Department of Computer Science and Engineering
University of Dhaka.

CSE-3211.
Operating System Lab

Assignment-2
OS161 FIFO and Preemptive Priority Scheduler

Submitted by:
1. Md. Ahasanul Alam(10)
2. Mustafizur Rahman(22)
3. Sree Sowmik Kumar Sarker(54)

Submission Date: 15/11/2019.

Submitted to:
Dr.Mosaddek Hossain Kamal Tushar.
Professor,Department of Computer Science and Engineering,
University of Dhaka.
Dr.Md Mamun-or-Rashid.
Professor,Department of Computer Science and Engineering,
University of Dhaka.

Theory :

Schedule:

When a computer is multiprogrammed, it frequently has multiple processes or threads competing for the CPU at the same time. This situation occurs whenever two or more of them are simultaneously in the ready state. If only one CPU is available, a choice has to be made which process to run next. The part of the operating system that makes the choice is called the scheduler, and the algorithm it uses is called the scheduling algorithm.

In our assignment you had to implement two scheduling Algorithm,

- First In First Out (**FIFO**)
- Preemptive priority

First In First Out:

With this algorithm, processes are assigned the CPU in the order they request it. Basically, there is a single queue of ready processes.When the first job enters the system, it is started immediately and allowed to run as long as it wants to. As other jobs come in, they are put onto the end of the queue. When the running process blocks, the first process on the queue is run next. When a blocked process becomes ready, like a newly arrived job, it is put on the end of the queue, behind all waiting processes.

Preemptive priority:

Each process is assigned a priority, and the runnable process with the highest priority(lowest priority value) is allowed to run.To prevent high-priority processes from running indefinitely,scheduler introduces Ageing to alleviate the priority.When a process takes more than 16-time slots, than increase its priority number by ’5’, which eventually decreases the priority of the process/thread. For the ageing, the priority of the process/thread would be improved by decreasing the priority number by one while the process is not running for 64 units of time.

Schedule Assigning and variable addition:

In src/kern/include/thread.h a macro is defined as SCHEDULE. If the value of SCHEDULE is assigned as 1 then the FIFO scheduling will run. If the value of SCHEDULE is assigned as 2 then the PRIORITY scheduling will run. For our convenience and requirement of the assignment, following variables are added with thread structure:

```
unsigned int priority; (Contains priority value of the thread)
unsigned int taken_time_slot; (To know how many time slot is taken by the
thread)
unsigned int not_taken_time_slot; (To know for how much time slot a thread
is waiting)
int thread_id;
struct timespec last_run_time,cur_run_time,wait_time,run_time;
struct timespec last_wait_time,cur_wait_time;
```

Implementing FIFO:

In src/kern/thread there exists thread.c which implements all the necessary things including create,fork,destroy,scheduling the thread.In it, The “curcpu” contains about current thread, runnable threads,ready threads,waiting threads etc.Here “curcpu->c_runqueue” contains the list of all the threads that are in runnable state.The first thread in the list is currently in execution. Function “thread_yield()” is used to switch it.This function removes the first thread (which were in execution) from the list and put it at the tail of the list so that the next thread can run. In our implementation we didn’t call thread_yield() at all . Therefore, the thread which created first, is at the head of the runqueue and able to run as long as it wants .As a result the scheduling algorithm is running just like FIFO.

Implementing Priority with ageing:

At first, each thread’s priority is assigned randomly during the creation of the thread. Then, In schedule() function we first run a loop over the runqueue to find highest priority thread. If highest priority thread is not the first one in the runqueue then we removed that thread from the runqueue and inserted after the firstnode. As a result, when thread_yield() will be called, firstnode in the runqueue will be removed and the node after the firstnode which is the highest priority thread will run now. On the other hand, if the current thread is the highest priority thread then nothing is done. As a result, the highest priority thread will run until it finishes. Each time the thread will run it’s taken_time_slot will be incremented by one. If it’s become multiple of 16 then that thread’s priority will be decremented by 5. Besides, that thread’s not_taken_time_slot will be assigned to 0. For those threads which will be unable to get the cpu, their not_taken_time_slot will be incremented by 1 . If any of the threads not_taken_time_slot becomes multiple of 64 then that thread’s priority will be increased by 1.

