Gray Houston
CS 354
Dr. Park

Lab 3 Answers

In this lab, I implemented the Solaris Time Share (TS) Scheduler for the XINU operating system. For this implementation, I made a few design choices which made the implementation significantly easier. I decided that all process, would receive a priority of 29 when they were created. Because we do not know whether a process will be CPU-Intensive or I/O-Intensive when it is created, the process should receive a median priority and a median time slice size so that it not only has a chance to be able to run, but so that the scheduler can quickly determine the nature (CPU or I/O bound) of the process.

Second, the scheduler does not treat the null process like any other process on the machine. Because the null process is the process of last resort (i.e. it will only run when no other processes are in the ready state), the null process is never added to the multilevel priority queue. By not actually being in the ready multi-queue, there is no possibility that the null process will run ahead of another ready process. The scheduler is set up in such a way that the null process will only be scheduled to run if the multi-queue is empty and there are no other ready processes.

A third design choice made was that the system calls and interrupt handlers were components responsible for upgrading or downgrading the priority of a process based on its behavior. If a processes depleted its time slice and was preempted by the clock handler, it was the clock handler's responsibility to downgrade the priority before calling the rescheduler. Conversely, it is the responsibility of the blocking system calls to upgrade the priority before invoking the scheduler. From a philosophical standpoint, it is not the job of the rescheduler to update or change the priority of a process. Its main and only purpose is to determine which jobs should run on the CPU. Therefore, in my implementation, it is the responsibility of the functions that invoke the scheduler to update a process' priority before calling the scheduler.

Additionally, the benchmark tests from the previous lab were run on the TS Scheduler. When testing with only CPU-Intensive processes, the behavior of decreasing priority but increasing time slices is evident. First, each process printed once with a priority of 29. Then, because of the longer time slices, each process prints 3 times before it is context switched off the CPU. Additionally, all the processes run in round robin order. Therefore, according to the definition of the Solaris Time Sharing philosophy, the processes are running in a “fair” manner. The difference between this test and the one in the previous lab is that as the processes run longer, their priority drops more and more and their time slices becomes bigger and bigger. Therefore, as opposed to the previous lab, the CPU-Intensive processes will run longer with longer time slices as time progresses.

In the case where all the processes are I/O-Intensive, the output of the test indicates that the scheduler is running the processes fairly. The processes execute in the order in which they were resumed (in round robin fashion), have the highest priority available to normal processes (priority 59 in the table is weird and you should check that), and have a very small time slice. In comparison to the previous lab, there is practically no difference. Because all of the processes are asleep almost all of the time, there is no noticeable difference between only I/O processes with this scheduler versus the one from lab 2.

Finally, in the case where there is a half and half mix between CPU and I/O bound processes, the scheduler still provides fair time sharing between processes. When the I/O-Intensive processes are

blocked/sleeping, the CPU-Intensive processes run for longer and can hog the CPU. However, when the I/O bound processes become ready, they immediately are switched on to the CPU to run and then subsequently become blocked. The outcome from this scheduler is similar to lab 2 in that I/O-Intensive processes run almost immediately after they become ready. The difference is that CPU-Intensive processes that run longer receive longer time slices and can perform more work between context switches than the scheduler allowed in lab 2.

If you wish to see the output of my tests, they are included at the bottom of this document. Additionally, if you would like to run the tests yourself, they are currently commented out in the main method.

Below is a list of files that I changed for this lab and a summary of their changes.

---

ts_disptb.h
- created and implemented struct
- added macros that all for changing of process priority and getting the time slice associated withthe process' priority

xinu.h
- added include statement for ts_disptb.h

initialize.c
- declaration & initialization for tsdtab (of length NUM_PRIO_LVLS)
- initialize every queue in the array multiqueue
- null process is initialized with the "maximum" priority (NUM_PRIO_LVLS - 1) so that it receives the smallest time slice

resched.c
- the updated quantum for the new process is now retrieved from tsdtab
- uses multilevel feedback queue instead of readylist
- the null process is never added to the multilevel queue
    - it is only every scheduled if all NUM_PRIO_LVLS queues are empty and there are no ready processes

clkhandler.c
- changed preempt behavior to downgrade priority when quantum is depleted

sleep.c
- upgrade process' priority when sleepms is called because CPU cycles have been yielded

yield.c
- upgrade process priority when yield is called

suspend.c
- upgrade process priority on success because it's a blocking system call

wait.c
- upgrade process priority on success because it's a blocking system call

receive.c
- upgrade process priority because of blocking I/O system call

recvtime.c
- upgrade process priority because of blocking I/O system call.

