

Práctica 2.2. Sistema de Ficheros

Objetivos

En esta práctica se revisan las funciones del sistema básicas para manejar un sistema de ficheros, referentes a la creación de ficheros y directorios, duplicación de descriptores, obtención de información de ficheros o el uso de cerrojos.

Contenidos

- Preparación del entorno para la práctica
- Creación y atributos de ficheros
- Redirecciones y duplicación de descriptores
- Cerrojos de ficheros
- Directorios

Preparación del entorno para la práctica

La realización de esta práctica únicamente requiere del entorno de desarrollo (compilador, editores y utilidades de depuración). Estas herramientas están disponibles en las máquinas virtuales de la asignatura y en la máquina física de los puestos del laboratorio.

Creación y atributos de ficheros

El inodo de un fichero guarda diferentes atributos de éste, como por ejemplo el propietario, permisos de acceso, tamaño o los tiempos de acceso, modificación y creación. En esta sección veremos las llamadas al sistema más importantes para consultar y fijar estos atributos así como las herramientas del sistema para su gestión.

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 y --color`. Estudiar el significado de la salida en cada caso.

man 1 ls

```
sansan@MSI: ~/ASOR/SO
LS(1) User Commands LS(1)
NAME
    ls - list directory contents
SYNOPSIS
    ls [OPTION]... [FILE]...
DESCRIPTION
    List information about the FILES (the current directory by default). Sort entries alphabetically if none of
    -cftuvSUX nor --sort is specified.

    Mandatory arguments to long options are mandatory for short options too.

    -a, --all
        do not ignore entries starting with .
    -A, --almost-all
        do not list implied . and ..
    --author
        with -l, print the author of each file
    -b, --escape
        print C-style escapes for nongraphic characters
    --block-size=SIZE
        with -l, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format below
    -B, --ignore-backups
        do not list implied entries ending with ~
    -c
        with -lt: sort by, and show, ctime (time of last modification of file status information); with -l:
        show ctime and sort by name; otherwise: sort by ctime, newest first
    -C
        list entries by columns
    --color[=WHEN]
        colorize the output; WHEN can be 'always' (default if omitted), 'auto', or 'never'; more info below
```

Ejercicio 2. El *modo* de un fichero es <tipo><rw_x_propietario><rw_x_grupo><rw_x_resto>:

- tipo: - fichero ordinario; d directorio; l enlace; c dispositivo carácter; b dispositivo bloque; p FIFO; s socket
- rw_x: r lectura (4); w escritura (2); x ejecución (1)

Comprobar los permisos de algunos directorios (con `ls -ld`).

```
sansan@MSI:~$ ls -lda
drwxr-xr-x 14 sansan sansan 4096 Nov 19 11:39 .
sansan@MSI:~$ cd ASOR
sansan@MSI:~/ASOR$ ls -ld
drwxr-xr-x 4 sansan sansan 4096 Nov 7 09:41 .
sansan@MSI:~/ASOR$ cd SO
sansan@MSI:~/ASOR/SO$ ls -ld
drwxr-xr-x 4 sansan sansan 4096 Nov 21 09:14 .
sansan@MSI:~/ASOR/SO$ cd P2.2
sansan@MSI:~/ASOR/SO/P2.2$ ls -ld
drwxr-xr-x 6 sansan sansan 4096 Nov 21 09:17 .
sansan@MSI:~/ASOR/SO/P2.2$
```

Ejercicio 3. Los permisos se pueden otorgar de forma selectiva usando la notación octal o la simbólica. Ejemplo, probar las siguientes órdenes (equivalentes):

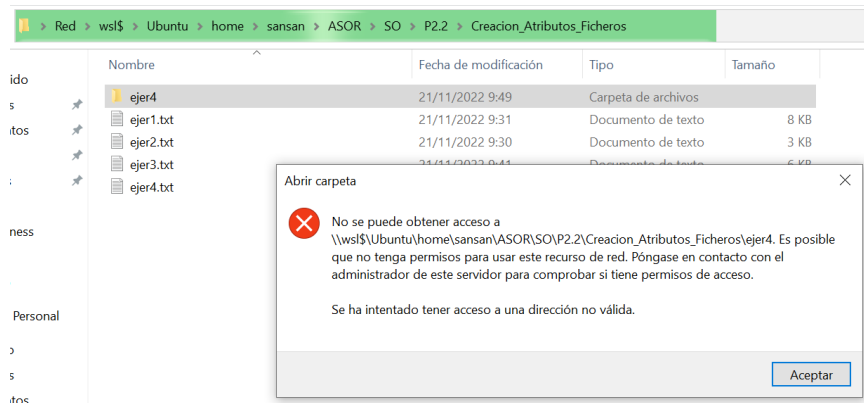
- `chmod 540 fichero`
- `chmod u=rx,g=r,o= fichero`

