Operating System Homework 5

Jinyan Xu, 3160101126, Information Security

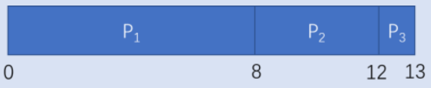
1. Operating System Concept Chapter 5 Exercises: 5.3, 5.4, 5.5, 5.8.

Answer:

5.3 Suppose that the following processes arrive for execution at the times indicated. Each processes will run for the amount of time listed.
In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made.

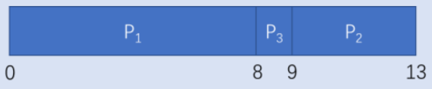
Process	Arrival Time	Burst Time
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

a. What is the average turnaround time for these processes with the FCFS scheduling algorithm?



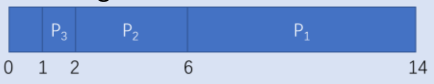
The turnaround time for P_1: $T(P_1) = 8 - 0 = 8$
The turnaround time for P_2: $T(P_2) = 12 - 0.4 = 11.6$
The turnaround time for P_3: $T(P_3) = 13 - 1 = 12$
The average turnaround time: $T_{ave} = (8 + 11.6 + 12) / 3 = 10.53$

b. What is the average turnaround time for these processes with the SJF scheduling algorithm?



The turnaround time for P_1: $T(P_1) = 8 - 0 = 8$
The turnaround time for P_2: $T(P_2) = 13 - 0.4 = 12.6$
The turnaround time for P_3: $T(P_3) = 9 - 1 = 8$
The average turnaround time: $T_{ave} = (8 + 12.6 + 8) / 3 = 9.53$

c. The SJF algorithm is supposed to improve performance but notice that we chose to run process P_1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processes P_1 and P_2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling.



The turnaround time for P_1: $T(P_1) = 14 - 0 = 14$
The turnaround time for P_2: $T(P_2) = 6 - 0.4 = 5.6$
The turnaround time for P_3: $T(P_3) = 2 - 1 = 1$
The average turnaround time: $T_{ave} = (14 + 5.6 + 1) / 3 = 6.87$

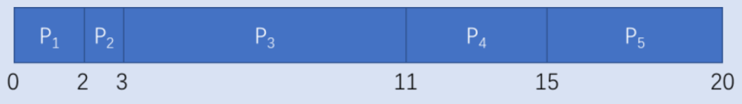
5.4 Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

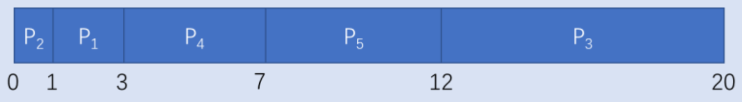
The processes are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5, all at time 0.

a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).

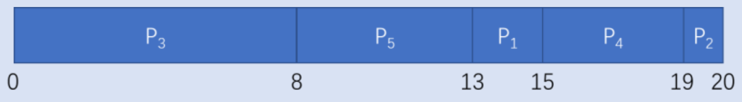
FCFS:



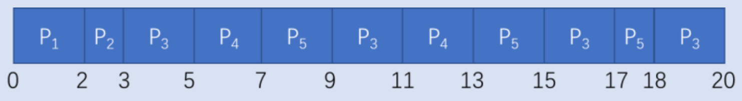
SJF:



Priority:



RR:



b. What is the turnaround time of each process for each of the scheduling algorithms in part a?

Process	FCFS	SJF	Priority	RR
P_1	2	3	15	2
P_2	3	1	20	3
P_3	11	20	8	20
P_4	15	7	19	13
P_5	20	12	13	18

c. What is the waiting time of each process for each of these scheduling algorithms?

Process	FCFS	SJF	Priority	RR
P_1	0	1	13	0
P_2	2	0	19	2
P_3	3	12	0	12
P_4	11	3	15	9
P_5	15	7	8	13

d. Which of the algorithms results in the minimum average waiting time (over all processes)?

From the table following, we can find that SJF algorithm is in the minimum average waiting time.

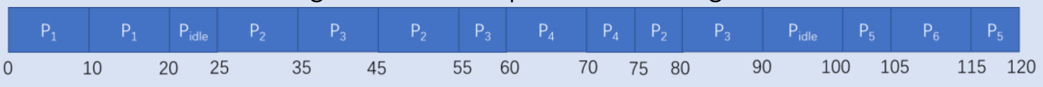
Algorithm	Average waiting time
FCFS	10.2
SJF	8.6
Priority	15
RR	11.2

5.5 The following processes are being scheduled using a preemptive, roundrobin scheduling algorithm.

Process	Priority	Burst	Arrival
P_1	40	20	0
P_2	30	25	25
P_3	30	25	30
P_4	35	15	60
P_5	5	10	100
P_6	10	10	105

Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed below, the system also has an idle task (which consumes no CPU resources and is identified as P_{idle}). This task has priority 0 and is scheduled whenever the system has no other available processes to run. The length of a time quantum is 10 units. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.

a. Show the scheduling order of the processes using a Gantt chart.



b. What is the turnaround time for each process?

Process	Turnaround time	Process	Turnaround time
P_1	20	P_4	15
P_2	55	P_5	20
P_3	60	P_6	10

c. What is the waiting time for each process?

Process	Waiting time	Process	Waiting time
P_1	0	P_4	0
P_2	30	P_5	10
P_3	35	P_6	0

d. What is the CPU utilization rate?

The CPU utilization rate = (105 / 120) * 100 % = 87.5 %

5.8 Suppose that a CPU scheduling algorithm favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs?

Answer:

An **I/O-bound program** typically has many short CPU bursts. A **CPU-bound program** might have a few long CPU bursts.

As the last paragraph of the 206 page says, if the algorithm favors CPU-bound program, it may cause a ***convoy effect*** as all the other processes wait for the one big process to get off the CPU. This effect results in lower CPU and device utilization than might be possible if the shorter processes were allowed to go first. So, it favors I/O-bound programs because these programs having short CPU bursts, and the CPU-bound programs will not starve because these programs won't use CPU all the time.

2. According to the ppt we have learnt in the class, explain the reason why the Mars Pathfinder had reset itself, and how to avoid that.

Answer:

In the Mars Pathfinder mission, NASA chooses **VxWorks**, a real-time embedded systems kernel. VxWorks provides preemptive priority scheduling of threads.

Pathfinder contains an **information bus**, which is a shared memory area used for passing information between different components of the spacecraft. An **information bus thread** runs frequently with **high priority** to move certain kinds of data in and out of the information bus.

The **weather data thread** is a **low priority** thread. When it uses the information bus to publish its data, it would acquire a mutex, do writes to the bus, and release the mutex. Before it releases the mutex, information bus thread is blocked to wait the mutex.

Unluckily, when an interrupt occurs, caused by the **medium priority communication thread**, the high priority information bus thread is blocked and waiting for the low priority weather data thread. Because the communication thread has higher priority than the weather data thread, CPU will choose communication thread to run, so the information bus thread is still blocked. After a long time, the **watchdog** notices that the information bus thread has not been executed for a long time, it believes that something wrong, and reset the system.

This scenario is called **priority inversion**. Scientists solved this problem by using priority inheritance. VxWorks mutex objects has a boolean parameter to decide whether priority inheritance should be performed, if it is on, the low priority weather data thread will inherit the priority of the high priority information bus thread blocked on it while it hold the mutex, causing it has higher priority than the medium priority communication thread, thus preventing the priority inversion.