

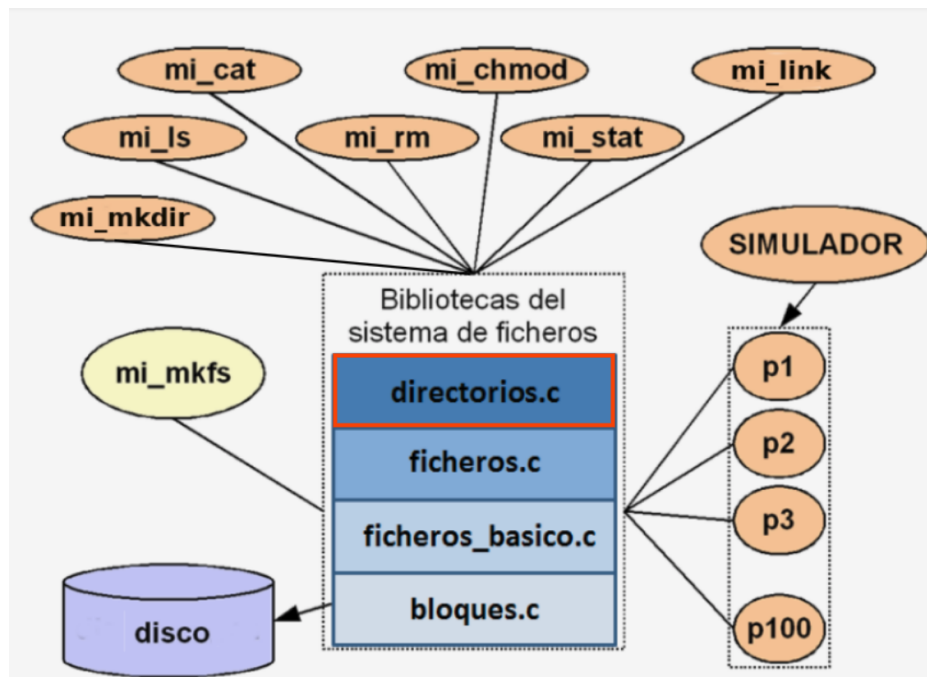
# Sistema de ficheros

## Nivel 8

**directorios.c** {mi\_creat(), mi\_dir(),  
mi\_chmod(), mi\_stat()}, y **mi\_mkdir.c**,  
[**mi\_touch.c**], **mi\_ls.c**, **mi\_chmod.c**,  
**mi\_stat.c**

## Nivel 8: directorios.c {mi\_creat(), mi\_dir(), mi\_chmod(), mi\_stat()}, y mi\_mkdir.c, [mi\_touch.c], mi\_ls.c, mi\_chmod.c, mi\_stat.c

En este nivel crearemos algunas funciones de la capa de directorios y los programas correspondientes (**comandos**) que permiten la ejecución de tales funcionalidades desde la consola. Todas las funciones llamarán a **buscar\_entrada()** para obtener el **nº de inodo**. Varias de ellas van directamente ligadas a la función correspondiente de la capa de ficheros, a la cual le pasan ese nº de inodo.



comandos:	directorios.c:	ficheros.c:
mi_mkdir, [mi_touch] mi_ls mi_chmod mi_stat  mi_escribir mi_cat  mi_link mi_rm, [mi_rmdir], [mi_rm_r]	mi_creat() mi_dir() mi_chmod() mi_stat()  mi_write() mi_read()  mi_link() mi_unlink()	mi_chmod_f() mi_stat_f()  mi_write_f() mi_read_f()  mi_truncar_f()

Tabla con la correlación de comandos y funciones de la capa de directorios y la capa de ficheros.

A continuación vamos a ir viendo (en este nivel y en los 2 siguientes) cada una de las funciones restantes de la capa de directorios junto con cada uno de los programas externos (comandos) que las llamarán.

Todos los programas externos tienen que incluir el fichero **directorios.h**, y montar y desmontar el dispositivo virtual.

