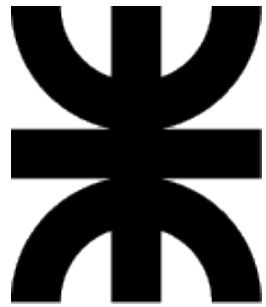
Informática II
Manejo de archivos a bajo nivel en lenguaje C

Gonzalo F. Perez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2024 –

Manejo de archivos en C (repaso)

- Cada programa en ejecución (proceso) tiene una cantidad de archivos asociados (descriptor de archivo).

Manejo de archivos en C (repaso)

- Cada programa en ejecución (proceso) tiene una cantidad de archivos asociados (descriptor de archivo).
- Descriptores de archivos: son números enteros pequeños que se pueden utilizar para acceder a estos archivos o dispositivos.

Manejo de archivos en C (repaso)

- Cada programa en ejecución (proceso) tiene una cantidad de archivos asociados (descriptor de archivo).
- Descriptores de archivos: son números enteros pequeños que se pueden utilizar para acceder a estos archivos o dispositivos.

Todo programa tiene abierto tres descriptores de archivos:

- 0: entrada estándar (`stdin`)
- 1: salida estándar (`stdout`)
- 2: error estándar (`stderr`)

Manejo de archivos en C – Ejemplos

escritura.c

```c
#include <unistd.h>

int main(void)
{
  if((write(1, "Hola mundo!\n", 12)) != 12)
      write(2, "ERROR\n", 6);

  return 0;
}
```

Manejo de archivos en C – Ejemplos

lectura.c

```c
#include <unistd.h>

int main(void) {
  char buffer[128];
  int nread;

  nread = read(0, buffer, 128);
  if(nread == -1)
    write(2, "Error de lectura\n", 17);

  if((write(1, buffer, nread)) != nread)
    write(2, "Error de escritura\n", 19);

  return 0;
}
```

Manejo de archivos en C – Ejemplos

lectura.c

```c
#include <unistd.h>

int main(void) {
  char buffer[128];
  int nread;

  nread = read(0, buffer, 128);
  if(nread == -1)
    write(2, "Error de lectura\n", 17);

  if((write(1, buffer, nread)) != nread)
    write(2, "Error de escritura\n", 19);

  return 0;
}
```

- Compilar y ejecutar (con/sin redirección)

Manejo de archivos en C – Ejemplos

lectura.c

```c
#include <unistd.h>

int main(void) {
  char buffer[128];
  int nread;

  nread = read(0, buffer, 128);
  if(nread == -1)
    write(2, "Error de lectura\n", 17);

  if((write(1, buffer, nread)) != nread)
    write(2, "Error de escritura\n", 19);

  return 0;
}
```

- Compilar y ejecutar (con/sin redirección)
- Ejecutar con redirección
  `> ./a.out < prueba.txt`

Manejo de archivos en C

Los programas pueden manejar archivos de disco, puerto serie, y otros dispositivos (excepto conexiones de red), todos de la misma forma.

Manejo de archivos en C

Los programas pueden manejar archivos de disco, puerto serie, y otros dispositivos (excepto conexiones de red), todos de la misma forma.

Archivos de dispositivos

Manejo de archivos en C

Los programas pueden manejar archivos de disco, puerto serie, y otros dispositivos (excepto conexiones de red), todos de la misma forma.

Archivos de dispositivos

- Los dispositivos de hardware se representan (mapean) por archivos.

Manejo de archivos en C

Los programas pueden manejar archivos de disco, puerto serie, y otros dispositivos (excepto conexiones de red), todos de la misma forma.

Archivos de dispositivos

- Los dispositivos de hardware se representan (mapean) por archivos.
- Los dispositivos se clasifican en: dispositivos de caracteres o de bloques.

Manejo de archivos en C

Los programas pueden manejar archivos de disco, puerto serie, y otros dispositivos (excepto conexiones de red), todos de la misma forma.

Archivos de dispositivos

- Los dispositivos de hardware se representan (mapean) por archivos.
- Los dispositivos se clasifican en: dispositivos de caracteres o de bloques.

```
> ls -l /dev
crw-rw-rw-  1 root root        1,   3 jul 10 13:54 null
```

Manejo de archivos en C

Los programas pueden manejar archivos de disco, puerto serie, y otros dispositivos (excepto conexiones de red), todos de la misma forma.

