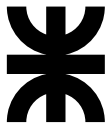


Informática II

Puerto serie

Gonzalo F. Pérez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2017 –

Contenido

- 1 **Introducción**
- 2 Hardware
- 3 Registros
- 4 Programación
- 5 Algunos ejemplos
- 6 Bibliografía

Introducción

Comunicación serie

- Se necesita conversor paralelo↔serie para “serializar/des-serializar” los datos
- Para la comunicación serie RS232 se utilizan las UARTs (Universal Asynchronous Receiver-Transmitter)

Comunicación serie

- Se necesita conversor paralelo↔serie para “serializar/des-serializar” los datos
- Para la comunicación serie RS232 se utilizan las UARTs (Universal Asynchronous Receiver-Transmitter)

Ventajas de la transmisión de datos serie (vs. paralelo):

- En la comunicación en paralelo se necesitan tantos cables como ancho de la palabra de comunicación
- Los cables pueden tener mayor longitud (RS232)
- Es posible reemplazar la conexión cableada por comunicación inalámbrica, como p.e. utilizando IR, láser, etc.
- Los microcontroladores en general cuentan con puertos de comunicación serie (UART, I²C, SPI, etc.)

Contenido

1 Introducción

2 Hardware

- Conector y señales
- Trama de comunicación
- Conexión null-módem
- UART, Universidad Asynchronous Receive-Transmitter
- Asignación de pines de la UART
- Dirección de puertos y registros

3 Registros

4 Programación

5 Algunos ejemplos

6 Bibliografía

Hardware

Las especificaciones eléctricas del **puerto serie** se definen en el estándar **EIA**¹, que para el **RS232C** son

- Un *espacio* (0 lógico) entre +3 y +25V
- Una *marca* (1 lógico) entre -3 y -25V
- Los valores entre +3V y -3V representan estados indefinidos
- Corriente de cortocircuito <500mA

(entre otros parámetros como: capacitancia máxima de línea, baudrate máximo, etc.)

¹EIA: Electronics Industry Association - RS: Recommended Standard
EIA-232C (1969), EIA-232D (1986), EIA-232 (1991)

Hardware

Las especificaciones eléctricas del **puerto serie** se definen en el estándar **EIA**¹, que para el **RS232C** son

- Un *espacio* (0 lógico) entre +3 y +25V
- Una *marca* (1 lógico) entre -3 y -25V
- Los valores entre +3V y -3V representan estados indefinidos
- Corriente de cortocircuito <500mA

(entre otros parámetros como: capacitancia máxima de línea, baudrate máximo, etc.)

Define una comunicación entre:

- **DTE**: Data Terminal Equipment
(p.e. PC)
- **DCE**: Data Communication
Equipment (p.e. módem)

¹EIA: Electronics Industry Association - RS: Recommended Standard
EIA-232C (1969), EIA-232D (1986), EIA-232 (1991)

Hardware

Las especificaciones eléctricas del **puerto serie** se definen en el estándar **EIA**¹, que para el **RS232C** son

- Un *espacio* (0 lógico) entre +3 y +25V
- Una *marca* (1 lógico) entre -3 y -25V
- Los valores entre +3V y -3V representan estados indefinidos
- Corriente de cortocircuito <500mA

(entre otros parámetros como: capacitancia máxima de línea, baudrate máximo, etc.)

Define una comunicación entre:

- **DTE**: Data Terminal Equipment
(p.e. PC)
- **DCE**: Data Communication
Equipment (p.e. módem)

Es posible comunicar dos PCs (DTE)
mediante cable **null-model**

¹EIA: Electronics Industry Association - RS: Recommended Standard
EIA-232C (1969), EIA-232D (1986), EIA-232 (1991)

Hardware

Las especificaciones eléctricas del **puerto serie** se definen en el estándar **EIA**¹, que para el **RS232C** son

- Un *espacio* (0 lógico) entre +3 y +25V
- Una *marca* (1 lógico) entre -3 y -25V
- Los valores entre +3V y -3V representan estados indefinidos
- Corriente de cortocircuito <500mA

(entre otros parámetros como: capacitancia máxima de línea, baudrate máximo, etc.)

