LESSON 8 – Posix Semaphores

Question 1

This program implement a LAN based Program.
One station sends the packets and the rest receive them.
The Receiving stations are distinguished by the dest field.

Every Receiving station has a number of processes waiting for input, they are distinguished by the SID field.

All packets use the following structure and are at a fixed size.

SOF	
Dest [7:4]	Src[3:0]
Fid[7:0]	
Sid[7:0]	
Data[0-1023]	
.....	
.....	

SOF =0x10 – Start of Packet
Fid – Frame Id per Destinations Service ID
Sid – Service Id per Destination
Target – destination to which frame is sent
Size- number of data bytes to be sent
CRC – LSB of sum of all data bytes

Write a program, with the following specifications:

1. Use the main function to create many read threads and one write thread.
2. Use the file /home/$user_name/a.txt as though it was the LAN (the buffer which holds the packets) the read threads and write thread will use it accordingly.
3. The amount of read threads depends on the number of Processes needed to be serviced per station (use 2^8 = 256 read threads).
4. The read threads should run all concurrently (since they do not modify the file)
5. Every read thread should call the read function endlessly
    a. The read function will read all packets,
    b. Check if the SID matches the one it has been waiting for
    c. When the thread recognizes its packet it should
        i. check if there where frames lost per Destination SID. if it recognizes missing packets it should print a message
        ii. Process (PRINT ON SCREEN) the data sent for it.
6. The read threads do not modify the file.
7. Since we are really using a single station, we need a write thread which (implements SENDING new packets in an endless loop).

Guide lines

1. Write a header file for read and write functions and structure implentation, use the appropriate prototypes, the header file will be called io.h
2. Write the source file implementing the read and write functions. the file will be called io.c
3. Write the main function which creates and synchronizes the threads. the file will be called main.c
4. Write a Makefile for the program, use variables (for -ggdb and –lpthreads etc..)