termiWin

~ A termios porting for Windows ~

Version 1.0

Introduction

termiWin is a library which purpose is to allow you to use on a Windows system, the same code used in Linux to communicate with a device through a serial port.

This is possible because termios's functions have been rewritten to be compatible with Windows's COM functions.

Library Architecture

The termios structure

This is the main structure of the library and it's often passed as argument to the functions, it has the following members:

- tcflag_t c_iflag; /*input modes*/
- tcflag_t c_oflag; /*output modes*/
- tcflag_t c_cflag; /*control modes*/
- tcflag_t c_lflag; /*local modes*/
- cc_t c_cc[NCCS]; /*special character*/

where tcflag_t is defined as an unsigned integer.

The members of termios structure are used to set and retrieve the serial port configuration parameters. There five types of flags, sorted by mode; they are implemented in the same way as they are in termios, except for some, which are not used.

Input modes flags

- **INPCK** – Not implemented, use PARENB instead.
- **IGNPAR** – Not implemented, disable PARENB instead.
- **PARMRK** – Not implemented, use PARENB instead.
- **ISTRIP** – Not implemented, use CS7 instead.
- **IGNBRK** – Not implemented.
- **IGNCR** – Not implemented.
- **ICRNL** – Not implemented.
- **INLCR** – Not implemented.
- **IXOFF** - If this bit is set, start/stop control on input is enabled. In other words, the computer sends STOP and START characters as necessary to prevent input from coming in faster than programs are reading it. The idea is that the actual terminal hardware that is generating the input data responds to a STOP character by suspending transmission, and to a START character by resuming transmission.
- **IXON** - If this bit is set, start/stop control on output is enabled. In other words, if the computer receives a STOP character, it suspends output until a START character is received. In this case, the STOP and START characters are never passed to the application program. If this bit is not set, then START and STOP can be read as ordinary characters.

Local modes flags

Since there’s no way to implement them in Windows, they have asbolutely no effect using termiWin, but you can keep the same you use in Linux for compatibilty.

Control modes flags

- **CSTOPB** - If this bit is set, two stop bits are used. Otherwise, only one stop bit is used.
- **PARENB** - If this bit is set, generation and detection of a parity bit are enabled.
- **PARODD** - This bit is only useful if PARENB is set. If PARODD is set, odd parity is used, otherwise even parity is used.
- **CSIZE** - This is a mask for the number of bits per character.

- **CS5** - This specifies five bits per byte.
- **CS6 –** This specifies six bits per byte.
- **CS7** – This specifies seven bits per byte.
- **CS8** – This specifies eight bits per byte.

Output modes flags

Since there’s no way to implement them in Windows, they have asbolutely no effect using termiWin, but you can keep the same you use in Linux for compatibilty.

Special character array

- **VEOF** – Is the EOF character to be used during the communication.
- **VEOL –** Not implemented.
- **VERASE –** Not implemented.
- **VINTR** – Interrupt character.
- **VKILL** – Not implemented.
- **VMIN** – If set to 0, the port is set to not-blocking, otherwise to blocking.
- **VQUIT –** Not implemented.
- **VSTART** – Not implemented.
- **VSTOP** – Not implemented.
- **VSUSP –** Not implemented.
- **VTIME** – Timeout for reading operations when COM is set to not-blocking.

Serial configurations functions

- Int **tcgetattr**(int fd, struct termios *termios_p);

  Sets in the internal DCB structures the current serial port parameters, it always has to be invoked before using tcsetattr. Returns 0 if succeded, otherwise -1.

- int **tcsetattr**(int fd, int optional_actions, struct termios *termios_p);

  Reads the flags set in the termios structure and sets the properly

  parameters in the DCB structure and eventually it applies the parameters to the serial port. Returns 0 if succeded, otherwise -1.

- int **tcsendbreak**(int fd, int duration);

  Sends a break character to the serial port; duration is not implemented.

  Returns 0 if succeded, otherwise -1.

- int **tcdrain**(int fd);

  Waits until all output written to the serial port has been transmitted.

  Returns 0 if succeded, otherwise -1.

- Int **tcflush**(int fd, int queue_selector);

  Discards data on serial port. queue_selector can assume the following values:

▪ **TCIFLUSH**	(discards data received but still not read).
▪ **TCOFLUSH**	(discards data written but still not transmitted),
▪ **TCIOFLUSH**	(discards both data received but still not read and data written but still not transmitted).

  Returns 0 if succeded, otherwise -1.

- Int **tcflow**(int fd, int action);

  Suspends transmission or receptions of data on serial port based on action. Action can assume the following values:

    - **TCOON** restarts suspended output.
    - **TCIOFF** transmits a STOP character.
    - **TCION** transmits a START character.
    - **TCOOF** suspends output.

  Returns 0 if succeded, otherwise -1.

- Void **cfmakeraw**(struct termios *termios_p);

  Sets but doesn't commit the following options for the serial port:

    - Charset: 8 bits
    - StopBits: one stop bit
    - Parity: no parity

  Use tcsetattr to commit them.

- speed_t **cfgetispeed**(const struct termios *termios_p);

  returns the input speed, speed can assume the same values of termios (B9600, B115200, ...).

- speed_t **cfgetospeed**(const struct termios *termios_p);

  returns the output speed, speed can assume the same values of termios (B9600, B115200, ...).

- int **cfsetispeed**(struct termios *termios_p, speed_t speed);

  Sets, but doesn't commits the parameter of speed for the serial port (in Windows there's no distinction between input/output/control), speed can assume the same values of termios (B9600, B115200, ...).
  Returns 0 if succeded, otherwise -1.

- int **cfsetospeed**(struct termios *termios_p, speed_t speed);

  Sets, but doesn’t commits the parameter of speed for the serial port (in Windows there’s no distinction between input/output/control), speed can assume the same values of termios (B9600, B115200, ...). Returns 0 if succeded, otherwise -1.

- int **cfsetsspeed**(struct termios *termios_p, speed_t speed);

  Sets, but doesn’t commits the parameter of speed for the serial port (in Windows there’s no distinction between input/output/control), speed can assume the same values of termios (B9600, B115200, ...). Returns 0 if succeded, otherwise -1.

Serial transmission/receiving - open/close Functions

- Int **openSerial**(char* portname, int opt);

  Open the serial port which name is *portname* with opt set for the port to be read only, write only or both read/write (**O_RDONLY, O_WRONLY, O_RDWR**). Returns the file descriptor (fd is actually useless in Windows with serial ports, but is set for compatibilty). **The function can be called using open instead of openSerial** (for termios compatibilty).

- Int **closeSerial**(int fd);

  Closes the serial port. Returns 0 if succeded, otherwise -1. **The function can be called using close instead of closeSerial** (for termios compatibilty).

- Int **writeToSerial**(int fd, char* buffer, int count);

  Writes to serial “count” characters contained in buffer. Returns the number of transmitted characters or -1 if transmission failed. **The**

function can be called using write instead of writeToSerial (for termios compatibilty).

- Int **readFromSerial**(int fd, char* buffer, int count);

  Reads “count” bytes from serial port and put them into buffer. Returns the number of read bytes or -1 if read failed. **The function can be called using read instead of readFromSerial** (for termios compatibilty).

- Int **selectSerial**(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);

  It behaves as termios select. Returns the file descriptor ready for the chosen operation or -1 if failed. **The function can be called using select instead of selectSerial** (for termios compatibility).

License