Statistics and Result:

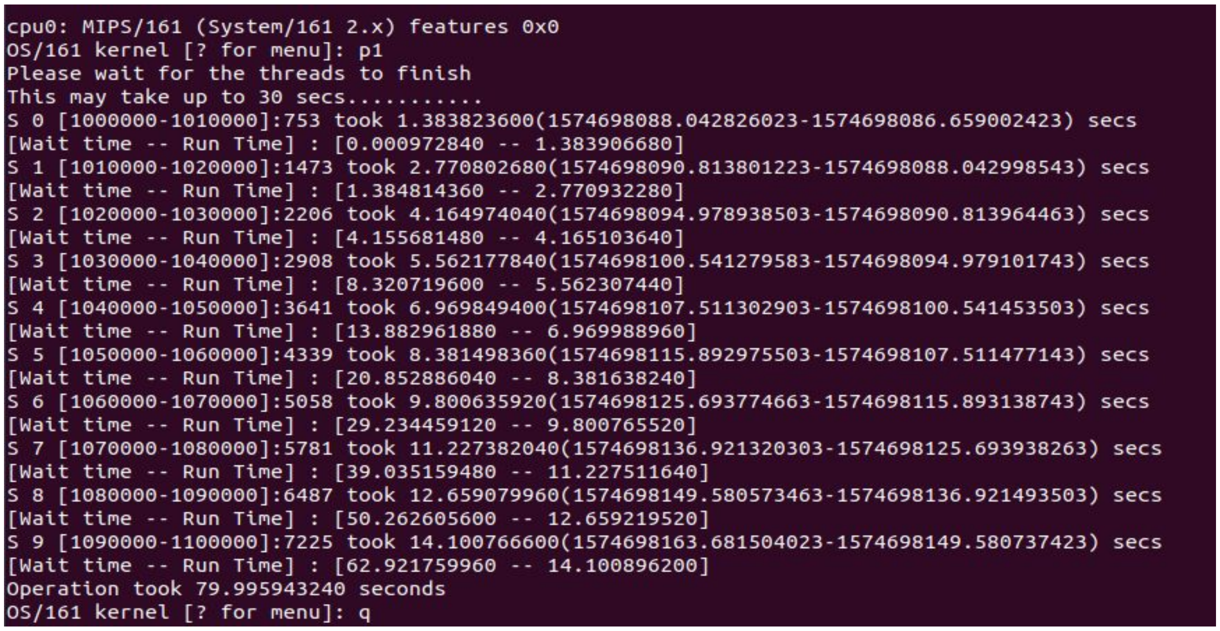
In this part we have printed the the results from each thread and some statistical information about the threads.The information was:

- Thread start time
- Thread run time
- Total run time
- Total waiting time

For total run time and waiting time we had to calculate the necessary calculation during each thread switch and and pass it to the prime_gen.c for printing.We have done this passing two parameter (run_time,wait_time) as call by reference.Every time a thread is in thread_switch() function we make the calculation:

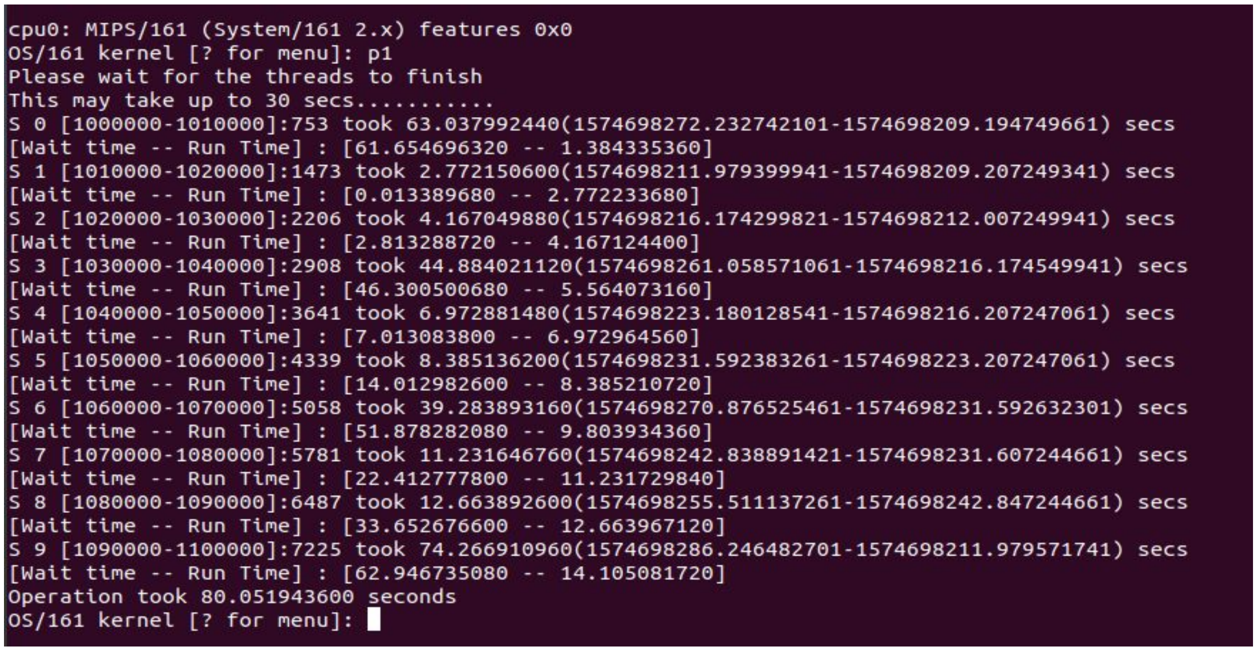
- Curthread is now going to sleep,(last_run_time - cur_time) this thread was in execution state,so add run_time for it.and update it’s last_wait_time = cur_time;
- For the next thread,it is going to be running now.(last_wait_time - cur_time) it was in waiting state.so add wait_time for it.and update it’s last_run_time = cur_time.

