# 7. Ficheros y comandos avanzados

Elena García-Morato, Felipe Ortega, Enrique Soriano

GSyC, ETSIT. URJC.

Laboratorio de Sistemas (LSIS)

24 abril, 2023


Universidad
Rey Juan Carlos

# Contenidos

# 7.1 Metadatos

# Metadatos

- Datos sobre un fichero, no los datos contenidos en un fichero.
- Ejemplos que ya conoces: nombre del fichero y permisos del fichero.
- Existen más metadatos.

# I-nodo

- En realidad, en un sistema tipo Unix un directorio es una **lista de entradas**.
- Una entrada de directorio es, básicamente, el nombre de un fichero y un número que identifica a dicho fichero.
- Es número se llama inodo, nodo-i o nodo índice.
- Distintas entradas pueden tener distintos nombres para el mismo i-nodo. Esto es, se pueden tener múltiples referencias al mismo fichero (i.e. un fichero puede tener distintos nombres).
- Entre las entradas, tiene la entrada de . (su i-nodo) y .. (i-nodo del padre).
- `ls -i`.

# I-nodo

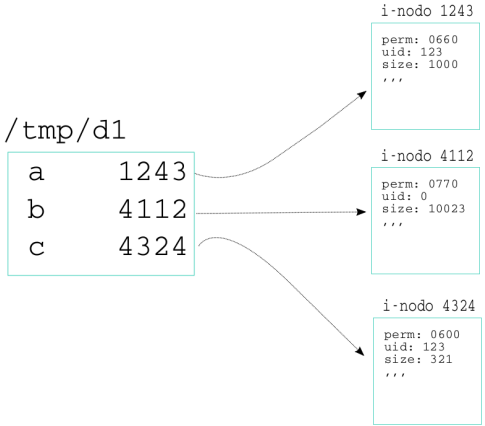


Figura 1: Ejemplo de identificación de tres archivos mediante inodos.

# Inodo

```
$ mkdir d
$ cd d
$ ls -di .
2371879 .
$ ls -di ..
2359297 ..
$ echo hola > a
$ echo adios > b
$ ls -i
2371884 a 2371885 b
```

# I-nodo

- El directorio raíz siempre tiene el número de i-nodo 2.
- El i-nodo es una estructura que está en la partición del sistema de ficheros. El número de i-nodo se usa para localizar la estructura (es el índice en un array).
- La estructura i-nodo es la que contiene los metadatos del fichero y la lista de bloques de disco que contienen los datos del fichero.

# I-nodo

- La estructura contiene:
  - permisos
  - tiempos
    - acceso a datos (`atime`)
    - modificación de los datos (`mtime`)
    - modificación del i-nodo (`ctime`)
  - tamaño
  - dueño
  - tipo
  - número de bloques
  - contador de referencias (*links*)
- **No contiene** el nombre de fichero.

# Enlaces duros

- Un enlace duro es otro nombre para el fichero, esto es, otra entrada de directorio para ese i-nodo.
- El contador de referencias del i-nodo se incrementa cada vez que se crea un nombre nuevo para el fichero.
- El contador de referencias se decrementa cada vez que se elimina una entrada de directorio que hace referencia al i-nodo.

# Enlaces duros

- Cuando el contador llega a 0, se puede eliminar el fichero (se libera el i-nodo y todos los bloques asociados a este, donde estaban los datos del fichero).
- En Linux no se permite crear enlaces duros para directorios: rompen la jerarquía, crean bucles y crea ambigüedad con .. (el padre).

# Enlaces duros

- El comando `ln` (sin la opción `-s`) permite crear un enlace duro.

```
$ echo hola > a
$ ls -i a
2371879 a
$ ln a b
$ ls
a b
$ ls -i
2371879 a 2371879 b
$ cat a
hola
$ cat b
hola
$ rm a
$ cat b
hola
```

# 7.2 Ficheros especiales

# Tipos de ficheros

- Ya sabes qué es un fichero normal y un directorio.
- Hay otros tipos:
  - Enlaces simbólicos (symlink, `'l'`).
  - Pipes con nombre (fifo, `'p'`).
  - Dispositivos (device, `'c'` o `'b'`).
  - Conexiones de red (sockets, `'s'`).

# Enlaces simbólicos (blandos)

- **No** se trata de otro nombre para el mismo i-nodo.
- Se trata de un fichero especial distinto (*symlink*), cuyos datos contienen la ruta al fichero enlazado.
- Pueden *romperse*: si el fichero enlazado se borra, el enlace está roto.
- También se crean con el comando `ln`, usando el modificador `-s`.