¿Cómo se podrían fijar los permisos `rw-r--r-x`, de las dos formas? Consultar la página de manual `chmod(1)` para ver otras formas de fijar los permisos (p.ej. los operadores `+` y `-`).

```
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ls -l
total 16
-rw-r--r-- 1 sansan sansan 7772 Nov 21 09:31 ejer1.txt
-rw-r--r-- 1 sansan sansan 2205 Nov 21 09:30 ejer2.txt
-rw-r--r-- 1 sansan sansan 527 Nov 21 09:32 ejer3.txt
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ chmod 540 ejer3.txt
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ls -l
total 16
-rw-r--r-- 1 sansan sansan 7772 Nov 21 09:31 ejer1.txt
-rw-r--r-- 1 sansan sansan 2205 Nov 21 09:30 ejer2.txt
-r-xr----- 1 sansan sansan 527 Nov 21 09:32 ejer3.txt
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ chmod u=rx,g=r,o= ejer3.txt
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ls -l
total 16
-rw-r--r-- 1 sansan sansan 7772 Nov 21 09:31 ejer1.txt
-rw-r--r-- 1 sansan sansan 2205 Nov 21 09:30 ejer2.txt
-r-xr----- 1 sansan sansan 527 Nov 21 09:32 ejer3.txt
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ chmod 645 ejer3.txt
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ls -l
total 16
-rw-r--r-- 1 sansan sansan 7772 Nov 21 09:31 ejer1.txt
-rw-r--r-- 1 sansan sansan 2205 Nov 21 09:30 ejer2.txt
-rw-r--r-x 1 sansan sansan 527 Nov 21 09:32 ejer3.txt
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ chmod u=rw,g=r,o=rx ejer3.txt
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ls -l
total 20
-rw-r--r-- 1 sansan sansan 7772 Nov 21 09:31 ejer1.txt
-rw-r--r-- 1 sansan sansan 2205 Nov 21 09:30 ejer2.txt
-rw-r--r-x 1 sansan sansan 4337 Nov 21 09:39 ejer3.txt
```

Ejercicio 4. Crear un directorio y quitar los permisos de ejecución para usuario, grupo y otros. Intentar cambiar al directorio.

```
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ mkdir ejer4
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ chmod 000 ejer4
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ls -l
total 28
-rw-r--r-- 1 sansan sansan 7772 Nov 21 09:31 ejer1.txt
-rw-r--r-- 1 sansan sansan 2205 Nov 21 09:30 ejer2.txt
-rw-r--r-- 1 sansan sansan 5257 Nov 21 09:41 ejer3.txt
d----- 2 sansan sansan 4096 Nov 21 09:49 ejer4
-rw-r--r-- 1 sansan sansan 1999 Nov 21 09:47 ejer4.txt
```



Ejercicio 5. Escribir un programa que, usando `open(2)`, cree un fichero con los permisos `rw-r--r-x`. Comprobar el resultado y las características del fichero con `ls(1)`.

```
● sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ gcc -o ejer5 ejer5.c
● sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ./ejer5
El descriptor del nuevo fichero es: 3
```