Define una comunicación entre:

- **DTE**: Data Terminal Equipment (p.e. PC)
- **DCE**: Data Communication Equipment (p.e. módem)

Es posible comunicar dos PCs (DTE) mediante cable **null-model**

Conector DB-9/DB-25 macho



¹EIA: Electronics Industry Association - RS: Recommended Standard
EIA-232C (1969), EIA-232D (1986), EIA-232 (1991)

Hardware

Conector y señales

DB-9 (DB-25) Pin No.	Abbrev.	Full name	
3 (2)	TD	Transmit Data	
2 (3)	RD	Receive Data	
7 (4)	RTS	Request To Send	
8 (5)	CTS	Clear To Send	
6 (6)	DSR	Data Set Ready	
5 (7)	SG	Signal Ground	
1 (8)	DCD	Data Carrier Detect	
4 (20)	DTR	Data Terminal Ready	
9 (22)	RI	Ring Indicator	

Hardware

Conector y señales

DB-9 (DB-25) Pin No.	Abbrev.	Full name	(PC) — (.)
3 (2)	TD	Transmit Data	DTE → DCE
2 (3)	RD	Receive Data	
7 (4)	RTS	Request To Send	
8 (5)	CTS	Clear To Send	
6 (6)	DSR	Data Set Ready	
5 (7)	SG	Signal Ground	
1 (8)	DCD	Data Carrier Detect	
4 (20)	DTR	Data Terminal Ready	
9 (22)	RI	Ring Indicator	

TD: Salida de datos seriales, TxD

Hardware

Conector y señales

DB-9 (DB-25) Pin No.	Abbrev.	Full name	(PC) — (.)
3 (2)	TD	Transmit Data	DTE → DCE
2 (3)	RD	Receive Data	DTE ← DCE
7 (4)	RTS	Request To Send	
8 (5)	CTS	Clear To Send	
6 (6)	DSR	Data Set Ready	
5 (7)	SG	Signal Ground	
1 (8)	DCD	Data Carrier Detect	
4 (20)	DTR	Data Terminal Ready	
9 (22)	RI	Ring Indicator	

RD: Entrada de datos seriales, RxD

Hardware

Conector y señales

DB-9 (DB-25) Pin No.	Abbrev.	Full name	(PC) — (.)
3 (2)	TD	Transmit Data	DTE → DCE
2 (3)	RD	Receive Data	DTE ← DCE
7 (4)	RTS	Request To Send	DTE → DCE
8 (5)	CTS	Clear To Send	
6 (6)	DSR	Data Set Ready	
5 (7)	SG	Signal Ground	
1 (8)	DCD	Data Carrier Detect	
4 (20)	DTR	Data Terminal Ready	
9 (22)	RI	Ring Indicator	

RTS: Le indica al módem que la UART está lista para comunicarse

Hardware

Conector y señales

DB-9 (DB-25) Pin No.	Abbrev.	Full name	(PC) — (.)
3 (2)	TD	Transmit Data	DTE → DCE
2 (3)	RD	Receive Data	DTE ← DCE
7 (4)	RTS	Request To Send	DTE → DCE
8 (5)	CTS	Clear To Send	DTE ← DCE
6 (6)	DSR	Data Set Ready	
5 (7)	SG	Signal Ground	
1 (8)	DCD	Data Carrier Detect	
4 (20)	DTR	Data Terminal Ready	
9 (22)	RI	Ring Indicator	

CTS: Indica que el módem está listo para comunicarse

Hardware

Conector y señales

DB-9 (DB-25) Pin No.	Abbrev.	Full name	(PC) — (.)
3 (2)	TD	Transmit Data	DTE → DCE
2 (3)	RD	Receive Data	DTE ← DCE
7 (4)	RTS	Request To Send	DTE → DCE
8 (5)	CTS	Clear To Send	DTE ← DCE
6 (6)	DSR	Data Set Ready	DTE ← DCE
5 (7)	SG	Signal Ground	
1 (8)	DCD	Data Carrier Detect	
4 (20)	DTR	Data Terminal Ready	
9 (22)	RI	Ring Indicator	

