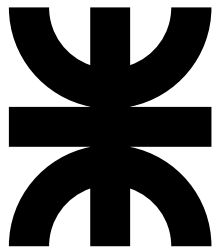
Programación en Linux embebido

Programación del puerto serie

Gonzalo F. Pérez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2017 –

Programación del puerto serie
Comunicación serie

Comunicación serie

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

Programación del puerto serie
Comunicación serie

Comunicación serie

- Se necesita conversor paralelo↔serie para "serializar/des-serializar" los datos
- Para la comunicación serie RS232 se utilizan las UARTs (Universal Asynchronous Reveiver-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, I^2C, SPI, etc.)

Programación del puerto serie
Programación

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`, `/dev/ttyUSBx`, `/dev/ttyACMx` (`x`: 0, 1, 2,...)

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`, `/dev/ttyUSBx`, `/dev/ttyACMx` (`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
- `ioctrl()`: Intercambiar información de control con el driver

Programación del puerto serie
Programación –Ejemplo–

```c
#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 del puerto serie
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 del puerto serie
Configuración del puerto serie (termios)

```
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);
```

Programación del puerto serie

Algunos ejemplos

Programación del puerto serie
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`

Programación del puerto serie
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`

Programación del puerto serie
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.

Programación del puerto serie
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.
- Ejemplo de código fuente de aplicación escritura de cadena