# Mid-Exam Sample

March 29, 2023

1. **Scheduling(15 points)**
   The most appropriate scheduling algorithm for an operating system can depend on the type of workloads expected to run on it. For each of the following three workloads, give the name of the most appropriate scheduling algorithm for such a workload, from among the following algorithms: first-come-first-served(FCFS), non-preemptive shorted-job-first(SJF), round-robin(RR), preemptive priority(PP), multilevel feedback (MFQ). For each system, write a sentence explaining why your choice of algorithm is the most appropriate.

   (a) An air-traffic control workstation which runs multiple processes for tracking planes, in which some processes must temporarily gain exclusive access to the CPU in response to critical conditions (e.g. ensuring two planes do not collide with one another).

   (b) A shared system on which multiple users perform interactive activities such as reading mail and browsing the web.

   (c) A batch-processing system in which we wish to minimise the average waiting time for running jobs.

   **Solution:**

   (a) PP. Certain critical processes that MUST have access to the CPU can be run at a higher priority.

   (b) RR. Give every interactive process a fair share of the processor. MFQ is also admissible, since it incorporates a round-robin component, but is strictly more complicated than the question requires.

   (c) SJF. Pre-emption only extends the processing time for the batch. Putting the shortest job first will (provably) reduce the average job waiting time.

2. **Mutual Exclusion(6 points)**

   (a) (2 points) Many operating systems designed to run only on uni-processors use disabling of interrupts to create critical sections in the code. Explain how this interrupts disabling prevents multiple threads from entering the critical section.

(b) (2 points) Explain why disabling of interrupts to implement critical sections will not work on a multiprocessor.

**Solution:**

(a) Disabling interrupts prevents interrupts (including the timer interrupt) from invoking the operating system scheduler that preempting the job during the critical section. So, the process will execute the critical section to completion before another job will be able to get the CPU and enter the critical section.

(b) Disabling interrupts only prevent concurrent execution on the same processor; it does not prevent another processor on a multiprocessor from entering critical sections.

3. **CPU Scheduling(10 points)**
Suppose you have the 3 processes running on one processor. Each process runs K iterations of computation and disk I/Os. In each iteration, a process runs its computation and a disk I/O in a serial fashion, but the disk I/O can overlap with the computation time of another prcoess. The following table lists the computation period as CPU time and the disk I/O operation as I/O time per iteration for each process.

| | CPU Time | I/O time | Iterations(K) |
|---|---|---|---|
| Process 1 | 11 ms | 10 ms | 4 |
| Process 2 | 21 ms | 10 ms | 3 |
| Process 3 | 31 ms | 10 ms | 2 |

Suppose a system uses a round-robin preemptive scheduling with a time slice quantum of 20ms and the order of the processes initially in the ready queue is process 1, process 2, and process 3. Assume that context-switch overhead is 0. Please answer the following questions:

A) Average turnaround time

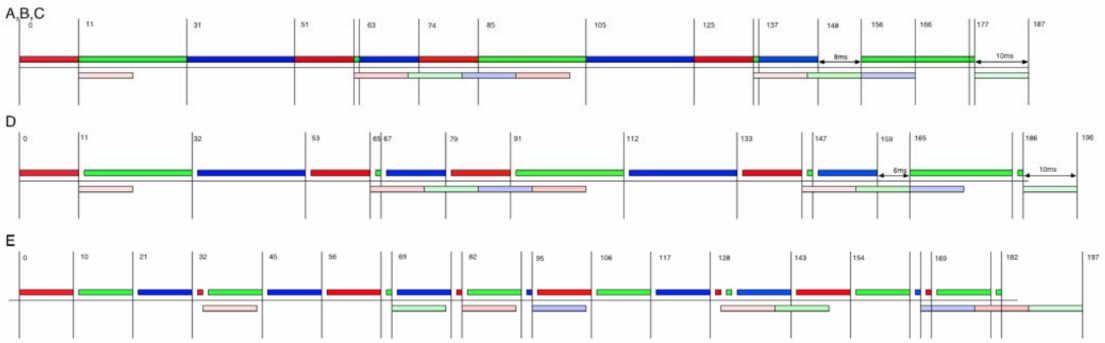
B) CPU utilization

C) Disk utilization

D) With a 1ms context switch, Disk utilization

E) With a 10ms quanta and 1ms context switch, Disk utilization

**Solution:**



A) Average turnaround time
Process $1 = 146 - 0 = 146ms$
Process $2 = 187 - 0 = 187ms$
Process $3 = 166 - 0 = 166ms$

B) CPU utilization
$\frac{187-8-10}{187} = \frac{169}{187}$

C) Disk utilization
$\frac{(4+3+2)\times 10}{187} = \frac{90}{187}$

D) With a 1ms context switch, Disk utilization
$\frac{90}{1\times 11+178} = \frac{90}{196}$

E) With a 10ms quanta and 1ms context switch, Disk utilization
$\frac{90}{197}$

4. **Threads(6 points)**

(a) (2 points) What needs to be saved and restored on a context switch between two threads in the same process? What if two are in different processes? Be brief and explicit.