DSR: Le indica a la UART que el módem está listo para establecer una conexión

Hardware

Conector y señales

DB-9 (DB-25) Pin No.	Abbrev.	Full name	(PC) — (.)
3 (2)	TD	Transmit Data	DTE → DCE
2 (3)	RD	Receive Data	DTE ← DCE
7 (4)	RTS	Request To Send	DTE → DCE
8 (5)	CTS	Clear To Send	DTE ← DCE
6 (6)	DSR	Data Set Ready	DTE ← DCE
5 (7)	SG	Signal Ground	DTE — DCE
1 (8)	DCD	Data Carrier Detect	DTE ← DCE
4 (20)	DTR	Data Terminal Ready	
9 (22)	RI	Ring Indicator	

DCD: Indica que el módem detecta “portadora”

Hardware

Conector y señales

DB-9 (DB-25) Pin No.	Abbrev.	Full name	(PC) — (.)
3 (2)	TD	Transmit Data	DTE → DCE
2 (3)	RD	Receive Data	DTE ← DCE
7 (4)	RTS	Request To Send	DTE → DCE
8 (5)	CTS	Clear To Send	DTE ← DCE
6 (6)	DSR	Data Set Ready	DTE ← DCE
5 (7)	SG	Signal Ground	DTE — DCE
1 (8)	DCD	Data Carrier Detect	DTE ← DCE
4 (20)	DTR	Data Terminal Ready	DTE → DCE
9 (22)	RI	Ring Indicator	

DTR: Opuesto a DSR. Le indica al módem que la UART está lista para establecer conexión

Hardware

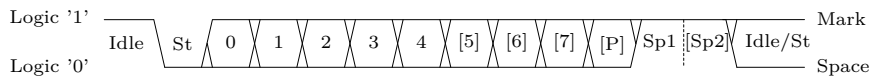
Conector y señales

DB-9 (DB-25) Pin No.	Abbrev.	Full name	(PC) — (.)
3 (2)	TD	Transmit Data	DTE → DCE
2 (3)	RD	Receive Data	DTE ← DCE
7 (4)	RTS	Request To Send	DTE → DCE
8 (5)	CTS	Clear To Send	DTE ← DCE
6 (6)	DSR	Data Set Ready	DTE ← DCE
5 (7)	SG	Signal Ground	DTE — DCE
1 (8)	DCD	Data Carrier Detect	DTE ← DCE
4 (20)	DTR	Data Terminal Ready	DTE → DCE
9 (22)	RI	Ring Indicator	DTE ← DCE

RI: Se activa ante la presencia de llamada

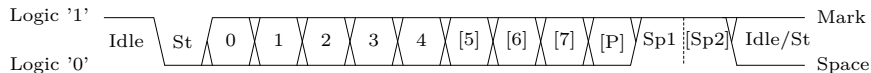
Hardware

Trama de comunicación



Hardware

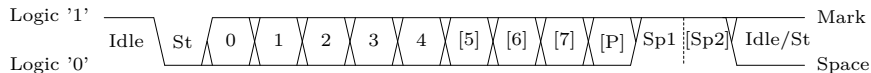
Trama de comunicación



- Idle: No hay transferencia de datos (TxD y Rx) <Logic 1>

Hardware

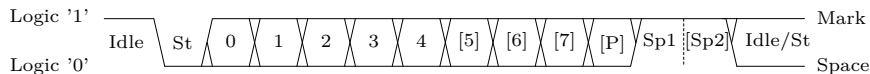
Trama de comunicación



- Idle: No hay transferencia de datos (TxD y Rx) <Logic 1>
- St: Start bit (bit de arranque) <Logic 0>

Hardware

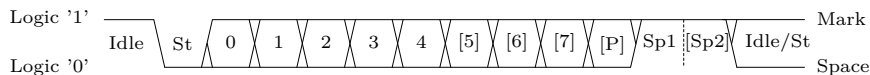
Trama de comunicación



- Idle: No hay transferencia de datos (TxD y RxD) <Logic 1>
- St: Start bit (bit de arranque) <Logic 0>
- (n): Bits de datos

