# Hoja de Ejercicios 2. Sistema Ficheros

### Contenidos

**Manejo de Ficheros y Directorios**

- Atributos de un fichero
- API para el manejo de ficheros y directorios

**Sistema de Ficheros**

## Ejercicios Prácticos

Los ejercicios marcados con el icono 💻 son prácticos y deben realizarse en el laboratorio. Los ejercicios prácticos de esta hoja requieren el entorno de usuario básico (shell) y de desarrollo (compilador, editores y depurador). Además algunos ejercicios necesitan acceso de superusuario que está disponible en las máquinas virtuales de la asignatura.

# Manejo de Ficheros y Directorios

## Atributos de un fichero

💻 ***Ejercicio 1.*** `ls(1)` muestra el contenido de directorios y los atributos básicos de los ficheros. Consultar la página de manual y estudiar el uso de las opciones `-a -l -d -h -i -R -1 -F`.

💻 ***Ejercicio 2.*** Considerar el directorio `/run`:

- Escribir un comando que obtenga las características detalladas de todos los ficheros.
- ¿Cuántos tipos de ficheros (directorio, fichero regular, enlace…) diferentes hay?
- Escribir una línea de comandos que muestre qué tipos diferentes hay en ese directorio. Los tipos se indicarán con el caracter (`d`, `-`, `l`...) usado por el comando `ls`. Ejemplo:

```
$ <línea de comandos ls ... | ... |>
-
d
l
p
s
```

Pista: El comando cut(1) puede utilizar se para *cortar* el primer byte de cada línea.

💻 ***Ejercicio 3.*** Crear un fichero en el directorio home del usuario con la orden:

```
echo “Fichero de prueba” > archivo.txt
```

Usando el comando stat(1) contestar a las siguientes preguntas:

- ¿Cuál es el inodo del fichero?
- ¿Qué usuario puede leer los contenidos del fichero? ¿y modificarlo?

- ¿Cuál es el tamaño del fichero? ¿Cuántos bloques ocupa en el sistema de ficheros?
- ¿Cuál es la ruta del disco (relativa a `/dev`) dónde se almacena el archivo?
- Determinar qué atributos del inodo del fichero cambian cuando se ejecutan los siguientes comandos:
    - `touch archivo.txt`
    - `echo "Otra línea más en el fichero" >> archivo.txt`
    - `cat archivo.txt`

💻 ***Ejercicio 4.*** Escribir un programa (`mistat`) que emule el comportamiento de `stat(1)`. El programa aceptará un único argumento que será la ruta del fichero del que se quiere obtener la información. Si el fichero no existe se informará del error. La información del fichero será:

- El número *major* y *minor* asociado al dispositivo, ver major(3)/minor(3).
- El número de inodo del fichero.
- El tipo de fichero (únicamente considerar directorio, enlace simbólico o fichero ordinario).
- La hora en la que se accedió el fichero por última vez. ¿Qué diferencia hay entre `st_mtime` y `st_ctime`?

Ejemplo de salida:

```
$ ./mistat /no/existe
Error en stat: No such file or directory

$ ./mistat archivo.txt
inodo: 256395
Número de dispositivo: [8,1]
Tipo: fichero regular
Último cambio de estado: Fri Feb 19 10:51:56 2025
Último acceso: Fri Feb 19 10:53:15 2025
```

**Nota:** Si el archivo es un enlace simbólico debe mostrar la información del enlace y no del fichero al que apunta.

***Ejercicio 5.*** Los permisos se pueden otorgar de forma selectiva usando la notación octal o la simbólica. Ejemplo, probar las siguientes órdenes (equivalentes):

- `chmod 540 fichero`
- `chmod u=rx,g=r,o= fichero`

Consultar la página de manual `chmod(1)` para ver otras formas de fijar los permisos (p.ej. los operadores + y -). Contesta las siguientes preguntas:

- ¿Cómo se podrían fijar los permisos `rw-r--r-x` usando notación octal?

- ¿Qué permisos debería fijar para que el usuario y grupo puedan leer y escribir y el resto de usuario nos tengan acceso?

