# First Lab Class

**Objective:** Exchange arrays of bytes between two computers using a serial cable and implement the establishment phase of the data link protocol.

**Steps:**

1. **Consider groups of 2 elements per workbench**

In each room of the NetLab (I320/I321) there are 6 workbenches, numbered from 1-6. Please consider groups of 2 elements per workbench. The group members and the workbench should remain the same across the semester.

2. **Select the correct computers**

Each workbench has a rack of four computers (tux1 – tux4), and a separate computer closer to the center of the room. In this work, we will use the serial port of this last computer and the one from tux3. To select tux3, you should press the number 3 on the KVM switch (room I320), or press Scroll Lock + Scroll Lock + 3 + Enter on the keyboard (room I321). These procedures and the username and password are written on the workbench desk.

3. **Ensure the serial connectivity between computers**

To check the connectivity, open GTKTerm on both computers of the workbench and write some characters. The text should appear on the other computer. If this test fails, please check if the selected serial port is the correct one and that the baud rate is the same on both computers. Please report if the problem persists.

**Note: the GTKTerm is only used to check the connectivity between the PCs. It must be closed while the data link protocol is running. When you are testing your programs make sure you do not have any GTKTerm instance running.**

4. **Use the serial port in non-canonical mode using the example code**

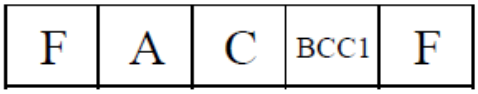
Please use the example code provided:

- writenoncanonical.c as the transmitter
- readnoncanonical.c as the receiver

You should compile the program using gcc and run the program on the terminal. Note that you should specify the serial port as an argument (/dev/ttySx, being x the index of the serial port and the default value 0). When running the example code, an array of “a” is sent through the serial port and printed on the receiver terminal.

5. **Implement the logic connection establishment**

Change the example code to send an array of bytes, defined in hexadecimal (declared as unsigned char), that performs the logic connection establishment. This phase considers the exchange of supervision frames that are composed by 5 bytes, starting with a flag, followed by an address field, a control field, a BCC and ending with a flag.




Each field, defined as unsigned char, has the following values:

FLAG: 01111110 (0x7E)

A (Address):

00000011 (0x03) – frames sent by the Sender or answers from the Receiver

00000001 (0x01) – frames sent by the Receiver or answers from the Sender

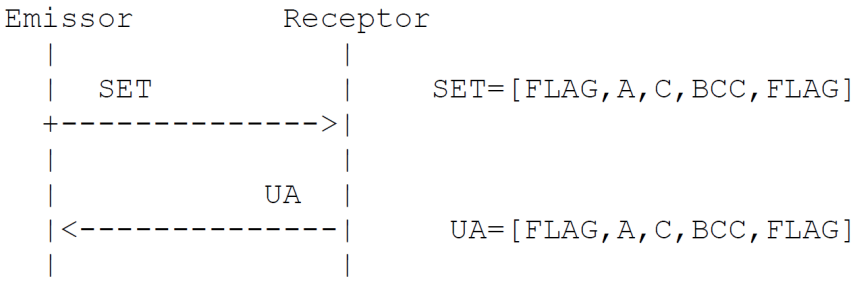
C (Control):

SET: 00000011 (0x03)

UA: 00000111 (0x07)

BCC1 (Block Check Character): XOR of all characters of the header (in this case A and C)

The establishment phase is completed once the transmitter sends a SET frame, the receiver answers with an UA frame after checking that the received SET frame is correct, and the transmitter correctly detects the UA frame.




For debugging purposes, you can print the hexadecimal values using:

*printf("var = 0x%02X\n", var);*

Or, if this does not work due to older versions of the compiler, this one:

*printf("var = 0x%02X\n", (unsigned int)(var & 0xFF));*

**Please report to the Professor once this step is completed.**

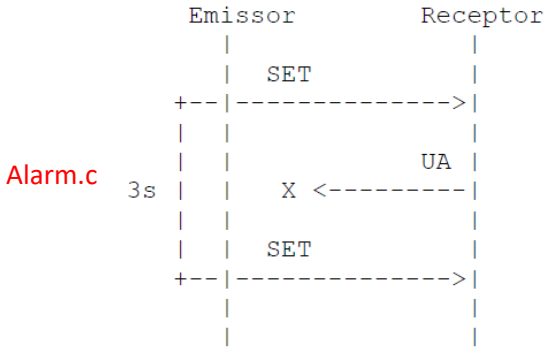
# Second Lab Class

**Objective:** Implement the retransmission mechanism and the state machine

**Steps:**

1. **Implement the retransmission mechanism**

Using the example code of alarm.c, implement a retransmission mechanism that retransmits the SET frame after a time-out is triggered (3 s for instance). Each frame can be retransmitted up to a maximum number of times (for instance, 3 times).




Alarm.c code available for download in Moodle. Here below some of the most important parts or commands to use:

```
void alarmHandler(int signal)          // user-defined function to handle alarms. May be changed. The
{                                      // important thing is that this function changes the value of an alarm
   alarmEnabled = FALSE;               // flag and that it increases the number of alarms
   alarmCount++;

   printf("Alarm #%d\n", alarmCount);
}

  (void)signal(SIGALRM, alarmHandler);        // install the OS function signal to be automatically
                                              // invoked when the timer expires, invoking in its turn
                                              // the user function alarmHandler


 alarm(t);            // activate alarm in t seconds

alarm(0);             // Disable pending alarms, if any
```

The alarm should be disabled - alarm(0) - once the UA frame is received.

The use of serial port in non canonical mode, enables the use of:

c_cc[VTIME] – inter-byte timeout

c_cc[VMIN] – minimum number of characters to be read

By modifying these values, different modes can be used:

- If MIN > 0 and TIME = 0 (blocking read), MIN sets the number of to receive before the read() function returns. As TIME is zero, the timer is not used.
- If MIN = 0 and TIME > 0 (read with timeout), TIME serves as a timeout value. The read() function will return if a single byte is received, or if TIME is exceeded (t = TIME *0.1 s). If TIME is exceeded, no byte will be returned.
- If MIN > 0 and TIME > 0 (read with inter-byte timeout), TIME serves as an inter-byte timeout. The read() function will return if MIN characters are received, the time between two bytes exceeds t = TIME * 0.1 s, or the number of requested bytes is received. The timer is restarted every time a byte is received and only becomes active after the first character has been received.
- If MIN = 0 and TIME = 0 (polling read), the read() function will return immediately, with either the number of bytes currently available in the receiver buffer, or the number of bytes requested.

You should set VTIME and VMIN accordingly.

2. **Implement the State Machine for receiving frames**

A state machine should be implemented to ensure that any frame is received correctly, in order, and without errors. In every situation in which a frame is being received/read, such a state machine needs to be used. Here below is an example of a state machine to receive the frame SET.