Hardware

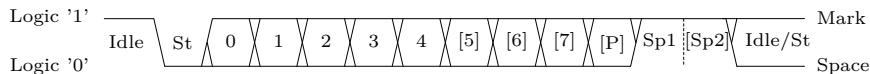
Trama de comunicación



- Idle: No hay transferencia de datos (TxD y RxD) <Logic 1>
- St: Start bit (bit de arranque) <Logic 0>
- (n): Bits de datos
- P: Parity bit (bit de paridad)

Hardware

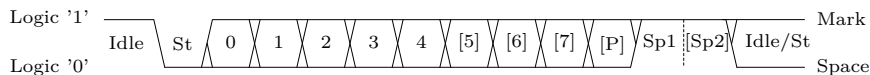
Trama de comunicación



- Idle: No hay transferencia de datos (TxD y RxD) <Logic 1>
- St: Start bit (bit de arranque) <Logic 0>
- (n): Bits de datos
- P: Parity bit (bit de paridad)
- Sp: Stop bit (bit de parada) <Logic 1>

Hardware

Trama de comunicación

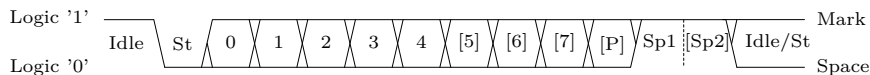


- Idle: No hay transferencia de datos (TxD y RxD) <Logic 1>
- St: Start bit (bit de arranque) <Logic 0>
- (n): Bits de datos
- P: Parity bit (bit de paridad)
- Sp: Stop bit (bit de parada) <Logic 1>

La comunicación es asíncrona (Bits menos significativo –LSb– se envía primero)

Hardware

Trama de comunicación



- Idle: No hay transferencia de datos (TxD y RxD) <Logic 1>
- St: Start bit (bit de arranque) <Logic 0>
- (n): Bits de datos
- P: Parity bit (bit de paridad)
- Sp: Stop bit (bit de parada) <Logic 1>

La comunicación es asíncrona (Bits menos significativo –LSb– se envía primero)

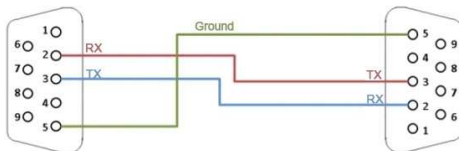
Algunas alternativas de tramas

- 8N1: 8 bits de datos, sin (N) bit de paridad, y 1 bit de stop.
- 5N2: 5 bits de datos, sin bit de paridad, y 2 bits de stop.

Hardware

Conexión null-módem

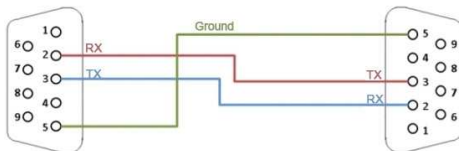
Cable Null-Modem Simple



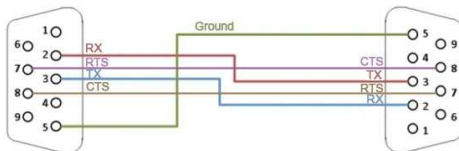
Hardware

Conexión null-módem

Cable Null-Modem Simple

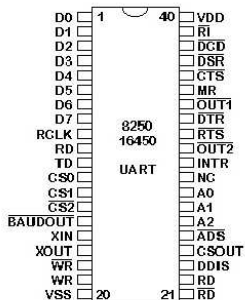
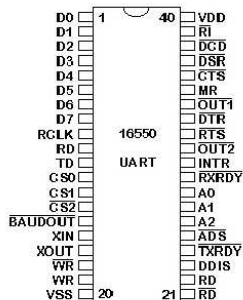