# Enlaces simbólicos (blandos)

```
$ echo hola > a
$ ln -s a b
$ ls -li
total 4
2371879 -rw-rw-r-- 1 esoriano esoriano 5 mar 21 13:42 a
2371883 lrwxrwxrwx 1 esoriano esoriano 1 mar 21 13:43 b -> a
$ cat b
hola
$ rm a
$ cat b
cat: b: No such file or directory
$ ls -li
total 0
2371883 lrwxrwxrwx 1 esoriano esoriano 1 mar 21 13:43 b -> a
```

# Pipes con nombre

- Se llaman fifo.
- Es un pipe con nombre que persiste en el tiempo.
- Se crean con el comando `mkfifo`.
- Se borran con el comando `rm`.

# Pipes con nombre

```
$ mkfifo p
$ echo uno dos > p  & sed 's/o/X/g' p
[1] 4459
unX dXs
[1]+ Done          echo uno dos > p
$
```

# Dispositivos

- En Unix, los dispositivos se presentan como ficheros.
- Ventaja: podemos usar las mismas herramientas que usamos con los ficheros convencionales.
- Ya conoces `/dev/null`, etc.
- Los dispositivos están en `/dev/`. Siguen un esquema de nombrado.
- P. ej.: los discos comienzan por sd seguido de la posición en el bus (a, b, etc.) seguido del número de partición: `/dev/sda2`, `/dev/sdb1`, etc.

# Dispositivos

Hay dos tipos de dispositivos:

- Un dispositivo de caracteres trabaja con flujos de bytes.
- Un dispositivo de bloques trabaja con trozos de datos de cierta longitud (bloque). El dispositivo tiene un tamaño determinado y se tiene acceso aleatorio a sus bloques (i.e. discos duros).
- Tienen asociado dos números que sirven al núcleo para identificar el dispositivo dentro del *kernel*:
  - *Major number*: indica la clase del dispositivo (i.e. su driver), todos los de la misma clase tienen el mismo.
  - *Minor number*: indica la instancia concreta dentro de esa clase de dispositivo.

# Dispositivos

```
$ cd /dev
$ ls -l sda1 sda2
brw-rw---- 1 root disk 8, 1 mar 21 16:39 sda1
brw-rw---- 1 root disk 8, 2 mar 21 16:39 sda2
$ ls -l tty1 tty2
crw--w---- 1 root tty 4, 1 mar 21 16:39 tty1
crw--w---- 1 root tty 4, 2 mar 21 16:39 tty2
```

# Dispositivos

- Los dispositivos se crean con el comando `mknod`. Hay que proporcionar el tipo (b,c) y los dos números (major y minor).
- No es común tener que crearlos, el sistema los crea automáticamente.
- El demonio udev se encarga de esta gestión.

# El comando dd

- El comando dd es muy útil para copiar datos entre dispositivos de bloques. Sus modificadores habituales son:
  - `if=file` fichero de entrada.
  - `of=file` fichero de salida.
  - `bs=size` tamaño de un bloque (p. ej. `bs=1k` o `bs=1024`).
  - `count=num` número de bloques a transferir.
  - `skip=num` número de bloques a saltar de la entrada.
  - `seek=num` número de bloques a saltar en la salida.

# El comando rsync

- Es una herramienta que permite copiar ficheros locales o remotos de manera rápida y versátil.
- En lugar de copiar ciegamente datos, compara los directorios fuente y destino y sólo se copian los elementos que han cambiado.

```
Local:
    rsync [OPTION...] SRC... [DEST]

Access via remote shell:
    Pull:
        rsync [OPTION...] [USER@]HOST:SRC... [DEST]
    Push:
        rsync [OPTION...] SRC... [USER@]HOST:DEST
```

# El comando rsync

- Ejemplo: comando para copia de respaldo de todos nuestros ficheros del `$HOME`.
- Suponemos que queremos almacenar la copia de respaldo en el directorio `/media/$USER/linuxbackup/home/`
- Se puede configurar una lista que excluye aquellos directorios con información superflua que no necesitamos en el respaldo (para ahorrar espacio).
  - Ejemplo de fichero con lista de elementos a excluir.

```
$ wget https://raw.githubusercontent.com/rubo77/rsync-homedir-excludes/master/rsync-homedir-excludes.txt
  -O /var/tmp/ignorelist
$ rsync -aP --exclude-from=/var/tmp/ignorelist /home/$USER/ /media/$USER/linuxbackup/home/
```

# 7.3 Inspección del sistema

# Inspección del sistema

- Desde el *shell*, tenemos dos formas de inspeccionar el sistema:
  - Navegando por sistemas de ficheros sintéticos como `/proc`.
  - Ejecutando comandos especializados como `ps`.

# /proc

- Es un sistema con ficheros generados *al vuelo*.
- Tiene un directorio por cada proceso en ejecución (p. ej. `/proc/323` para el proceso con PID 323).
  - Además, tiene otros ficheros y directorios importantes.
  - `man 5 proc`.

