo do *ficheiro*. Para cada caracter do ficheiro é imprimido o seu código em octal (-o), decimal (-d), hexadecimal (-x), ou como caracter ASCII (-c).

cut -cf*lista* [-d*char*] [*ficheiro ...*] Permite seleccionar colunas de um ficheiro (equivalente à projecção em bases de dados).

lista representa a lista de colunas (separadas por ";") que deve ser passada para *stdout*.

O conceito de coluna pode ser:

- de comprimento fixo (opção **-c**)
- de comprimento variável (opção **-f**) delimitado por um caracter especificado na opção **-d** (TAB por defeito).

Por exemplo

cut -c10-20 *fich* escreve em *stdout* os caracteres que estão desde a 10ª à 20ª posição em cada linha do ficheiro *fich*.

cut -f1;3 -d: /etc/passwd dá um ficheiro com os nomes e números dos utilizadores.

paste [-d*lista*] *fich1 fich2 ...* Cada linha enviada para *stdout* resulta da concatenação (separadas pelos caracteres de *lista* - TAB por defeito) das correspondentes linhas dos ficheiros argumentos.

O comando **paste** trata cada ficheiro como uma tabela, produzindo um ficheiro que é a justaposição das várias tabelas.

Para significar que um dos ficheiros é *stdin* deve-se fornecer −.

Por exemplo:

ls | paste − − − produz uma listagem da directoria em três colunas.

tee [-a] [*ficheiro*] Copia *stdin* para *stdout* e para *ficheiro*.

É muito usado para fazer derivações (**t**'s) em pipes. Por exemplo, o comando

who | tee fich | wc -l mostra no ecran quantos utilizadores estão no sistema e escreve informação sobre eles no ficheiro *fich*

df [*file system*] (**d**isk **f**ree) Dá informação sobre o nível de ocupação da parte do disco alocada ao sistema de ficheiros especificado.

du [*ficheiro ...*] (**d**isk **u**sed) Dá informação acerca do espaço ocupado pelos ficheiros.

wc [-lwc] [*ficheiro ...*] (**w**ord **c**ount) Conta os caracteres (**-c**), palavras (**-w**) e linhas (**-l**) dos ficheiros.

whereis *ficheiro* Procura *ficheiro* em directorias pré-definidas.

find *path-list expressão* procura (recursivamente) a partir das directorias existentes em *path-list* os ficheiros que verifiquem *expressão*. Expressão pode ser:

-name *nome* verdadeiro se o nome do ficheiro for *nome*

-perm *ooo* verdadeiro se o modo de acesso do ficheiro for o número octal *ooo*.

-links *n* verdadeiro se o ficheiro tiver *n* *link*s.

-user *nome* verdadeiro se o dono do ficheiro for *nome*.

-size *n* verdadeiro se o ficheiro tiver *n* blocos.

-atime *n* verdadeiro se o ficheiro tiver sido acedido nos últimos *n* dias

-exec *comando* Executa o *comando*. O nome do ficheiro actual pode ser referido nesse comando usando "{}". O fim do comando deve ser explicitado por "\;".

O resultado é o resultado da execução de *comando* (verdadeiro se *comando* retornar **0**).

-ok *comando* Idêntico ao anterior só que antes de executar *comando* pede para confirmar.

-print Sempre verdadeiro.

Imprime o *pathname* do ficheiro.

Estes comandos podem ser combinados da seguinte forma:

! *comando* (negação) verdadeiro se *comando* for falso.

com1* o *com2 (disjunção) verdadeiro se um deles o for.

com1 com2 (conjunção) verdadeiro se os dois o forem.

Note-se ainda que nos comandos onde aparece *n* pode ser especificado **+n** significando "$> n$" ou **-n** significando "$< n$".

tr [*string1* [*string2*]] Copia para *stdout* o ficheiro *stdin* substituindo todas as ocorrências de caracteres de *string1* pelo correspondente caracter de *string2*.