Cable Null-Modem con Handshake



Hardware

UART, Universidad Asynchronous Receive-Transmitter



Hardware

Asignación de pines de la UART

Nro. Pin	Nombre	Descripción
1:8	D0:D7	Bus de datos
9	RCLK	Entrada de reloj ($f = 16$ baudrate)
10	RD	Recepción de datos
11	TD	Transmisión de datos
12	CS0	Chip Select 0
13	CS1	Chip Select 1
14	nCS2	Chip Select 2
15	nBAUDOUT	Salida BaudRate programable
16	XIN	Entrada cristal externo
17	XOUT	Salida cristal externo
18	nWR	Línea de escritura (invertida)
19	WR	Línea de escritura
20	VSS	Masa general
21	RD	Línea de entrada
22	nRD	Línea de entrada (invertida)
23	DDIS	Driver Disable
24	nTXRDY	Transmisor listo
25	nADS	Address Strobe
26:28	A0:A2	Bits de dirección

Hardware

Asignación de pines de la UART

Nro. Pin	Nombre	Notas
29	nRXRDY	Receptor listo
30	INTR	Salida Interrupción
31	nOUT2	Salida de usuario 2
32	nRTS	Request To Send
33	nDTR	Data Terminal Ready
34	nOUT1	Salida de usuario 1
35	MR	Master Reset
36	nCTS	Clear To Send
37	nDSR	Data Set Ready
38	nDCD	Data Carrier Detect
39	nRI	Ring Indicator
40	VDD	Alimentación (+5V)

Hardware

Dirección de puertos y registros

Direcciones de los puertos

- **0x3F8**: Puerto serie 1 (COM1)
- **0x2F8**: Puerto serie 2 (COM2)
- **0x3E8**: Puerto serie 3 (COM3)
- **0x2E8**: Puerto serie 4 (COM4)

Hardware

Dirección de puertos y registros

Direcciones de los puertos

- **0x3F8**: Puerto serie 1 (COM1)
- **0x2F8**: Puerto serie 2 (COM2)
- **0x3E8**: Puerto serie 3 (COM3)
- **0x2E8**: Puerto serie 4 (COM4)

Los **Registros** van desde **BASE+0** hasta **BASE+7**

BASE+	DLAB	R/W	Arb.	Nombre
+0	0	W	-	Transmitter Holding Buffer
	0	R	-	Receiver Buffer
	1	R/W	-	Divisor Latch Low Byte (DLLB)
+1	0	R/W	IER	Interrupt Enable Register
	1	R/W	-	Divisor Latch High Byte (DLHB)
+2	-	R	IIR	Interrupt Identification Register
	-	W	FCR	FIFO Control Register
+3	-	R/W	LCR	Line Control Register
+4	-	R/W	MCR	Modem Control Register
+5	-	R	LSR	Line Status Register
+6	-	R	MSR	Modem Status Register
+7	-	R/W	-	Scratch Register

Contenido

1 Introducción

2 Hardware

3 Registros

- Bit DLAB, y velocidad de comunicación
- Line Control Register (LCR)
- Otros registros

4 Programación

5 Algunos ejemplos

6 Bibliografía

Registros

Bit DLAB, y velocidad de comunicación

DLAB: **D**ivision **L**atch **A**ccess **B**it

- Permite tener 12 registros en 8 direcciones
- Cuando **DBAL=1** a través del **Line Control Register** se tiene acceso a dos registros que permite fijar la velocidad de comunicación (bps)

Registros

Bit DLAB, y velocidad de comunicación

DLAB: Division Latch Access Bit

- Permite tener 12 registros en 8 direcciones
- Cuando **DBAL=1** a través del **Line Control Register** se tiene acceso a dos registros que permite fijar la velocidad de comunicación (bps)

La UART (oscilador de 1.8432MHz)

- Realiza una división por 16 \implies velocidad máxima de 115.200Hz (115.200bps)
- Para velocidades menores cuenta con un **Generador de Baud Rate Programmable** controlado por dos registros

Registros

Bit DLAB, y velocidad de comunicación

DLAB: Division Latch Access Bit

- Permite tener 12 registros en 8 direcciones
- Cuando **DBAL=1** a través del **Line Control Register** se tiene acceso a dos registros que permite fijar la velocidad de comunicación (bps)

La UART (oscilador de 1.8432MHz)

- Realiza una división por 16 \implies velocidad máxima de 115.200Hz (115.200bps)
- Para velocidades menores cuenta con un **Generador de Baud Rate Programmable** controlado por dos registros

