DAT 103
Datamaskiner og operativsystemer (Computers and Operating Systems)

Supplementary exercises (Set 3)

Problem 1

Consider an operating system running on a computer system with 16 cores uses the *many-to-many* model to map user threads to kernel threads. If N user threads are created for a particular process and $N > 16$, how many kernel threads will be allocated to this process?

Solution.
At most N

Problem 2

Describe the differences among short-term, and long-term scheduling.

Solution.
Short-term (CPU scheduler) selects from jobs in memory those jobs that are ready to execute and allocates the CPU to them.

Long-term (job scheduler) determines which jobs are brought into memory for processing.

The primary difference is in the frequency of their execution. The short-term must select a new process quite often. Long-term is used much less often since it handles placing jobs in the system and may wait a while for a job to finish before it admits another one

Problem 3

Describe the actions taken by a kernel to context-switch between processes.

Solution.
In general, the operating system must save the state of the currently running process and restore the state of the process scheduled to be run next. Saving the state of a process typically includes the values of all the CPU registers in addition to memory allocation. Context switches must also perform many architecture-specific operations,including flushing data and instruction caches.

Problem 4

What resources are used when a thread is created? How do they differ from those used when a process is created?
Solution.
Because a thread is smaller than a process, thread creation typically uses fewer resources than process creation. The latter requires allocating a process control block (PCB), a rather large data structure, while the former involves allocating a small data structure to hold a register set, stack, and priority.

Problem 5

What advantage is there in having different time-quantum sizes at different levels of a multilevel queueing system?

Solution.
Processes that need more frequent servicing – for instance, interactive processes such as editors – can be in a

queue with a small time quantum; otherwise, processes can be in a queue with a larger quantum, requiring fewer context switches to complete the processing and thus making more efficient use of the computer.

Problem 6

Explain the how the following scheduling algorithms discriminate either in favor of or against short processes:

(a) FCFS

(b) RR

(c) Multilevel feedback queues

Solution.

(a) against short jobs, since any short job arriving after a long job will have a longer waiting time.

(b) treats all jobs equally, so short jobs will be able to leave the system faster, since they will finish first.

(c) work similarly to the RR algorithm

Problem 7

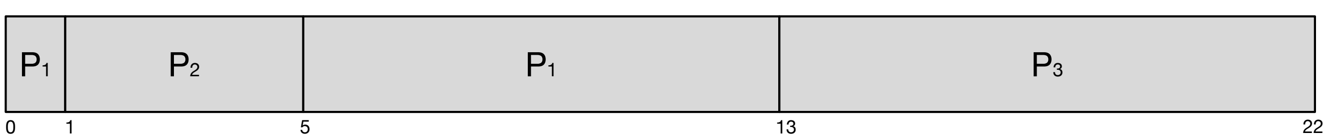
Consider the following table of arrival time and burst time for three processes P_1, P_2 and P_3:

Process	Arrival Time	Burst Time
P_1	0	9
P_2	1	4
P_3	2	9

The shortest-remaining-time-first (i.e., preemptive shortest job first) scheduling algorithm is used. Scheduling is carried out only at arrival or completion of processes.

(a) Draw the Gantt Chart of the execution.

Solution.



(b) What is the average waiting time for the three processes?

Solution.
5

Process P_1 is allocated processor at 0 ms as there is no other process in ready queue. P_1 is preempted after 1 ms as P_2 arrives at 1 ms and burst time for P_2 is less than remaining time of P_1. P_2 runs for 4ms. P_3 arrived at 2 ms but P_2 continued as burst time of P_3 is longer than P_2. After P_2 completes, P_1 is scheduled again as the remaining time for P_1 is less than the burst time of P_3. P_1 waits for 4 ms, P_2 waits for 0 ms and P_3 waits for 11 ms. So average waiting time is $(0+4+11)/3 = 5$.