Archivos de dispositivos

- Los dispositivos de hardware se representan (mapean) por archivos.
- Los dispositivos se clasifican en: dispositivos de caracteres o de bloques.

```
> ls -l /dev
crw-rw-rw-  1 root root        1,   3 jul 10 13:54 null
brw-rw----  1 root disk        8,   0 jul 19 13:51 sda
brw-rw----  1 root disk        8,   1 jul 19 13:51 sda1
brw-rw----  1 root disk        8,   2 jul 19 13:51 sda2
brw-rw----  1 root disk        8,   3 jul 19 13:51 sda3
```

Manejo de archivos en C

Los programas pueden manejar archivos de disco, puerto serie, y otros dispositivos (excepto conexiones de red), todos de la misma forma.

Archivos de dispositivos

- Los dispositivos de hardware se representan (mapean) por archivos.
- Los dispositivos se clasifican en: dispositivos de caracteres o de bloques.

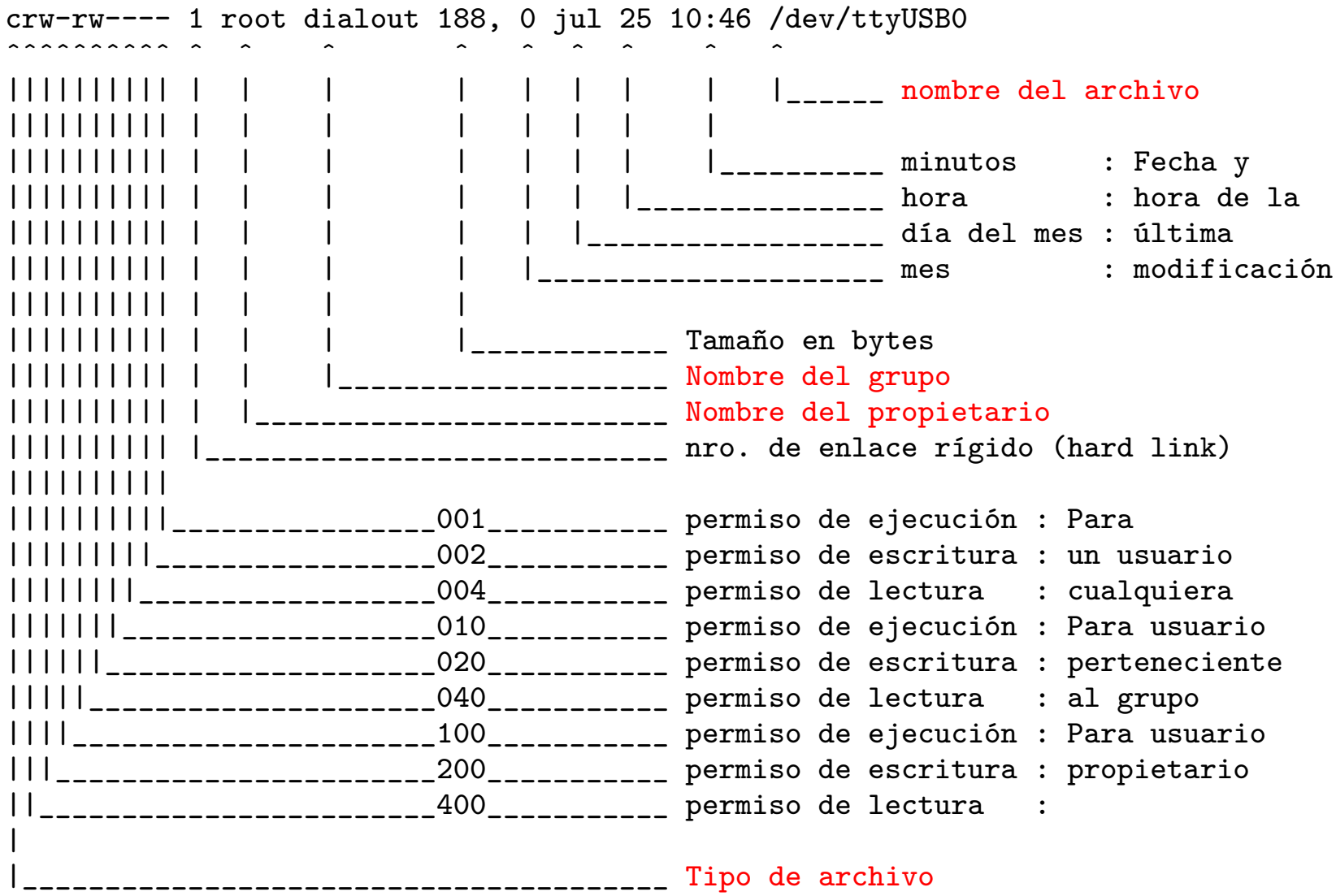
```
> ls -l /dev
crw-rw-rw-  1 root root        1,   3 jul 10 13:54 null
brw-rw----  1 root disk        8,   0 jul 19 13:51 sda
brw-rw----  1 root disk        8,   1 jul 19 13:51 sda1
brw-rw----  1 root disk        8,   2 jul 19 13:51 sda2
brw-rw----  1 root disk        8,   3 jul 19 13:51 sda3
crw-rw----  1 root dialout     4,  64 jul 10 13:53 ttyS0
crw-rw----  1 root dialout   188,   0 jul 19 19:10 ttyUSB0
```

