# Introducción a Sistemas Operativos: Ficheros

*Clips xxx*
*Francisco J Ballesteros*

## 1. Montajes y nombres

En UNIX tienes todos los ficheros dispuestos en un único árbol como ya sabes. El directorio raíz es "`/`" y en dicho directorio están contenidos todos los ficheros a que puedes acceder. Esto es así incluso si tienes varias particiones con ficheros o varios discos.

Una vez más, estamos ante otra abstracción suministrada por el sistema, la idea de un **espacio de nombres** único que podemos adaptar para crear la ilusión de un sólo árbol, aunque tengamos múltiples árboles de ficheros. Normalmente cada árbol está guardado en una partición utilizando un formato concreto de sistema de ficheros, pero recuerda que existen árboles de ficheros implementados en software que no tienen almacenamiento en disco (como `/proc`).

La implementación es simple: UNIX dispone de una *tabla de montajes* que hace que, al recorrer paths, el kernel pueda saltar de un directorio en el árbol al directorio raíz de de otro árbol.

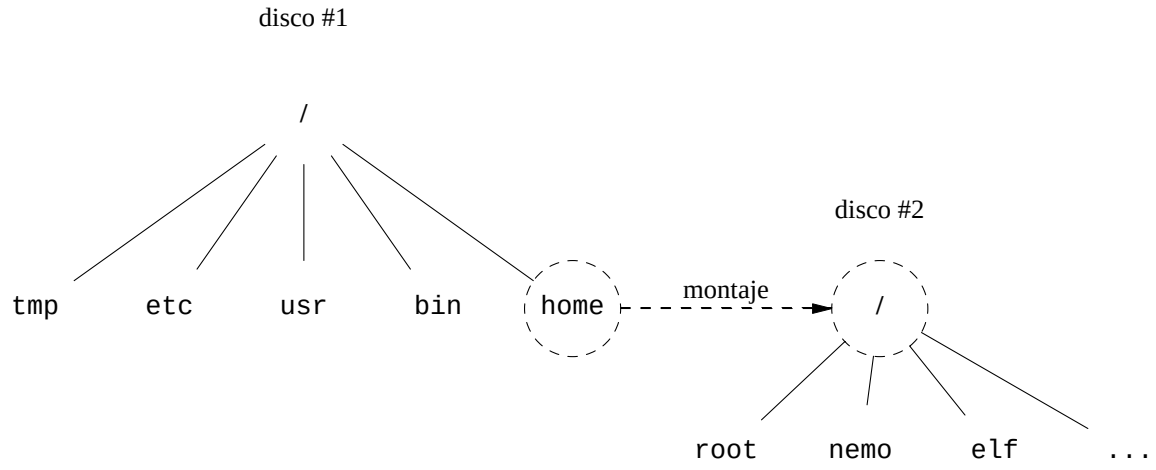


**Figura 1:** *Un montaje hace que UNIX salte de un directorio en el árbol al raíz de otro árbol creando el efecto de un único árbol.*

El efecto puedes verlo en la figura 1. En ella puedes ver que el administrador de este sistema ha utilizado dos discos (o dos particiones, a UNIX le da lo mismo) y ha hecho que en el directorio `/home` se monte el segundo disco (o la segunda partición). Tras el montaje, cuando el kernel resuelva paths y alcance "`/home`", saltará al directorio raíz del segundo disco, por lo que aparentemente tenemos paths como "`/home/nemo`", lo que es una ilusión.

A la vista de esto puedes imaginar que si `/home` contenía ficheros o directorios antes del montaje, dicho contenido resulta ahora inaccesible. Aparentemente `/home` contiene los directorios `root`, `nemo`, etc. y eso es todo lo que pueden ver los usuarios del sistema.

El comando `mount(1)` muestra los montajes en el sistema:

```
unix$ mount
/dev/disk1 on / (hfs, local, journaled, noatime)
devfs on /dev (devfs, local, nobrowse)
elf@fs.lsub.org:/home/dump on /dump (osxfusefs, nodev, nosuid)
elf@fs.lsub.org:/home/lsub on /zx (osxfusefs, nodev, nosuid)
elf@fs.lsub.org:/home/once on /once (osxfusefs, nodev, nosuid)
unix$
```

El resultando dependerá mucho no sólo del tipo de UNIX sino también de cómo esté administrado. En este caso vemos que hay una sóla partición en el primer disco que está montada en el directorio raíz. Por otra parte, no hay `/proc` en este sistema (OS X) y `/dev` es un sistema de ficheros sintético (igual que lo es `/proc` en otros UNIX). Puede verse también que los directorios `/dump`, `/zx` y `/once` están montados y proceden de otra máquina llamada `fs.lsub.org`.

Existe otro comando que resulta útil no sólo para ver qué tenemos montado sino también para ver cuánto espacio libre resta en cada uno de los sistemas de ficheros. hablamos de *df(1)*. Para que puedamos ver otro ejemplo, vamos a utilizar un sistema OpenBSD esta vez:

```
unix$ mount
/dev/sd2a on / type ffs (local, noatime, softdep)
unix$ df -h
Filesystem     Size    Used   Avail Capacity  Mounted on
/dev/sd2a      1.8T    239G    1.5T    14%    /
unix$
```

En este sistema hay un único sistema de ficheros montado en el raíz, procedente de la primera partición (la "`a`" en "`sd2a`") del tercer disco (el "`2`" en "`sd2a`", empezando a contar en cero). Con el flag `-h` (*human readable sizes*) `df` informa que en dicha partición de 1.8TiB estamos usando 239GiB.

El flag `-i` de `df` es muy útil e informa de cuántos *i-nodos* estamos usando en cada sistema de ficheros.

```
unix$ df -ih
Filesystem     Size    Used   Avail Capacity iused   ifree  %iused  Mounted on
/dev/sd2a      1.8T    239G    1.5T    14% 1727771 58895843     3%   /
```

Como podrás suponer, una vez hemos usado todos los *i-nodos* de que dispone un sistema de ficheros ya no es posible crear nuevos ficheros dentro del mismo.

Para montar sistemas de ficheros hay que ser *root* (a no ser que dicho usuario se ocupe de configurar el sistema para que un usuario normal pueda realizar ciertos montajes, como por ejemplo sucede con los discos USB hoy día). Un ejemplo es

```
unix$ mount -t mfs -o rw /dev/sd0b /tmp
```

que monta (en un sistema OpenBSD) en `/tmp` un sistema de ficheros en memoria virtual, respaldado por una partición de swap en disco. (Dicha partición se utiliza para mantener en ella la parte de la memoria virtual que no nos cabe en memoria física).

Para deshacer el efecto de *mount(8)* disponemos de *umount(8)*. Por ejemplo,

```
unix$ umount /tmp
```

deshace el montaje del ejemplo anterior. El sistema se negará a desmontar un sistema de ficheros que esté utilizándose. Por ejemplo, si un proceso tiene su directorio actual dentro de dicho sistema de ficheros o tiene ficheros abiertos procedentes del mismo, o está paginando su código desde un ejecutable almacenado en él. Si lo intentamos...

```
unix$ umount /
/dev/sd2a: device busy
unix$
```

Tenemos llamadas al sistema en *mount(2)* que utilizan los comandos que hemos visto, pero es muy poco probable que las necesites.