ISCTE IUL
Instituto Universitário de Lisboa
Escola de Tecnologias e Arquitectura
Dep. de Ciências e Tecnologias da Informação

Sistemas Operativos
Ano letivo 2013/2014
1º Semestre

Conteúdo

Linha de comandos, comandos de utilizador

1 Ficheiros

Um ficheiro é caracterizado por um nome (presente num determinado directório), um conjunto de propriedades e um conteúdo. Para ver a *lista* de ficheiros que se encontram em determinado directório usa-se o comando:

```
ls
```

A maior parte das vezes para criar um novo ficheiro usamos um editor de texto. É normal numa instalação Linux encontrar vários editores de texto disponíveis, por exemplo o `vi` ou o `emacs`. O comando seguinte pode ser usado para criar um novo ficheiro de texto com o `vi`, sendo `novo.txt` o nome do ficheiro que se pretende criar.

```
vi novo.txt
```

Para criar um texto simples podemos fazer a seguinte sequência, depois de entrar no `vi`:

1. carregar em `i` passando o `vi` para o modo de edição de texto
2. escrever o texto
3. carregar na tecla `Esc`
4. escrever `ZZ` para sair do `vi`, gravando o conteúdo inserido.

Exercício 1. Use o `vi` para criar/alterar outros ficheiros de texto. O guia de utilização do `vi`, disponível no site, descreve os principais comandos e mecanismos de trabalho neste editor.

O comando `cat` permite ver o conteúdo de um ficheiro no ecrã. Por exemplo para ver o conteúdo do ficheiro novo.txt pode usar o comando:

```
cat novo.txt
```

Se o ficheiro for muito grande é mais prático usar, em alternativa, o comando `more`, que mostra o ficheiro página a página. Por exemplo o comando:

```
more /etc/passwd
```

mostra o ficheiro mencionado, começando por exibir apenas o conteúdo da "primeira página". Para mudar de página carrega na tecla de espaço. Se quiser sair carrega na tecla q (de "quit").

Exercício. Pode também experimentar executar: `less /etc/passwd`

2 Sistema de ficheiros

No Linux, o sistema de ficheiros encontra-se organizado numa árvore com directórios ("ramos") e ficheiros ("folhas"). O primeiro desses directórios representa-se por / e designa-se "raiz". Como se disse, o comando `ls -l` mostra o conteúdo de um directório. Por exemplo, o comando

```
ls -l /
```

mostra o conteúdo do directório / ("raíz") que será qualquer coisa do género:

```
drwxr-xr-x    2 root     root         4096 2004-01-13 16:48 bin
drwxr-xr-x    3 root     root         1024 2004-01-13 17:23 boot
drwxr-xr-x   29 root     root        61440 2004-09-18 11:26 dev
drwxr-xr-x   65 root     root         8192 2005-02-18 09:48 etc
drwxr-xr-x   28 root     root         4096 2005-02-04 15:28 home
drwxr-xr-x    7 root     root         4096 2003-09-09 18:37 lib
drwxr-xr-x    2 root     root         4096 2004-03-26 17:11 local
drwx------    2 root     root        16384 2003-09-09 17:36 lost+found
drwxr-xr-x    5 root     root         4096 2003-03-18 17:49 media
drwxr-xr-x    2 root     root         4096 2003-03-18 17:49 mnt
drwxr-xr-x    4 root     root         4096 2003-09-11 10:19 net
drwxr-xr-x    8 root     root         4096 2003-09-09 18:10 opt
dr-xr-xr-x   77 root     root            0 2004-09-18 12:23 proc
drwx------   16 root     root         4096 2005-02-18 09:48 root
drwxr-xr-x    3 root     root         8192 2003-11-01 16:36 sbin
drwxrwxrwt   31 root     root         8192 2005-02-28 13:45 tmp
drwxr-xr-x   12 root     root         4096 2003-09-09 17:47 usr
drwxr-xr-x   15 root     root         4096 2003-09-09 18:17 var
```