- Considera la siguiente salida del comando ls:

```
$ ls -ld /tmp/ /usr/bin/passwd

drwxrwxrwt 12 root root  4096 Fre 12 06:48 /tmp/
-rwsr-xr-x  1 root root 59976 Nov 24  2022 /usr/bin/passwd
```

¿Qué permisos tiene el directorio /tmp y el fichero /usr/bin/passwd en notación octal? ¿Cuál es su significado?

- ¿Qué significan los permisos de ejecución en un directorio? ¿Qué comando se puede utilizar para dar permisos de ejecución?

- 💻 Verifica las respuestas anteriores en la máquina del laboratorio.

💻 ***Ejercicio 6.*** Considera el archivo.txt creado en el ejercicio 3:

- Crea un enlace simbólico (symlink.txt) y otro duro (hardlink.txt) con el comando `ln(1)`.
- Completar la siguiente tabla usando el comando `ls(1)` (opciones `-l y -i`) o `stat(1)`:

| Archivo | i-nodo | Número de enlaces | Tamaño |
|---|---|---|---|
| `archivo.txt` | | | |
| `symlink.txt` | | | |
| `hardlink.txt` | | | |

- ¿Qué sucede con los enlaces si se borra el archivo.txt, siguen los contenidos disponibles? ¿Cómo cambia el número de enlaces?

***Ejercicio 7.*** Considera la siguiente salida del comando `stat(1)`. ¿Qué tipo de fichero se trata, cuánto tamaño ocupa en disco?:

```
$ stat ejercicio7.file
  File: ejercicio7.file
  Size: 105906176    Blocks: 2048       IO Block: 4096   regular file
  Device: 801h/2049d Inode: 256409      Links: 1
...
```

## API para el manejo de ficheros y directorios

💻 ***Ejercicio 8.*** Implementar una versión simplificada del comando `dd`. El programa recibirá 5 argumentos posicionales, en :

- **`input_file`**: El archivo de entrada del que se leerá. Si el archivo es '-' se leerá de la entrada estándar.
- **`output_file`**: El archivo de salida en el que se escribirá. Si el archivo es '-' se escribirá en la salida estándar.
- **`block_size`**: El número de bytes que se leerán o escribirán en cada llamada al sistema. (entero, int)
- **`block_count`**: El número de bloques que se copiarán (entero, int)
- **seek:** Número de bloques que se saltarán de la salida antes de escribir.

Notas sobre la implementación:

- Tratar los errores en las llamadas al sistema adecuadamente. Asumir que el programa siempre se ejecuta con 5 argumentos.

- La función atoi(3) puede usarse para convertir una cadena de caracteres a un entero.
- El fichero de salida se creará (con permisos `rw- rw- r--`) si no existe, en caso contrario se truncará el contenido.
- El programa usará un buffer estático de 8192 bytes. Si el tamaño de bloque indicado es mayor se usará el tamaño máximo del buffer como tamaño de bloque.
- Tratar adecuadamente el valor que devuelven las llamadas `read(2)` y `write(2)`. Además considerar que la llamada read()/write() pueden no leer un bloque completo en la llamada.

Ejemplo de salida del comando:

```
$ ./midd /dev/random ./prueba1 512 10 0
Copiados 10 bloques de 512 bytes

$ ls -l prueba1
-rw-r----- 1 ubuntu ubuntu 5120 Feb 3 20:57 prueba1
```

**Cuestiones**

- Comprobar el funcionamiento de lectura de bloques incompletos mostrado a continuación ( ¿Qué hace la orden de comandos indicada?). Para el ejemplo el programa debería escribir con dos llamadas write de 8 y 2 bytes, comprobarlo con la orden `strace` (Tema 1) :

```
$ ((echo "0123"; sleep 1; echo "4567")&) |  ./midd - prueba2 8 2 0
Copiados 2 bloques de 8 bytes

$ ls -l prueba2
-rw-r----- 1 ubuntu ubuntu 10 Feb 3 21:00 prueba2

$ cat prueba2
0123
4567
```

- Finalmente comprobar si el comando puede generar un archivo *sparse* de 100MB. Elegir los argumentos adecuados para midd y verificar que el fichero es *sparse* con los comandos: `ls -lh`, `du -h` y `stat`.

