O Comando `find` em Unix (breve resumo)

Aviso: existem versões do **find** para outros sistemas operativos, o texto refere-se ao **find** da GNU versão 4.1, outras versões do comando **find** podem não possuir as mesmas funcionalidades, este documento não é de forma alguma exaustivo, e se quer mesmo saber como o **find** funciona, nada melhor do que fazer: `man find`.

O comando **find** serve para encontrar ficheiros, o exemplo de utilização mais simples é:

```
find .
```

ou em versões mais antigas:

```
find . -print
```

Este comando encontra todos os ficheiros existentes debaixo do directório corrente,
e imprime o seu nome (acção por defeito).

Assim, o primeiro argumento do comando **find** é sempre o directório no qual queremos procurar os ficheiros.

No entanto a maior utilidade do comando **find** vem do facto de podermos seleccionar os ficheiros que queremos de muitas formas segundo as várias opções:

Ficheiros com um certo nome:

-name nome_do_ficheiro
-iname nome_do_ficheiro (não distingue entre maiúsculas e minúculas)

Exemplos:

```
find . -name core -print
find . -name \*.c -print
```

Nota: Para encontrar um ficheiro com um determinado nome é mais rápido usar o comando **locate** se o seu sistema unix estiver bem configurado. Associado ao comando **locate** existe o comando **updatedb** para fazer a actualização do ficheiro que o locate utiliza. Assim o **locate** consulta apenas um ficheiro, enquanto o **find** pode procurar em toda a árvore de directórios.

Ficheiros com um certo path:

-path pathname
-ipath pathname (não distingue entre maiúsculas e minúculas)

Exemplo:

```
find . -path misc -print
```

Pode encontrar (se existirem) os ficheiros:

```
./misc.c
./src/misc/test.c
./src/prog1/misc3.c
```

Ficheiros de um certo tipo:

-type tipo

O tipo poder ser:

b	block device
c	char device
d	directório
f	ficheiro "normal"
l	link simbólico
p	named pipe

s socket

Exemplos:

```
find . -type d -print
find . -type f -print
```

Ficheiros de um determinado grupo ou utilizador:

-user username
-uid user_id
-group nome_do grupo
-gid group_id
-nouser não pertencem a nenhum utilizador (já não existe o utilizador)
-nogroup não pertencem a nenhum grupo (já não existe o grupo)

Exemplos:

```
find . -username  joe -print
find . -uid 1024  -print
find . -nouser -print
```

Ficheiros de um certo tamanho:

-size n

Nota: o tamanho é escrito em blocos, que correspondem a 512 bytes nalguns sistemas unix e a 1024 bytes noutros… Para evitar problemas pode-se usar os seguintes sufixos:

c - chars=bytes
w -words
b - blocos
k - kbytes

Exemplos:

```
find . -size 100 -print                 ficheiros com 100 blocos
find . -size +100c -print               ficheiros com mais de 100 bytes
find . -size -100k -print               ficheiros com menos de 100 kbytes
```

Ficheiros que correspondem a um certo inode:

-inum n

Exemplo:

```
find . -inum 226 -print
```

Ficheiros com uma certa data/tempo:

Data de acesso dos ficheiros:
-atime -amin -anewer
Data de "mudança" (change) do ficheiro (pode ser mudança de dono ou atributos !!!)
-ctime -cmin -cnewer
Data de modificação (do conteúdo) do ficheiro
-mtime -mmin -newer

Nota: **time** - dias **min** - minutos

Exemplos:

```
find . -atime 2 -print          ficheiros acedidos há dois dias
find . -amin 1 -print           ficheiros acedidos há um minuto
find . -mtime -5 -print                 ficheiros modificados há menos de 5 dias
```

`find . -mmin +30 -print` ficheiros modificados há mais de 30 minutos
`find . -newer fich1.c -print` ficheiros modificados depois do ficheiro **fich1.c**

Ficheiros com certas permissões (exactamente) **-perm n**
Ficheiros com certos bits das permissões todos activados **-perm -n**
Ficheiros com certos bits das permissões activados **-perm +n**

Nota: as permissões são escritas em octal

Exemplos:

`find . -perm 700 -print` ficheiros com as permissões a 700
ficheiros com `rwx------` apenas!

`find . -perm -600 -print` com os dois primeiros bits activados
servem : `rw-------`
`rwx------`
`rw-rw----`
`rwxrwxrwx`, etc.
`find . -perm +600 -print` com um dos dois primeiros bits activados
servem: `r--------`
`-w-------`
`rw-rw----`
`rwxrwxrwx`, etc.

Podemos usar certas opções antes dos comandos:

-daystart medir os dias desde o inicio do dia e não desde há 24 horas

Exemplo:

`find . -mtime 1 -print` ficheiros modificados entre 24 e 48 horas atrás
`find . -daystart -mtime 1 -print` ficheiros modificados ontem

-xdev excluir outros "devices" (serve para excluir outros discos ou servidores da rede)
-mount excluir outros "devices" (o mesmo que **-xdev** para compatibilidade)
-maxdepth **n** descer no máximo **n** níveis de subdirectórios
-mindepth **n** considerar apenas os ficheiros a **n** ou mais níveis de subdirectórios
-follow seguir links simbólicos
-depth pesquisar em profundidade (ficheiros antes dos directórios onde eles estão)

Podemos usar mais do que uma opção:

```
find . -xdev -mindepth 3 -name core -print
```

ou podemos usar mais do que uma condição:

```
find . -xdev  -user joe -name core -print
```

Por defeito, o **find** faz o **"and"** das condições, ou seja, o comando só e executado se todas as condições se verificarem, embora se possam usar outros operadores lógicos

```
-a        -and
-o        -or
-not      !
```

e podemos usar parêntesis para agrupar as condições (protegidos da shell)

Nota: a avaliação das expressões é feita em "curto-circuito", isto é se num **or** a primeira expressão é verdadeira, ou num **and** a primeira expressão é falsa, a segunda expressão não é avaliada.