Ejercicio 6. Cuando se crea un fichero, los permisos por defecto se derivan de la máscara de usuario (`umask`). El comando interno de la *shell* `umask` permite consultar y fijar esta máscara. Usando este comando, fijar la máscara de forma que los nuevos ficheros no tengan permiso de escritura para el grupo y no tengan ningún permiso para otros. Comprobar el funcionamiento con `touch(1)`, `mkdir(1)` y `ls(1)`.

```
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ umask
0022
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ umask 027
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ umask
0027
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ mkdir Ejer6
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ touch ejer6b.txt
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ls
Ejer6 ejer1.txt ejer2.txt ejer3.txt ejer4.txt ejer5.c ejer6.txt ejer6b.txt ejer7.c
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ls -l
total 44
drwxr-x--- 2 sansan sansan 4096 Nov 21 11:08 Ejer6
-rw-r--r-- 1 sansan sansan 7772 Nov 21 09:31 ejer1.txt
-rw-r--r-- 1 sansan sansan 2205 Nov 21 09:30 ejer2.txt
-rw-r--r-- 1 sansan sansan 5257 Nov 21 09:41 ejer3.txt
-rw-r--r-- 1 sansan sansan 2173 Nov 21 09:54 ejer4.txt
-rw-r--r-- 1 sansan sansan 1071 Nov 21 10:24 ejer5.c
-rw-r--r-- 1 sansan sansan 5861 Nov 21 10:46 ejer6.txt
-rw-r----- 1 sansan sansan 0 Nov 21 11:08 ejer6b.txt
-rw-r--r-- 1 sansan sansan 1063 Nov 21 10:46 ejer7.c
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$
```

Ejercicio 7. Modificar el ejercicio 5 para que, antes de crear el fichero, se fije la máscara igual que en el ejercicio 6. Comprobar el resultado con `ls(1)`. Comprobar que la máscara del proceso padre (la `shell`) no cambia.

```
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ gcc ejer7.c -o ejer7
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ./ejer7
El descriptor del nuevo fichero es: 3
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ls -l
total 60
-rw-r--r-- 1 sansan sansan 7772 Nov 21 09:31 ejer1.txt
-rw-r--r-- 1 sansan sansan 2205 Nov 21 09:30 ejer2.txt
-rw-r--r-- 1 sansan sansan 5257 Nov 21 09:41 ejer3.txt
-rw-r--r-- 1 sansan sansan 2173 Nov 21 09:54 ejer4.txt
-rw-r--r-- 1 sansan sansan 1071 Nov 21 10:24 ejer5.c
-rw-r--r-- 1 sansan sansan 5861 Nov 21 11:17 ejer6.txt
-rwxr-xr-x 1 sansan sansan 16880 Nov 21 11:18 ejer7
-rw-r--r-- 1 sansan sansan 1379 Nov 21 11:18 ejer7.c
-rw-r----- 1 sansan sansan 0 Nov 21 11:18 ejer7.txt
```

Ejercicio 8. `ls(1)` puede mostrar el inodo con la opción `-li`. El resto de información del inodo puede obtenerse usando `stat(1)`. Consultar las opciones del comando y comprobar su funcionamiento.

```
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ls -li
total 48
186926 -rw-r--r-- 1 sansan sansan 7772 Nov 21 09:31 ejer1.txt
186931 -rw-r--r-- 1 sansan sansan 2205 Nov 21 09:30 ejer2.txt
186937 -rw-r--r-- 1 sansan sansan 5257 Nov 21 09:41 ejer3.txt
186951 -rw-r--r-- 1 sansan sansan 2173 Nov 21 09:54 ejer4.txt
186956 -rw-r--r-- 1 sansan sansan 1071 Nov 21 10:24 ejer5.c
159675 -rw-r--r-- 1 sansan sansan 5861 Nov 21 11:17 ejer6.txt
186978 -rw-r--r-- 1 sansan sansan 1378 Nov 21 11:21 ejer7.c
159677 -rw-r--r-- 1 sansan sansan 4237 Nov 21 11:23 ejer8.txt
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ stat -c %i ejer8.txt
159677
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ stat ejer8.txt
  File: ejer8.txt
  Size: 4237          Blocks: 16          IO Block: 4096   regular file
Device: 810h/2064d   Inode: 159677       Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/  sansan)   Gid: ( 1000/  sansan)
Access: 2022-11-21 11:23:24.443887242 +0100
Modify: 2022-11-21 11:23:24.433887242 +0100
Change: 2022-11-21 11:23:24.433887242 +0100
 Birth: -
```

Ejercicio 9. Escribir un programa que emule el comportamiento de `stat(1)` y muestre:

- El número *major* y *minor* asociado al dispositivo.
- El número de inodo del fichero.
- El tipo de fichero (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`?

```
• sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ./ejer9
El descriptor del nuevo fichero es: 3
El Major asociado al dispositivo es: 8
El Minor asociado al dispositivo es: 16
El Inodo del fichero es: 163896
El tipo de archivo es: Fichero Regular
Último acceso al archivo: Mon Nov 21 17:07:06 2022
Última modificación en el archivo: Mon Nov 21 17:07:06 2022
Última modificación del status del archivo: Mon Nov 21 17:07:06 2022
```

