OPERATING SYSTEM

Lecture 6

Process Management: Scheduling

Dr Victoria Mukami

INTRODUCTION

This lecture looks at the processor. Specifically, we shall review scheduling managers and the process scheduler. We will review some of the most common scheduling policies and algorithms including First Come First Serve, Priority Scheduling, Shortest Job Next, Shortest Remaining Time, and Round Robin. Finally, we will review how the processor manages interrupts.

Learning objectives

By the end of this topic, you should be able to:

1. Describe how scheduling managers work
2. Show how process scheduling algorithms work
3. How the processor handles interrupts

OVERVIEW

So far, we have worked with memory management within the operating system. Memory is used to hold information for the processor to aid in faster processing. Within this lecture, we now concentrate on how processing is conducted within the CPU. We will review processor management scheduling. We initially begin with scheduling and how jobs are assigned to the CPU.

SCHEDULING SUBMANAGERS

The process manager oversees jobs ensuring they are scheduled and processing takes place correctly. It is made up of the job scheduler and the process scheduler [1]. The job scheduler oversees initiating a job based on certain criteria [1]. A job also referred to as a program goes through a hierarchy of managers [1]. Any job first encounters the job scheduler. The job scheduler selects jobs from a queue of incoming jobs and places them in the processor queue. The main role of the job scheduler is to place jobs based on specifications set by the system.

For example, jobs sometimes interact with input/output (I/0) devices during processing. The job scheduler must optimize the CPU's time to ensure that the CPU is not dealing with jobs that have many I/O requests. The reverse is also true where the job scheduler should not only send jobs that have no I/O requests, thereby leaving the I/O devices idle. The job scheduler, therefore, needs to review the requirements

of each job and provide a mix. The job scheduler is known as a high-level scheduler [1].

Once the job scheduler completes scheduling jobs, then the process scheduler takes over. The process scheduler determines which jobs (processes) will get to the CPU, at what point and for how long [1]. It is also in charge of CPU interrupts based on incoming higher priority jobs. It is a low-level scheduler and works with jobs that are assigned a READY state.

SCHEDULING ALGORITHMS

A scheduling algorithm is normally based on a specific scheduling policy within the system. Scheduling policies are created by the system administrator to determine the criteria that are most important within a system [1]. Several scheduling algorithms are used to schedule processes or threads if allowed by an operating system. The algorithms include First-Come, First-Served, Shortest Job Next, Shortest Remaining Time, Priority Scheduling and Round Robin. We now review each of these algorithms and their inner working. There are two main types of policies. Pre-emptive scheduling policies are those that interrupt the processing of a CPU and transfer it to another job. Non-pre-emptive scheduling policy functions without external interrupts. The only interrupt would be that of an I/0 request.

First-Come, First-Served

This is a non-pre-emptive scheduling algorithm that handles all the incoming jobs according to their arrival time [1]. The earlier the job arrives, the earlier it is served. This algorithm is commonly used for batch systems. FCFS does not run WAIT queues because a job runs to completion once it is sent for processing. Remember this is the non-pre-emptive nature of the algorithm.

Turnaround time is the time required to execute a job and return the output to the user [1]. Review the following jobs in Table 1 that enter the CPU one after the other.

Table 1: List of Jobs to be processed using FCFS

Job	Processing Time (Milliseconds)
A	13
B	7
C	2
D	5

Processing occurs so that a timeline is shown in Figure 1.

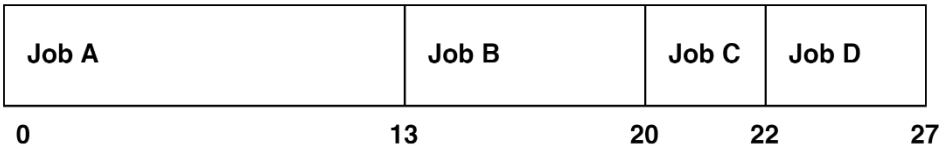

Figure 1: FCFS job timeline [1]

As shown, the jobs arrive and are immediately processed. Job A ends at 13ms while Job B starts at 13ms and ends at 20ms and so on. The entire time it takes to process the jobs is 27ms. If our 4 jobs arrived simultaneously at time 0, then the turnaround time is calculated by subtracting the arrival time from the finish time of each job. In this case Job, A is 13, Job B is 20, Job C is 22, and Job D is 27ms. The turnaround time is 20.5ms.

$$\frac{(13-0)+(20-0)+(22-0)+(27-0)}{4}=20.5$$

Had the jobs arrived in a different order? That is Job C first then Job D, Job B and finally job E the timeline would have looked like Figure 2 while the turnaround time is calculated below.