clkinit.c
- changed use of default QUANTUM macro to custom macro that uses TS Dispatch Table
- currently, the QUANTUM macro is never used; my own macro is used

queue.h
- changed the default number of entries in the global table of queues so there is enough room for multilevel queue scheduler

ready.c
- declared multilevel queue as array of queues of length NUM_PRIO_LVLS
- in ready() method, insert process into the queue associated with priority

create.c
- changed functionality so that all created processes have a priority of DEFAULT_PRIO (which has a value of 29)

kprintf.c
- disabled interupts during the print (so that the prints from tests aren't interrupted)

multilevelfbq.h
- header file containing an extern definition of the multiqueue data structure
    - the multiqueue is simply an array of process queues where the index corresponds to the priority level of the queue

---

------- CPU-Intensive Muthafuckas -------

PID = 3 CPU-Intensive "cpu1"
Outer Loop = 0
Priority = 29
Time Slice Remaining = 58

PID = 4 CPU-Intensive "cpu2"
Outer Loop = 0
Priority = 29
Time Slice Remaining = 58

PID = 5 CPU-Intensive "cpu3"
Outer Loop = 0
Priority = 29
Time Slice Remaining = 58

PID = 6 CPU-Intensive "cpu4"
Outer Loop = 0
Priority = 29
Time Slice Remaining = 58

PID = 3 CPU-Intensive "cpu1"
Outer Loop = 1
Priority = 19
Time Slice Remaining = 154

PID = 3 CPU-Intensive "cpu1"
Outer Loop = 2
Priority = 19
Time Slice Remaining = 91

PID = 3 CPU-Intensive "cpu1"
Outer Loop = 3
Priority = 19
Time Slice Remaining = 27

PID = 4 CPU-Intensive "cpu2"
Outer Loop = 1
Priority = 19
Time Slice Remaining = 154

PID = 4 CPU-Intensive "cpu2"
Outer Loop = 2
Priority = 19
Time Slice Remaining = 90

PID = 4 CPU-Intensive "cpu2"
Outer Loop = 3
Priority = 19
Time Slice Remaining = 27

PID = 5 CPU-Intensive "cpu3"
Outer Loop = 1
Priority = 19
Time Slice Remaining = 154

PID = 5 CPU-Intensive "cpu3"
Outer Loop = 2
Priority = 19
Time Slice Remaining = 90

PID = 5 CPU-Intensive "cpu3"
Outer Loop = 3
Priority = 19
Time Slice Remaining = 27

PID = 6 CPU-Intensive "cpu4"
Outer Loop = 1
Priority = 19
Time Slice Remaining = 154

PID = 6 CPU-Intensive "cpu4"
Outer Loop = 2
Priority = 19
Time Slice Remaining = 90

PID = 6 CPU-Intensive "cpu4"
Outer Loop = 3
Priority = 19
Time Slice Remaining = 26

PID = 3 CPU-Intensive "cpu1"
Outer Loop = 4
Priority = 9
Time Slice Remaining = 163

PID = 3 CPU-Intensive "cpu1"
Outer Loop = 5
Priority = 9
Time Slice Remaining = 100

PID = 4 CPU-Intensive "cpu2"
Outer Loop = 4
Priority = 9
Time Slice Remaining = 162

PID = 4 CPU-Intensive "cpu2"
Outer Loop = 5
Priority = 9
Time Slice Remaining = 98

PID = 5 CPU-Intensive "cpu3"
Outer Loop = 4
Priority = 9
Time Slice Remaining = 163

PID = 5 CPU-Intensive "cpu3"
Outer Loop = 5
Priority = 9

Time Slice Remaining = 100

PID = 6 CPU-Intensive "cpu4"
Outer Loop = 4
Priority = 9
Time Slice Remaining = 162

PID = 6 CPU-Intensive "cpu4"
Outer Loop = 5
Priority = 9
Time Slice Remaining = 98

------- I/O-Intensive Muthafuckas -------

PID = 7 I/O-Intensive "io1"
Outer Loop = 0
Priority = 58
Time Slice Remaining = 40

PID = 8 I/O-Intensive "io2"
Outer Loop = 0
Priority = 58
Time Slice Remaining = 40

PID = 9 I/O-Intensive "io3"
Outer Loop = 0
Priority = 58
Time Slice Remaining = 40

PID = 10 I/O-Intensive "io4"
Outer Loop = 0
Priority = 58
Time Slice Remaining = 40

PID = 7 I/O-Intensive "io1"
Outer Loop = 1
Priority = 58
Time Slice Remaining = 40

PID = 8 I/O-Intensive "io2"
Outer Loop = 1
Priority = 58
Time Slice Remaining = 40