Manejo de archivos en C (repaso)

Cada archivo tiene un nombre y algunas propiedades como la fecha de creación/modificación, permisos, etc.

Manejo de archivos en C (repaso)

Cada archivo tiene un nombre y algunas propiedades como la fecha de creación/modificación, permisos, etc.



Manejo de archivos en C

Acceso a archivos

- Funciones de bajo nivel (llamada al sistema y drives de dispositivos)
- Funciones de alto nivel (biblioteca de entrada/salida) [buffers]

Manejo de archivos en C

Acceso a archivos

- Funciones de bajo nivel (llamada al sistema y drives de dispositivos)
- Funciones de alto nivel (biblioteca de entrada/salida) [buffers]

- Bajo nivel: Archivo de cabecera `unistd.h`
  En los lenguajes C y C++ este archivo de cabecera define la API que brinda acceso al sistema operativo POSIX (descriptor de archivo).

Manejo de archivos en C

Acceso a archivos

- Funciones de bajo nivel (llamada al sistema y drives de dispositivos)
- Funciones de alto nivel (biblioteca de entrada/salida) [buffers]

- Bajo nivel: Archivo de cabecera `unistd.h`
  En los lenguajes C y C++ este archivo de cabecera define la API que brinda acceso al sistema operativo POSIX (descriptor de archivo).
  (`open`, `close`, `write`, `read`, etc.)

Manejo de archivos en C

Acceso a archivos

- Funciones de bajo nivel (llamada al sistema y drives de dispositivos)
- Funciones de alto nivel (biblioteca de entrada/salida) [buffers]

- Bajo nivel: Archivo de cabecera `unistd.h`
  En los lenguajes C y C++ este archivo de cabecera define la API que brinda acceso al sistema operativo POSIX (descriptor de archivo).
  (`open`, `close`, `write`, `read`, etc.)

- Alto nivel: Archivo de cabecera `stdio.h` (ANSI C)
  Manejo de *stream*, se implementa como puntero a estructura de tipo `FILE*`.

Manejo de archivos en C

Acceso a archivos

- Funciones de bajo nivel (llamada al sistema y drives de dispositivos)
- Funciones de alto nivel (biblioteca de entrada/salida) [buffers]

- Bajo nivel: Archivo de cabecera `unistd.h`
  En los lenguajes C y C++ este archivo de cabecera define la API que brinda acceso al sistema operativo POSIX (descriptor de archivo).
  (`open`, `close`, `write`, `read`, etc.)

- Alto nivel: Archivo de cabecera `stdio.h` (ANSI C)
  Manejo de *stream*, se implementa como puntero a estructura de tipo `FILE*`.
  (`fopen`, `fclose`, `fwrite`, `fread`, `fseek`, `fflush`, etc.)

Manejo de archivos en C – Funciones de bajo nivel

Se utilizan cinco funciones: `open`, `close`, `write`, `read`, e `ioctrl`.