Ejercicio 10. Los enlaces se crean con `ln(1)`:

- Con la opción `-s`, se crea un enlace simbólico. Crear un enlace simbólico a un fichero ordinario y otro a un directorio. Comprobar el resultado con `ls -l` y `ls -li`. Determinar el inodo de cada fichero.
- Repetir el apartado anterior con enlaces rígidos. Determinar los inodos de los ficheros y las propiedades con `stat` (observar el atributo número de enlaces).
- ¿Qué sucede cuando se borra uno de los enlaces rígidos? ¿Qué sucede si se borra uno de los enlaces simbólicos? ¿Y si se borra el fichero original?

```
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ stat -L namelink
File: namelink
Size: 0                Blocks: 0                IO Block: 4096   regular empty file
Device: 810h/2064d    Inode: 173477           Links: 2
Access: (0644/-rw-r--r--)  Uid: ( 1000/   sansan)   Gid: ( 1000/   sansan)
Access: 2022-11-28 09:11:58.424261688 +0100
Modify: 2022-11-28 09:11:58.424261688 +0100
Change: 2022-11-28 09:17:09.184257116 +0100
Birth: -
```

```
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ls -li
total 60
drwxr-xr-x 2 sansan sansan 4096 Nov 28 09:12 ej10
-rw-r--r-- 1 sansan sansan 7772 Nov 21 09:31 ejer1.txt
-rw-r--r-- 1 sansan sansan 732 Nov 21 17:13 ejer10.txt
-rw-r--r-- 1 sansan sansan 2205 Nov 21 09:30 ejer2.txt
-rw-r--r-- 1 sansan sansan 5257 Nov 21 09:41 ejer3.txt
-rw-r--r-- 1 sansan sansan 2173 Nov 21 09:54 ejer4.txt
-rw-r--r-- 1 sansan sansan 1071 Nov 21 10:24 ejer5.c
-rw-r--r-- 1 sansan sansan 5861 Nov 21 11:17 ejer6.txt
-rw-r--r-- 1 sansan sansan 1378 Nov 21 11:21 ejer7.c
-rw-r--r-- 1 sansan sansan 5731 Nov 21 11:35 ejer8.txt
-rw-r--r-- 1 sansan sansan 2747 Nov 21 17:12 ejer9.c
lrwxrwxrwx 1 sansan sansan 4 Nov 28 09:15 link_name -> ej10
lrwxrwxrwx 1 sansan sansan 8 Nov 28 09:15 name_link -> ej10.txt
-rw-r--r-- 1 sansan sansan 0 Nov 28 09:11 namelink
```

Ejercicio 11. `link(2)` y `symlink(2)` crean enlaces rígidos y simbólicos, respectivamente. Escribir un programa que reciba una ruta a un fichero como argumento. Si la ruta es un fichero regular, creará un enlace simbólico y rígido con el mismo nombre terminado en `.sym` y `.hard`, respectivamente. Comprobar el resultado con `ls(1)`.

```
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ gcc ejer11.c -o ejer11
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ ./ejer11 ejemplo.txt
Enlace duro: ejemplo.txt
Enlace simbolico: ejemplo.txt
Es un fichero regular
Se ha creado enlace duro.
Se ha creado enlace simbólico.
```

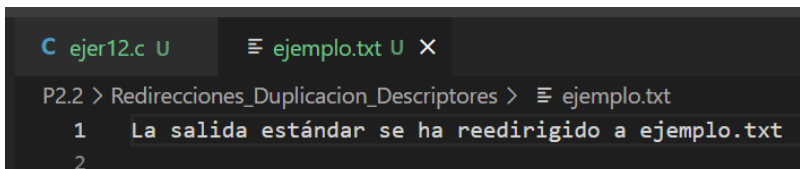
```
▼ Creacion_Atributos_Ficheros ●
  ≡ ejemplo.txt.hard      U
  ≡ ejemplo.txt.sym      U, ↵
```

Redirecciones y duplicación de descriptores

La *shell* proporciona operadores (>, >&, >>) que permiten redirigir un fichero a otro, ver los ejercicios propuestos en la práctica opcional. Esta funcionalidad se implementa mediante `dup(2)` y `dup2(2)`.