## 1) Creación de ficheros y directorios

### 1a) **mi\_mkdir.c**

**Sintaxis:** `./mi_mkdir <disco> <permisos> </ruta>`

Programa (**comando**) que crea un fichero o directorio, llamando a la función **mi\_creat()**. Dependiendo de si la ruta acaba en `/` o no se estará indicando si hay que crear un directorio o un fichero.

Hay que comprobar que **permisos** sea un nº válido (0-7).

Vuestro sistema de ficheros no ha de permitir crear un directorio o fichero dentro de un fichero!!! (eso lo ha de controlar **buscar\_entrada()**).

Opcionalmente se puede crear el comando **mi\_touch**, (programa adicional **mi\_touch.c**) para crear un fichero, separando la funcionalidad de **mi\_mkdir** que se limitaría a crear directorios, comprobando la sintaxis de camino acabado en `'/'`.

### 1b) **int mi\_creat(const char \*camino, unsigned char permisos);**

Función de la capa de **directorios** que crea un fichero/directorio y su entrada de directorio. Se basa, principalmente, en la función **buscar\_entrada()** con **reservar=1**.

(Realmente habría que leer el superbloque para pasarle la posición del inodo del directorio raíz, aunque por simplicidad podemos suponer directamente que **p\_inodo\_dir** es 0<sup>1</sup>).

Los directorios intermedios han de existir!!! (eso lo controla **buscar\_entrada()**)

Otros posibles errores que devolvería al usuario a través de **buscar\_entrada()**:

- Que algún directorio no tenga permiso de lectura.
- Que el directorio padre no tenga permiso de escritura.
- Que la entrada ya exista.

## 2) Listado del contenido de un directorio

---

<sup>1</sup> En realidad habría que leer el superbloque y asignarle el campo **posInodoRaiz**.

### 2a) `mi_ls.c`

Sintaxis: `./mi_ls <disco> </ruta_directorio>`

Programa (**comando**) que lista el contenido de un directorio (**nombres** de las entradas), llamando a la función `mi_dir()` de la capa de directorios, que es quien construye el *buffer*<sup>2</sup> que mostrará `mi_ls.c`. **Indicaremos el total de entradas.**

#### Mejoras opcionales:

- Listar no solo los nombres de las entradas de directorio sino varios **datos del inodo** de cada fichero o directorio (**tipo**, **permisos**, **mtime**, **tamEnBytesLog**) tabulados. En tal caso, en `mi_dir()` cuando se obtiene el nº de inodo asociado a cada entrada, hay que leer el inodo correspondiente e incorporar al *buffer* los datos de tal inodo que vayan a aparecer en el listado. Hay que **imprimir las cabeceras** cuando el total de entradas > 0.

Ejemplo:

<code>\$/mi_ls disco /di1/dir11/</code>				
Total: 2				
Tipo	Permisos	mTime	Tamaño	Nombre
-----				
-----				
f	rw-	2018-04-18 13:59:45	0	fic111
f	rw-	2018-04-18 14:01:09	0	fic112

- Utilizar **colores** para diferenciar el listado de un fichero del de un directorio (la información del color la aportaría `mi_dir()` al buffer).
- Admitir el comando `mi_ls` también para ficheros, y en tal caso, mostrar los datos del mismo. **No habrá total de entradas.**

<code>\$/mi_ls disco /di1/dir11/fic111</code>				
Tipo	Permisos	mTime	Tamaño	Nombre
-----				
-----				
f	rw-	2018-04-18 14:01:10	0	fic111

<sup>2</sup> Si vamos a listar los datos de cada entrada en una línea, podemos suponer que cada línea tiene 100 caracteres y prever un máximo de 1000 líneas:

```
#define TAMFILA 100
#define TAMBUFFER (TAMFILA*1000) //suponemos un máx de 1000 entradas, aunque debería ser
SB.totInodos
```