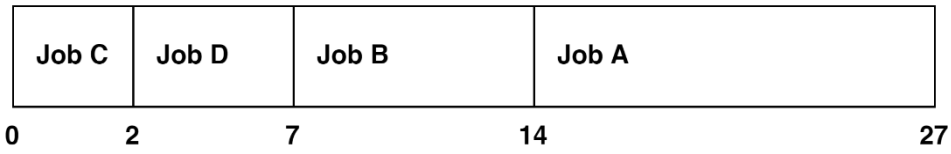

Figure 2: FCFS job timeline [1]

The turnaround time is 12.5 when the jobs are re-ordered.

$$\frac{(2-0)+(7-0)+(14-0)+(27-0)}{4} = 12.5$$

As seen with the two timelines Figure 1 and Figure 2, there is a disparity in the turnaround time based on when the jobs first arrived. This brings up the biggest disadvantage that the timelines vary. The FCFS does not provide the chance of processing the jobs at the most optimum time.

Shortest Job Next

SJN is also a non-pre-emptive algorithm that selects jobs based on the length of the CPU cycle time (processing time) [1]. The algorithm is also known as the shortest job first. This means that before any job can be run, the CPU time must be known in advance. Let us use the same jobs from Table 1 with slightly different CPU cycles.

Table 2: SJN jobs

Job	CPU Cycle
A	8
B	6
C	2
D	4

The timeline for the jobs is shown in Figure 3.

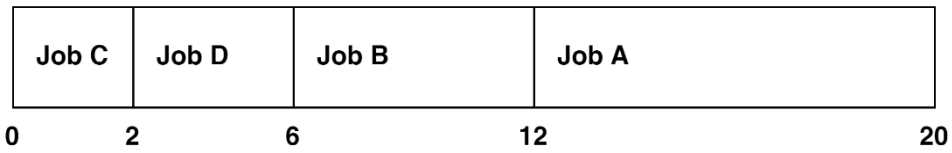

Figure 3: SJN Jobs Timeline

The final CPU time is 20 while the turnaround time is 10ms

$$\frac{(2-0)+(6-0)+(12-0)+(20-0)}{4} = 10$$

Based on the SJN the job with the shortest time would run first, followed by the next shortest and so on. This algorithm is very effective when all the jobs are available at the same time and the CPU estimates are accurate.

Priority Scheduling

This is a non-pre-emptive algorithm that gives preferential treatment to important jobs [1]. A job that has the highest priority is processed until its CPU cycle runs out or a wait occurs due to I/O requests. If two jobs with similar priorities happen to arrive in the READY queue, the job that arrived first is processed first.

Priorities are determined by the processor manager based on

- Memory requirements
- Number and type of peripheral devices
- Total CPU time
- Amount of time already spent in the system

Shortest Remaining Time

This is a pre-emptive version of the Shortest Job Next [1]. Here the processor is assigned the job that is closest to completion. Where several jobs are waiting and have similar remaining time, the job that has been waiting for the longest goes next. This algorithm requires a lot more overhead than SJN as the operating system must keep monitoring the CPU for any job in the READY queue. Table 3 shows a list of jobs that are in the READY queue for processing.

Table 3: Jobs for Processing

Job	CPU Cycle	Arrival Time
A	8	0
B	6	1
C	2	2
D	4	3

Figure 4 shows the timeline for the jobs. The CPU pre-empts the jobs as processing is going on. Job A will go in, but after one cycle, the CPU will check and see that Job B has a shorter CPU cycle than A which after 1 cycle will still have 7 cycles left. This goes on where a cycle 3 Job C gets in and since it has a shorter time than B it gets in. The CPU continues pre-empting until all jobs are processed.

