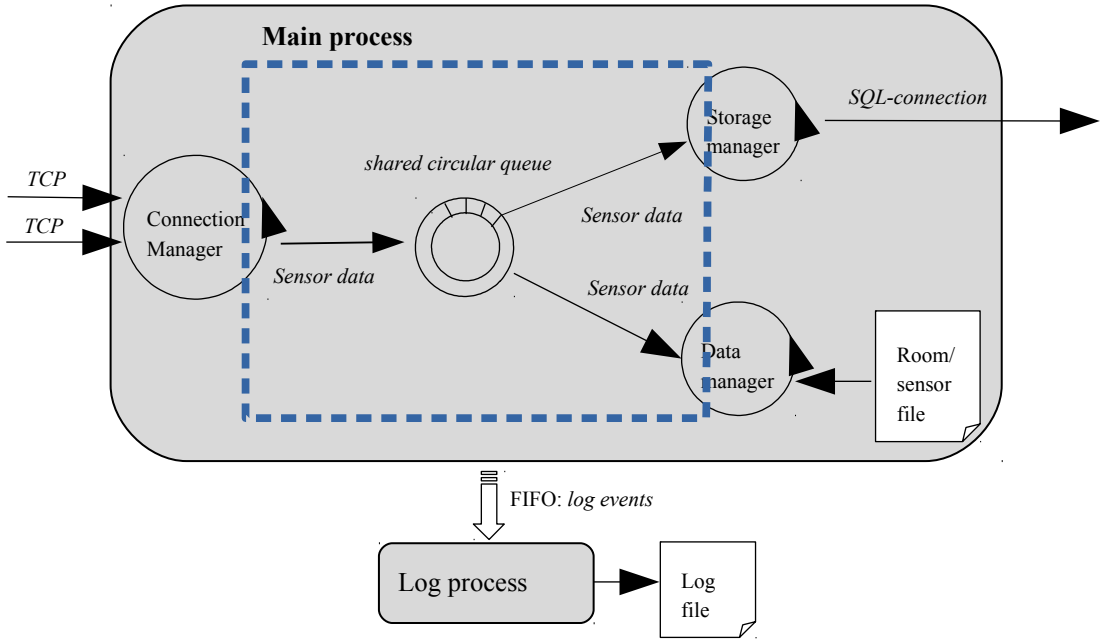
Lab 9 – Programming with Threads

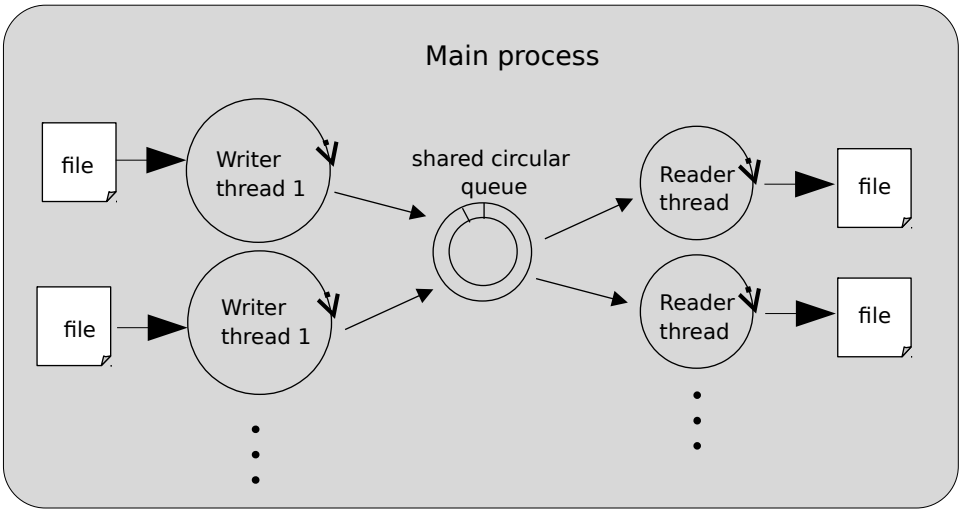
For your information

The picture below visually sketches the final assignment of this course. The relationship of this lab to the final assignment is indicated by the dashed blue line.



Exercise: A thread-safe circular queue

You already implemented a circular queue in lab 4. But this implementation is not safe when multiple processes or threads access (read, write,, update, …) the circular queue. Therefore, re-implement the circular queue operators such that multiple threads can access the data in a thread-safe way. Carefully think about an efficient data locking strategy (locking granularity). A simplistic implementation would lock the entire data structure for every operation, but is this really needed? For instance, is there a problem if two threads only 'read' data at the same time? Try to avoid that the execution of queue operations is in fact 'serialized' such that real concurrency is actually lost.



For testing your implementation, the following program should be implemented. The program starts up several reader and writer threads. A writer thread reads sensor data from file (see lab 6, 7, 8) and writes sensor date as one packet to the circular queue. The reader threads read sensor data packet by packet from the circular queue and write the data to file again.
The number of reader and writer threads that the program needs to run, should be given as a command-line argument.

THE SOLUTION OF THIS EXERCISE NEEDS TO BE UPLOADED ON TOLEDO BEFORE THE END OF NEXT WEEK.