PID = 9 I/O-Intensive "io3"
Outer Loop = 1
Priority = 58
Time Slice Remaining = 40

PID = 10 I/O-Intensive "io4"
Outer Loop = 1
Priority = 58
Time Slice Remaining = 40

PID = 7 I/O-Intensive "io1"
Outer Loop = 2
Priority = 58
Time Slice Remaining = 40

PID = 8 I/O-Intensive "io2"
Outer Loop = 2
Priority = 58
Time Slice Remaining = 40

PID = 9 I/O-Intensive "io3"
Outer Loop = 2
Priority = 58
Time Slice Remaining = 40

PID = 10 I/O-Intensive "io4"
Outer Loop = 2
Priority = 58
Time Slice Remaining = 40

PID = 7 I/O-Intensive "io1"
Outer Loop = 3
Priority = 58
Time Slice Remaining = 40

PID = 8 I/O-Intensive "io2"
Outer Loop = 3
Priority = 58
Time Slice Remaining = 40

PID = 9 I/O-Intensive "io3"
Outer Loop = 3
Priority = 58
Time Slice Remaining = 40

PID = 10 I/O-Intensive "io4"
Outer Loop = 3
Priority = 58
Time Slice Remaining = 40

PID = 7 I/O-Intensive "io1"
Outer Loop = 4
Priority = 58
Time Slice Remaining = 40

PID = 8 I/O-Intensive "io2"
Outer Loop = 4
Priority = 58
Time Slice Remaining = 40

PID = 9 I/O-Intensive "io3"
Outer Loop = 4
Priority = 58
Time Slice Remaining = 40

PID = 10 I/O-Intensive "io4"
Outer Loop = 4
Priority = 58
Time Slice Remaining = 40

------- Half & Half Muthafuckas -------

PID = 11 CPU-Intensive "cpuA"
Outer Loop = 0
Priority = 29
Time Slice Remaining = 57

PID = 12 CPU-Intensive "cpuB"
Outer Loop = 0
Priority = 29
Time Slice Remaining = 119

PID = 12 CPU-Intensive "cpuB"
Outer Loop = 1
Priority = 29
Time Slice Remaining = 119

PID = 12 CPU-Intensive "cpuB"
Outer Loop = 2
Priority = 29
Time Slice Remaining = 119

PID = 13 I/O-Intensive "ioA"
Outer Loop = 0
Priority = 58
Time Slice Remaining = 40

PID = 14 I/O-Intensive "ioB"
Outer Loop = 0
Priority = 58
Time Slice Remaining = 40

PID = 12 CPU-Intensive "cpuB"

Outer Loop = 3
Priority = 29
Time Slice Remaining = 119

PID = 12 CPU-Intensive "cpuB"
Outer Loop = 4
Priority = 29
Time Slice Remaining = 119

PID = 12 CPU-Intensive "cpuB"
Outer Loop = 5
Priority = 29
Time Slice Remaining = 119

PID = 11 CPU-Intensive "cpuA"
Outer Loop = 1
Priority = 19
Time Slice Remaining = 159

PID = 13 I/O-Intensive "ioA"
Outer Loop = 1
Priority = 58
Time Slice Remaining = 40

PID = 14 I/O-Intensive "ioB"
Outer Loop = 1
Priority = 58
Time Slice Remaining = 40

PID = 11 CPU-Intensive "cpuA"
Outer Loop = 2
Priority = 19
Time Slice Remaining = 158

PID = 11 CPU-Intensive "cpuA"
Outer Loop = 3
Priority = 19
Time Slice Remaining = 158

PID = 11 CPU-Intensive "cpuA"
Outer Loop = 4
Priority = 19
Time Slice Remaining = 158

PID = 13 I/O-Intensive "ioA"
Outer Loop = 2
Priority = 58
Time Slice Remaining = 40

PID = 14 I/O-Intensive "ioB"
Outer Loop = 2
Priority = 58
Time Slice Remaining = 40

PID = 11 CPU-Intensive "cpuA"
Outer Loop = 5
Priority = 19
Time Slice Remaining = 158

PID = 13 I/O-Intensive "ioA"
Outer Loop = 3
Priority = 58
Time Slice Remaining = 40

PID = 14 I/O-Intensive "ioB"
Outer Loop = 3
Priority = 58
Time Slice Remaining = 40

PID = 13 I/O-Intensive "ioA"
Outer Loop = 4
Priority = 58
Time Slice Remaining = 40

PID = 14 I/O-Intensive "ioB"
Outer Loop = 4
Priority = 58
Time Slice Remaining = 40