Assim por exemplo, o comando

tr [a-z] [A-Z] < fich1 > fich2 coloca em *fich2* o texto que se encontra em *fich1* convertendo todas as letras minúsculas para maiúsculas.

nl [*opções*] *ficheiro* Envia para *stdout* o *ficheiro* (*stdin* por defeito) com as linhas numeradas.

pr [*opções*] [*ficheiro ...*] produz (em *stdout*) uma listagem de *ficheiro* (*stdin* por defeito).

Esta listagem está dividida em páginas em que cada uma tem um cabeçalho.

Alguns dos parâmetros (cabeçalho, nº de linhas por página, ...) podem ser alterados (usando as opcções disponíveis).

uniq [***opções***] [***fich*in** [***fich*out**]] escreve em *fichout* as linhas repetidas.

sort [***opções***] [***ficheiro ...***] Ordena as linhas dos ficheiros.

As várias opções permitem controlar qual ou quais os campos de ordenação , se a ordem é crescente ou decrescente, etc.

grep [***opções***] ***expreg*** [***ficheiro ...***] Procura nos ficheiros especificados as linhas onde aparece uma ocorrência de *expreg* e escreve-as em *stdout*.

As várias opções permitem imprimir apenas os nomes dos ficheiros onde tal expressão ocorre, imprimir o número de ocorrências da expressão, etc.

Por exemplo a opção **-v** faz com que sejam seleccionadas apenas as linhas onde não aparece *expreg*.

head [-*n*] [***ficheiro ...***] Escreve em *stdout* as ***n*** (por defeito **10**) primeiras linhas de cada um dos ficheiros.

tail [-*n*] [***opções***] [***ficheiro ...***] Escreve em *stdout* as ***n*** últimas linhas de cada um dos ficheiros.

cmp [-**l**] [-s] ***fich1 fich2*** Escreve em *stdout* as diferenças entre os dois ficheiros ("–" para significar *stdin*).

Para cada *byte* diferente, escreve a posição (linha, coluna) e os *byte*s respectivos.

diff [**-efbh**] ***fich1 fich2*** Escreve em *stdout* as diferenças entre os dois ficheiros.

A diferença entre este comando e o anterior é que o *output* deste comando é um ficheiro de comandos ***sed*** que permite de *fich1* obter *fich2*.

lp [***opções***] [***ficheiro ...***] Envia para a fila de espera da impressora os *ficheiro*s.

lpstat [***opções***] Mostra o estado da impressora (fila de espera, etc).

more [***opções***] [***ficheiro ...***] Mostra os *ficheiro*s página a página.

sed [**-n**] [**-e *fich***] [**-f *fich***] [***ficheiro ...***] (stream **ed**itor) é um editor de texto especialmente desenhado para funcionar em *batch*.

Os comandos *sed* são da forma

zona comando

Quando a **zona** de aplicação do **comando** não é especificada, esse comando é aplicado a cada uma das linhas.

Zona pode ser especificada das seguintes formas:

linha significando apenas uma linha

linha1,linha2 significando as linhas desde *linha1* a *linha2*.

linha pode ser especificada ou pelo número ou por uma expressão regular (padrão contido numa linha).

Um resumo dos comandos disponíveis encontra-se na tabela seguinte.

a\	acrescenta linhas ao *output* até que apareça em *stdin* uma linha **não** terminada por \
b *label*	salta para a instrução *: label*
c\	altera linhas; o fim do input é detectado como em a\
d	apaga a linha e substitui-a pela próxima linha de *stdin*
i\	insere texto até aparecer uma linha **não** terminada por \
p	escreve a linha
q	abandona
r *fich*	lê *fich* e escreve em *stdout*
s/*old*/*new*/***f***	Substitui a expressão regular *old* por *new*. **f**=g substitui todas as ocorrências **f**=p escreve a linha em *stdout* **f**=w *fich* escreve a linha em *fich*
t *label*	salta para a instrução *:label* se alguma substituição foi feita na linha corrente
w *fich*	escreve a linha no ficheiro *fich*
y/*str1*/*str2*/	substitui cada caracter de *str1* pelo correspondente em *str2*
=	imprime a linha actual e o seu número
! *comando*	Executa o *comando* "sed" se a linha não for seleccionada
: *label*	
{ *comandos* }	trata *comandos* como um só

Lista dos Comandos *sed*