Exemplos:

```
find . -user joe -perm -002 -print
find . -user joe -a -perm -002 -print
find . -nouser -o -nogroup -o \( -name core -a -type f \) -print
```

A maior utilidade do comando find vem do facto de em vez de apenas imprimir o nome dos ficheiros que obedecem ao nosso critério, podermos fazer outras coisas:

`-print`	imprimir nomes dos ficheiros
`-fprint file`	imprimir nomes dos ficheiros num ficheiro
`-printf format`	imprimir nomes dos ficheiros num certo formato
`-fprintf file format`	imprimir nomes num ficheiro com um certo formato

Exemplos:

```
find . -fprint lista.lst
```

A string de formato é escrita nos moldes normais da linguagem C com:

Sequências de formatação:

%b tamanho do ficheiro em blocos de 512 bytes
%k tamanho do ficheiro em blocos de 1kbyte
%s tamanho do ficheiro em bytes
%a tempo de acesso no format standard
%A tempo de acesso formatado (ver na man page as opções)
%c tempo do status
%C tempo do status formatado (ver na man page as opções)
%F tipo do filesystem
%p nome completo do ficheiro (com path)
%f nome do ficheiro (sem path)
%u username
%g groupname

Sequências de escape:

\a alarm bell
\b backspace
\f form feed
\n newline
\c carriage return
\t horizontal tab
\v vertical tab
**** backslash
\c pára de imprimir e faz o "flush" da saída

Exemplo:

```
find . -printf 'Nome: %f  Dono: %u %s bytes\n'
```

Nota: tanto o **printf** como o **fprintf** não dão um newline por cada ficheiro!

-ls	listar os ficheiros como se usássemos as opções -idls do comando ls
-fls file	idem só que escrever num ficheiro

Exemplos:

```
find . -ls
find . -name \*.c -fls lista-c.txt
```

Por último podemos executar comandos nos ficheiros que escolhemos com:

-exec comando {} \; executar um comando
-ok comando {} \; executar um comando condicionalmente (pergunta ok ?)

O **find** substitui **{}** pelo nome de cada ficheiro que vai encontrando e o **\;** serve para dizermos ao **find** onde acaba o nosso comando.

Exemplos:

```
find / -user joe -exec rm -rf {} \;
```
apagar todos os ficheiros do utilizador joe

```
find . -type f -name \*.bak -ok rm {} \;
```
apagar condicionalmente os ficheiros *.bak

Notas:

Nalgumas shells convém fazer o escape das chavetas, para saber em quais, experimentar ou rtfm. Porque é que temos de assinalar o fim de um comando? Porque as acções para o **find** comportam-se como condições e podemos ter mais de uma acção no mesmo **find**.

Exemplo:

```
find . -name \*.c -exec cp {}  ~/backup \; -size +10k -ok rm {} \;
```

Encontrar todos os ficheiros ***.c** , copiá-los todos para o directório **~/backup** e perguntar se o utilizador quer apagar os que têm mais de 10 kbytes.

Exercicios

1) Descubra todos os ficheiros e directórios que lhe pertencem.

2) Descubra todos os comandos (ficheiros) da directoria /bin e /usr/bin que não foram acedidos à mais de 90 dias.

3) Descubra todos os ficheiros no seu directório corrente que foram modificados nos últimos 3 dias.

4) Descubra todos os directórios que lhe pertencem a si ou ao utilizador ixxxxxx (*loginid* diferente do seu) e que se encontram no directório pai do seu directório **home**

5) Liste todos os ficheiros que estão no directório /etc cujas permissões são:

"donos" com permissão para leitura, escrita e execução;

"outros" apenas com permissão para leitura e escrita.

6) Copie os ficheiros começados por "f", "p" ou "c" do seu directório corrente para o directório /tmp usando o comando **find**.

7) Elimine qualquer ficheiro do directório /tmp que seja seu e que tenha sido acedido a menos de 2 dias. Assegure-se de que vai poder confirmar antes de proceder à eliminação.

8) Explique o significado das seguinte linha de comando:

8.1) find $HOME -name "*.c" -exec mv \{\} $HOME/cdir \;

9) O comando apropriado para procurar a partir da root (/) o ficheiro compras e mostrar a sua "path" é :

a) find /root/compras* -path
b) find –file "compras" / -print
c) find / -name compras –print
d) find . –name "compras" –path

10) Escreva um comando que permita apagar todos os ficheiros do directório **/lixo**, que não tenham sido acedidos à mais de 30 dias e cujo tamanho exceda os 5000 caracteres.

11) Escreva um comando que permita mover dos directórios da nossa área de trabalho, todos os ficheiros que não nos pertencem, para o directório "nao_meus" que se encontra na nossa home directory.