NTNU | Norwegian University of Science and Technology

Operating Systems

Example solutions for Theoretical Exercise 5

Michael Engel

5.1 Scheduling

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

Process	Burst time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

5.1 Scheduling

Process	Burst time	Priority	
P1	10	3	🟥
P2	1	1	🟦
P3	2	3	🟨
P4	1	4	🟩
P5	5	2	🟧

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

a. Draw four Gantt charts illustrating the execution of these processes using FCFS, SJF, a nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1) scheduling.

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
FCFS	P1										P2	P3		P4	P5				
SJF	P2	P4	P3		P5					P1									
Prio	P2	P5					P1										P3		P4
RR	P1	P2	P3	P4	P5	P1	P3	P5	P1	P5	P1	P5	P1	P5	P1	P1	P1	P1	P1

RR: P2 and P4 are finished here!

P3 is finished here!

P5 is finished here!

5.1 Scheduling

Process	Burst time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

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

	FCFS	RR	SJF	Priority
P_1	10	19	19	16
P_2	11	2	1	1
P_3	13	7	4	18
P_4	14	4	2	19
P_5	19	14	9	6

5.1 Scheduling

Process	Burst time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

c. What is the waiting time of each process for each of the scheduling algorithms in part a?

	FCFS	RR	SJF	Priority
P_1	0	9	9	6
P_2	10	1	0	0
P_3	11	5	2	16
P_4	13	3	1	18
P_5	14	9	4	1

d. Which of the schedules in part a results in the minimal average waiting time (over all processes)?

SJF (FCFS: 38/5 = 7.6; RR: 27/5 = 5.4; SJF: 16/5 = 3.2)

5.2 Starvation

Which of the following scheduling algorithms could result in starvation?

Explain why the starvation can occur.

- First-come, first-served
- Shortest job first
- Round robin
- Priority

Shortest job first and priority-based scheduling algorithms could result in starvation.

SJF: a long-running job can starve if a stream of short-running jobs comes in

Priority: a low-priority job can always be preempted by incoming higher-priority jobs

5.3 Round-Robin

Consider a variant of the RR scheduling algorithm where the entries in the ready queue are pointers to the PCBs.

a. What would be the effect of putting two pointers to the same process in the ready queue?

In effect, that process will have increased its priority since by getting time more often it is receiving preferential treatment.

5.3 Round-Robin

Consider a variant of the RR scheduling algorithm where the entries in the ready queue are pointers to the PCBs.

b. What would be the major advantages and disadvantages of this scheme?

The advantage is that more important jobs could be given more time, in other words, higher priority in treatment. The consequence, of course, is that shorter jobs will suffer.

5.3 Round-Robin

Consider a variant of the RR scheduling algorithm where the entries in the ready queue are pointers to the PCBs.

c. How would you modify the basic RR algorithm to achieve the same effect without the duplicate pointers?

Allot a longer amount of time to processes deserving higher priority. In other words, have two or more quantums possible in the Round-Robin scheme.

5.4 Scheduling behavior

Explain the differences in the degree to which the following scheduling algorithms discriminate in favor of short processes:

- FCFS

discriminates against short jobs since any short jobs arriving after long jobs will have a longer waiting time.

- RR

treats all jobs equally (giving them equal bursts of CPU time) so short jobs will be able to leave the system faster since they will finish first.

- Multilevel feedback queues

Multilevel feedback queues work similar to the RR algorithm — they discriminate favorably toward short jobs.

5.5 Corner cases

Consider a preemptive priority scheduling algorithm based on dynamically changing priorities. Larger priority numbers imply higher priority. When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate α; when it is running, its priority changes at a rate β. All processes are given a priority of 0 when they enter the ready queue. The parameters α and β can be set to give many different scheduling algorithms.

- What is the algorithm that results from $\beta > \alpha > 0$?

FCFS (the priority of a running process grows faster than the ones of the waiting processes)

- What is the algorithm that results from $\alpha < \beta < 0$?

LIFO (last-in, first-out; the most recently arrived process has the highest priority of 0, all others that have waited or executed have negative prios)