SIMPLE DISK DEVICE DRIVER

Operating systems

CSCE 611

FALL 2021

SIRI NAMBURI

In this machine problem, the aim is to implement a simple disk device driver on top of a I/O programmed device. Initially the code was implemented in a busy waiting loop, where the thread for reading and writing to the disk will wait in a busy loop until the read and write operations are completed. The target is to modify the code such that instead of being in the busy waiting loop , the cpu checks if the disk is ready and if it is not ready it should yield to other processes.

The basic code is implemented in this commit: **ebb807488bb4c454446cf238ba0331b291460ca6** where only the wait_until_ready function in blocking function is modified. Basically the code added checks if the disk is ready and if not it adds the disk thread to the end of the ready queue in the scheduler. That way other threads won't be obstructed. This works well enough as the other threads don't take much time before yielding. A version corresponding to bonus question 3 and 4 is implemented in the next commit. In this we maintain a separate queue for threads which utilize the disk and everytime the cpu yields the disk is checked whether it is ready or not. If the disk is ready, first the thread corresponding to blocking disk is implemented and then it is transferred to the next thread . Detailed explanation is given below.

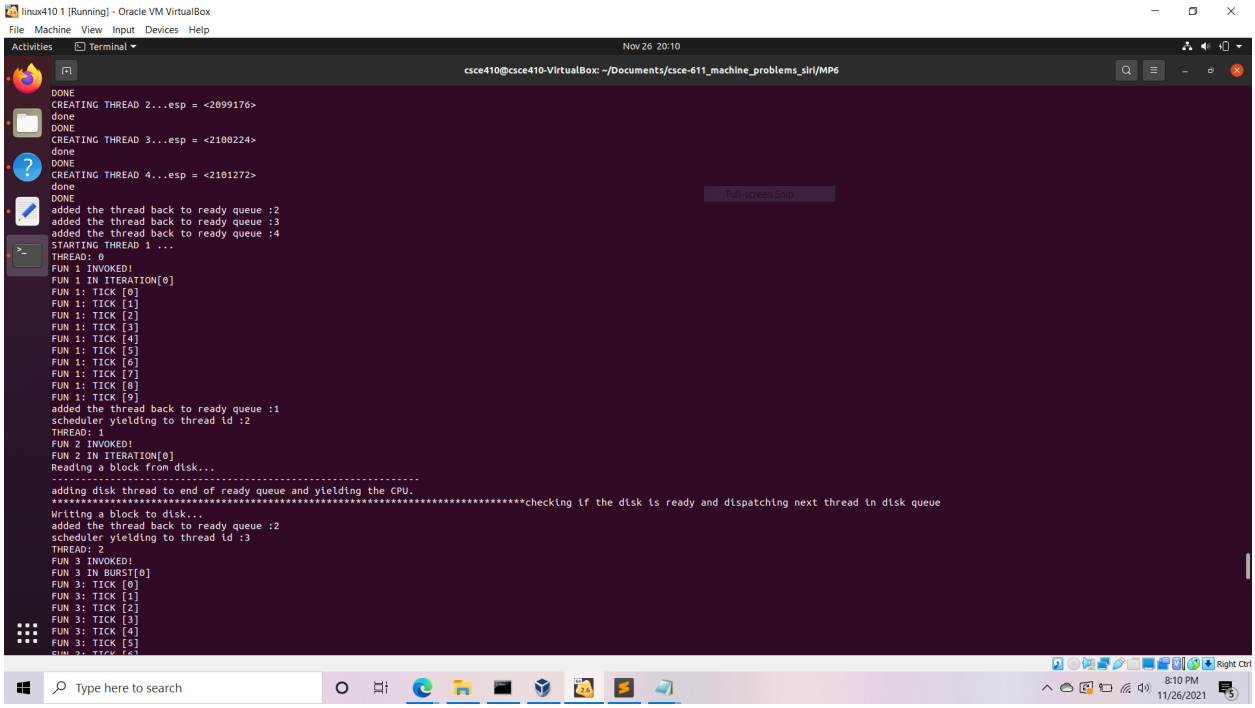
Option3 , design: In order to accommodate multiple threads which can raise a write or read request to the disk, a queue is created in the disk class itself to handle the threads corresponding to these requests. In this way, we can deal with them without concurrency without facing race conditions.

Option4: the code changes adopted (for the above design)
In scheduler class, an extra variable is added to maintain information about the blocking disk ,checking if the disk is ready, and yielding cpu if the disk is ready

In blocking disk class following functions are implemented/modified:

1. Add_disk_thread: takes in a new thread and adds it to the disk queue
2. Return_first_disk_thread : returns the first thread in the queue
3. Wait_until_ready: if the disk is not ready the thread is added to the disk queue and cpu is yielded.

As we can see from the screenshot below, the implemented code is working .


linux410 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Nov 26 20:10
csce410@csce410-VirtualBox: ~/Documents/csce-611_machine_problems_siri/MP6
DONE
CREATING THREAD 2...esp = <2099176>
done
DONE
CREATING THREAD 3...esp = <2100224>
done
DONE
CREATING THREAD 4...esp = <2101272>
done
DONE
added the thread back to ready queue :2
added the thread back to ready queue :3
added the thread back to ready queue :4
STARTING THREAD 1 ...
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]
added the thread back to ready queue :1
scheduler yielding to thread id :2
THREAD: 1
FUN 2 INVOKED!
FUN 2 IN ITERATION[0]
Reading a block from disk...
adding disk thread to end of ready queue and yielding the CPU.
checking if the disk is ready and dispatching next thread in disk queue
Writing a block to disk...
added the thread back to ready queue :2
scheduler yielding to thread id :3
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]
Right Ctrl
Type here to search
8:10 PM
11/26/2021