CSCE 611 Operating Systems Summer 2017

825008857 Chaiwei Chang

MP6

Design for disk I/O

Originally in simple_disk.C, it use the method of busy waiting that checking whether disk is available or not in the while loop. In this way, the CPU resource is block there waiting for this read/ write operation to execute. This is not desirable. However, we can write additional blocking_disk class that inherit simple disk to fix this problem. I simply override only method wait_until_ready, since read/write method all stuck here in the while loop to block the CPU. All it does is that it puts the thread in the waiting kernel scheduler's ready queue and passes the CPU onto another thread. This way, the CPU is not blocked. In other words, we just allow for multiprogramming in wait_until_ready. But I have comment out these lines of code since I was going for option3 and option4 which is design a thread save infrastructure. Also, I import my previous scheduler.C and thread.C to do the FIFO scheduling job. If multiple threads can access these items concurrently, how do we take care of race conditions. I add blockdisk pointer and an variable inside our kernel system_scheduler so that the scheduler can access the information about the disk. And then I create a ready_queue storing thread that want to wirte of read on the same disk at some conflict time. This is a FIFO queue, so who ever come first while be executed first. The latter one will wait inside the queue but not blocking the CPU since this is a non-blocking system in previous design. Overall, I implement 2 level schedulers, one is the kernel scheduler for dispatching threads which I consider outer level. The other one is inner level scheduler is will trigger a queue when a disk is being read or write. In order to test this, I modified the kernel.C to both thread2 and thread 3 will do the same things that thread 2 originally does which is read and write. Last but not least, I change putch to puti inside the kernel for debugging purpose and I file in some buff to the disk before. Furthermore, I did try to implement mirrordisk, but fail~~

Finally, I wrote some command on my code, but if grader of TA have any question about my logic feel free to contact me at hunkywei@tamu.edu