Manejo de archivos en C – Funciones de bajo nivel

Se utilizan cinco funciones: `open`, `close`, `write`, `read`, e `ioctrl`.

1. `open()`: Abrir archivo o dispositivo

Manejo de archivos en C – Funciones de bajo nivel

Se utilizan cinco funciones: `open`, `close`, `write`, `read`, e `ioctrl`.

1. `open()`: Abrir archivo o dispositivo
2. `close()`: Cerrar archivo o dispositivo

Manejo de archivos en C – Funciones de bajo nivel

Se utilizan cinco funciones: `open`, `close`, `write`, `read`, e `ioctrl`.

1. `open()`: Abrir archivo o dispositivo
2. `close()`: Cerrar archivo o dispositivo
3. `read()`: Leer archivo o dispositivo

Manejo de archivos en C – Funciones de bajo nivel

Se utilizan cinco funciones: `open`, `close`, `write`, `read`, e `ioctrl`.

1. `open()`: Abrir archivo o dispositivo
2. `close()`: Cerrar archivo o dispositivo
3. `read()`: Leer archivo o dispositivo
4. `write()`: Escribir archivo o dispositivo

Manejo de archivos en C – Funciones de bajo nivel

Se utilizan cinco funciones: `open`, `close`, `write`, `read`, e `ioctrl`.

1. `open()`: Abrir archivo o dispositivo
2. `close()`: Cerrar archivo o dispositivo
3. `read()`: Leer archivo o dispositivo
4. `write()`: Escribir archivo o dispositivo
5. `ioctrl()`: Intercambiar información de control con el driver

Manejo de archivos en C – Funciones de bajo nivel

Se utilizan cinco funciones: `open`, `close`, `write`, `read`, e `ioctrl`.

1. `open()`: Abrir archivo o dispositivo
2. `close()`: Cerrar archivo o dispositivo
3. `read()`: Leer archivo o dispositivo
4. `write()`: Escribir archivo o dispositivo
5. `ioctrl()`: Intercambiar información de control con el driver

En el Kernel están los *drivers de dispositivos* (device drivers): interfaz de bajo nivel para el control de hardware.

Manejo de archivos en C – Funciones de bajo nivel

Se utilizan cinco funciones: `open`, `close`, `write`, `read`, e `ioctrl`.

1. `open()`: Abrir archivo o dispositivo
2. `close()`: Cerrar archivo o dispositivo
3. `read()`: Leer archivo o dispositivo
4. `write()`: Escribir archivo o dispositivo
5. `ioctrl()`: Intercambiar información de control con el driver

En el Kernel están los *drivers de dispositivos* (device drivers): interfaz de bajo nivel para el control de hardware.

Descriptores de archivos abiertos en cualquier programa: `0`, `1` y `2`

Manejo de archivos en C – Funciones de bajo nivel

Abrir archivo

```c
#include <unistd.h> /* UNIX standard function definitions */
#include <fcntl.h> /* File control definitions */

int open(const char* path, int oflags);
```

Manejo de archivos en C – Funciones de bajo nivel

Abrir archivo

```c
#include <unistd.h> /* UNIX standard function definitions */
#include <fcntl.h> /* File control definitions */

int open(const char* path, int oflags);
```

- `path`: nombre del archivo o dispositivo, p.e.: `"/dev/ttyUSB0"`

Manejo de archivos en C – Funciones de bajo nivel

Abrir archivo

```c
#include <unistd.h> /* UNIX standard function definitions */
#include <fcntl.h> /* File control definitions */

int open(const char* path, int oflags);
```

- `path`: nombre del archivo o dispositivo, p.e.: `"/dev/ttyUSB0"`
- `oflags`: indica acciones al abrir el archivo
  - `O_RDONLY`: abrir para solo lectura

Manejo de archivos en C – Funciones de bajo nivel

Abrir archivo

```c
#include <unistd.h> /* UNIX standard function definitions */
#include <fcntl.h> /* File control definitions */

int open(const char* path, int oflags);
```

- `path`: nombre del archivo o dispositivo, p.e.: `"/dev/ttyUSB0"`
- `oflags`: indica acciones al abrir el archivo
  - `O_RDONLY`: abrir para solo lectura
  - `O_WRONLY`: abrir para solo escritura

