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
Proyecto: Comando `ls` extendido

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.

En la realización de las prácticas se puede usar cualquier editor gráfico o de terminal. Además se puede usar tanto el lenguaje C (compilador `gcc`) como C++ (compilador `g++`). Si fuera necesario compilar varios ficheros se recomienda el uso de alguna herramienta para la compilación de proyectos como `make`. Finalmente, el depurador recomendado en las prácticas es `gdb`. **No está permitido** el uso de IDEs como Eclipse.

Creación y atributos de ficheros

El i-nodo 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. La herramienta principal para consultar el contenido y atributos básicos de un fichero es `ls`. 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.

-a -> No ignoran las entradas que empiecen por .
-l -> Muestra con formato de lista larga
-d -> Lista el directorio en el que está, no su contenido
-h -> Junto con -l o -s pone el tamaño de los archivos (ya no en bytes)
-i -> Pone el nº de i-nodo de los ficheros
-R -> Lista subdirectorios recursivamente
-1 -> Lista 1 archivo por línea (mete un '\n')
-F -> Añade al final un indicador a las entradas (*/>\@|)
--color -> Colorear la salida (se le puede poner siempre, nunca etc..)

Ejercicio 2. El *modo* de un fichero es `<tipo><rwx_propietario><rwx_grupo><rwx_resto>`:

- `tipo`: `-` fichero ordinario; `d` directorio; `l` enlace; `c` dispositivo carácter; `b` dispositivo bloque; `p` FIFO; `s` socket

- `rwx`: `r` lectura (4); `w` escritura (2); `x` ejecución (1)

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

En el directorio usuarioso:
Comando: `ls -ld`
`Salida:`
drwxr-xr-x 20 usuarioso usuarioso 4096 dic 13 17:43 .

Es un directorio, donde:

El propietario (usuarioso) tiene permisos de escritura, lectura y ejecución
El grupo (usuarioso) tiene permisos de ejecución y lectura pero no escritura
Los demás tienen permisos de ejecución y lectura, pero no de escritura

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-wx,o-wxr fichero`

¿Cómo se podrían fijar los permisos `rw-r--r-x`, de las dos formas?

chmod 540 fichero
ls -l
total 52
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Descargas
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Documentos
drwxr-xr-x 3 usuarioso usuarioso 4096 jul 13 2018 eclipse
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Escritorio
-rw-r--r-- 1 usuarioso usuarioso 8980 jul 10 2018 examples.desktop
-r-xr----- 1 usuarioso usuarioso 0 dic 13 18:01 fichero
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Imágenes
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Música
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Plantillas
-rw-rw-r-- 1 usuarioso usuarioso 0 dic 13 17:43 pr2.c
drwxrwxr-x 2 usuarioso usuarioso 4096 nov 29 13:46 Practica1ASOR
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Público
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Vídeos

Borramos el fichero, lo creamos nuevo y hacemos:
chmod u+rx,g+r-wx,o-wxr fichero
ls -l
total 52
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Descargas
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Documentos
drwxr-xr-x 3 usuarioso usuarioso 4096 jul 13 2018 eclipse
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Escritorio
-rw-r--r-- 1 usuarioso usuarioso 8980 jul 10 2018 examples.desktop
-rwxr----- 1 usuarioso usuarioso 0 dic 13 18:04 fichero
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Imágenes
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Música
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Plantillas

```
-rw-rw-r-- 1 usuarioso usuarioso    0 dic 13 17:43 pr2.c
drwxrwxr-x 2 usuarioso usuarioso 4096 nov 29 13:46 Practica1ASOR
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Público
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Vídeos
```

Con la primera forma:

Comando: chmod 645 fichero
ls -l
Salida:

```
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Descargas
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Documentos
drwxr-xr-x 3 usuarioso usuarioso 4096 jul 13  2018 eclipse
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Escritorio
-rw-r--r-- 1 usuarioso usuarioso 8980 jul 10  2018 examples.desktop
-rw-r--r-x 1 usuarioso usuarioso    0 dic 13 18:04 fichero
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Imágenes
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Música
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Plantillas
-rw-rw-r-- 1 usuarioso usuarioso    0 dic 13 17:43 pr2.c
drwxrwxr-x 2 usuarioso usuarioso 4096 nov 29 13:46 Practica1ASOR
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Público
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Vídeos
```