En tal caso, en la llamada a `mi_dir()` desde `mi_ls.c`, podéis utilizar un parámetro adicional de `tipo`, basado en la sintaxis del camino (acabado en '/' para directorios).

```
int mi_dir(const char *camino, char *buffer, char tipo);
```

Para obtener el valor de ese parámetro basta examinar la sintaxis del último carácter del camino:

```
if(camino[strlen(camino)-1]=='/') //es un directorio
...
```

Cuando dentro de `mi_dir()` se lee el inodo se tendría que comparar ese parámetro con el tipo guardado en el inodo, y si no coincide indicar **Error: la sintaxis no concuerda con el tipo** y salir.

Podéis hacer uso del dato `*p_entrada` que os ha devuelto `buscar_entrada()` y emplearlo en `mi_read_f()` para obtener el `offset` adecuado para leer directamente esa entrada y obtener el nombre del fichero o bien utilizar un buffer de entradas al igual que en la función `buscar_entrada()` y así sólo acceder al disco cuando sea necesario

- Utilizar una opción (parámetro) para distinguir entre mostrar formato simple y expandido de `mi_ls`.

2b) `int mi_dir(const char *camino, char *buffer);` o  
`int mi_dir(const char *camino, char *buffer, char tipo);`

Función de la capa de **directorios** que pone el contenido del directorio en un *buffer* de memoria (el nombre de cada entrada puede venir separado por '|' o por un tabulador) y devuelve el número de entradas. Implica leer de forma secuencial el contenido de un inodo de tipo directorio, con `mi_read_f()` leyendo sus entradas.<sup>3</sup>

Buscamos la entrada correspondiente a `*camino` para **comprobar que existe** y leemos su inodo, comprobando que se trata de un **directorio** y que tiene **permisos de lectura**.

Para cada entrada concatenamos (mediante la función `strcat()`) su nombre al *buffer* con un separador.

Como ya se ha indicado, opcionalmente podemos leer también el inodo asociado a cada entrada e incorporar al *buffer* la información acerca de su tipo, permisos, tamaño y mtime.

Para incorporar la información acerca de los permisos:

```
if (inodo.permisos & 4) strcat(buffer, "r"); else strcat(buffer, "-");
if (inodo.permisos & 2) strcat(buffer, "w"); else strcat(buffer, "-");
```

<sup>3</sup> Aquí también se podría utilizar un **buffer de n entradas**, siendo  $n = \text{BLOCKSIZE} / \text{sizeof}(\text{struct entrada})$ , para no acceder al dispositivo cada vez que hay que leer una entrada

```
if (inodo.permisos & 1) strcat(buffer, "x"); else strcat(buffer, "-");
```

Para incorporar la información acerca del tiempo:

```
struct tm *tm; //ver info: struct tm
tm = localtime(&inodo.mtime);
sprintf(tmp, "%d-%02d-%02d %02d:%02d:%02d", tm->tm_year + 1900, tm->tm_mon + 1,
tm->tm_mday, tm->tm_hour, tm->tm_min, tm->tm_sec);
strcat(buffer, tmp);
```

Si queremos ampliar la utilidad de `mi_dir()` para aplicarla también a ficheros, podemos añadir un parámetro que indique el **tipo**<sup>4</sup> y que nos lo pasará `mi_ls.c`, para luego poder comparar la sintaxis con el tipo real del inodo que obtendremos al leer el inodo.

### 3) Cambio de permisos de un fichero o directorio

#### 3a) `mi_chmod.c`

**Sintaxis:** `./mi_chmod <disco> <permisos> </ruta>`

Cambia los permisos de un fichero o directorio, llamando a la función `mi_chmod()` de la capa de **directorios**. Los permisos se indican en octal, será 4 para sólo lectura (r-), 2 para sólo escritura (-w-), 1 para sólo ejecución (--x)...

Hay que comprobar que `permisos` sea un nº válido (0-7).