O `ls -l` assinala os directórios com um `d` na primeira coluna (os ficheiros normais são assinalados com -). Vemos assim que dentro deste directório `/` existem, por sua vez, vários outros directórios como o `bin`, o `etc` ou o `home`. Por exemplo, o comando seguinte mostra o conteúdo do directório `/etc`. Dentro deste existem por sua vez vários ficheiros e outros directórios, e assim sucessivamente formando a mencionada "árvore".

```
ls -l /etc
```

3 Ficheiros: nomes relativos e absolutos

Considere o seguinte exemplo:

```
/
|-- etc
|   `-- passwd
|-- home
|   |-- a1000
|   |-- a1001
|   |   |-- notas.txt
|   |   `-- trabalho.java
|   `-- labso
|       |-- ac
|       |   `-- programa.mac
|       |-- geral.tab
|       `-- so
|           |-- ficheiro.txt
|           `-- outro_dir
|               `-- ficheiro.txt
`-- proc
    `-- cpuinfo
```

Considerando o exemplo representado, vemos que dentro do directório `/` existem, neste caso, mais três directórios: `etc`, `home` e `proc`. Por sua vez, dentro do directório home existem três outros directórios, com nomes `a1000`, `a1001` e `labso`. Dentro deste último, existe um ficheiro de nome `geral.tab` e mais dois directórios chamados `ac` e `so`. E assim sucessivamente...

Um ficheiro é identificado pela sua posição na árvore de directórios, a partir da raiz. O nome completo de um ficheiro obtém-se traçando o caminho deste o directório `/` até ao ficheiro em causa. No traçado deste caminho os directórios percorridos são também separados por uma `/`. Assim, por exemplo, os seguintes nomes são válidos:

```
/home/a1001/trabalho.java
/home/labso/so/ficheiro.txt
/home/labso/so/outro_dir
/home/labso/so/outro_dir/ficheiro.txt
```

Note a diferença entre a primeira `/` e as seguintes. A primeira representa o nome do directório principal ("raiz"). As seguintes separam os directórios percorridos no percurso até ao ficheiro.

Na realidade, os nomes de ficheiros podem ser dados de duas maneiras:

- **absoluto**: partindo da `/` (“raiz”)
- **relativo**: partindo do directório corrente actual.

O nome é **absoluto** (ou "completo) se começar por `/`; caso contrário é **relativo**. Para indicar um nome absoluto traça-se o caminho da raiz até ao ficheiro. Os exemplos anteriores são todos nomes absolutos. Para indicar um nome relativo traça-se, na mesma, o caminho até ao ficheiro, mas partindo do directório corrente. Para este efeito são válidos os seguintes elementos sintácticos:

. directório corrente

.. directório de cima (do nível anterior)

Exemplo. seja o directório corrente `/home/labso/so`. Nestas condições

```
ficheiro.txt                m.q.   /home/labso/so/ficheiro.txt
outro_dir/ficheiro.txt      m.q.   /home/labso/so/outro_dir/ficheiro.txt
../../a1001/trabalho.java   m.q.   /home/a1001/trabalho.java
```

4 Comandos relacionados com ficheiros e directorias

Directório corrente

O directório corrente é o directório de trabalho em cada momento. Para saber qual o directório corrente actual usa o comando `pwd` (de "print working directory"). Por exemplo:

```
tigre:~> pwd
/home/a88888  //indica que o directório corrente actual é /home/a88888
```

Para mudarmos de directório corrente usa-se comando `cd` (de change directory). Por exemplo:

```
tigre:~> cd /home/labso  //  directório corrente passa a ser /home/labso
tigre:~> pwd             //  confirmação com pwd
/home/labso
tigre:~> cd ..           // directório corrente passa a ser o "de cima"
tigre:~> pwd             // confirmação com pwd
/home
```

O directório corrente inicial quando entra no sistema, depois de fazer o login, designa-se directório de utilizador (ou "de trabalho").

Exercício. O directório de utilizador é indicado no ficheiro /etc/passwd. Localize o seu "username" neste ficheiro e o confirme o respectivo directório de utilizador.