Con la segunda forma:
Comando: chmod u+rw-x,g+r-xw,o+rx-w fichero

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

Comando: chmod u-x,g-x,o-x directorio

Al intentar acceder al directorio:
cd directorio
bash: cd: directorio: Permiso denegado

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

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv ){
        if(argc == 1){
                printf("Error, faltan argumentos\n");
                exit(1);
```

```
        }

        int fd= open(argv[1], O_CREAT, 0645);
        if(fd == -1){
                perror("Open");
                exit(1);
        }
        return 0;
}
```

Ejecución: ./pr2 ficheroPrueba
Comando: ls -l
Salida:

```
total 68
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Descargas
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Documentos
drwxr-xr-x 3 usuarioso usuarioso 4096 jul 13  2018 eclipse
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Escritorio
-rw-r--r-- 1 usuarioso usuarioso 8980 jul 10  2018 examples.desktop
-rw-r--r-x 1 usuarioso usuarioso    0 dic 13 18:04 fichero
-rw-r--r-x 1 usuarioso usuarioso    0 dic 13 18:22 ficheroPrueba
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Imágenes
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Música
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Plantillas
-rwxrwxr-x 1 usuarioso usuarioso 8376 dic 13 18:22 pr2
-rw-rw-r-- 1 usuarioso usuarioso  222 dic 13 18:22 pr2.c
drwxrwxr-x 2 usuarioso usuarioso 4096 nov 29 13:46 Practica1ASOR
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Público
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Vídeos
```

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 ningún permiso para otros. Comprobar el funcionamiento con los comandos `touch`, `mkdir` y `ls`.

Comando: umask 027
umask
0027

touch fichero2
ls -l

```
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Descargas
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Documentos
drwxr-xr-x 3 usuarioso usuarioso 4096 jul 13  2018 eclipse
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Escritorio
-rw-r--r-- 1 usuarioso usuarioso 8980 jul 10  2018 examples.desktop
-rw-r--r-x 1 usuarioso usuarioso    0 dic 13 18:04 fichero
-rw-r----- 1 usuarioso usuarioso    0 dic 13 18:32 fichero2
-rw-r--r-x 1 usuarioso usuarioso    0 dic 13 18:30 ficheroPrueba
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Imágenes
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Música
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Plantillas
-rwxrwxr-x 1 usuarioso usuarioso 8376 dic 13 18:22 pr2
-rw-rw-r-- 1 usuarioso usuarioso  222 dic 13 18:22 pr2.c
```

```
drwxrwxr-x 2 usuarioso usuarioso 4096 nov 29 13:46 Practica1ASOR
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Público
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Vídeos

El nuevo fichero se crea sin los permisos que hemos impedido con umask
```

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 el comando `ls`. Comprobar que la máscara del proceso padre (la *shell*) no cambia.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>

int main(){
        umask(027);
        int fd= open("ficheroPruebaEjer7", O_CREAT, 0645);
        if(fd == -1){
                perror("Open");
                exit(1);
        }
        close(fd);
        return 0;
}

Hacemos ls -l:
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Descargas
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Documentos
drwxr-xr-x 3 usuarioso usuarioso 4096 jul 13  2018 eclipse
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Escritorio
-rw-r--r-- 1 usuarioso usuarioso 8980 jul 10  2018 examples.desktop
-rw-r--r-x 1 usuarioso usuarioso    0 dic 13 18:04 fichero
-rw-r----- 1 usuarioso usuarioso    0 dic 13 18:32 fichero2
-rw-r----- 1 usuarioso usuarioso    0 dic 13 18:37 ficheroPruebaEjer7
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Imágenes
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Música
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Plantillas
-rwxr-xr-x 1 usuarioso usuarioso 8416 dic 13 18:37 pr2
-rw-rw-r-- 1 usuarioso usuarioso  240 dic 13 18:36 pr2.c
drwxrwxr-x 2 usuarioso usuarioso 4096 nov 29 13:46 Practica1ASOR
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Público
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10  2018 Vídeos

Comprobamos que la máscara del proceso padre (shell) no ha cambiado.
umask
0022
```