Manejo de archivos en C – Funciones de bajo nivel

Abrir archivo

```c
#include <unistd.h> /* UNIX standard function definitions */
#include <fcntl.h> /* File control definitions */

int open(const char* path, int oflags);
```

- `path`: nombre del archivo o dispositivo, p.e.: `"/dev/ttyUSB0"`
- `oflags`: indica acciones al abrir el archivo
  - `O_RDONLY`: abrir para solo lectura
  - `O_WRONLY`: abrir para solo escritura
  - `O_RDWR`: abrir para lectura y escritura

Manejo de archivos en C – Funciones de bajo nivel

Abrir archivo

```c
#include <unistd.h> /* UNIX standard function definitions */
#include <fcntl.h> /* File control definitions */

int open(const char* path, int oflags);
```

- `path`: nombre del archivo o dispositivo, p.e.: `"/dev/ttyUSB0"`
- `oflags`: indica acciones al abrir el archivo
  - `O_RDONLY`: abrir para solo lectura
  - `O_WRONLY`: abrir para solo escritura
  - `O_RDWR`: abrir para lectura y escritura

Devuelve el *descriptor de archivo* (entero no negativo) si tuvo éxito, o -1 si falló. El descriptor de archivo se puede usar en lectura, escritura y otras llamadas al sistema.

Manejo de archivos en C – Funciones de bajo nivel

Abrir archivo

```c
#include <unistd.h> /* UNIX standard function definitions */
#include <fcntl.h> /* File control definitions */

int open(const char* path, int oflags);
```

- `path`: nombre del archivo o dispositivo, p.e.: `"/dev/ttyUSB0"`
- `oflags`: indica acciones al abrir el archivo
  - `O_RDONLY`: abrir para solo lectura
  - `O_WRONLY`: abrir para solo escritura
  - `O_RDWR`: abrir para lectura y escritura

Devuelve el *descriptor de archivo* (entero no negativo) si tuvo éxito, o -1 si falló. El descriptor de archivo se puede usar en lectura, escritura y otras llamadas al sistema.

Las opciones fijadas con `oflags` se pueden combinar mediante el uso de OR con `O_APPEND`, `O_TRUNC` y `O_CREAT`.

Manejo de archivos en C – Funciones de bajo nivel

Cerrar archivo

```c
#include <unistd.h> /* UNIX standard function definitions */

int close(int fildes);
```

- `fildes`: descriptor de archivo (devuelto por `open`)

Manejo de archivos en C – Funciones de bajo nivel

Cerrar archivo

```c
#include <unistd.h> /* UNIX standard function definitions */

int close(int fildes);
```

- `fildes`: descriptor de archivo (devuelto por `open`)

Devuelve 0 si tiene éxito y –1 en caso de error.

Manejo de archivos en C – Funciones de bajo nivel

Escribir archivo

```c
#include <unistd.h> /* UNIX standard function definitions */

size_t write(int fildes, const void *buf, size_t nbytes);
```

Manejo de archivos en C – Funciones de bajo nivel

Escribir archivo

```
#include <unistd.h> /* UNIX standard function definitions */

size_t write(int fildes, const void *buf, size_t nbytes);
```

Envía los primeros `nbytes` de `buf` para que se escriban en el archivo asociado con el descriptor de archivo `fildes`.

Manejo de archivos en C – Funciones de bajo nivel

Escribir archivo

```c
#include <unistd.h> /* UNIX standard function definitions */

size_t write(int fildes, const void *buf, size_t nbytes);
```

Envía los primeros `nbytes` de `buf` para que se escriban en el archivo asociado con el descriptor de archivo `fildes`.

- Devuelve la cantidad total de bytes realmente escritos.

Manejo de archivos en C – Funciones de bajo nivel

Escribir archivo

```
#include <unistd.h> /* UNIX standard function definitions */

size_t write(int fildes, const void *buf, size_t nbytes);
```

Envía los primeros `nbytes` de `buf` para que se escriban en el archivo asociado con el descriptor de archivo `fildes`.

- Devuelve la cantidad total de bytes realmente escritos.
- Puede ser menos que `nbytes` si ha habido algún error en el descriptor de archivo.