# /proc

- Dentro del directorio para un proceso tenemos (entre otros):
  - `cmdline`: línea de comandos que ejecuta
  - `cwd`: enlace simbólico a su directorio actual
  - `environ`: variables de entorno
  - `exe`: enlace simbólico al ejecutable
  - `fd`: ficheros que tiene abiertos
  - `io`: información sobre entrada/salida
  - `maps`: regiones de memoria
  - `mem`: memoria del proceso

# /proc

- `/proc` tiene otros ficheros y directorios de interés:
  - `cpuinfo`: información sobre la CPU
  - `kmsg`: *log* del *kernel*
  - `meminfo`: información sobre el uso de la memoria
  - `modules`: módulos cargados
  - `net`: directorio con información sobre la red
  - `uptime`: tiempo que lleva levantado el sistema
  - ...

# /sys

- Es otro directorio con ficheros sintéticos como `/proc`, es la misma idea.
- Es una interfaz para acceder a las estructuras internas del kernel.
- `man 5 sysfs`.

# /sys

Entre otros, ofrece:

- `block`: enlaces simbólicos para cada dispositivo de bloques.
- `class`: directorios por cada clase de dispositivo.
- `devices`: representación del árbol de dispositivos.
- `firmware`: interfaz para manipular objetos y atributos del firmware.
- `fs`: interfaz para controlar los sistemas de ficheros
- `kernel`: manipulación variada del kernel
- `module`: módulos cargados
- `power`: manipulación de la gestión de energía

# ps

- `ps` lista los procesos del sistema.
- Comúnmente se ejecuta con los modificadores `aux`: mostrar todos los procesos (Sintaxis BSD).
  - **Precaución**: los comandos `ps -aux` (sintaxis UNIX) y `ps aux` (sintaxis BSD) no son equivalentes. Aquí nos referimos al segundo.
- Hay muchos otros modificadores para mostrar distintas columnas, hilos de los procesos, etc. Ya hemos visto muchos de ellos en los Temas 1 y 2.
  - `man ps`

# top

- `top` muestra los procesos refrescando sus datos.
- Es útil para ver el consumo de CPU y de memoria de los procesos.
- Las filas se pueden ordenar de distinta forma.
- Pulsando `h` (o también `?`) nos ofrece algunas de las opciones interactivas. Se sale pulsando `q`.
  - Pulsando `P` ordena por uso de CPU
  - Pulsando `M` ordena por consumo de memoria, etc.
- Hay alternativas más intuitivas y fáciles de usar, como `htop`.

# lsof

- `lsof` muestra los ficheros que tienen abiertos los procesos.
- Como ya sabemos, hay ficheros de múltiples tipos (*sockets*, *pipes*, etc.).
- Por omisión nos muestra columnas con el comando, PID, usuario, descriptor de fichero, tipo, dispositivo que lo sirve, tamaño, posición y nombre del fichero.

# mount

- `mount` sin argumentos muestra los sistemas de ficheros que están montados.
- Indica: dispositivo, punto de montaje, tipo de sistema de ficheros y opciones de montaje.
- El fichero `/etc/fstab` tiene algunos de los sistemas de ficheros que se pueden montar.

# df

- `df` muestra el espacio libre de los distintos sistemas de ficheros montados.
- La opción `-h` da los tamaños en un formato más legible y en potencias de $2^10$.
- La opción `-i` Lista la información de los i-nodos en lugar de el uso de bloques en sistemas de ficheros.

```
$ df -h
S.ficheros     Tamaño Usados  Disp Uso% Montado en
tmpfs            3,2G   2,6M  3,2G   1% /run
/dev/nvme0n1p2   916G   405G  465G  47% /
tmpfs             16G   184K   16G   1% /dev/shm
tmpfs            5,0M   4,0K  5,0M   1% /run/lock
/dev/nvme1n1p1   916G   132G  738G  16% /data
/dev/nvme0n1p1   511M   6,1M  505M   2% /boot/efi
tmpfs            3,2G   4,0M  3,2G   1% /run/user/1000
```

# netstat

- `netstat` ofrece información sobre las conexiones de red.
- Tiene muchas opciones, las más habituales:
  - `-a`: lista todas las conexiones.
  - `-at`: lista solo conexiones TCP.
  - `-au`: lista solo conexiones UDP.
  - `-tnl`: lista los puertos de nuestra máquina en los que están escuchando servicios.
  - `-nr`: muestra la tabla de enrutamiento de la máquina.

# dmesg

- `dmesg` muestra el *kernel ring buffer*, esto es, los mensajes que escribe el *kernel*.
- El kernel escribe mensajes de diagnóstico de distinto nivel de importancia (información, avisos, errores de distinta gravedad, etc.).
- El opción –w hace que se quede esperando nuevos mensajes.