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

Ejemplo de uso del comando stat que nos da la información del fichero:

stat fichero

```
  Fichero: fichero
  Tamaño: 0                Bloques: 0          Bloque E/S: 4096   fichero regular vacío
Dispositivo: 801h/2049d    Nodo-i: 264727      Enlaces: 1
Acceso: (0645/-rw-r--r-x)  Uid: ( 1001/usuarioso)   Gid: ( 1001/usuarioso)
Acceso: 2020-12-13 18:04:09.338833161 +0100
Modificación: 2020-12-13 18:04:09.338833161 +0100
     Cambio: 2020-12-13 18:09:37.490827161 +0100
    Creación: -
```

Ejercicio 9. Escribir un programa que emule el comportamiento del comando `stat` y muestre:

- El número *major* y *minor* asociado al dispositivo.
- El número de i-nodo 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`?

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/sysmacros.h>
#include <time.h>

int main(int argc, char** argv){
	struct stat statbuf;
	if(argc == 1){
		printf("Error, faltan argumentos\n");
		exit(1);
	}
	int st = stat(argv[1], &statbuf);
	if(st == -1){
		perror("Stat");
		exit(1);
	}
	printf("Major: %i\n", major(statbuf.st_dev));
	printf("Minor: %i\n", minor(statbuf.st_dev));
	printf("Nº de i-nodo: %li\n", statbuf.st_ino);
	if(S_ISREG(statbuf.st_mode)){
		printf("Es un fichero regular\n");
	}
	else if(S_ISDIR(statbuf.st_mode)){
		printf("Es un directorio\n");
	}
	else if(S_ISLNK(statbuf.st_mode)){
		printf("Es un enlace simbólico\n");
	}
	struct tm * time = localtime(&statbuf.st_atime);
	printf("Ultimo acceso al fichero: %i:%i \n", time->tm_hour, time->tm_min);
```