Este directório é também, por vezes, relacionado com o conceito de "área de trabalho": é um directório que pertence ao utilizador e que o utilizador pode organizar internamente como entender (possivelmente criando uma sub-árvore de directórios) para guardar os seus ficheiros. O directório de utilizador é designado simbolicamente por `~`. Tal como o `..` e o `.` este símbolo representa o nome de um directório e pode ser usado como tal na generalidade dos comandos. Assim, por exemplo:

```
cp /etc/passwd ~
```

copia o ficheiro `/etc/passwd` para o directório de utilizador, onde fica com o mesmo nome (passwd). Por omissão os comandos de manipulação de ficheiros operam no directório corrente. Por exemplo, fazendo a sequência:

```
cd /etc
ls -l
```

o `ls -l` lista o conteúdo do directório corrente, na circunstância o `/etc`. O mesmo resultado teria o comando

```
ls -l /etc
```

onde se explicita o mesmo directório.

Exercício. Qual o efeito de cada uma das seguintes variantes do `ls` ? Qual delas dá o mesmo resultado que, apenas: `ls -l`

```
ls -l ..
ls -l .
ls -l ~
```

Copiar, mover e apagar: cp, mv e rm

O comando cp (de "copy") permite copiar um ficheiro para outro. Em caso de sucesso a operação cria um novo ficheiro com nome diferente e conteúdo igual ao do ficheiro original. Por exemplo, admitindo que o directório de trabalho é /home/labso, o comando

```
cp geral.tab clone.tab
```

cria um novo ficheiro, com nome `clone.tab`, no mesmo directório do ficheiro original.

Exercício. Considere a árvore de ficheiros do ponto 3. Qual o efeito dos seguintes comandos:

```
cd /home/labso/so
cp /etc/passwd .
cp /etc/passwd ..
cp /etc/passwd outro_dir/pass
cp ficheiro.txt ..
cp ../ac/programa.mac /home/labso
```

O comando mv (de "move") permite alterar o nome de um ficheiro. Por exemplo o comando:

```
mv programa.mac teste.mac
```

altera o nome de um ficheiro, de `programa.mac` para `teste.mac`, mantendo-se o ficheiro no mesmo directório. Já o comando:

```
mv programa.mac ..
```

movimenta o ficheiro de um directório para outro - neste caso, concretamente, para o directório de cima. Em qualquer dos casos o efeito é o mesmo é criado um novo ficheiro com outro nome (completo) e destruído o ficheiro original.

Exercício. Considere a árvore de ficheiros do ponto 3. Qual o efeito dos seguintes comandos:

```
cd /home/labso/so
mv ficheiro.txt ../a1000/copia.txt
mv ../ac/programa.mac ../ac/teste.mac
```

O comando `rm` (de “remove”) permite eliminar um ficheiro. Por exemplo, o comando

```
rm programa.txt
```

apaga o ficheiro mencionado (o ficheiro com nome programa.txt no directório corrente).

links: ln / ln -s

O comando `ln` permite criar links entre ficheiros. O link é uma espécie de atalho ou “shortcut”. Na forma simples, `ln`, permite criar os chamados "hard link".

```
ln ficheiro.txt copia.txt
```

A linha enterior cria um *hard link* `copia.txt` para o ficheiro original `ficheiro.txt`. Um hard link é parecido com uma cópia no sentido em que passam a existir dois nomes; mas apenas o nome é duplicado, o conteúdo é único e comum aos dois ficheiros. Se alterar o ficheiro.txt verá a mesma alteração reflectida no copia.txt. O comando `ls -l` que, como vimos, lista o conteúdo de um directório mostra também o número de "links" que cada ficheiro contém. Depois do comando anterior ambos os ficheiros (ficheiro.txt e copia.txt) ficarão com 2 links. Se apagar um dos nomes, por exemplo

```
rm copia.txt
```

não apaga o conteúdo do ficheiro, apenas um dos links. O número de links do ficheiro.txt diminuirá, em correspondência, de novo para 1.

A forma `ln -s` do mesmo comando cria os designados "soft link". Um *soft link* é um tipo de ficheiro especial que "aponta" para um outro ficheiro. Por exemplo:

```
ln -s /etc/passwd soft
```

cria no directório corrente um ficheiro de nome soft para que "aponta" para o ficheiro `/etc/passwd`. Se vir agora a lista de ficheiros com `ls -l` fica clara a natureza especial do ficheiro soft e o ficheiro para que "aponta". A criação de um “soft link” é idêntica à criação de um “atalho” (alias) no Windows.