Manejo de archivos en C – Funciones de bajo nivel

Escribir archivo

```c
#include <unistd.h> /* UNIX standard function definitions */

size_t write(int fildes, const void *buf, size_t nbytes);
```

Envía los primeros `nbytes` de `buf` para que se escriban en el archivo asociado con el descriptor de archivo `fildes`.

- Devuelve la cantidad total de bytes realmente escritos.
- Puede ser menos que `nbytes` si ha habido algún error en el descriptor de archivo.
- Si la función devuelve:
  - 0: significa que no se han escritos datos algunos
  - -1: ha habido un error en la llamada de escritura

Manejo de archivos en C – Funciones de bajo nivel

Leer archivo

```c
#include <unistd.h> /* UNIX standard function definitions */

size_t read(int fildes, void *buf, size_t nbytes);
```

Manejo de archivos en C – Funciones de bajo nivel

Leer archivo

```
#include <unistd.h> /* UNIX standard function definitions */

size_t read(int fildes, void *buf, size_t nbytes);
```

Lee hasta `nbytes` de datos del archivo asociado con el descriptor de archivo `fildes` y los coloca en el área de datos `buf`.

Manejo de archivos en C – Funciones de bajo nivel

Leer archivo

```
#include <unistd.h> /* UNIX standard function definitions */

size_t read(int fildes, void *buf, size_t nbytes);
```

Lee hasta `nbytes` de datos del archivo asociado con el descriptor de archivo `fildes` y los coloca en el área de datos `buf`.

- Devuelve el número de bytes de datos realmente leídos, que puede ser menor que el número solicitado.

Manejo de archivos en C – Funciones de bajo nivel

Leer archivo

```c
#include <unistd.h> /* UNIX standard function definitions */

size_t read(int fildes, void *buf, size_t nbytes);
```

Lee hasta `nbytes` de datos del archivo asociado con el descriptor de archivo `fildes` y los coloca en el área de datos `buf`.

- Devuelve el número de bytes de datos realmente leídos, que puede ser menor que el número solicitado.
- La llamada devuelve:
  - 0: no tenía nada que leer; llegó al final del archivo
  - -1: error en la llamada

Manejo de archivos en C – Funciones de bajo nivel

I/O control

```c
#include <unistd.h> /* UNIX standard function definitions */

int ioctl(int fildes, int cmd, ...);
```

Manejo de archivos en C – Funciones de bajo nivel

I/O control

```c
#include <unistd.h> /* UNIX standard function definitions */

int ioctl(int fildes, int cmd, ...);
```

Proporciona una interfaz para controlar el comportamiento de los dispositivos y sus descriptores y configurar los servicios subyacentes.

Manejo de archivos en C – Funciones de bajo nivel

I/O control

```c
#include <unistd.h> /* UNIX standard function definitions */

int ioctl(int fildes, int cmd, ...);
```

Proporciona una interfaz para controlar el comportamiento de los dispositivos y sus descriptores y configurar los servicios subyacentes.

- Realiza la función indicada por `cmd` en el objeto al que hace referencia el descriptor `fildes`.

Manejo de archivos en C – Funciones de bajo nivel

I/O control

```c
#include <unistd.h> /* UNIX standard function definitions */

int ioctl(int fildes, int cmd, ...);
```

Proporciona una interfaz para controlar el comportamiento de los dispositivos y sus descriptores y configurar los servicios subyacentes.

- Realiza la función indicada por `cmd` en el objeto al que hace referencia el descriptor `fildes`.
- Puede tomar un tercer argumento opcional, dependiendo de las funciones admitidas por un dispositivo en particular.

Manejo de archivos en C – Ejemplos c/Arduino

```c
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

const char port[] = "/dev/ttyUSB0";

int main(void)
{
  int fd = open(port, O_WRONLY | O_NOCTTY | O_SYNC); /* Abre archivo */
  if(fd < -1)
  {
    printf("ERROR: no se pudo abrir el archivo\n");
    return -1;
  }

  for(;;) /* Blucle */
  {
    printf("Parpadeando el LED...\n");
    write(fd, "t", 2); /* Envía un caracter */
    sleep(1);
  }
  return 0;
}
```