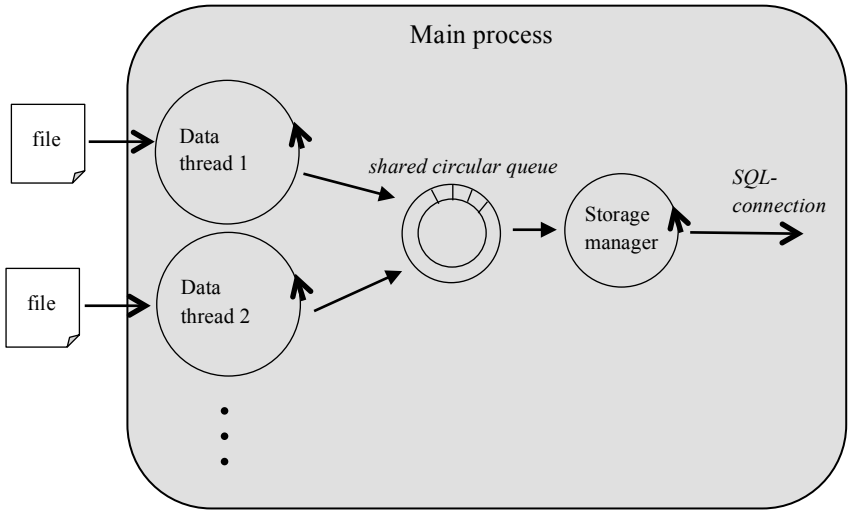
Lab 9 – Programming with Threads

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 data threads and one storage manager thread. A data thread reads sensor data from file (see exercise 1 of lab 7) and writes this date as one packet to the circular queue. The storage manager thread reads packet by packet from the circular queue and connects to an SQL database to store the sensor data (exercise 3 of lab 7).



The number of data threads that the program needs to run, should be given as a command-line argument. For each data thread there must a separate sensor data file. Assume that these files have names like 'sensor1.data', 'sensor2.data', etc. These sensor data files can be generated the same way as you did in exercise 1 of lab 7.