💻 ***Ejercicio 9.*** Escribir un programa que tenga un comportamiento similar a `ls`. El programa `mils` mostrará el contenido de un directorio cuya ruta se proporciona como argumento. Para ello, el programa:

- Comprobará que el argumento es un directorio y que tiene acceso con la llamada al sistema adecuada.

- Recorrerá las entradas del directorio y escribirá su nombre de fichero. Además:
  - Si es un fichero regular y tiene permiso de ejecución para usuario, grupo u otros, escribirá el carácter `*` después del nombre.
  - Si es un directorio escribirá el carácter `/` después del nombre.
  - Si es un enlace simbólico, escribirá `->` y el nombre del fichero enlazado obtenido con `readlink(2)`.

**Nota:** la variable `d_name` de las estructuras `dirent` sólo contienen el nombre para obtener los atributos del archivo es necesario especificar la ruta completa concatenando el nombre del

directorio, en el primer argumento.

Para *concatenar* ambas cadenas, directorio y nombre del archivo, definir un buffer de tamaño `PATH_MAX` (`#include <linux/limits.h>`) y la llamada `snprintf(3)`.

Ejemplo de uso:

```
$ ./mils dir_prueba/
Contenidos del direcotrio dir_prueba/
--------------------------------------
programa1*
subdir1/
archivo1
../
archivo2
link1 -> /etc/passwd
./

#Comprobación comparando la salida con el comando ls
$ ls -l dir_prueba/
total 12
drwxr-xr-x 3 ubuntu ubuntu 4096 Feb 3 22:02 .
drwxr-xr-x 3 ubuntu ubuntu 4096 Feb 3 22:02 ..
-rw-r--r-- 1 ubuntu ubuntu    0 Feb 3 22:02 archivo1
-rw-r--r-- 1 ubuntu ubuntu    0 Feb 3 22:02 archivo2
lrwxrwxrwx 1 ubuntu ubuntu   11 Feb 3 22:02 link1 -> /etc/passwd
-rwxr-xr-x 1 ubuntu ubuntu    0 Feb 3 22:02 programa1
drwxr-xr-x 2 ubuntu ubuntu 4096 Feb 3 22:02 subdir1
```

# Sistema de Ficheros

***Ejercicio 10.*** Un dispositivo de memoria flash de 64 MB de capacidad y bloques de 1KB, contiene un sistema de ficheros FAT. Describa la estructura de la tabla y cómo se representa la asignación de bloques a un fichero. ¿Cuántos bytes son necesarios para almacenar la tabla FAT?

***Ejercicio 11.*** En la siguiente figura se representa una tabla FAT y el contenido de cierto directorio: que incluye: el nombre del archivo, el tipo (F=archivo, D=directorio) y el número del bloque inicial.

| | Bloque |
|---|---|
| 0 | |
| 1 | |
| 2 | |

| | Bloque |
|---|---|
| 10 | |
| 11 | |
| 12 | |

| Nombre | Tipo | Bloque |
|---|---|---|
| DATA.TXT | F | 3 |
| | | |
| | | |

| | |
|---|---|
| 3 | 15 |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |

| | |
|---|---|
| 13 | |
| 14 | |
| 15 | [EOF] |
| 16 | |
| 17 | |
| 18 | |
| 19 | |

| | | |
|---|---|---|
| | | |
| | | |

El tamaño de bloque en este sistema de ficheros es de 512 bytes y que el sistema operativo siempre asigna los bloques empezando por el primer bloque libre (número inferior) disponible. Completar el estado final de las tablas tras realizar (en orden) las siguientes operaciones:

1. Creación del fichero `DATA1.TXT` de tamaño 10 bytes.
2. Creación del fichero `DATA2.TXT` de tamaño 1200 bytes.
3. Se añaden datos al archivo `DATA.TXT` que requieren 2 bloques más.
4. Creación del directorio `LOGS`.
5. Creacion del fichero `RESULTS.JPG` de tamaño 2 Kbytes.

