CSCE - 611

MP6: Primitive Disk Device Driver

DESIGN DOCUMENT

Submitted by: RAJENDRA SAHU (731008796)
github-repo : CSCE-410-611-fall-2021/rpsahu_CSCE611

OPTIONS ATTEMPTED

- **Option 1: Implementing a mirrored disk**.
- **Option 2: Design of a Thread Safe system:** Described in the design pdf later.

Uncomment the macro **_DISK_MIRRORING** in *kernel.C* to enable the disk mirroring(Option 1) & run the emulation.

INTRODUCTION:

This MP intends to replace the busy wait in SimpleDisk by a non-blocking wait method.

SOURCE CODES MODIFIED/ADDED:

1. *kernel.C: Creating* **_DISK_MIRRORING** *macro to selctively instantiate BlockingDisk vs MirroredDisk*
2. *scheduler.C: Included from previous MP*
3. *scheduler.H: Included from previous MP*
4. *simple_disk.H: Moved issue_operation() & is_ready() to protected so that they can be inherited from BlockingDisk & MirroredDisk*
5. *blocking_disk.H : Core Implementation ; derived from SimpleDisk*
6. *blocking_disk.C : Core Implementation*
7. *mirrored_disk.H : Core implementation for option 2; Derived from SimpleDisk*
8. *mirrored_disk.C: Core implementation for option 2*
9. *makefile: To include the compilation of mirrored_disk*

ABSTRACTION INTRODUCED: A structure *rw_request_s* as the datatype of the io request in *BlockingDisk*. A similar datatype mirr_request_s added in *MirroredDisk*.

CORE IMPLEMENTATION (Only the incremental additions/modifications have been mentioned, the rest of the code stays same as provided):

1. ***BlockingDisk.C constructor:***
   Allocates memory for the *request* variable; a placeholder of the request.
2. ***BlockingDisk::push_request():*** Stores all the incoming request relevant info in the *request* variable.
3. ***BlockingDisk::read():*** Overriding function over the SimpleDisk:: read. This function handles the incoming read commands. It does 3 things.
   a. save the request by calling *push_request().*
   b. Issues a read command by calling *SimpleDisk::issue_operation()*
   c. Call the non-blocking wait i.e. *nonblock_wait_and_process()*
4. ***BlockingDisk::write():*** This follows the same function flow of *read()* except is issues a write command.
5. ***BlockingDisk:: nonblock_wait_and_process():*** This function does the actual waiting & data transfer. It waits id the device is ready by calling **SimpleDisk::*is_ready()*** . If the device is ready then it proceeds to process the data transfer. If the device is not ready then the thread gets added to the ready queue and the CPU is

yielded to the next thread in the ready queue. This is how the thread doesn't get blocked on the io. The data transfer code has been borrowed from the *SimpleDisk* class.

OPTION 1 MirroredDisk :

This class is almost a replica of the SimpleDisk except for some mechanisms. This class gets derived from SimpleDisk to keep the inheritance flow simple.
These are the differences between MirroedDisk & BlockigDisk

1. It has 2 SimpleDisk instances (master & dependent).
2. Same command is issued twice to both the master & dependent.
3. The device ready wait is done on both of the drives(master & dependent) ; within the thread yield mechanism

OPTION 2 Thread Safe System DESIGN:

1. Create a request queue to store requests from multiple threads. Add provision to store the thread info too.
2. Enable interrupts
3. Implement a handler that will process the actual io transfer once device is ready.
4. Keep a mutex like mechanism (a pair of enable_interrupts & disable_interrupts) handy.
5. Add every new request to this queue.
6. Issue the commands & yield the thread. We need not wait for the device ready status; it will be interrupted.
7. Interrupt 14 got triggered & handler caught the interrupt. Process the io transfer
8. Pop the head io request that just got handled & push the owner thread to the ready queue again
9. Apply the mutex in both push & pop operations to synchronize the threads.

This way multiple threads can operate concurrently without having to bother about the requests being lost.

TESTING:

_USES_SCHEDULING has been uncommented to use the scheduler by default.
One observation from the BlockingDisk output is that the thread2 operates in half of the bandwidth. The thread2 iteration is half of the other 3 thread's burst number. This is because of the yielding of thread2.
Uncomment _DISK_MIRRORING to test the mirrored drive.

GENERIC INSTRUCTIONS:

This code has been developed on the new code environment.
The TA is expected to unzip the file and the run *make.*

CODE MAINTENANCE:

The entire source code base has been pushed to the student's github repository. This have been done progressively over the course of development. This should help the TA in grading the assignment. Apart from the code base provided in the zip file submitted on canvas, the code can also be compiled from the github repo code base.
solution to make the code complete.