For fifo we got the following output:



```
cpu0: MIPS/161 (System/161 2.x) features 0x0
OS/161 kernel [? for menu]: p1
Please wait for the threads to finish
This may take up to 30 secs..........
S 0 [1000000-1010000]:753 took 1.383823600(1574698088.042826023-1574698086.659002423) secs
[Wait time -- Run Time] : [0.000972840 -- 1.383906680]
S 1 [1010000-1020000]:1473 took 2.770802680(1574698090.813801223-1574698088.042998543) secs
[Wait time -- Run Time] : [1.384814360 -- 2.770932280]
S 2 [1020000-1030000]:2206 took 4.164974040(1574698094.978938503-1574698090.813964463) secs
[Wait time -- Run Time] : [4.155681480 -- 4.165103640]
S 3 [1030000-1040000]:2908 took 5.562177840(1574698100.541279583-1574698094.979101743) secs
[Wait time -- Run Time] : [8.320719600 -- 5.562307440]
S 4 [1040000-1050000]:3641 took 6.969849400(1574698107.511302903-1574698100.541453503) secs
[Wait time -- Run Time] : [13.882961880 -- 6.969988960]
S 5 [1050000-1060000]:4339 took 8.381498360(1574698115.892975503-1574698107.511477143) secs
[Wait time -- Run Time] : [20.852886040 -- 8.381638240]
S 6 [1060000-1070000]:5058 took 9.800635920(1574698125.693774663-1574698115.893138743) secs
[Wait time -- Run Time] : [29.234459120 -- 9.800765520]
S 7 [1070000-1080000]:5781 took 11.227382040(1574698136.921320303-1574698125.693938263) secs
[Wait time -- Run Time] : [39.035159480 -- 11.227511640]
S 8 [1080000-1090000]:6487 took 12.659079960(1574698149.580573463-1574698136.921493503) secs
[Wait time -- Run Time] : [50.262605600 -- 12.659219520]
S 9 [1090000-1100000]:7225 took 14.100766600(1574698163.681504023-1574698149.580737423) secs
[Wait time -- Run Time] : [62.921759960 -- 14.100896200]
Operation took 79.995943240 seconds
OS/161 kernel [? for menu]: q
```

For Priority we got the following output:



```
cpu0: MIPS/161 (System/161 2.x) features 0x0
OS/161 kernel [? for menu]: p1
Please wait for the threads to finish
This may take up to 30 secs..........
S 0 [1000000-1010000]:753 took 63.037992440(1574698272.232742101-1574698209.194749661) secs
[Wait time -- Run Time] : [61.654696320 -- 1.384335360]
S 1 [1010000-1020000]:1473 took 2.772150600(1574698211.979399941-1574698209.207249341) secs
[Wait time -- Run Time] : [0.013389680 -- 2.772233680]
S 2 [1020000-1030000]:2206 took 4.167049880(1574698216.174299821-1574698212.007249941) secs
[Wait time -- Run Time] : [2.813288720 -- 4.167124400]
S 3 [1030000-1040000]:2908 took 44.884021120(1574698261.058571061-1574698216.174549941) secs
[Wait time -- Run Time] : [46.300500680 -- 5.564073160]
S 4 [1040000-1050000]:3641 took 6.972881480(1574698223.180128541-1574698216.207247061) secs
[Wait time -- Run Time] : [7.013083800 -- 6.972964560]
S 5 [1050000-1060000]:4339 took 8.385136200(1574698231.592383261-1574698223.207247061) secs
[Wait time -- Run Time] : [14.012982600 -- 8.385210720]
S 6 [1060000-1070000]:5058 took 39.283893160(1574698270.876525461-1574698231.592632301) secs
[Wait time -- Run Time] : [51.878282080 -- 9.803934360]
S 7 [1070000-1080000]:5781 took 11.231646760(1574698242.838891421-1574698231.607244661) secs
[Wait time -- Run Time] : [22.412777800 -- 11.231729840]
S 8 [1080000-1090000]:6487 took 12.663892600(1574698255.511137261-1574698242.847244661) secs
[Wait time -- Run Time] : [33.652676600 -- 12.663967120]
S 9 [1090000-1100000]:7225 took 74.266910960(1574698286.246482701-1574698211.979571741) secs
[Wait time -- Run Time] : [62.946735080 -- 14.105081720]
Operation took 80.051943600 seconds
OS/161 kernel [? for menu]: █
```