Ejercicio 12. Escribir un programa que redirija la salida estándar a un fichero cuya ruta se pasa como primer argumento. Probar haciendo que el programa escriba varias cadenas en la salida estándar.

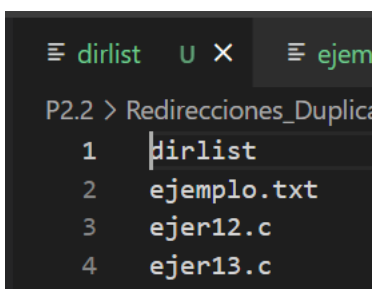
```
sansan@MSI:~/ASOR/SO/P2.2/Redirecciones_Duplicacion_Descriptores$ gcc ejer12.c -o ejer12
sansan@MSI:~/ASOR/SO/P2.2/Redirecciones_Duplicacion_Descriptores$ ./ejer12 ejemplo.txt
```



```
C ejer12.c U  ejemplo.txt U X
P2.2 > Redirecciones_Duplicacion_Descriptores > ejemplo.txt
1  La salida estándar se ha reedirigido a ejemplo.txt
2
```

Ejercicio 13. Modificar el programa anterior para que también redirija la salida estándar de error al fichero. Comprobar el funcionamiento incluyendo varias sentencias que impriman en ambos flujos. ¿Hay diferencia si las redirecciones se hacen en diferente orden? ¿Por qué `ls > dirlist 2>&1` es diferente a `ls 2>&1 > dirlist`?

```
sansan@MSI:~/ASOR/SO/P2.2/Redirecciones_Duplicacion_Descriptores$ ./ejer13 ejemplo.txt
sansan@MSI:~/ASOR/SO/P2.2/Redirecciones_Duplicacion_Descriptores$ ls > dirlist 2>&1
sansan@MSI:~/ASOR/SO/P2.2/Redirecciones_Duplicacion_Descriptores$ ls 2>&1
dirlist ejemplo.txt ejer12.c ejer13 ejer13.c
```



```
dirlist U X  ejemplo.txt U X
P2.2 > Redirecciones_Duplicacion_Descriptores
1  dirlist
2  ejemplo.txt
3  ejer12.c
4  ejer13.c
```

Cerrojos de ficheros

El sistema de ficheros ofrece cerrojos de ficheros consultivos.

Ejercicio 14. El estado y cerrojos de fichero en uso en el sistema se pueden consultar en el fichero `/proc/locks`. Estudiar el contenido de este fichero.

```
sansan@MSI:~/ASOR/SO/P2.2/Creacion_Atributos_Ficheros$ cd
sansan@MSI:~$ cat /proc/locks
1: POSIX ADVISORY READ 12584 08:10:173469 128 128
sansan@MSI:~$ |
```

Ejercicio 15. Escribir un programa que intente bloquear un fichero usando `lockf(3)`:

- Si lo consigue, mostrará la hora actual y suspenderá su ejecución durante 10 segundos con `sleep(3)`. A continuación, desbloqueará el fichero, suspenderá su ejecución durante otros 10 segundos y terminará.
- Si no lo consigue, el programa mostrará el error con `perror(3)` y terminará.

```
sansan@MSI:~/ASOR/SO/P2.2/Cerrojos_de_Ficheros$ ./ejer15 ejemplo
Hoy estamos a Monday, 28 de November de 2022, 11:31
sansan@MSI:~/ASOR/SO/P2.2/Cerrojos_de_Ficheros$
```

Ejercicio 16 (Opcional). `flock(1)` proporciona funcionalidad de cerrojos antiguos BSD en guiones *shell*. Consultar la página de manual y el funcionamiento del comando.

Directorios

Ejercicio 17. Escribir un programa que muestre el contenido de un directorio:

- El programa tiene un único argumento que es la ruta a un directorio. El programa debe comprobar la corrección del argumento.
- El programa 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 después del nombre. Usar `readlink(2)`.
- Al final de la lista, el programa escribirá el tamaño total que ocupan los ficheros (no directorios) en kilobytes.

```
sansan@MSI:~/ASOR/S0/P2.2/Directorios$ ./ejer17 ejemplo
Directory
../
Directory
./
El total de KB de los ficheros es: 0.000000
sansan@MSI:~/ASOR/S0/P2.2/Directorios$
```