```
        return 0;
}
```

Llamada: ./pr2 fichero
Salida:
Major: 8
Minor: 1
Nº de i-nodo: 264727
Es un fichero regular
Ultimo acceso al fichero: 18:4

`st_mtime da fecha de la última modificación del fichero donde entra por ejemplo una escritura, un truncate etc...`

`st_ctime da la fecha del último cambio en la ifnormación del i-nodo (propietario, grupo, permisos etc..)`

Ejercicio 10. Los enlaces se crean con la orden `ln`:

- La opción `-s` 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 -i`. Determinar el i-nodo de cada fichero.
- Repetir el apartado anterior con enlaces rígidos. Determinar los i-nodos 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?

Comandos:

ln -s fichero ficherolink
ln -s descargas descargaslink

ls -l

rwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Descargas
lrwxrwxrwx 1 usuarioso usuarioso 9 dic 13 19:31 descargaslink -> descargas
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Documentos
drwxr-xr-x 3 usuarioso usuarioso 4096 jul 13 2018 eclipse
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Escritorio
-rw-r--r-- 1 usuarioso usuarioso 8980 jul 10 2018 examples.desktop
-rw-r--r-x 1 usuarioso usuarioso 0 dic 13 18:04 fichero
-rw-r----- 1 usuarioso usuarioso 0 dic 13 18:32 fichero2
lrwxrwxrwx 1 usuarioso usuarioso 7 dic 13 19:31 ficherolink -> fichero
-rw-r----- 1 usuarioso usuarioso 0 dic 13 18:37 ficheroPruebaEjer7
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Imágenes
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Música
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Plantillas
-rwxr-xr-x 1 usuarioso usuarioso 8720 dic 13 19:21 pr2
-rw-rw-r-- 1 usuarioso usuarioso 858 dic 13 19:21 pr2.c
drwxrwxr-x 2 usuarioso usuarioso 4096 nov 29 13:46 Practica1ASOR
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Público
drwxr-xr-x 2 usuarioso usuarioso 4096 jul 10 2018 Vídeos

Con ls -i descubrimos que tienen los i-nodos:

358340 descargaslink
268993 Descargas

264727 fichero
358338 ficherolink

ln fichero ficherorigido
(Para directorios no se permiten enlaces rígidos)

ls -i
264727 fichero
264727 ficherorigido

stat fichero
Fichero: fichero
  Tamaño: 0 Bloques: 0 Bloque E/S: 4096 fichero regular vacío
Dispositivo: 801h/2049d Nodo-i: 264727 Enlaces: 2
Acceso: (0645/-rw-r--r-x) Uid: (1001/usuarioso) Gid: (1001/usuarioso)
Acceso: 2020-12-13 18:04:09.338833161 +0100
Modificación: 2020-12-13 18:04:09.338833161 +0100
  Cambio: 2020-12-13 19:34:20.896551103 +0100
  Creación: -

(Tiene 2 enlaces)

Si se borra un enlace rígido sólo desciende el contador de enlaces, si se borra el simbólico, desaparece este fichero (su i-nodo) pero sobre el original no pasa nada y si se borra el original siempre que haya algún rígido más no pasa nada y si no se borra el i-nodo.

Ejercicio 11. Las llamadas `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 la orden `ls`.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/sysmacros.h>
#include <time.h>
#include <string.h>

int main(int argc, char** argv){
	if(argc == 1){
		printf("Error, faltan argumentos\n");
		exit(1);
	}
	struct stat statbuf;
```

```
	int st = stat(argv[1], &statbuf);
	if(st == -1){
		perror("stat");
		exit(1);
	}
	if(!S_ISREG(statbuf.st_mode)){
		printf("Error, no es fichero regular\n");
		exit(1);
	}

	char * cadena1 = malloc(strlen(argv[1]) + 4);
	char * cadena2 = malloc(strlen(argv[1]) + 5);

	strcpy(cadena1, argv[1]);
	strcpy(cadena2, argv[1]);

	int symlnk = symlink(argv[1], strcat(cadena1, ".sym"));
	if(symlnk == -1){
		perror ("symlink");
		exit(1);
	}
	int lnk = link(argv[1], strcat(cadena2, ".hard"));
	if(lnk == -1){
		perror ("link");
		exit(1);
	}
	return 0;
}

./pr2 ejemploEjer11

ls -l
total 72
drwxr-xr-x 2 usuarioso usuarioso  4096 jul 10  2018 Descargas
drwxr-xr-x 2 usuarioso usuarioso  4096 jul 10  2018 Documentos
drwxr-xr-x 3 usuarioso usuarioso  4096 jul 13  2018 eclipse
-rw-r--r-- 3 usuarioso usuarioso     0 dic 13 19:52 ejemploEjer11
-rw-r--r-- 3 usuarioso usuarioso     0 dic 13 19:52 ejemploEjer11.hard
lrwxrwxrwx 1 usuarioso usuarioso    13 dic 13 20:02 ejemploEjer11.sym -> ejemploEjer11
drwxr-xr-x 2 usuarioso usuarioso  4096 jul 10  2018 Escritorio
-rw-r--r-- 1 usuarioso usuarioso  8980 jul 10  2018 examples.desktop
-rw-r--r-x 2 usuarioso usuarioso     0 dic 13 18:04 fichero
-rw-r----- 1 usuarioso usuarioso     0 dic 13 18:32 fichero2
lrwxrwxrwx 1 usuarioso usuarioso     7 dic 13 19:31 ficherolink -> fichero
-rw-r----- 1 usuarioso usuarioso     0 dic 13 18:37 ficheroPruebaEjer7
lrwxrwxrwx 1 usuarioso usuarioso    22 dic 13 19:50 ficheroPruebaEjer7.sym ->
ficheroPruebaEjer7.sym
-rw-r--r-x 2 usuarioso usuarioso     0 dic 13 18:04 ficherorigido
drwxr-xr-x 2 usuarioso usuarioso  4096 jul 10  2018 Imágenes
drwxr-xr-x 2 usuarioso usuarioso  4096 jul 10  2018 Música
drwxr-xr-x 2 usuarioso usuarioso  4096 jul 10  2018 Plantillas
-rwxr-xr-x 1 usuarioso usuarioso 12824 dic 13 20:02 pr2
-rw-rw-r-- 1 usuarioso usuarioso   843 dic 13 20:02 pr2.c
drwxrwxr-x 2 usuarioso usuarioso  4096 nov 29 13:46 Practica1ASOR
drwxr-xr-x 2 usuarioso usuarioso  4096 jul 10  2018 Público
drwxr-xr-x 2 usuarioso usuarioso  4096 jul 10  2018 Vídeos
```