# Usuarios

- `w` muestra la lista de usuarios que están usando el sistema ahora mismo, en qué terminal están, etc. El comando `who` sirve para lo mismo (es menos completo).
- `last` muestra la lista de los últimos usuarios que han entrado en el sistema. Muestra tanto entradas locales como remotas (p. ej. por SSH).

# Usuarios

- `w` muestra la lista de usuarios que están usando el sistema ahora mismo, en qué terminal están, etc. El comando `who` sirve para lo mismo (es menos completo).
- `last` muestra la lista de los últimos usuarios que han entrado en el sistema. Muestra tanto entradas locales como remotas (p. ej. por SSH).

# Logs

- Los fichero de bitácora (*logs*) principales del sistema están en `/var/log`.
- Cuidado: algunos son binarios, otros son de texto.
- Se van **rotando**: cuando un fichero llega a un cierto tamaño (o tiempo desde su creación) se comprime y se renombra (asignándole un número). P. ej.: `kern.log.2.gz`.
- `systemd` también mantiene un *log*, se puede inspeccionar con el comando `journalctl`.

# Logs

Algunos *logs* relevantes:

- `syslog`: información general del sistema, todo menos temas de autenticación. Un demonio `syslog` recopila los mensajes de distintos servicios y programas y los escribe en este log. Syslog es configurable (/etc/syslog.conf).
- `auth.log`: información sobre autenticaciones (correctas y fallidas).
- `kern.log`: mensajes del *kernel*.
- `wtmp`: contiene la información que muestra el comando `last`.
- `dpkg.log`: mensajes sobre la instalación y gestión de paquetes Debian.
- `Xorg.N.log`: mensajes de la interfaz gráfica en la sesión `N`.

# 7.4 Trabajo en remoto

# SSH

- `ssh` permite trabajar con uns *shell* en un sistema remoto de forma segura (túnel de comunicación cifrado).
- Es la versión segura de otros programas antiguos (inseguros, envían la información sin cifrar) como `telnet`, `rsh`, etc.

```
$ ssh egarciam@maquina01.gsyc.urjc.es
```

- Se recomienda encarecidamente seguir las instrucciones para configurar un *certificado seguro* (p.ej. RSA) para identificar a los usuarios en la máquina remota y **deshabilitar el acceso remoto mediante contraseña**.

# El comando scp

- El comando scp permite copiar un fichero desde el servidor remoto a la máquina local (por ejemplo, del laboratorio a casa):

```
scp tu-login@servidor:fichero-origen fichero-destino
```

- Ejemplos:

(copia el fichero `expr.p` de tu directorio personal del servidor al directorio actual en tu PC)

```
scp supercoco@alpha.aulas.gsyc.urjc.es:expr.p expr.p
```

(copia el fichero `a.txt` de tu directorio Escritorio en el servidor al directorio actual en tu PC)

```
scp supercoco@alpha.aulas.gsyc.urjc.es:Escritorio/a.txt a.txt
```

# El comando scp

- Para copiar un fichero local a la máquina remota (por ejemplo al laboratorio desde casa):

```
scp fichero-origen tu-login@servidor:fichero-destino
```

- Ejemplos:

(copia el fichero `expr.p` del directorio actual de tu PC en tu directorio personal en el servidor)

```
scp expr.p supercoco@alpha.aulas.gsyc.urjc.es:expr.p
```

(copia el fichero `func.p` del directorio actual de tu PC en tu directorio `Documentos` en el servidor)

```
scp func.p supercoco@alpha.aulas.gsyc.urjc.es:Documentos/func.p
```

# Para saber más

- El capítulo 2 de [1] contiene información adicional sobre sistemas del *kernel*.
- El libro de referencia sobre administración en Unix/Linux [2] incluye mucha información adicional y excelentes explicaciones:
  - Capítulo 3: El sistema de ficheros.
  - Capítulo 11: Archivos de registro de eventos del sistema (*syslog*).
  - Capítulo 29: Utilización de herramientas en este capítulo para análisis de rendimiento del sistema.
- El capítulo 8 de [3] describe herramientas para inspección de procesos y utilización de recursos del sistema; el capítulo 12 describe herramientas para transferir archivos (como `rsync` o `scp`).

# Referencias I

[1] M. Hausenblas. *Learning Modern Linux*. O'Reilly Media, abr. de 2022.

[2] E. Nemeth y col. *Unix and Linux System Administration Handbook*. 4ª ed. Pearson, 2010.

[3] B. Ward. *How Linux Works: What Every Superuser Should Know*. 3ª ed. No Starch Press, abr. de 2021.