Velocidad (BPS)	Divisor (dec)	DLHB	DLLB
50	2304	0x09	0x00
300	384	0x01	0x80
600	192	0x00	0xC0
2400	48	0x00	0x30
4800	24	0x00	0x18
9600	12	0x00	0x0C
19200	6	0x00	0x06
38400	3	0x00	0x03
57600	2	0x00	0x02
115200	1	0x00	0x01

Registros

Line Control Register (LCR)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DLAB	SBE	PS2	PS1	PS0	LSB	WL1	WL0

- DLAB: Divisor Latch Access Bit
- SBE: Set Break Enable
- PS: Parity Select
 - ▶ xx0: No parity
 - ▶ 001: Odd parity
 - ▶ 011: Even parity
 - ▶ 101: High parity
 - ▶ 111: Low parity
- LSB: Length of Stop Bit (0: 1 stop bit, 1: 2 stop bit)
- WL: Word length
 - ▶ 00: 5 bits
 - ▶ 01: 6 bits
 - ▶ 10: 7 bits
 - ▶ 11: 8 bits

La descripción de los demás registro se puede ver en los documentos de bibliografía

Contenido

1 Introducción

2 Hardware

3 Registros

4 Programación

- Archivos de dispositivos – acceso de bajo nivel
- Llamadas al sistema para archivos de dispositivos
- Abrir y cerrar el puerto serie
- Escribir y leer datos
- Configuración del puerto serie (termios)
- Configuración de comunicación

5 Algunos ejemplos

6 Bibliografía

Programación

Archivos de dispositivos – acceso de bajo nivel

Dispositivos en sistemas UNIX

- Los dispositivos (p.e. puerto serie) se acceden mediante **archivos de dispositivos**, los mismo se encuentran en `/dev/`
- En GNU/Linux los puertos series son: `/dev/ttySx` ($x: 0, 1, 2, \dots$)

Dispositivos en sistemas UNIX

- Los dispositivos (p.e. puerto serie) se acceden mediante **archivos de dispositivos**, los mismo se encuentran en `/dev/`
- En GNU/Linux los puertos series son: `/dev/ttySx` (`x`: 0, 1, 2,...)

Manejo de archivos de dispositivos (`/dev/`) en Linux (llamadas al SO):

- `open()`: Abrir archivo o dispositivo

Dispositivos en sistemas UNIX

- Los dispositivos (p.e. puerto serie) se acceden mediante **archivos de dispositivos**, los mismo se encuentran en `/dev/`
- En GNU/Linux los puertos series son: `/dev/ttySx` (`x`: 0, 1, 2,...)

Manejo de archivos de dispositivos (`/dev/`) en Linux (llamadas al SO):

- `open()`: Abrir archivo o dispositivo
- `close()`: Cerrar archivo o dispositivo

Dispositivos en sistemas UNIX

- Los dispositivos (p.e. puerto serie) se acceden mediante **archivos de dispositivos**, los mismo se encuentran en `/dev/`
- En GNU/Linux los puertos series son: `/dev/ttySx` (`x`: 0, 1, 2,...)

Manejo de archivos de dispositivos (`/dev/`) en Linux (llamadas al SO):

- `open()`: Abrir archivo o dispositivo
- `close()`: Cerrar archivo o dispositivo
- `read()`: Leer archivo o dispositivo

Dispositivos en sistemas UNIX

- Los dispositivos (p.e. puerto serie) se acceden mediante **archivos de dispositivos**, los mismo se encuentran en `/dev/`
- En GNU/Linux los puertos series son: `/dev/ttySx` (`x`: 0, 1, 2,...)

Manejo de archivos de dispositivos (`/dev/`) en Linux (llamadas al SO):

- `open()`: Abrir archivo o dispositivo
- `close()`: Cerrar archivo o dispositivo
- `read()`: Leer archivo o dispositivo
- `write()`: Escribir archivo o dispositivo

Dispositivos en sistemas UNIX

- Los dispositivos (p.e. puerto serie) se acceden mediante **archivos de dispositivos**, los mismo se encuentran en `/dev/`
- En GNU/Linux los puertos series son: `/dev/ttySx` (`x`: 0, 1, 2,...)