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 las llamadas `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.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/sysmacros.h>
#include <time.h>
#include <string.h>

int main(int argc, char** argv){
	if(argc < 2){
		printf("Error, faltan argumentos\n");
		exit(1);
	}

	int fdold = open(argv[1], O_CREAT | O_RDWR, 0666);
	if(fdold == -1){
		perror("open");
		exit(1);
	}
	dup2(fdold, 1);
	printf("Probando redirección\n");
	return 0;
}

En el fichero redirección está la frase:
Probando redirección
```

Ejercicio 13. Modificar el programa anterior para que además de escribir en el fichero la salida estándar también se escriba la salida estándar de error. Comprobar el funcionamiento incluyendo varias sentencias que impriman en ambos flujos. ¿Hay alguna diferencia si las redirecciones se hacen en diferente orden? ¿Por qué no es lo mismo "`ls > dirlist 2>&1`" que "`ls 2>&1 > dirlist`"?

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/sysmacros.h>
#include <time.h>
```

```
#include <string.h>

int main(int argc, char** argv){
	if(argc < 2){
		printf("Error, faltan argumentos\n");
		exit(1);
	}

	int fdold = open(argv[1], O_CREAT | O_RDWR, 0666);
	if(fdold == -1){
		perror("open");
		exit(1);
	}
	dup2(fdold, 1);
	dup2(fdold, 2);
	printf("Probando redirección\n");
	fprintf(stderr, "Redireccionando error\n");
	return 0;
}
```

```
ls > dirlist 2>&1
```

Redirecciona la salida estándar a un fichero que se llama dirlist. Luego redirecciona la salida de error al fichero con descriptor de fichero 1, es decir a dirlist. Ambas salida acaban en dirlist.

```
ls 2>&1 > dirlist
```

Redirecciona la salida de error a la estándar. Redirecciona la estándar al fichero dirlist Con lo cual al escribir por la salida de error acaba en la estándar y en el fichero.

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.

```
1: FLOCK  ADVISORY  WRITE 1941 00:31:29 0 EOF
2: POSIX  ADVISORY  WRITE 1546 08:01:358271 0 EOF
3: POSIX  ADVISORY  WRITE 1541 08:01:358269 0 EOF
4: POSIX  ADVISORY  WRITE 1536 08:01:358268 0 EOF
5: POSIX  ADVISORY  WRITE 1526 08:01:358264 0 EOF
6: FLOCK  ADVISORY  WRITE 1098 00:18:6 0 EOF
7: POSIX  ADVISORY  WRITE 1020 00:16:628 0 EOF
8: FLOCK  ADVISORY  WRITE 901 00:16:600 0 EOF
9: POSIX  ADVISORY  WRITE 898 08:01:393265 0 EOF

FLOCK -> Bloqueo hecho con flock
```

POSIX -> Bloqueo hecho con fcntl

Advisory -> El acceso a los datos está permitido y evita otros bloqueos

Write -> El proceso está escribiendo y por ello no se puede leer ni escribir en el área bloqueada.

Luego viene el pid del proceso que está bloqueando el fichero

Luego número que identifica el proceso

Y finalmente los bytes de inicio y fin del bloqueo

Ejercicio 15. Escribir un programa que consulte y muestre en pantalla el estado del cerrojo sobre un fichero. El programa mostrará el estado del cerrojo (bloqueado o desbloqueado). Además:

- Si está desbloqueado, fijará un cerrojo y escribirá la hora actual. Después suspenderá su ejecución durante 30 segundos (con `sleep(3)`) y a continuación liberará el cerrojo.
- Si está bloqueado, terminará el programa.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
 #include <time.h>

int main(int argc, char** argv){
	if(argc < 2){
		printf("Error, faltan argumentos\n");
		exit(1);
	}
	int fd = open(argv[1], O_RDWR);
	if(fd == -1){
		perror("open");
		exit(1);
	}
	int tam = lseek(fd, 0, SEEK_END);
	lseek(fd, 0, SEEK_SET);
	int bloqueado = lockf(fd, F_TEST, tam);
	if(bloqueado == 0){
		printf("Fichero no bloqueado\n");
		printf("Bloqueamos...\n");
		lockf(fd, F_LOCK, tam);
		time_t tiempo = time(NULL);
		struct tm* fecha = localtime(&tiempo);
		printf("Son las: %i:%i\n", fecha->tm_hour, fecha->tm_min);
		sleep(30);
		lockf(fd, F_ULOCK, tam);
		printf("Desbloqueamos...\n");
	}
	else {
		printf("Fichero bloqueado\n");
```

