Project report

Afonso Pereira 201505870
David Cunha 201604317
João Loureiro 201604453

We have implemented a poll-based device driver as well as an interrupt driven one with all the features enumerated up to Lab 4. Furthermore, regarding the interrupt driven Device Driver, we have implemented 7 of the enhancements described in Lab 5: **1)** we have tried to minimized the use of global variables; **2)** it uses semaphores to eliminate race conditions; **3)** it is able to avoid block or busy waiting using the O_NONBLOCK flag; **4)** ioctl operations were added; **5)** it allows the user to interrupt a process inside a read syscall; **6)** access control was implemented in order to prevent more than one "user" to access the serial port; **7)**it supports the select/poll system call.
It is worth mentioning that, although the seri implementation allows access to 4 device drivers, only seri0 has updates regarding received and transmitted information, being the only one operational.

Enhancements:
3.1) The usage of global variables consists of 2 integers that hold the number of drivers possible and the Major number, and a pointer of type *struct seri_dev* which is the data structure each device has. This variable is then allocated in memory containing an array of 4 elements. This data structure holds several types need for the implementation of each device. Besides these variables, there is also a variable of type *struct file_operations* used to indicate which functions represent each operation. (seri.c, lines 41-568)

3.2) We eliminated race conditions using semaphores for the read and write system call. This is implemented using the variable *struct semaphore mutex*, which is a variable present in the *struct seri_dev* of each device. This variable is initialized with 1. (seri.c, line 249 and 309).

3.3) On the condition statement beginning in line 255 of the file seri.c we honour the O_NONBLOCK flag. This flag is only important in the read system call. Basically the if statement checks for the flag, and if so just returns from the function in order not to block. The file to test this is test/ononblockTest.

3.4) We created the function seri_ioctl, lines 155-218 of the file seri.c, which implements ioctl operations. The implemented operations are as follows: to get and set the bitrate, the number of information bits, parity and number of stop bits. In order to execute these operations, the message received and returned by the ioctl is composed of 24 bits with the format: DLM DLL LCR. This message is the decoded by either the user or device driver. The file to test this is test/ioctlTest. In this test the users is expected to call the function with the arguments: GET or SET BITRATE CHAR_WIDTH STOP_BITS PARITY.

3.5) In lines 262-268 of the file seri.c, the possibility for user to terminate reading is implemented. This implementation makes use of the function wait_event_interruptible (…) which returns the error -ERESTARTSYS if there was an interruption. This is then checked, and the system call returns if it was indeed the return of the function. The file to test this is test/readTest where the user simply needs to execute Ctrl-C.

3.6) By implementing the condition statement in line 129 of the file seri.c, we prevent more than one user to open the device. The variable *n_users* which is assigned a 0 at initialization, is checked to see if

it’s 1 or 0. 1 means that there is a user already using the device, otherwise the device is free to use. The file to test this is test/nusersTest which tries to open the same device twice.

3.8) Select/poll operations were added in function seri_poll (lines 220-238 of the file seri.c). this function simply checks if the size of the transmitter FIFO is full and if the size of the receiver FIFO is empty. It then returns a mask containing with information on whether the device is readable or writable. The file to test this is test/selectTest.