Problem 8

Consider the following table of processing time and period for two periodic processes P_1 and P_2:

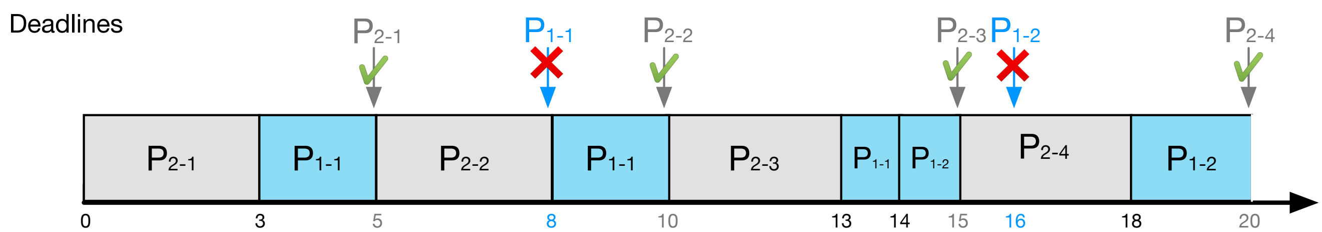
Process	Processing time	Period
P_1	5	8
P_2	3	5

The completion deadlines of each process are the beginning of its next period. For example, the completion deadlines of P_1 are at time $8, 16, 24, \ldots$, while the completion deadlines of P_2 are at time $5, 10, 15, 20, \ldots$.

8.1 Use *Rate-Monotonic Scheduling* to schedule the two processes, where the priority is assigned based on the inverse of each process' period. At $time = 20$, can the two processes meet all their deadlines? Draw the Gantt Chart of the execution to justify your answer.

Solution.

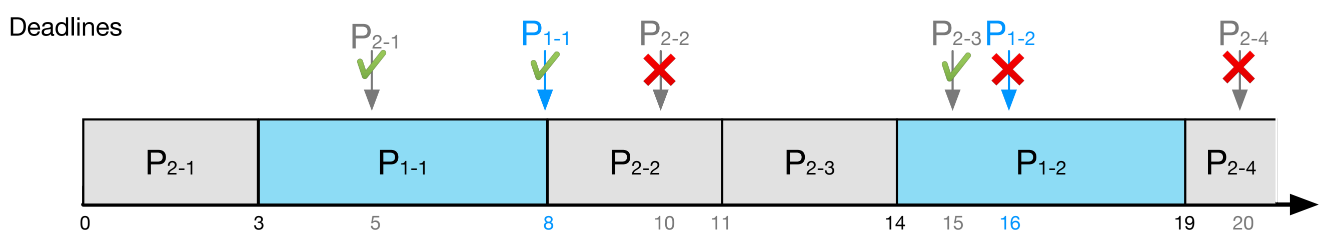
Rate-Monotonic Scheduling schedules periodic tasks using a ***static*** priority policy with preemption, i.e., the priority does not change over time. Since the priority is assigned based on the inverse of each process' period, that means P_2 has higher priority than P_1, and this priority does not change. In the following gantt charts, P_{i-j} is used to indicate the j-th occurance of P_i. As you can see, P_2 always meets its deadlines, while P_1 always misses its deadlines.



8.2 Use *Earliest Deadline First* to schedule the two processes. At $time = 20$, can the two processes meet all their deadlines? Draw the Gantt Chart of the execution to justify your answer.

Solution.

Earliest Deadline First schedules periodic tasks using a ***dynamic*** priority policy with preemption, i.e., the priority *does change* over time. In this subquestion, P_2 has higher priority than P_1 in the beginning, however, it will change dynamically over time. From the gantt chart below, both P_1 and P_2 meet their first deadline, but have problem in keeping the rest. Note that P_{2-3} is scheduled prior to P_{1-2} is due to the former has an earlier deadline.



Hint: Both Rate-Monotonic Scheduling and Earliest Deadline First are priority scheduling with preemption.