***Ejercicio 12.*** El sistema de ficheros ext2 usa una asignación de bloques indexada con las siguientes características:

- Los puntero de bloque ocupan 4 bytes
- El inodo guarda los siguientes punteros:
  - 12 punteros directos a bloque
  - 1 puntero a bloque indirecto.
  - 1 puntero a bloque indirecto doble.
  - 1 puntero a bloque indirecto triple.
- El tamaño de bloque es de 4KB (por defecto)

¿Cuál es el tamaño máximo de fichero si solo se usan los bloques directos? ¿Y usando además los bloques indirectos simples y dobles? ¿Cuál es el tamaño máximo del sistema de ficheros?

💻 ***Ejercicio 13.*** Crear un sistema de ficheros ext2 con la siguiente estructura de ficheros y directorios. **Nota:** los comandos mostrados a continuación se ejecutarán como root (cambiar a ejecutando `sudo -i`):

1. Crear un fichero de 100MB.

```
# truncate --size 100M ext2.img
```

2. Crear el sistema de ficheros ext2 en el fichero anterior. **Nota:** normalmente los sistemas de ficheros se crean en dispositivos en modo bloque. También es posible usar un fichero como soporte.

```
# mkfs.ext2 ext2.img
```

   ¿Qué tamaño de bloque se ha usado? ¿Cuántos inodos hay disponibles?

3. Montar el sistema de ficheros (la opción especial loop, permite montar el fichero accediendo mediante un dispositivo en modo bloque especial, *loopdevice* en `/dev/loop[0-9]`)

```
# mount -t ext2 -o loop ext2.img /mnt
```

4. En /mnt crear los siguientes contenidos:

```
# mkdir /mnt/dir
# echo "123456789" > /mnt/dir/small.txt
# dd if=/dev/random of=/mnt/dir/big bs=1024 count=5120
# dd if=/dev/random of=/mnt/dir/sparse bs=1024 count=1 seek=5192
```

Comprobar que se han creado correctamente los ficheros:

```
# ls -lhis /mnt/dir/
total 5.1M
15 5.1M -rw-r--r-- 1 root root 5.0M Feb 31 13:31 big
14 4.0K -rw-r--r-- 1 root root   10 Feb 31 13:29 small.txt
16  12K -rw-r--r-- 1 root root 5.1M Feb 31 14:15 sparse
```

5. Usando la herramienta debugfs estudiar la estructura de los inodos ext2 de los tres ficheros (comando stat). El fichero se referencia con el número de inodo en la forma <inodo>:

```
$ debugfs -R "stat <14>" /dev/sda1
Inode: 14   Type: regular    Mode:  0644   Flags: 0x0
Generation: 3900547475    Version: 0x00000000:00000001
User:     0   Group:     0   Project:     0   Size: 10
File ACL: 0
Links: 1   Blockcount: 8
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x68b44e22:7ba162a8 -- Sun Aug 31 13:29:06 2025
 atime: 0x68b44e22:7ba162a8 -- Sun Aug 31 13:29:06 2025
 mtime: 0x68b44e22:7ba162a8 -- Sun Aug 31 13:29:06 2025
crtime: 0x68b44e22:7ba162a8 -- Sun Aug 31 13:29:06 2025
Size of extra inode fields: 32
BLOCKS:
(0):24576
TOTAL: 1
```

La salida del comando `ls` muestra el número de inodo del archivo y entre paréntesis el tamaño que ocupa la entrada en la estructura del directorio (`rec_len`) (**Nota:** la última entrada consume todo el espacio restante de los 4096 bytes del bloque).

En este caso el tamaño es 10 bytes (correspondiente a los bytes de la cadena “123456789\n”) que se almacenan en el bloque 24576 que usa el primer puntero directo (0).

**Cuestiones**

- Estudiar la estructura de bloques del fichero big:

    - Bloques directos usados
    - Bloque dónde se guarda el puntero indirecto (`IND`). Y los bloques usados en el primer nivel de indirecciń (12-1035)
    - Bloque dónde se guarda el puntero doble indirecto (`DIND`). ¿Cuántos bloques indirectos (`IND`) hay en la tabla?. ¿Y los bloques usados para los bloques restantes (`1036-1279`)?
    - ¿Cuántos bloques consume en total el archivo? ¿Cuántos bloques se usan para guardar datos útiles?
    - Dibujar un esquema con las tablas usadas para almacenar el fichero.