Manejo de archivos de dispositivos (`/dev/`) en Linux (llamadas al SO):

- `open()`: Abrir archivo o dispositivo
- `close()`: Cerrar archivo o dispositivo
- `read()`: Leer archivo o dispositivo
- `write()`: Escribir archivo o dispositivo
- `ioctl()`: Intercambiar información de control con el driver

Programación

Llamadas al sistema para archivos de dispositivos

Abrir archivo

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

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

- **path**: nombre del archivo o dispositivo
- **oflags**: indica acciones al abrir el archivo (O_RDONLY, O_WRONLY, O_RDWR)

Devuelve el *descriptor de archivo* (entero no negativo) si tuvo éxito, o -1 si falló.

Programación

Llamadas al sistema para archivos de dispositivos

Abrir archivo

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

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

- **path**: nombre del archivo o dispositivo
- **oflags**: indica acciones al abrir el archivo (O_RDONLY, O_WRONLY, O_RDWR)

Devuelve el *descriptor de archivo* (entero no negativo) si tuvo éxito, o -1 si falló.

Cerrar archivo

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

int close(int fildes);
```

- **fildes**: descriptor de archivo

Programación

Llamadas al sistema para archivos de dispositivos

Leer archivo

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

Escribir archivo

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

I/O control

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

- **buf**: Buffer para la lectura/escritura
- **nbytes**: Cantindad de bytes a leer/escribir
- **cmd**: Acción a realizar sobre el archivo

Programación

Abrir y cerrar el puerto serie

```
#include <stdio.h>      /* Standard input/output definitions */
#include <fcntl.h>      /* File control definitions */
#include <unistd.h>     /* UNIX standard function definitions */

int main(void)
{
    int fd;             /* File descriptor for the port */

    fd = open("/dev/ttyUSB0", O_RDWR | O_NOCTTY | O_NDELAY);

    if(fd == -1)
        /* ERROR */

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

Flags:

- O_RDWR: Lectura/Escritura
- O_NOCTTY: Para que no sea una terminal de control (?)
- O_NDELAY: Ignora la señal DCD (sino, el proceso se duerme hasta activarse DCD)

Programación

Escribir y leer datos

```
#include <stdio.h>      /* Standard input/output definitions */
#include <fcntl.h>      /* File control definitions */
#include <unistd.h>     /* UNIX standard function definitions */

int main(void)
{
    int n, fd;
    char* data;

    fd = open("/dev/ttyUSB0", O_RDWR | O_NOCTTY | O_NDELAY);
    if(fd == -1)
        /* ERROR */

    /* Writing data to the port */
    n = write(fd, "ATZ\n", 4);
    if(n < 0)
        /* ERROR */

    /* Reading data from the port */
    fcntl(fd, F_SETFL, 0);          /* Blocing */
    fcntl(fd, F_SETFL, FNDELAY);   /* Not blocking */
    /* n = read(fid, data, 4); */
    /* if(n < 4) */

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

Programación

Configuración del puerto serie (termios)

Terminales en sistemas POSIX (Portable Operating System Interface (UNIX))

- `termios.h`: estructura de control de terminal y funciones de control
- Cambiar parámetros tales como velocidad de comunicación, tamaño trama, etc.

Las dos funciones más importantes son: `tcgetattr()`, `tcsetattr()`

Miembros de la [estructura termios](#):

- `c_cflags`: Opciones de control
- `c_lflags`: Opciones de línea
- `c_iflags`: Opciones de entrada
- `c_oflags`: Opciones de salida
- `c_cc`: Caracteres de control
- `c_ispeed`: Baudrate de entrada (interfaz nueva)
- `c_ospeed`: Baudrate de salida (interfaz nueva)

Programación

Configuración de comunicación

```
struct termios options;
/* Get the current options for the port */
tcgetattr(fd, &options);

/* Set the baud rates to 19200 */
cfsetispeed(&options, B19200);
cfsetospeed(&options, B19200);

/* Enable the receiver and set local mode */
options.c_cflag |= (CLOCAL | CREAD);

