MP6: Primitive Disk Device Driver

The NonBlockingDisk class implements a non-blocking disk device driver for an operating system. It provides asynchronous disk operations by utilizing a queue to manage disk requests and a polling mechanism to handle disk I/O without blocking the calling thread.

nonblocking_disk.h

The header file for the NonBlockingDisk class defines the essential structure of the non-blocking disk driver, including private members for the disk state, and function declarations for disk operations.

NonBlockingDisk Class: The NonBlockingDisk class extends the SimpleDisk class to provide non-blocking disk I/O operations.

Private Members:

- DiskOperation Queue:
    - **DiskOperation**: Represents a disk operation (read/write) with associated metadata.
    - **head**: Pointer to the first operation in the queue.
    - **tail**: Pointer to the last operation in the queue.
    - **disk_busy**: Boolean flag indicating whether the disk is currently processing an operation.

Protected Members:

- **is_ready()**: Checks if the disk is ready to process the next operation. Overrides the base class function.

Public Members:

- Constructor: NonBlockingDisk(DISK_ID _disk_id, unsigned int _size).
- read(unsigned long _block_no, unsigned char* _buf): Initiates a non-blocking read operation.
- write(unsigned long _block_no, unsigned char* _buf): Initiates a non-blocking write operation.
- check_status(): Processes queued operations and updates the disk status.

nonblocking_disk.C

The implementation of the NonBlockingDisk class provides the functional logic for non-blocking disk operations. It defines the constructor, the polling mechanism for disk readiness, and the core functions for handling reads, writes, and queued operations.

DiskOperation Struct
Represents a single disk operation, including:

- **block_no:** The block number to read/write.
- **buffer:** The buffer for data transfer.
- **type:** The operation type (read or write).
- **thread:** Pointer to the thread that issued the operation.
- **next:** Pointer to the next operation in the queue.

Class Implementation

Static Initialization: The disk_busy flag is initialized to false in the constructor to indicate the disk is idle.

Constructor:Initializes the base SimpleDisk with the specified disk ID and size.
Sets the queue pointers (head and tail) to nullptr.

is_ready()

- Queries the disk controller to determine readiness.
- Returns true if the disk is ready to process the next operation.

read(unsigned long _block_no, unsigned char* _buf)

- Issues a read operation using the base class function.
- Polls the disk readiness and yields control to the scheduler while waiting.
- Transfers 512 bytes from the disk to the buffer once ready.

write(unsigned long _block_no, unsigned char* _buf)

- Issues a write operation using the base class function.
- Polls the disk readiness and yields control to the scheduler while waiting.
- Transfers 512 bytes from the buffer to the disk once ready.

check_status()

- Processes the next operation in the queue if the disk is idle.
- Resumes the thread that issued the operation once complete.

Thread Queue Implementation

In order to manage the threads for this machine problem, I have included my thread scheduler from MP5. It has a queue of the running threads and allows me to yield the cpu when necessary within the nonblocking_disk class.

Debugging

Debugging print statements are used in critical areas:

- Operation Queue: Logs operations being enqueued and dequeued for debugging disk operation management.
- Disk Status: Logs readiness status to ensure polling and readiness checks function correctly.
- Scheduler Integration: Verifies that threads yield and resume correctly during disk operations.