Exercício. Crie um novo ficheiro e um "soft link" para ele. Qual o efeito no "soft link" se apagar o ficheiro original?

directórios

O comando `mkdir` (de "make directory") permite criar um novo directório.

```
mkdir pasta
mkdir pasta/sub_pasta
```

Por exemplo os comandos anteriores criam um novo directório de nome pasta no directório corrente e um outro, de nome `sub_pasta`, dentro desse.

O comando `rmdir` apaga um directório que, para esse efeito, deve estar vazio. Para apagar um directório não vazio, utilizamos `rm -rf`. Este comando é muito perigoso e deve ser utilizado com cuidado.

5 Elementos breves sobre a sintaxe dos comandos

A generalidade dos comandos aceita opções e argumentos. As opções definem ou alteram o funcionamento normal do comando. Normalmente, escrevem-se a seguir ao comando, antecedidos de sinal `-`. Por exemplo, a seguinte variante do comando `ls` é usada muitas vezes para listar os ficheiros:

```
ls -l
```

Os argumentos são os dados sobre o qual o comando trabalha. Por exemplo:

```
ls -l /etc
```

tem um argumento /etc. O comando lista o conteúdo de um directório - justamente o que é indicado no argumento.

Em muitos casos os argumentos podem ser identificados por omissão. Por exemplo

```
ls -l
```

não tem argumento explícito. Por omissão vale o directório corrente, ou seja dá o mesmo que:

```
ls -l .
```

A generalidade dos comandos tem variantes mais ou menos intuitivas. Por exemplo, muitos comandos aceitam vários ficheiros em lugar de apenas um. Assim, por exemplo:

```
cp ficheiro.txt copia.txt ..
```

copia dois ficheiros existentes no directório corrente para o directório de cima.

O comando `rm` tem algumas variantes de interesse. De um lado uma defensiva `rm -i` que pede uma confirmação antes de apagar o ficheiro. Ao contrário, a variante `rm -r`, útil mas particularmente perigosa, apaga recursivamente todos os ficheiros de um directório e dos subdirectórios nele contidos.

6 Permissões

As permissões de um ficheiro dividem-se em três grupos:

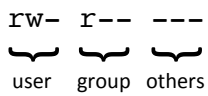
- *dono* (owner), *grupo* (group) ou *outros* (others)

Para cada um destes três grupos podem ser especificadas três operações:

- `r` (ler), `w` (escrever, alterar) ou `x` (executar).

O comando `ls -l` mostra as permissões de cada ficheiro nos 9 caracteres que se situam a seguir à primeira coluna (que como se mencionou anteriormente indica a natureza do "nome" presente no directório: por exemplo - para um ficheiro normal, d para um directório ou l para um "soft link"). Mostra também os nomes do dono e do grupo a que o ficheiro pertence.

Do conjunto dos 9 caracteres que representam as permissões o primeiro grupo de 3 caracteres indica as permissões que se aplicam ao dono do ficheiro. O segundo grupo indica as permissões que se aplicam aos utilizadores que pertencem ao grupo do ficheiro. E o terceiro grupo indica as permissões que se aplicam ao conjunto dos utilizadores. Assim, por exemplo, se tiver: `rw-r-----`, isso significa que ao dono do ficheiro aplicam-se as permissões `rw-`, ao grupo aplicam-se as permissões `r--` e ao conjunto de todos os utilizadores aplicam-se as permissões `---`.



Relativamente a ficheiros comuns as permissões indicam o direito de:

- `r`: ler (ver) o conteúdo do ficheiro;
- `w`: escrever (alterar) o conteúdo do ficheiro;
- `x`: executar o ficheiro como um programa.

Assim, por exemplo se tiver as permissões `rw-` isso significa que é permitido ler o ficheiro, escrever (alterar) o ficheiro mas não executá-lo. Se for `r---` pode-se apenas ler. Se for `---` não se pode fazer nenhuma das operações. Se o ficheiro for um "programa" poderá por exemplo, ter permissões `r-x`: pode-se ler e executar mas não alterar.