```
        }
        close(fd);
        return 0;
}


./pr2 redireccion

Salida:
Fichero no bloqueado
Bloqueamos...
Son las: 11:56
Desbloqueamos…

Si desde otra consola ejecutamos el mismo programa antes de que este acabe:
Fichero bloqueado
```

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

Proyecto: Comando `ls` extendido

Escribir un programa que cumpla las siguientes especificaciones:

- 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 de forma que:
    - Si es un fichero normal, escribirá el nombre.
    - Si es un directorio, escribirá el nombre seguido del carácter `'/'`.
    - Si es un enlace simbólico, escribirá su nombre seguido de `'->'` y el nombre del fichero enlazado. Usar `readlink(2)` y dimensionar adecuadamente el buffer.
    - Si el fichero es ejecutable, escribirá el nombre seguido del carácter `'*'`.
- Al final de la lista el programa escribirá el tamaño total que ocupan los ficheros (no directorios) en kilobytes.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <dirent.h>
#include <string.h>

int main(int argc, char** argv){
        if(argc < 2){
                printf("Error, faltan argumentos\n");
                exit(1);
        }
        DIR * dirp = opendir(argv[1]);
```

```
    if(dirp == NULL){
            perror("opendir");
            exit(1);
    }
    struct dirent * dire = readdir(dirp);
    long int tam = 0;
    while(dire != NULL){
            char* copiaDir;
            strcpy(copiaDir, argv[1]);
            char* ruta = strcat(copiaDir, "/");
            ruta= strcat(copiaDir, dire->d_name);


            if(dire->d_type == DT_REG){
                    struct stat sb;
                    if (lstat(ruta, &sb) == -1) {
                       perror("lstat");
                       exit(1);
                    }
                    if(sb.st_mode & (S_IXUSR | S_IXGRP | S_IXOTH)){
                            printf("%s*\n", dire->d_name);
                    }
                    else printf("%s\n", dire->d_name);


                    tam += sb.st_size;
            }
            else if(dire->d_type == DT_DIR){
                    printf("%s/\n", dire->d_name);
            }
            else if(dire->d_type == DT_LNK){
                    struct stat sb;
                    char *buf;
 ssize_t nbytes, bufsiz;
                    if (lstat(ruta, &sb) == -1) {
                       perror("lstat");
                       exit(1);
                    }
                    tam += sb.st_size;

                    bufsiz = sb.st_size + 1;

    if (sb.st_size == 0)
       bufsiz = PATH_MAX;

    buf = malloc(bufsiz);
    if (buf == NULL) {
       perror("malloc");
       exit(EXIT_FAILURE);
    }
    nbytes = readlink(ruta, buf, bufsiz);
    if (nbytes == -1) {
       perror("readlink");
       exit(EXIT_FAILURE);
    }

    printf("%s -> %s\n", dire->d_name, buf);
```

```
        free(buf);
                }
                dire = readdir(dirp);
        }
        closedir(dirp);
        printf("\nTamaño total en bytes: %li\n", tam);
        return 0;
}
```

Al ejecutarlo sobre el directorio actual:

```
./pr2 .
fichero*
Descargas/
.bashrc
.ssh/
../
.vboxclient-display.pid
redireccion
ejemploEjer11
Imágenes/
.vboxclient-draganddrop.pid
.eclipse/
.p2/
ficheroPruebaEjer7.sym -> ficheroPruebaEjer7.sym
Escritorio/
.bash_history
.gnupg/
.config/
./
ficheroPruebaEjer7
.swt/
.cache/
Plantillas/
ficherorigido*
.profile
examples.desktop
.bash_logout
.sudo_as_admin_successful
eclipse/
pr2.c
.Xauthority
.dmrc
Vídeos/
Practica1ASOR/
fichero2
.xsession-errors
pr2*
Documentos/
.ICEauthority
ejemploEjer11.hard
.vboxclient-clipboard.pid
.vboxclient-seamless.pid
Público/
.pam_environment
.xsession-errors.old
ejemploEjer11.sym -> ejemploEjer11Ejer7.sym
```

```
ficherolink -> fichero
.lesshst
.local/
Música/
.viminfo

Tamaño total en bytes: 57225
```