#### 3b) `int mi_chmod(const char *camino, unsigned char permisos);`

Buscar la entrada `*camino` con `buscar_entrada()` para obtener el nº de inodo (`p_inodo`). Si la entrada existe llamamos a la función correspondiente de `ficheros.c` pasándole el `p_inodo`:

```
mi_chmod_f(p_inodo, permisos);
```

### 4) Visualización metadatos del inodo

#### 4a) `mi_stat.c`

**Sintaxis:** `./mi_stat <disco> </ruta>`

Programa (comando) que muestra la información acerca del inodo de un fichero o directorio, llamando a la función `mi_stat()` de la capa de **directorios**, que a su vez llamará a `mi_stat_f()` de la capa de ficheros.

---

<sup>4</sup> `int mi_dir(const char *camino, char *buffer, char tipo);`

Ejemplo de ejecución del comando `stat` del bash en Ubuntu:

```
$ stat bloques.c
```

```
Fichero: bloques.c
```

```
Tamaño: 1600    Bloques: 8    Bloque E/S: 4096  fichero regular
```

```
Dispositivo: 802h/2050d  Nodo-i: 1839652  Enlaces: 1
```

```
Acceso: (0664/-rw-rw-r--) Uid: ( 1000/   uib)  Gid: ( 1000/   uib)
```

```
Acceso: 2021-04-26 11:04:25.511916890 +0200
```

```
Modificación: 2021-03-03 10:07:24.067776000 +0100
```

```
    Cambio: 2021-04-15 19:25:34.652026933 +0200
```

```
    Creación: -
```

4b) `int mi_stat(const char *camino, struct STAT *p_stat);`

Buscar la entrada `*camino` con `buscar_entrada()` para obtener el `p_inodo`. Si la entrada existe llamamos a la función correspondiente de `ficheros.c` pasándole el `p_inodo`:

```
mi_stat_f(p_inodo, stat);
```

Mostrar también el **nº de inodo**.

Ejemplo:

```
$ ./mi_stat disco /dir1/dir11/
```

```
Nº de inodo: 2
```

```
tipo: d
```

```
permisos: 6
```

```
atime: Mon 2021-04-26 11:58:08
```

```
ctime: Mon 2021-04-26 11:58:08
```

```
mtime: Mon 2021-04-26 11:58:08
```

```
nlinks: 1
```

```
tamEnBytesLog: 128
```

```
numBloquesOcupados: 1
```

### TESTS DE PRUEBA <sup>5</sup>

Podéis ejecutar línea a línea el script `test8.sh`<sup>6</sup>. Si no habéis implementado el comando `mi_touch`, substituirlo en el test por `mi_mkdir`.

```
$ ./test8.sh
#####
$ ./mi_mkfs disco 100000

#####
$ ./mi_mkdir #comprobar sintaxis
Sintaxis: ./mi_mkdir <nombre_dispositivo> <permisos> </ruta_directorio/>

#####
$ ./mi_mkdir disco 7 / #no ha de dejar crear la raíz al usuario

#####
$ ./mi_mkdir disco 6 dir1/
Error: Camino incorrecto.

#####
$ ./mi_mkdir disco 6 /dir1/ #permiso lectura/escritura
[buscar_entrada()→ inicial: dir1, final: /, reservar: 1]
[buscar_entrada()→ reservado inodo 1 tipo d con permisos 6 para dir1]
[buscar_entrada()→ creada entrada: dir1, 1]

$ ./mi_mkdir disco 6 /dir1/dir11/ #permiso lectura/escritura
[buscar_entrada()→ inicial: dir1, final: /dir11/, reservar: 1]
[buscar_entrada()→ inicial: dir11, final: /, reservar: 1]
[buscar_entrada()→ reservado inodo 2 tipo d con permisos 6 para dir11]
[buscar_entrada()→ creada entrada: dir11, 2]

#####
$ ./mi_chmod #comprobar sintaxis
Sintaxis: ./mi_chmod <nombre_dispositivo> <permisos> </ruta>

#####
$ ./mi_chmod disco 1 /dir1/dir11/ #permiso ejecución
[buscar_entrada()→ inicial: dir1, final: /dir11/, reservar: 0]
[buscar_entrada()→ inicial: dir11, final: /, reservar: 0]
```