- Considerar ahora el fichero `sparse` y repetir la cuestión anterior. Comparar los punteros inicializados en ambos casos.

***Ejercicio 14.*** Las entradas de directorio en un sistema de ficheros ext2 incluyen: el número de inodo (32 bits), la longitud del nombre y el nombre del archivo, que puede tener un máximo de 256 caracteres. ¿Cuántas entradas de directorio se pueden almacenar en un bloque de disco de 4 KB?

***Ejercicio 15.*** Considere un sistema de ficheros con los siguientes contenidos de inodos y bloques de disco.

| **inodo** | 2 |
|---|---|
| Enlaces | |
| Tipo | D |
| Bloque | |

| **inodo** | 3 |
|---|---|
| Enlaces | |
| Tipo | F |
| Bloque | 6 |

| **inodo** | 4 |
|---|---|
| Enlaces | |
| Tipo | F |
| Bloque | 12 |

| **inodo** | 5 |
|---|---|
| Enlaces | |
| Tipo | D |
| Bloque | 0 |

| **inodo** | 9 |
|---|---|
| Enlaces | |
| Tipo | D |
| Bloque | |

| **Bloque** | 0 |
|---|---|
| . | 5 |
| .. | 2 |
| C | 9 |
| D | 4 |

| **Bloque** | 3 |
|---|---|
| . | 2 |
| .. | |
| A | 3 |
| B | 5 |
| E | 4 |

| **Bloque** | 5 |
|---|---|
| . | 9 |
| .. | 5 |

1. Rellene los huecos para que el sistema sea consistente.

2. Dibuje de forma esquemática el árbol del directorio empleando.

***Ejercicio 16.*** Un usuario desea dar formato a una partición de disco para almacenar su colección de fotografías. Cada fotografía se almacena en un fichero con un tamaño fijo de 7000 bytes. El sistema de ficheros usado tiene las siguientes características:

- Bloques indexados con 2 enlaces directos y 1 indirecto simple por inodo.
- 4 bytes para identificar un inodo
- Tamaño de puntero a bloque de 32 bits.

El sistema de ficheros se puede formatear con tres tamaños de bloque: 1024 bytes, 2048 bytes y

4096 bytes. Analiza las ventajas de cada tamaño contestando a las siguientes preguntas:

- ¿Cuál de las tres opciones presenta menos fragmentación interna? Para cada tamaño de bloque, indique el porcentaje de ocupación real de cada uno de los bloques de datos asignados a un fichero.
- ¿Cuál de las tres opciones ofrece un mejor aprovechamiento de los bloques de disco disponibles? Para cada tamaño de bloque, indique qué porcentaje de bloques de disco se utilizarán para almacenar los datos de un determinado fichero.
- ¿Cuántos accesos a disco requerirá una lectura secuencial completa de una fotografía en cada caso?
- ¿Cómo afectarán a la selección de un tamaño de bloque determinado al tamaño de la tabla de inodos y al mapa de bits de bloques libres?

***Ejercicio 17.*** Un sistema de ficheros UNIX tiene las siguientes características:

- Bloques de 512 bytes y direcciones de bloque de disco de 16 bits.
- Bloques indexados con 10 punteros directos a bloque, 1 puntero indirecto simple y 1 puntero indirecto doble.

Conteste de manera razonada a las siguientes cuestiones:

- ¿Cuál es el tamaño máximo de un fichero en este sistema?
- Un programa UNIX crea un fichero en este sistema e inmediatamente después escribe un byte de datos en la posición 1.000 y otro en la posición 10.000. ¿Cuántos bloques de datos ocupa este nuevo fichero en disco?

***Ejercicio 18.*** Describe detalladamente qué operaciones relativas al sistema de ficheros realiza el sistema operativo al ejecutar leer los contenidos de `/home/ubuntu/.bashrc` en un sistema de ficheros tipo ext con tamaño de bloque de 4K. Suponer que el nodo-i del directorio raíz está ya en memoria y que el resto de las cachés del VFS están vacías.