# MACHINE PROBLEM 6

**OBJECTIVE:** In this machine problem we will investigate kernel-level device drivers on top of a Simple programmed-I/O block device (programmed-I/O LBA disk controller).
**Bonus Option 3 and 4 are attempted.**

**OVERVIEW:**
The following functions needs to be implemented in blocking_disk.C

1) BlockingDisk() -> constructor for class BlockingDisk.
2) read() & write() -> instead of changing these functions will redefine the wait_until_ready function to give up the CPU.
3) Is_ready() -> will be redefined in the derived class blocking disk.

The following functions needs to be changed in scheduler.C(from MP5)

1) yield() -> updated to dispatch the blocked threads.

**IMPLEMENTATION:**
**Changes in blocking_disk.C and blocking_disk.H-**

1) Added a queue for blocked threads named BD_queue, the queue implementation is same as MP5 which was used for ready queue.
2) Queue size variable added named BD_queue_size.
3) Added  wait_until_ready function -> re-defined in BlockingDisk class, i didnt changed the blocking disk read and write function, instead changed the wait_until_ready function called inside the read and write functions. This function instead of busy waiting, checks if the disk is ready and if it is not it adds the current thread to the blocked queue and yields the CPU by calling the scheduler's yield function which dispatches the next thread from the FIFO scheduler.
4) is_ready() -> just calls the simple_disk is_ready() function.
5) add_queue function -> to add a thread to blocked queue.

**Changes in scheduler.C -**

1) Updated the yield function to check if there are any blocked threads and whether the disk is ready, if yes the blocked thread is dispatched to do read or write from the blocked queue otherwise if the disk is not ready or the blocked queue is empty, a thread is dispatched from ready queue of the scheduler.
2) Added a pointer of BlockingDisk type named BD.
3) Added function add_BD to assign a BlockingDisk object to BD pointer in scheduler class.

**Changes in kernel.C -**

1) Changed SYSTEM_DISK from a object of simple disk to blocking disk.
2) Added SYSTEM_DISK to SYSTEM_SCHEDULER using add_BD function.

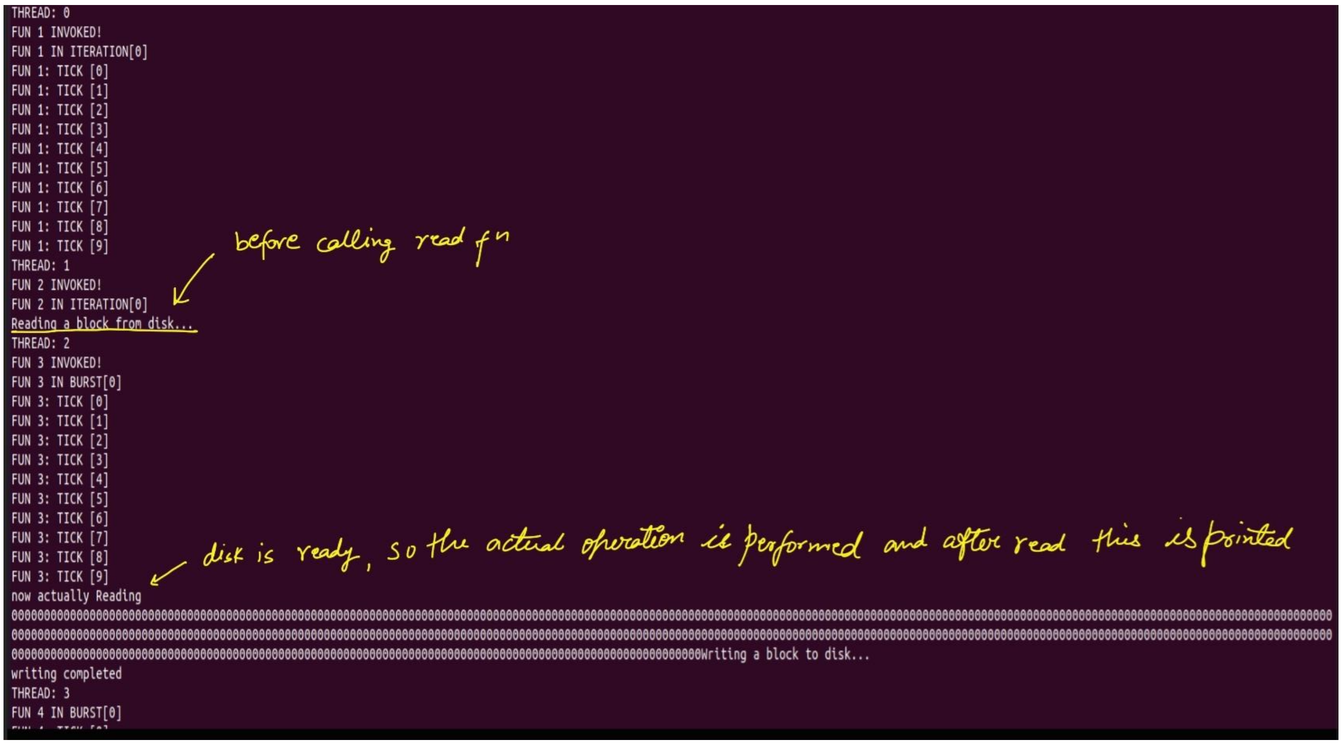
**queue.H is added from MP5 which is queue implementation.**
**Makefile is changed as queue.H and scheduler.C/H are added.**