<sup>5</sup> Aquí ya se puede omitir la visualización de `fprintf()` de `traducir_bloque_inodo()`

<sup>6</sup> Ejecutado línea a línea manualmente observaréis que varían los sellos de tiempo entre sí

```
$ ./mi_touch disco 6 /dir1/dir11/fic111 #Error: Permiso denegado de lectura.
```

```
[buscar_entrada()→ inicial: dir1, final: /dir11/fic111, reservar: 1]
```

```
[buscar_entrada()→ inicial: dir11, final: /fic111, reservar: 1]
```

```
[buscar_entrada()→ inicial: fic111, final: , reservar: 1]
```

```
[buscar_entrada()→ El inodo 2 no tiene permisos de lectura]
```

```
Error: Permiso denegado de lectura.
```

```
#####
```

```
./mi_chmod disco 2 /dir1/dir11/ #permiso escritura
```

```
[buscar_entrada()→ inicial: dir1, final: /dir11/, reservar: 0]
```

```
[buscar_entrada()→ inicial: dir11, final: /, reservar: 0]
```

```
$ ./mi_touch disco 6 /dir1/dir11/fic111 #Error: Permiso denegado de lectura.
```

```
[buscar_entrada()→ inicial: dir1, final: /dir11/fic111, reservar: 1]
```

```
[buscar_entrada()→ inicial: dir11, final: /fic111, reservar: 1]
```

```
[buscar_entrada()→ inicial: fic111, final: , reservar: 1]
```

```
[buscar_entrada()→ El inodo 2 no tiene permisos de lectura]
```

```
Error: Permiso denegado de lectura.
```

```
#####
```

```
$ ./mi_chmod disco 6 /dir1/dir11/ #permiso lectura/escritura
```

```
[buscar_entrada()→ inicial: dir1, final: /dir11/, reservar: 0]
```

```
[buscar_entrada()→ inicial: dir11, final: /, reservar: 0]
```

```
$ ./mi_touch disco 6 /dir1/dir11/fic111 #permiso lectura/escritura
```

```
[buscar_entrada()→ inicial: dir1, final: /dir11/fic111, reservar: 1]
```

```
[buscar_entrada()→ inicial: dir11, final: /fic111, reservar: 1]
```

```
[buscar_entrada()→ inicial: fic111, final: , reservar: 1]
```

```
[buscar_entrada()→ reservado inodo 3 tipo f con permisos 6 para fic111]
```

```
[buscar_entrada()→ creada entrada: fic111, 3]
```

```
$ ./mi_touch disco 6 /dir1/dir11/fic112 #permiso lectura/escritura
```

```
[buscar_entrada()→ inicial: dir1, final: /dir11/fic112, reservar: 1]
```

```
[buscar_entrada()→ inicial: dir11, final: /fic112, reservar: 1]
```

```
[buscar_entrada()→ inicial: fic112, final: , reservar: 1]
```

```
[buscar_entrada()→ reservado inodo 4 tipo f con permisos 6 para fic112]
```

```
[buscar_entrada()→ creada entrada: fic112, 4]
```

```
#####
```

```
$ ./mi_ls disco /
```

```
Total: 1
```

