Real-Time Scheduling and Analysis

Due: Monday, April 11 at 11:59pm, with 2 hour grace period. This due date is firm.

NO LATE ASSIGNMENTS accepted.

You can use any language of your choosing.
Submit via Github (use "RTOS" tag)

Part 1: Static Scheduling Tool

Write a simple software tool to read a periodic task set and generate a non-preemptive uni-processor static schedule for that workload. The output is a table that identifies the time each job is executed within the hyperperiod.

To generate the table, **simulate a non-preemptive earliest deadline scheduling algorithm for one hyperperiod. Use a deadline monotonic priority assignment for tasks.** Implement a simulated ready queue, in which jobs are placed at a rate matching their period. Simulate the dispatching based on priority of these periodic tasks at appropriate scheduling points (e.g. when a job is complete or released). When the processor is free, the scheduler selects the highest priority job in the queue and executes it non-preemptively until it completes. Another job can only be selected after that. If you run this simulation for one hyperperiod, you will get a series of start and complete times for every job for every task. You can use any built-in priority queue to maintain your ready queue or use one from the web. If you use the web, please cite your resource in your code.

Write a verification subroutine that, once a table has been created, confirms that each job has met its deadline. Have it list all jobs that will miss their deadline. The utilization of these workloads is quite low, thus it isn't likely that you would get any missed deadlines, but the functionality should still exist.

Run the following workloads through the tool. If a workload cannot be feasibly scheduled by your tool, you are permitted to re-write the workload by hand to split tasks with large worst-case execution times into two or more tasks whose combination performs the same amount of work within the same timing constraints.

Workload 1

period	WCET	offset	deadline
10	2	0	10
10	2	0	10
20	1	5	20
20	2	5	20
40	2	5	30
40	2	5	30
80	2	10	60
80	2	10	60

Workload 2

period	WCET	offset	deadline
20	4	0	15
20	1	5	20
30	2	5	30
30	1	5	30
50	1	10	40
50	1	10	40
50	2	25	50
50	5	25	50

Start of the Table for Workload 2 (Some tasks have equal priority, thus results might look slightly different):

TIME	JOB
0	J11
4	Idle
5	J21
6	J31

8	J41
9	Idle
…..	

Part 2: Time Demand Analysis Tool

Write a simple software tool to perform a uni-processor time demand analysis on a periodic task set that is scheduled using a preemptive fixed priority discipline. Include a capability to model user-specified deadlines that are smaller than the period. Include a capability to model non-preemptable activities and mutually exclusive access within tasks (i.e. include support for blocking times, including self-suspension).

In the generated output report,

- Show the input model (i.e. the task set).
- For each task, list blocking times used in the analysis.
- Results of the analysis (i.e. the response time for each task).
- Where a workload is infeasible, indicate which subset of the tasks will always meet their deadlines and which subset is at risk of missing deadlines.

Given the following workload:

T1 = (3,1), T2 = (5,1), T3 = (7,2.5), T4 = (16,1)
Your algorithm should produce: w1 = 1, w2 = 2, w3 = 7.5 (and/or missed deadline), w4 = 14

Use this software to analyze the following workloads to determine which are feasible and which are not. For all workloads, use a deadline monotonic priority assignment.

Workload 1
Task 1 has period and deadline 25ms, execution time 8ms.
Task 2 has period and deadline 50ms, execution time 13ms.
Task 3 has period and deadline 100ms, execution time 40ms.

Workload 2
Task 1 has period and deadline 20ms, execution time 5ms, with a 1ms self-suspension.
Task 2 has period and deadline 30ms, execution time 12ms.
Task 3 has period and deadline 50ms, execution time 15ms.

Workload 3
Task 1 has period 25ms, deadline 25ms, execution time 5ms.
Task 2 has period 30ms, deadline 23ms, execution time 12ms.
Task 3 has period 50ms, deadline 45ms, execution time 15ms.

Workload 4
This is identical to workload 1, except tasks 2 and 3 access a shared data structure and contain mutually exclusive critical sections that may execute for 2ms. (The 2ms is included in the execution times, i.e. task 2 has a maximum execution time of 13 ms, but it may be in a mutually exclusive section for up to 2ms at a time

during that 13ms.) Assume mutual exclusion is achieved by masking interrupts, i.e. non-preemption is used to insure mutual exclusion.