**OPTION 3 and OPTION 4 explanation:**
We queue the read/write requests to a blocked queue in BlockingDisk class and make the thread requesting read/write to give up the CPU, next time when some thread yields the CPU using the yield function, it checks if there is a blocked read/write request in the blocked queue and whether the disk is

ready, if yes the thread from blocked queue is dispatched. In this way we have to only handle one read/write request in fifo order at a given time.

In our case only one thread can execute at a time because we have only one processor. In case of multiple processors multiple threads can try to access our queue, in that case we need to design a thread safe queue which can be used in a multi-threaded environment without any loss of data.

**TESTING:**
Testing is done using kernel.C, as shown in the below snapshot “Reading a block from disk” is printed before calling the read function in fun 2, in the simple disk read function it will call the wait_until_ready function which we have redefined in our derived class BlockingDisk, this function will check if disk is ready or not, if not ready it will add the thread in blocked queue and simply give up the CPU by calling the yield function. The yield function then checks if there is a blocked request in blocked queue and whether the disk is ready, if yes it is dispatched to CPU, this happens when “now actually reading” is printed which is printed after the read function completes and then it prints what it read, i changed the putch function to puti so that it prints 0 instead of nothing which makes it easy to debug.


before calling read fn
disk is ready, so the actual operation is performed and after read this is printed

```
THREAD: 0
FUN 1 INVOKED!
FUN 1 IN ITERATION[0]
FUN 1: TICK [0]
FUN 1: TICK [1]
FUN 1: TICK [2]
FUN 1: TICK [3]
FUN 1: TICK [4]
FUN 1: TICK [5]
FUN 1: TICK [6]
FUN 1: TICK [7]
FUN 1: TICK [8]
FUN 1: TICK [9]
THREAD: 1
FUN 2 INVOKED!
FUN 2 IN ITERATION[0]
Reading a block from disk...
THREAD: 2
FUN 3 INVOKED!
FUN 3 IN BURST[0]
FUN 3: TICK [0]
FUN 3: TICK [1]
FUN 3: TICK [2]
FUN 3: TICK [3]
FUN 3: TICK [4]
FUN 3: TICK [5]
FUN 3: TICK [6]
FUN 3: TICK [7]
FUN 3: TICK [8]
FUN 3: TICK [9]
now actually Reading
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000Writing a block to disk...
writing completed
THREAD: 3
FUN 4 IN BURST[0]
```