Tipo	Modo	mTime	Tamaño	Nombre
d	rw-	2021-04-26 11:58:08	64	dir1



# Sistema de ficheros

## Nivel 8

`directorios.c` {`mi_creat()`, `mi_dir()`,  
`mi_chmod()`, `mi_stat()`}, y `mi_mkdir.c`,  
`[mi_touch.c]`, `mi_ls.c`, `mi_chmod.c`, `mi_stat.c`

```
#####
```

```
$ ./mi_stat disco /dir1/
```

```
[buscar_entrada()→ inicial: dir1, final: /, reservar: 0]
```

```
Nº de inodo: 1
```

```
tipo: d
```

```
permisos: 6
```

```
atime: Mon 2021-04-26 11:58:08
```

```
ctime: Mon 2021-04-26 11:58:08
```

```
mtime: Mon 2021-04-26 11:58:08
```

```
nlinks: 1
```

```
tamEnBytesLog: 64
```

```
numBloquesOcupados: 1
```

```
$ ./mi_ls disco /dir1/
```

```
[buscar_entrada()→ inicial: dir1, final: /, reservar: 0]
```

```
Total: 1
```

Tipo	Modo	mTime	Tamaño	Nombre
d	rw-	2021-04-26 11:58:08	128	dir11

```
#####
```

```
$ ./mi_stat disco /dir1/dir11/
```

```
[buscar_entrada()→ inicial: dir1, final: /dir11/, reservar: 0]
```

```
[buscar_entrada()→ inicial: dir11, final: /, reservar: 0]
```

```
Nº de inodo: 2
```

```
tipo: d
```

```
permisos: 6
```

```
atime: Mon 2021-04-26 11:58:08
```

```
ctime: Mon 2021-04-26 11:58:08
```

```
mtime: Mon 2021-04-26 11:58:08
```

```
nlinks: 1
```

```
tamEnBytesLog: 128
```

```
numBloquesOcupados: 1
```

```
$ ./mi_ls disco /dir1/dir11/
```

```
[buscar_entrada()→ inicial: dir1, final: /dir11/, reservar: 0]
```

```
[buscar_entrada()→ inicial: dir11, final: /, reservar: 0]
```

```
Total: 2
```

Tipo	Modo	mTime	Tamaño	Nombre
f	rw-	2021-04-26 11:58:08	0	fic111
f	rw-	2021-04-26 11:58:08	0	fic112

```
#####
```

```
./mi_ls disco /dir1/dir12/ #Error: No existe el archivo o el directorio.
```

```
[buscar_entrada()→ inicial: dir1, final: /dir12/, reservar: 0]
```

```
[buscar_entrada()→ inicial: dir12, final: /, reservar: 0]
```

**Error: No existe el archivo o el directorio.**

```
#####
```

```
$ ./mi_touch disco 6 /dir1/dir11/fic111 #Error: El archivo ya existe.
```

```
[buscar_entrada()→ inicial: dir1, final: /dir11/fic111, reservar: 1]
```

```
[buscar_entrada()→ inicial: dir11, final: /fic111, reservar: 1]
```

```
[buscar_entrada()→ inicial: fic111, final: /, reservar: 1]
```

**Error: El archivo ya existe.**

```
#####
```

```
$ ./mi_mkdir disco 6 /dir1/dir11/fic111/dir12/ #Error: No es un directorio.
```

```
[buscar_entrada()→ inicial: dir1, final: /dir11/fic111/dir12/, reservar: 1]
```

```
[buscar_entrada()→ inicial: dir11, final: /fic111/dir12/, reservar: 1]
```

```
[buscar_entrada()→ inicial: fic111, final: /dir12/, reservar: 1]
```

```
[buscar_entrada()→ inicial: dir12, final: /, reservar: 1]
```

**Error: No es un directorio.**

```
#####
```

```
$ ./mi_touch disco 6 /dir1/dir11/dir12/fic111 #Error: No existe algún directorio  
intermedio.
```

```
[buscar_entrada()→ inicial: dir1, final: /dir11/dir12/fic111, reservar: 1]
```

```
[buscar_entrada()→ inicial: dir11, final: /dir12/fic111, reservar: 1]
```

```
[buscar_entrada()→ inicial: dir12, final: /fic111, reservar: 1]
```

**Error: No existe algún directorio intermedio.**

```
#####
```

```
$ ./mi_mkdir disco 9 /dir2/ #Error: modo inválido: <<9>>
```

**Error: modo inválido: <<9>>**