ENSC 351: Real-time and Embedded Systems

Craig Scratchley, Fall 2017

Multipart Project Part 3

Please continue working with a partner. My intention was to give you almost two weeks for this part of the multipart project.

In part 2 of the project you wrote code to both send and receive blocks and hooked in a simulator for a communication medium provided by a pair of telephone modems and the telephone system between them. The medium corrupted the data in some ways, but did not as configured inject spurious characters from simulated noise on the simulated telephone line. In particular, in the medium we had not enabled the ability for the medium to send an extra ACK character to the XMODEM sender while a block was being transmitted. We will change that in Part 3.

As I have discussed in class, before the XMODEM sender sends the last byte in a block, it should dump any "glitch" characters that may have arrived at the sender due to noise on the telephone line. It dumps characters by calling myReadcond() specifying that the read should timeout immediately. If any characters are available, they are read in and discarded. If no characters are available, the myReadcond() call times out immediately (it returns the value 0 immediately).

How does the XMODEM sender know that all previously written characters have actually been transmitted to the modem and that now is the time for it to dump glitch characters? Well, in the code, we are calling myTcdrain() for it to know. For a terminal (i.e. console) device like a serial port, tcdrain() blocks the calling thread until all previously written characters have actually been sent. However, tcdrain() cannot be used with sockets, which we are using in part to simulate serial ports. In fact, our myTcdrain() currently does nothing but return 0. See the following diagram from the Part 2 instructions:

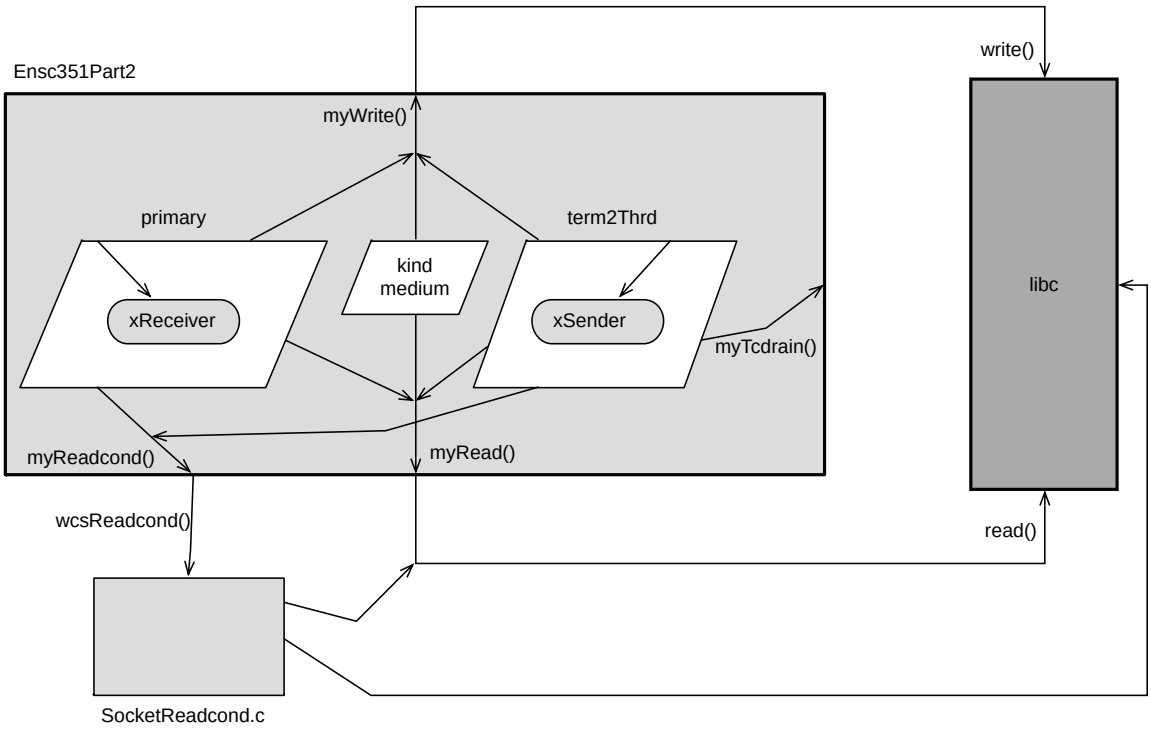

Figure 1: Collaboration Graph showing function calls for socketpair communication

You can enable the sending of extra ACKs in the medium by removing in Medium.cpp the comment on the line:

#define SEND_EXTRA_ACKS

If you try this, you will see that troubles currently arise transferring the file with the XMODEM protocol.

So, your job is to make modifications in myIO.* files to get myTcdrain() working, similar to how it works for terminal devices, but when using sockets in socketpairs for communication. I recommend that you use condition variables, as discussed in Chapter 4, in order to achieve this. For our immediate needs, we only have two threads using each socketpair. However, try to make your code as general as possible, where more than two threads might be using a socketpair. For example, thread A might write data to a socket and then call myTcdrain(), and then a thread B might write data to a socket and call myTcdrain(), and then a thread C might come along and read from the paired socket all the data written by both writing threads, and then a thread D might also come along wanting to read data from the paired socket. Both the writing threads should be notified, but presumably those threads should each be notified only once.

As long as there is memory available, the myIO.* files should work with as many descriptors as needed.

Please ask on Piazza if you have any questions or need any clarifications.

One suggestion: for a socket in a socketpair you can cast a call to myRead() into a call to myReadcond():

```
ssize_t myRead( int des, void* buf, size_t nbyte )
{
        … // maybe deal with descriptors for files
        // myRead (for our socketpairs) reads a minimum of 1 byte
        return myReadcond(des, buf, nbyte, 1, 0, 0);
}
```

That might mean putting the condition variable stuff in less places in the code.