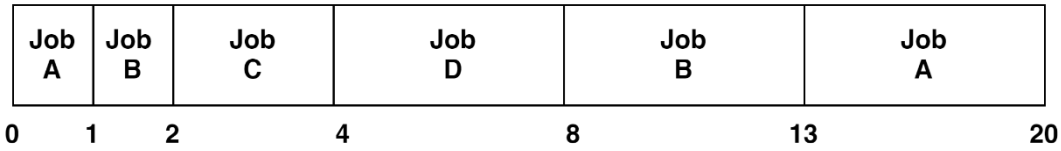

Figure 4: SRT Job Timeline

The turnaround time for each job is calculated as its completion time minus its arrival time. This is shown below.

$$\frac{(20-0)+(13-1)+(4-2)+(8-3)}{4}=9.75$$

Round Robin

This is another pre-emptive algorithm that is used in lots of interactive systems [1]. This algorithm is easy to implement as it is not based on job characteristics but on a determined slice of time [1]. The slice of time is called a time quantum and the size is very crucial to the system performance. Jobs are placed using the FCFS scheme however, the processor scheduler selects the first job sets the timer and when the time expires the CPU moves to the next job.

If the job's CPU cycle is shorter than the time quantum, then all the resources are released for the next job. Let us use the same table 3 that shows the incoming jobs. The time quantum is 5ms.

Job	CPU Cycle	Arrival Time
A	8	0
B	6	1
C	2	2
D	4	3

The jobs end at 20 but each job gets a run through the CPU. Job A runs for 5 cycles and by that time Job, B is already in the queue, so it goes in for processing. By 10ms Job C is in the READY queue and it goes in for processing. However, Job C is only 2ms therefore Job D comes in and it is only 4ms. Once Job D is complete, then Job A goes in, completes in 3ms and Job B goes in for 1ms.

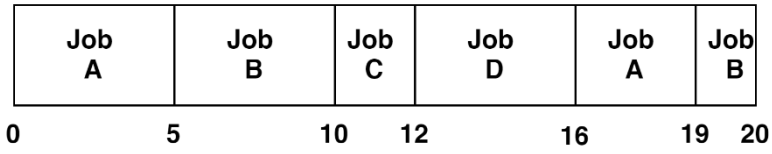

Figure 5: Round Robin Job Timeline

The average turnaround time is calculated as shown below.

$$\frac{(19-0)+(20-1)+(12-2)+(16-3)}{4}=15.25$$

The efficiency of the round robin depends on the time quantum with the average CPU cycle [1]. If the time quantum is too large, then time is wasted and if the time quantum is too short then the CPU overhead increases. General rules used for selecting an optimum time quantum include:

1. It should be long enough to allow 80% of the CPU cycle to run to completion [1]
2. It should be at least 100 times longer than the time required to perform one switch [1]

MANAGING INTERRUPTS

Interrupts occur when the time quantum expires and during I/O interrupts. Interrupts also occur when illegal operations occur such as dividing a sum with zero or trying to access password-protected files. The interrupt handler deals with interruption events. When there is an error that cannot be recovered, the following happens:

1. The type of interrupt is passed on to the user and a copy is stored [1]
2. The state of the interrupted process is saved [1]
3. The interrupt is processed, execution is halted, resources are released, and the job exits [1]
4. The processor resumes normal operation [1]

SUMMARY

This lecture looked at the processor. Specifically, we reviewed scheduling managers and the process scheduler. We also reviewed some of the most common scheduling policies and algorithms including First Come First Serve, Priority Scheduling, Shortest Job Next, Shortest Remaining Time, and Round Robin. Finally, we reviewed how the processor manages interrupts.

DISCUSSION TOPIC

Considering the number of multicore CPUs that currently exist, one of the scheduling algorithms is used to manage jobs in the cores. Do web research and find the most common algorithm used in either quad or dual-core CPUs and discuss the findings with your peers.

REFERENCES

[1] McHoes, A., & Flynn, I., Understanding Operating Systems. Boston: Cengage Learning, 2018

[2] Stallings, W., Operating Systems: Internals and Design Principles. Harlow: Pearson Education Limited, 2018.