Relativamente a directórios as permissões indicam o direito de:

- `r,x`: poder "ler" (listar) o conteúdo do directório, ou seja, saber a lista dos ficheiros nele contidos:
- `w`: poder alterar ou seja criar/apagar nomes de ficheiros nesse directório;

Assim por exemplo, se um directório tiver permissões `r-x` pode-se "entrar" nele, fazer `ls -l` sobre ele mas não alterá-lo: ou seja não se podem criar ou apagar ficheiros nesse directório. Se tiver rwx podem-se fazer todas estas operações e se tiver `---` não se podem fazer quaisquer operações.

O dono de um ficheiro (ou directório) pode mudar as respectivas permissões usando o comando chmod (de “change mode”). Por exemplo, o comando seguinte muda as permissões do ficheiro novo.txt de acordo com o padrão indicado 744.

```
chmod 744 novo.txt
```

Este padrão indica as permissões através de 3 algarismos em octal, o primeiro para o dono, o segundo para o grupo e o terceiro para todos os utilizadores. O algarismo octal é formado fazendo corresponder a permissão r ("read") ao 4 a permissão w ("write") ao 2 e a permissão x ("execute") ao 1. Assim, por exemplo, r-x corresponde a 4+0+1 ou seja 5. Ao 6, ou seja, 4+2+0 corresponde rw-. Exemplos:

```
rwxrw---- → 760
rwxr-xr-x → 755
```

No exemplo anterior, com 744 estamos então a atribuir ao ficheiro as permissões:

```
rwxr--r--
```

Observação. Entretanto, o comando `chmod` tem outra forma sintáctica em que as permissões são indicadas através de letras. Por exemplo, o comando seguinte acrescenta a permissão x em todos os grupos. Veja a página do manual para mais pormenores.

```
chmod +x
```

7 Ajuda

Para saber mais pormenores sobre o funcionamento de cada comando pode consultar o manual do sistema usando o comando man. Por exemplo, o comando seguinte mostra a página do manual correspondente ao comando ls.

```
man ls
```

Exercício. Procure no manual uma variante do `ls`, que mostre os ficheiros por ordem cronológica inversa.

Por vezes, não sabemos o comando mas sabemos a operação que queremos efectuar. Por exemplo, se quisermos copiar um ficheiro, o comando “man copy” não devolveria a informação pretendida. Se precisarmos de pesquisar um comando pela sua descrição (por exemplo, copy), podemos fazer:

```
apropos copy
```

O `apropos` devolve a lista de comandos em cuja descrição esteja a palavra especificada. Um comando equivalente seria: `man -k copy`

8 Expansão. "Wildcards"

Um dos mecanismos mais usados da shell é a possibilidade de "apanhar" nomes de ficheiros usando o "*" ou, mais raramente, o "?". De forma simples, o significado do "*" é "qualquer coisa". Mais exatamente, ao ler um nome com * a interpretação é: no lugar o * pode figurar qualquer sequência de caracteres, seja vazio ("nada"), um caractere qualquer, dois, etc. A utilização mais vulgar do * é no ls. Por exemplo,

```
ls -l *.txt
```

lista os nomes de ficheiro cuja nome adira ao padrão "qualquer coisa seguido de `.txt`" (ou seja, os ficheiros com extensão `.txt`). Mas pode ser usado em muitas outras circunstâncias como forma de apanhar um grupo de ficheiros. Por exemplo o comando seguinte cria um ficheiro zip com todos os `.txt` existentes na diretoria corrente.

```
zip textos.zip *.txt
```

9 Escapes

O * e o ? são caracteres especiais para a shell. Como estes há outros, por exemplo $, & ou >. Se tivermos a ideie não muito feliz, de dar nomes a ficheiros contendo caracteres deste tipo podemos ter alguns problemas. Por exemplo, supondo que tínhamos a ideia não muito feliz de chamar a um ficheiro

```
***star***.txt
```

a utilização simples do nome do ficheiro em comandos, por exemplo

```
vi ***star***.txt
```

seria ambígua. Nesta situação teríamos que enquadrar o nome do ficheiro entre ' ':

```
vi '***star***.txt'
```