(b) (2 points) Why is switching threads less costly than switching processes?

(c) (2 points) Suppose a thread is running in a critical section of code, meaning that it has required all the locks through proper arbitration. Can it get context switched? Why or not?

(d) (2 points) Why would two processes want to use shared memory for communication instead of using message passing?

**Solution:**

(a) Need to save the registers, stack pointer, and program counter into the thread control(TCB) of the thread that is no longer running. Need to reload the registers, stack pointer, and program counter from the TCB of the new thread.
When the threads are from different processes, need to not only save and resolve what was given above, but also need to load the pointer for the top-level page table of the new address space. (Note that this top-level page table pointer from the old process does not need to be saved since it does not change and is already contained in the process control block(PCB)).

(b) Less state need to be saved and restored. Futhermore, switching between threads benefits from catching; whereas, switching between processes invalidates the cache and TLB.

(c) Yes, a process holding a lock can get context switched. Locks(especially user-level locks) are independent of the scheduler. (Note that threads running in the kernel with interrupts disabled would not get context-switched).

(d) Performance. Communicating via shared memory is often faster and more efficient since there is no kernel intervention and no copying.

5. **CPU Scheduling(24 points total**
Here is a table of processes and their associated arrival and running times.

| Process ID | Arrival Time | Expected CPU Running Time |
|---|---|---|
| Process 1 | 0 | 5 |
| Process 2 | 1 | 5 |
| Process 3 | 5 | 3 |
| Process 4 | 6 | 2 |

(a) (12 points) Show the scheduling order for these processes under First-In-First-Out(FIFO), Shortest-Job First(SJF), and Round-Robin(RR) with a quantum = 1 time unit. Assume that the context switch overhead is 0 and new processes are added to the head of the queue except for FIFO.

(b) (12 points) For each process in each schedule above, indicate the queue wait time and turnaround time (TRT).

**Solution:**

(a) In the following:

| Process\Time | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | | | | | | | | | | | | | | | | |
| B | | | | | | | | | | | | | | | | |
| C | | | | | | | | | | | | | | | | |
| D | | | | | | | | | | | | | | | | |

FCFS

| Process\Time | 0(A) | 1(B) | 2 | 3 | 4 | 5(C) | 6(D) | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | | | | | | | | | | | | | | | | |
| B | | | | | | | | | | | | | | | | |
| C | | | | | | | | | | | | | | | | |
| D | | | | | | | | | | | | | | | | |

SJF

| Process\Time | 0(A) | 1(B) | 2 | 3 | 4 | 5(C) | 6(D) | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | | | | | | | | | | | | | | | | |
| B | | | | | | | | | | | | | | | | |
| C | | | | | | | | | | | | | | | | |
| D | | | | | | | | | | | | | | | | |
| Ready | A | AB | BA | AB | BA | ABC | CAB | DCA | BDC | ABD | CAB | CAB | BCA | BC | B | |
| CPU | A | B | A | B | A | C | D | B | A | C | D | B | A | C | B | |

RR, q = 1

(b) The queue wait time is the total time a process spends in the wait queue. The turnaround time is defined as the time a process takes to complete after it first arrives.

| Process\Time | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | Queue Wait | Turanound Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | | | | | | | | | | | | | | | | | 0 | 5 |
| B | | | | | | | | | | | | | | | | | 4 | 10-1= 9 |
| C | | | | | | | | | | | | | | | | | 10-5=5 | 13-5=8 |
| D | | | | | | | | | | | | | | | | | 13-6=7 | 15-6=9 |

FCFS

| Process\Time | 0(A) | 1(B) | 2 | 3 | 4 | 5(C) | 6(D) | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | Queue Wait | Turanound Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | | | | | | | | | | | | | | | | | 0 | 5 |
| B | | | | | | | | | | | | | | | | | 10-1=9 | 15-1=14 |
| C | | | | | | | | | | | | | | | | | 5-5=0 | 8-5=3 |
| D | | | | | | | | | | | | | | | | | 8-6=2 | 10-6=4 |

SJF

| Process\Time | 0(A) | 1(B) | 2 | 3 | 4 | 5(C) | 6(D) | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | Queue Wait | Turanound Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | | | | | | | | | | | | | | | | | 8 | 13 |
| B | | | | | | | | | | | | | | | | | 9 | 15-1=14 |
| C | | | | | | | | | | | | | | | | | 6 | 14-5=9 |
| D | | | | | | | | | | | | | | | | | 3 | 11-6=5 |
| Ready | A | AB | BA | AB | BA | ABC | CAB | DCA | BDC | ABD | CAB | CAB | BCA | BC | B | | | |
| CPU | A | B | A | B | A | C | D | B | A | C | D | B | A | C | B | | | |

RR, q = 1