MP6: Primitive Device Driver
Chonglin Zhang
UIN: 833003072
CSCE611: Operating System

Assigned Task:

The objective of this project is to enhance the basic disk driver by implementing read and write functions that operate without busy waiting, instead using blocking methods.

Design

Blocking Disk Driver Implementation

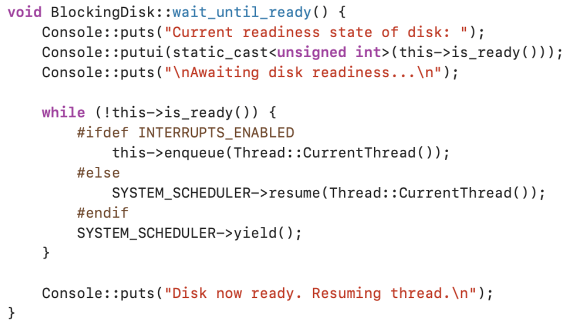
The goal of this project is to develop a blocking disk driver, derived from the SimpleDisk class, which facilitates read and write activities without engaging in busy waiting. Additionally, this disk driver efficiently utilizes the scheduler to yield CPU usage when the disk is unprepared, thereby permitting other threads to run.

In the BlockingDisk class, a new function wait_until_ready() is introduced to assess whether the driver is primed for disk activities. If it's not ready, the function cedes the CPU, facilitating the operation of other threads. The scheduler then puts the current thread on hold in a waiting queue, while it manages the execution of other threads. This waiting thread is intermittently reactivated by the scheduler to check if the driver has reached a state of readiness, continuing this routine until the driver is equipped for disk actions.

The BlockingDisk class, derived from the SimpleDisk class, retains the read, write, and is_ready methods, guaranteeing smooth execution of fundamental disk operations. Furthermore, it incorporates the scheduler and utilizes the wait_until_ready() method, providing an advanced and efficient approach. This integration effectively removes busy waiting, leading to an enhancement in the system's overall performance.

Implementation

blocking_disk.C: wait until ready(): In my implementation, I've adapted the method from SimpleDisk to assess whether the blocking disk is ready, and if not, to relinquish control of the CPU. This strategy effectively resolves the busy waiting problem inherent in the SimpleDisk. Moreover, it involves placing the thread into the main queue through the invocation of the resume method before yielding the CPU.



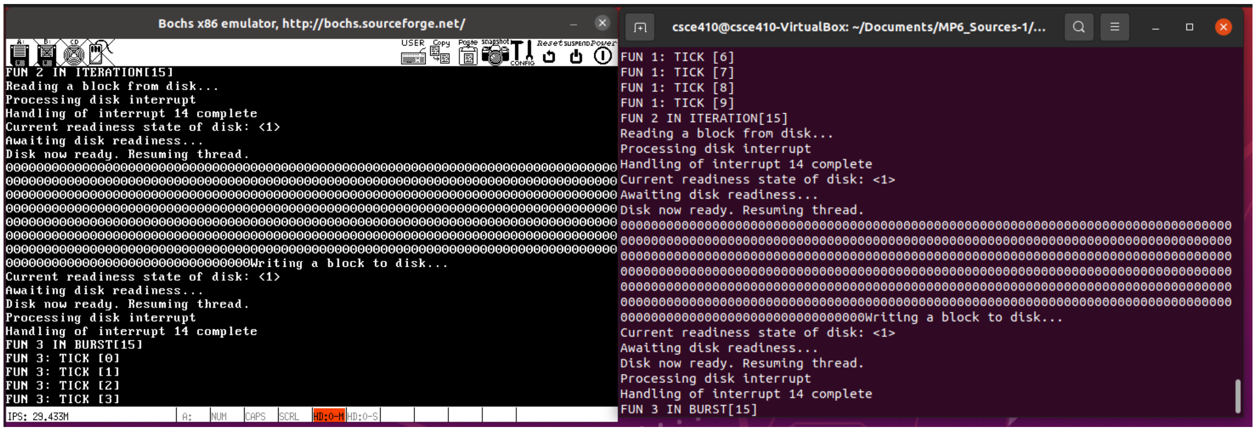
```
void BlockingDisk::wait_until_ready() {
    Console::puts("Current readiness state of disk: ");
    Console::putui(static_cast<unsigned int>(this->is_ready()));
    Console::puts("\nAwaiting disk readiness...\n");

    while (!this->is_ready()) {
        #ifdef INTERRUPTS_ENABLED
            this->enqueue(Thread::CurrentThread());
        #else
            SYSTEM_SCHEDULER->resume(Thread::CurrentThread());
        #endif
        SYSTEM_SCHEDULER->yield();
    }

    Console::puts("Disk now ready. Resuming thread.\n");
}
```

Testing

Ensure that the 'INTERRUPTS ENABLED' macro in scheduler.H is uncommented, and the 'ENABLE BLOCKING DISK' macro in kernel.C is uncommented, prior to initiating tests for this basic functionality. Below figure is the output for kernel and console:


Bochs x86 emulator, http://bochs.sourceforge.net/
A:
B:
CD
USER
Copy
Paste
snapshot
CONFIG
Reset
SUSPEND
Power
IPS: 29.433M
A:
NUM
CAPS
SCRL
HD:0-M
HD:0-S
csce410@csce410-VirtualBox: ~/Documents/MP6_Sources-1/...

```
FUN 2 IN ITERATION[15]
Reading a block from disk...
Processing disk interrupt
Handling of interrupt 14 complete
Current readiness state of disk: <1>
Awaiting disk readiness...
Disk now ready. Resuming thread.
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000Writing a block to disk...
Current readiness state of disk: <1>
Awaiting disk readiness...
Disk now ready. Resuming thread.
Processing disk interrupt
Handling of interrupt 14 complete
FUN 3 IN BURST[15]
FUN 3: TICK [0]
FUN 3: TICK [1]
FUN 3: TICK [2]
FUN 3: TICK [3]
```

```
FUN 1: TICK [6]
FUN 1: TICK [7]
FUN 1: TICK [8]
FUN 1: TICK [9]
FUN 2 IN ITERATION[15]
Reading a block from disk...
Processing disk interrupt
Handling of interrupt 14 complete
Current readiness state of disk: <1>
Awaiting disk readiness...
Disk now ready. Resuming thread.
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000Writing a block to disk...
Current readiness state of disk: <1>
Awaiting disk readiness...
Disk now ready. Resuming thread.
Processing disk interrupt
Handling of interrupt 14 complete
FUN 3 IN BURST[15]
```

Files Modified:

I modified some files for this project:

1. Kernel.C
2. scheduler.C/H: For this implementation, I utilized the same scheduler code from MP5 and introduced the 'ENABLE INTERRUPT' macro within the header field.
3. blocking disk.C/H: Implemented for the current project
4. thread.C: in start_thread method, allowing interrupt
5. makefile: add some compile statement for above file.