/* No parity 8N1 */
options.c_cflag &= ~PARENB; /* No parity */
options.c_cflag &= ~CSTOPB; /* 1 stop bit */
options.c_cflag &= ~CSIZE; /* Mask the character size bits */
options.c_cflag |= CS8;

/* Set the new options for the port */
tcsetattr(fd, TCSANOW, &options);
```

Contenido

- 1 Introducción
- 2 Hardware
- 3 Registros
- 4 Programación
- 5 Algunos ejemplos**
- 6 Bibliografía

Algunos ejemplos

Algunos ejemplos

- **stty**: ver y modificar configuración de terminal
 - ▶ Ver configuración: `$stty -a -F /dev/ttyUSBx`
 - ▶ Enviar: `$echo "Hola" > /dev/ttyUSBx`
 - ▶ Recibir: `$cat /dev/ttyUSBx`

Algunos ejemplos

- **stty**: ver y modificar configuración de terminal
 - ▶ Ver configuración: `$stty -a -F /dev/ttyUSBx`
 - ▶ Enviar: `$echo "Hola" > /dev/ttyUSBx`
 - ▶ Recibir: `$cat /dev/ttyUSBx`
- Uso de **socat** para generación de puertos “locales”
 - ▶ `$socat PTY,link=/tmp/ttyS0 PTY,link=/tmp/ttyS1`

Algunos ejemplos

- **stty**: ver y modificar configuración de terminal
 - ▶ Ver configuración: `$stty -a -F /dev/ttyUSBx`
 - ▶ Enviar: `$echo "Hola" > /dev/ttyUSBx`
 - ▶ Recibir: `$cat /dev/ttyUSBx`
- Uso de **socat** para generación de puertos “locales”
 - ▶ `$socat PTY,link=/tmp/ttyS0 PTY,link=/tmp/ttyS1`
- Algunas terminales: **screen**, **cutecom**, etc.

Algunos ejemplos

- **stty**: ver y modificar configuración de terminal
 - ▶ Ver configuración: `$stty -a -F /dev/ttyUSBx`
 - ▶ Enviar: `$echo "Hola" > /dev/ttyUSBx`
 - ▶ Recibir: `$cat /dev/ttyUSBx`
- Uso de **socat** para generación de puertos “locales”
 - ▶ `$socat PTY,link=/tmp/ttyS0 PTY,link=/tmp/ttyS1`
- Algunas terminales: **screen**, **cutecom**, etc.
- Comunicación PC↔Celular (**cutecom**, **Free USB Serial Terminal**)
 - ▶ Adaptador USB/serie (dispositivo ‘`/dev/ttyUSB0`’) de ambos extremos
 - ▶ USB OTG (On-The-Go)

Algunos ejemplos

- **stty**: ver y modificar configuración de terminal
 - ▶ Ver configuración: `$stty -a -F /dev/ttyUSBx`
 - ▶ Enviar: `$echo "Hola" > /dev/ttyUSBx`
 - ▶ Recibir: `$cat /dev/ttyUSBx`
- Uso de **socat** para generación de puertos “locales”
 - ▶ `$socat PTY,link=/tmp/ttyS0 PTY,link=/tmp/ttyS1`
- Algunas terminales: **screen**, **cutecom**, etc.
- Comunicación PC↔Celular (**cutecom**, **Free USB Serial Terminal**)
 - ▶ Adaptador USB/serie (dispositivo ‘`/dev/ttyUSB0`’) de ambos extremos
 - ▶ USB OTG (On-The-Go)
- Ejemplos de código fuente

Contenido

- 1 Introducción
- 2 Hardware
- 3 Registros
- 4 Programación
- 5 Algunos ejemplos
- 6 Bibliografía**

Bibliografía

- *Interfacing the Serial / RS232 Port*, V5.0 (web)
- *Serial Programming Guide for POSIX Operating Systems*, 5th Edition– 3rd Revision. (web)
- *Beginning Linux Programming*, 4th Edition, Neil Matthew, Richard Stones (ISBN: 978-0-470-14762-7) (Libro)
- *GNU C Library –Terminal modes–* ([link](#))