O mesmo aliás se usarmos nomes de ficheiros com espaços. Por exemplo se tivermos a ideia também não muito boa de chamar a um ficheiro

```
exemplo 1.txt
```

a utilização simples do nome do ficheiro, por exemplo:

```
vi exemplo 1.txt
```

é também ambígua e resolve-se com

```
vi 'exemplo 1.txt'
```

10 Redirecionamento

A maioria dos comandos mandam o resultado para o ecrã. Por exemplo, ao fazer:

```
ls -l
```

o resultado do comando, a lista ficheiros existentes no directório corrente, vai para o ecrã. Se fizermos a seguinte variante

```
ls -l > lista.txt
```

nada aparece no ecrã; o resultado do comando vai, sim, para o ficheiro *teste.txt*. Dizemos que o resultado do comando é redireccionado para o ficheiro *lista.txt*. Este é um exemplo de um mecanismo geral que a shell implementa e que permite redireccionar os canais "standard" de entrada e saída de um programa.

Normalmente a saída de um programa vai para o canal "standard de saída" (*stdout*) ou, em caso de erro, para o canal "standard de erros" (*stderr*). Um e outro podem ser redireccionados separadamente. A forma básica deste mecanismo é o normal redireccionamento da saída. Exemplos:

```
cat /etc/passwd > x   # saída para o ficheiro x
cat /etc/passwd >> x  # saída acrescenta-se ao ficheiro x
```

No exemplo seguinte, a saída normal é redireccionada para `/dev/null`; e a de erros continua no ecrã. Se x não existir, a mensagem de erro vai para o ecrã;

```
cat x 1> /dev/null    # ou apenas: cat x > /dev/null
```

No exemplo seguinte, a saída de erros é redireccionada para /dev/null. Se o ficheiro existir, aparece no ecrã; caso contrário, não aparece nada no ecrã;

```
cat x 2> /dev/null
```

O exemplo seguinte, redirecciona a saída standard para /dev/null e a saída de erros para o mesmo sítio;

```
cat x > /dev/null 2>&1
```

Resta acrescentar que `/dev/null` é um ficheiro muito especial: representa um "poço sem fundo" (ou buraco negro) para onde se pode redireccionar qualquer output, que é descartado pelo Unix (e Linux);

Existe também o canal “standard de entrada” (*stdin*) que normalmente está associado ao teclado.

Nos exemplos seguintes, 1) é criado um ficheiro x; 2) o ficheiro x é usado como canal de entrada para o comando cat; 3) copia-se o conteúdo do ficheiro x, para um novo ficheiro y.

```
echo "teste" > x
cat < x
cat < x > y          ou          cat > y < x
```

Finalmente o `|` (designado "pipe") permite encadear comandos fazendo com que o standard output de um comando seja encaminhado para input do programa seguinte. O exemplo seguinte mostra o encadeamento de 3 comandos: o primeiro mostra o conteúdo do ficheiro `/etc/passwd`; o segundo filtra esse conteúdo mostrando apenas as linhas que contêm a palavra root; o terceiro substitui ocorrências da palavra "root" pela palavra "raiz".

```
cat /etc/passwd | grep "root" | sed 's/root/raiz/'
```

11 "Hello World"

Considere o seguinte script, muito simples, que apenas escreve a mensagem “Hello World!” e "Adeus"

```
#!/bin/bash
echo "Hello world!"
echo -n "data e hora: "
date
echo "Adeus"
```

Para executar este script tem que:

1. escrever o programa num ficheiro de texto com extensão `.sh`; por exemplo `hello.sh`. Pode fazer isso em qualquer editor de texto como o vi, emacs, ou kwrite.
2. mudar as permissões do ficheiro

   ```
   chmod +x hello.sh
   ```

3. executar o script, indicando que ele se encontra na diretório atual.

Tipicamente na consola (janela de comandos) seria feita a seguinte sequência:

```
vi hello.sh                  # escrever o programa
chmod +x hello.sh            # mudar as permissões
./hello.sh                   # executar o script
```