CSCE 611

Pranav Anand Taukari (UIN: 433003781)

MP6 Design Document

OBJECTIVE:

The objective of machine problem 6 is to add a layer on top of simple device driver to support the same blocking read and write operations as the basic implementation, but without busy waiting in the device driver code and bonus points for **options 1,2,3,4** have been done.

Bonus Options Implemented:

1. Basic Blocking disk implementation where threads aren't waiting until I/O is complete
2. [BONUS 1: IMPLEMENTED]: Mirroring Disk Implementation
3. [BONUS 2 : IMPLEMENTED] : Interrupt based Disk Handling
4. [BONUS 3: DESIGNED] : Thread safe system
5. [BONUS 4: IMPLMENTED]: Implemented a Thread Safe system.

Following are the files modified:

1. Kernel.C
2. Blocking_disk.C
3. Blocking_disk.H
4. Mirroring_disk.C
5. Mirroring_disk.H
6. makefile

1. Basic and design of [Bonus:3]

In this while designing the basic functionality of the Blocking disk system, measures were taken to make it a Thread Safe system, thus simultaneously satisfying problem statement 1, 3, 4.

To make the blocking disk we design a locking system using test and set.

When a thread calls a read or write operation it acquires the lock , this way in case any other function comes in to access the disk will not be allowed to until the previous thread releases the lock. In case if the disk is busy, we resume the thread by adding it to the back of the ready queue and yielding the CPU to next thread. This way we make sure that the thread is not busy waiting until the I/O operations are finished.

The read, write and is_ready methods are used from the simple disk as the blocking disk inherits the methods of simple disk.

2. Implementation of Mirroring Disk [Bonus:1]

I have implemented mirroring disk by which inherits from blocking disk wherein we write data to both master and dependent disk. For read we read from whichever disk is ready first.

Design Implementation:

1. Mirroring disk inherits blocking disk
2. It creates two instances of blocking disk for master and slave in the constructor.
3. When write is requested by the user, mirroring disk writes to both master and slave.
4. When read is requested by the user, mirroring disk issues command to master and slave disk and it wait for either of the disk to complete and gives the data to the user.

3. Implementation of Interrupt Based Disk Handling [Bonus:2]

I have implemented interrupts where the interrupt is invoked once the disk is ready and the data can be read/written to the disk.

Design Implementation:

1.Register the interrupt handler function
2.If the disk is not ready, push the thread into the blocking_queue and yield the CPU
3.Once the interrupt handler function is called, pop the thread from the local blocking queue and resume the thread.

4. Thread Safe System [Bonus:4]

Implemented the design as stated in 1 by defining a locking system based on test and set and then acquiring the lock before any read or write operation in blocking_disk.C

Note:

To test the code, do the following:

1. To test blocking disk, define BlockingDisk object in kernel.C
2. To test mirroing disk (**option 1**), define MirroringDisk object in kernel.C
3. To test interrupts feature (**option 2)**, uncomment INTERRUPTS_ENABLED macro in blocking_disk.H and kernel.C
4. To test thread safe feature,(**option 3 & 4)** define BlockingDisk object in kernel.C (Blocking disk implements thread safe system)