GSM 0710 tty multiplexor HOWTO

This line discipline implements the GSM 07.10 multiplexing protocol detailed in the following 3GPP document:

https://www.3gpp.org/ftp/Specs/archive/07_series/07.10/0710-720.zip

This document give some hints on how to use this driver with GPRS and 3G modems connected to a physical serial port.

How to use it

1. config initiator

1.1 initialize the modem in 0710 mux mode (usually AT+CMUX= command) through its serial port. Depending on the modem used, you can pass more or less parameters to this command.

1.2 switch the serial line to using the n_gsm line discipline by using TIOCSETD ioctl.

1.3 configure the mux using GSMIOC_GETCONF / GSMIOC_SETCONF ioctl.

1.4 obtain base gsmtty number for the used serial port.

Major parts of the initialization program : (a good starting point is util-linux-ng/sys-utils/ldattach.c):

```
#include <stdio.h>
#include <stdint.h>
#include <linux/gsmmux.h>
#include <linux/tty.h>
#define DEFAULT_SPEED B115200
#define SERIAL_PORT   /dev/ttyS0

      int ldisc = N_GSM0710;
      struct gsm_config c;
      struct termios configuration;
      uint32_t first;

      /* open the serial port connected to the modem */
      fd = open(SERIAL_PORT, O_RDWR | O_NOCTTY | O_NDELAY);

      /* configure the serial port : speed, flow control ... */

      /* send the AT commands to switch the modem to CMUX mode
         and check that it's successful (should return OK) */
      write(fd, "AT+CMUX=0\r", 10);

      /* experience showed that some modems need some time before
         being able to answer to the first MUX packet so a delay
         may be needed here in some case */
      sleep(3);

      /* use n_gsm line discipline */
      ioctl(fd, TIOCSETD, &ldisc);

      /* get n_gsm configuration */
      ioctl(fd, GSMIOC_GETCONF, &c);
      /* we are initiator and need encoding 0 (basic) */
      c.initiator = 1;
      c.encapsulation = 0;
      /* our modem defaults to a maximum size of 127 bytes */
      c.mru = 127;
      c.mtu = 127;
      /* set the new configuration */
      ioctl(fd, GSMIOC_SETCONF, &c);
      /* get first gsmtty device node */
      ioctl(fd, GSMIOC_GETFIRST, &first);
      printf("first muxed line: /dev/gsmtty%i\n", first);

      /* and wait for ever to keep the line discipline enabled */
      daemon(0,0);
      pause();
```

1.5 use these devices as plain serial ports.

for example, it's possible:

- and to use gnokii to send / receive SMS on ttygsm1
- to use ppp to establish a datalink on ttygsm2

1.6 first close all virtual ports before closing the physical port.

Note that after closing the physical port the modem is still in multiplexing mode. This may prevent a successful re-opening of the port later. To avoid this situation either reset the modem if your hardware allows that or send a disconnect command frame manually before initializing the multiplexing mode for the second time. The byte sequence for the disconnect command frame is:

```
0xf9, 0x03, 0xef, 0x03, 0xc3, 0x16, 0xf9.
```

2. config requester

2.1 receive string "AT+CMUX= command" through its serial port,initialize mux mode config

2.2 switch the serial line to using the n_gsm line discipline by using TIOCSETD ioctl.

2.3 configure the mux using GSMIOC_GETCONF / GSMIOC_SETCONF ioctl.

2.4 obtain base gsmtty number for the used serial port:

```
#include <stdio.h>
#include <stdint.h>
#include <linux/gsmmux.h>
#include <linux/tty.h>
#define DEFAULT_SPEED B115200
#define SERIAL_PORT   /dev/ttyS0

      int ldisc = N_GSM0710;
      struct gsm_config c;
      struct termios configuration;
      uint32_t first;

      /* open the serial port */
      fd = open(SERIAL_PORT, O_RDWR | O_NOCTTY | O_NDELAY);

      /* configure the serial port : speed, flow control ... */

      /* get serial data and check "AT+CMUX=command" parameter ... */

      /* use n_gsm line discipline */
      ioctl(fd, TIOCSETD, &ldisc);

      /* get n_gsm configuration */
      ioctl(fd, GSMIOC_GETCONF, &c);
      /* we are requester and need encoding 0 (basic) */
      c.initiator = 0;
      c.encapsulation = 0;
      /* our modem defaults to a maximum size of 127 bytes */
      c.mru = 127;
      c.mtu = 127;
      /* set the new configuration */
      ioctl(fd, GSMIOC_SETCONF, &c);
      /* get first gsmtty device node */
      ioctl(fd, GSMIOC_GETFIRST, &first);
      printf("first muxed line: /dev/gsmtty%i\n", first);

      /* and wait for ever to keep the line discipline enabled */
      daemon(0,0);
      pause();
```

Additional Documentation

More practical details on the protocol and how it's supported by industrial modems can be found in the following documents :

- http://www.telit.com/module/infopool/download.php?id=616
- http://www.u-blox.com/images/downloads/Product_Docs/LEON-G100-G200-MuxImplementation_ApplicationNote_%28GSM%20G1-CS-10002%29.pdf
- http://www.sierrawireless.com/Support/Downloads/AirPrime/WMP_Series/~/media/Support_Downloads/AirPrime/Application_notes/CMUX_Feature_ Rev004.ashx
- http://wm.sim.com/sim/News/photo/2010721161442.pdf

11-03-08 - Eric BÃ©nard - <eric@eukrea.com>