```c
/*
  libxbee - a C library to aid the use of Digi's Series 1 XBee modules
            running in API mode (AP=2).

  Copyright (C) 2009  Attie Grande (attie@attie.co.uk)

  This program is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/

/* ####################################################################### */
/* ### Linux Code ######################################################## */
/* ####################################################################### */

/*  this file contains code that is used by Linux ONLY */
#ifndef __GNUC__
#error "This file should only be used on a Linux system"
#endif

#include "linux.h"

int init_serial(xbee_hnd xbee, int baudrate) {
  struct flock fl;
  struct termios tc;
  speed_t chosenbaud;

  /* select the baud rate */
  switch (baudrate) {
  case 1200:  chosenbaud = B1200;   break;
  case 2400:  chosenbaud = B2400;   break;
  case 4800:  chosenbaud = B4800;   break;
  case 9600:  chosenbaud = B9600;   break;
  case 19200: chosenbaud = B19200;  break;
  case 38400: chosenbaud = B38400;  break;
  case 57600: chosenbaud = B57600;  break;
  case 115200:chosenbaud = B115200; break;
  default:
    fprintf(stderr,"%s(): Unknown or incompatiable baud rate specified... (%d)\n",__FUNCTION__,baudrate);
    return -1;
  };

  /* open the serial port as a file descriptor */
  if ((xbee->ttyfd = open(xbee->path,O_RDWR | O_NOCTTY | O_NONBLOCK)) == -1) {
    perror("xbee_setup():open()");
    xbee_mutex_destroy(xbee->conmutex);
    xbee_mutex_destroy(xbee->pktmutex);
    xbee_mutex_destroy(xbee->sendmutex);
    Xfree(xbee->path);
    return -1;
  }

  /* lock the file */
  fl.l_type = F_WRLCK | F_RDLCK;
  fl.l_whence = SEEK_SET;
  fl.l_start = 0;
  fl.l_len = 0;
  fl.l_pid = getpid();
  if (fcntl(xbee->ttyfd, F_SETLK, &fl) == -1) {
    perror("xbee_setup():fcntl()");
    xbee_mutex_destroy(xbee->conmutex);
    xbee_mutex_destroy(xbee->pktmutex);
    xbee_mutex_destroy(xbee->sendmutex);
    Xfree(xbee->path);
    close(xbee->ttyfd);
    return -1;
  }

  /* open the serial port as a FILE* */
  if ((xbee->tty = fdopen(xbee->ttyfd,"r+")) == NULL) {
    perror("xbee_setup():fdopen()");
    xbee_mutex_destroy(xbee->conmutex);
    xbee_mutex_destroy(xbee->pktmutex);
    xbee_mutex_destroy(xbee->sendmutex);
    Xfree(xbee->path);
    close(xbee->ttyfd);
```

```
  86:     return -1;
  87:   }
  88:
  89:   /* flush the serial port */
  90:   fflush(xbee->tty);
  91:
  92:   /* disable buffering */
  93:   setvbuf(xbee->tty,NULL,_IONBF,BUFSIZ);
  94:
  95:   /* setup the baud rate and other io attributes */
  96:   tcgetattr(xbee->ttyfd, &tc);
  97:   /* input flags */
  98:   tc.c_iflag &= ~ IGNBRK;           /* enable ignoring break */
  99:   tc.c_iflag &= ~(IGNPAR | PARMRK); /* disable parity checks */
 100:   tc.c_iflag &= ~ INPCK;            /* disable parity checking */
 101:   tc.c_iflag &= ~ ISTRIP;           /* disable stripping 8th bit */
 102:   tc.c_iflag &= ~(INLCR | ICRNL);   /* disable translating NL <-> CR */
 103:   tc.c_iflag &= ~ IGNCR;            /* disable ignoring CR */
 104:   tc.c_iflag &= ~(IXON | IXOFF);    /* disable XON/XOFF flow control */
 105:   /* output flags */
 106:   tc.c_oflag &= ~ OPOST;            /* disable output processing */
 107:   tc.c_oflag &= ~(ONLCR | OCRNL);   /* disable translating NL <-> CR */
 108:   tc.c_oflag &= ~ OFILL;            /* disable fill characters */
 109:   /* control flags */
 110:   tc.c_cflag |=   CREAD;            /* enable reciever */
 111:   tc.c_cflag &= ~ PARENB;           /* disable parity */
 112:   tc.c_cflag &= ~ CSTOPB;           /* disable 2 stop bits */
 113:   tc.c_cflag &= ~ CSIZE;            /* remove size flag... */
 114:   tc.c_cflag |=   CS8;              /* ...enable 8 bit characters */
 115:   tc.c_cflag |=   HUPCL;            /* enable lower control lines on close - hang up */
 116:   /* local flags */
 117:   tc.c_lflag &= ~ ISIG;             /* disable generating signals */
 118:   tc.c_lflag &= ~ ICANON;           /* disable canonical mode - line by line */
 119:   tc.c_lflag &= ~ ECHO;             /* disable echoing characters */
 120:   tc.c_lflag &= ~ ECHONL;           /* ??? */
 121:   tc.c_lflag &= ~ NOFLSH;           /* disable flushing on SIGINT */
 122:   tc.c_lflag &= ~ IEXTEN;           /* disable input processing */
 123:   /* control characters */
 124:   memset(tc.c_cc,0,sizeof(tc.c_cc));
 125:   /* i/o rates */
 126:   cfsetspeed(&tc, chosenbaud);     /* set i/o baud rate */
 127:   tcsetattr(xbee->ttyfd, TCSANOW, &tc);
 128:   tcflow(xbee->ttyfd, TCOON|TCION); /* enable input & output transmission */
 129:
 130:   return 0;
 131: }
 132:
 133: static int xbee_select(xbee_hnd xbee, struct timeval *timeout) {
 134:   fd_set fds;
 135:
 136:   FD_ZERO(&fds);
 137:   FD_SET(xbee->ttyfd, &fds);
 138:
 139:   return select(xbee->ttyfd+1, &fds, NULL, NULL, timeout);
 140: }
 141:
 142: #define xbee_sem_wait(a) xbee_sem_wait2(&(a))
 143: static inline int xbee_sem_wait2(xbee_sem_t *sem) {
 144:   struct timespec to;
 145:   clock_gettime(CLOCK_REALTIME,&to);
 146:   to.tv_sec++;
 147:   return sem_timedwait(sem,&to);
 148: }
```