HW # 5 p 1

Thursday, August 29, 2024 5:44 PM

Q6-6: Using RMS Calculations

a) $U = \sum_{i=1}^{2} \frac{T_i}{\tau_i} = \frac{10ms}{1} + \frac{10ms}{100ms} = 0.11$

b) $U = \sum_{i=1}^{3} \frac{T_i}{\tau_i} = \frac{25ms}{100ms} + \frac{15ms}{80ms} + \frac{5ms}{40ms} = 0.5625$

c) $U = \sum_{i=1}^{3} \frac{T_i}{\tau_i} = \frac{1ms}{10ms} + \frac{0.2ms}{1ms} + \frac{0.05ms}{0.2ms} = 0.55$

Q6-9: Ready and waiting states determine the ability for a process to run. When an operating system (OS) is deciding which process to run next, it needs to know the process state, where a process in the ready state is currently able to execute and a process in the waiting state is not able to due to whichever reason (e.g. waiting for I/O, a timer preventing execution at the moment).

Q6-12: Shortest interval refers to the hyperperiod (LCM). So the shortest interval for processes with deadlines:

a) 2,5,10 is **10**
b) 2,4,5,10 is **20**
c) 3,4,5,6,10, is **60**

Q6-13: What we know:

- Hyperperiod is 200
- Necessary # of runs given hyperperiod: P1: 200/200=1 1, P2: 200/10=20, P3: 200/40=5, P4: 200/50 = 4
- Needed execution time: P1: 1*4=4, P2: 20*1=20, P3: 2*5=10, P4=6*4=24

As it stands, 200 - (4 + 20 + 10 + 24 + x) = 142 - x, where x is the proposed needed execution time of another P1 instance

With an added P1, we need an additional 4 time units for run P1, which brings us to 142-4=**138 remaining time units.**
THEREFORE (FINAL ANSWER): **Yes, we can add another instance and meet all deadlines**

Q6-14:
Current Hyperperiod: 100
Current utilization: 1/10 + 3/25 + x/50 + 10/100 = 10 + 12 + 2x + 10/100 = 32 + 2x/100

To stay within this utilization, we need U_{max} = 100 ≥ 32 + 2x, which if solved gives (100-32)/2 = 34 ≥ x
THEREFORE (FINAL ANSWER): **Max execution time has to be 34**

HW # 5 p 2

Thursday, August 29, 2024 5:44 PM

Q6-17: LCM = 12 [parts a) and b) done together]
RMS policy needed times: P1: 1*(12/3)=4, P2: 1*(12/4)=3, P3:1*(12/12)=1 => 4+3+1=8
EDF policy utilizations = 1/3 + 1/4 + 1/12 = 8/12 = 0.6667

RMS Policy				EDF Policy			
Time	Running Process	Queue	Fully Done	Time	Running Process	Deadlines	Fully Done
0	P1			0	P1		
1	P2			1	P2		
2	P3	P1		2	P3	P1	
3	P1	P2	P3	3	P1	P2	P3
4	P2		P3	4	P2		P3
5	Idle	P1	P3	5	P1	P1	P3
6	P1		P3	6	P1		P3
7	Idle	P2	P3	7	P2	P2	P3, P1
8	P2	P1	P3	8	Idle	P1	P3, P2, P1
9	P1		P3,P2	9	Idle		P3, P2, P1
10	Idle		P3,P2,P1	10	Idle		P3, P2, P1
11	Idle	P1,P2,P3	P3,P2,P2	11	Idle	P1,P2,P3	P3, P2, P1

HW # 5 p 2 - Spreadsheet

HW # 5 p 3

Thursday, August 29, 2024 5:44 PM

Q6-19: LCM = 30 [parts a) and b) done together]
RMS policy needed times: P1: 1*(30/2)=15, P2: 1*(30/3)=10, P3: 2*(30/10)=6 => 15+10+6 = 31
EDF policy utilizations: 1/2 + 1/3 + 2/10 = 31/30 = 1.0333

HW # 5 p 2 - Spreadsheet

RMS Policy				EDF Policy			
Time	Running Process	Queue	Fully Done	Time	Running Process	Deadlines	Fully Done
0	P1			0	P1		
1	P2	P1		1	P2	P1	
2	P1	P2		2	P1	P2	
3	P2	P1		3	P2	P1	
4	P1			4	P1		
5	P3 - 1/2	P1, P2		5	P3	P1, P2	
6	P1			6	P1		
7	P2	P1		7	P2	P1	
8	P1	P2		8	P1	P2	
9	P2	P1, P3		9	P3	P1, P3	
10	P1			10	P1		
11	P3 - 2/2	P1		11	P2	P1, P2	
12	P1	P2		12	P1		
13	P2	P1		13	P2	P1	
14	P1			14	P1	P2	
15	P3 - 1/2 (from t=9)	P1, P2		15	P3	P1	
16	P1			16	P1		
17	P2	P1		17	P2	P1, P2	
18	P1	P2		18	P1		
19	P2	P1, P3		19	P3	P1, P3	
20	P1			20	P1	P2	
21	P3 - 2/2	P1		21	P2	P1	
22	P1	P2		22	P1		
23	P2	P1		23	P3	P1, P2	
24	P1			24	P1		
25	P3 - 1/2 (from t=19)	P1, P2		25	P2	P1	
26	P1			26	P1	P2	
27	P2	P1		27	P2	P1	
28	P1	P2		28	P1		
29	P2	P1, P2, P3	P1	29	P3	P1, P2, P3	

HW # 5 p 4

Thursday, August 29, 2024 5:44 PM

Q6-21: LCM is 120

a) Needed time for RMS: P1: 2*4=8, P2: 5*3=15, P3: 7*1=1, P4: 5*2=10, P5: 1*8=8 => 8+15+1+10+8=**42**
b) With one unit for context switch:
    a. P1: (2+1)*4=12
    b. P2: (5+1)*3=18
    c. P3: (7+1)*1=8
    d. P4: (5+1)*2=12
    e. P5: (1+1)*8=16

    Total: 12+18+8+12+16=**66**

Q6-24: In this case, the OS will check if P2 is requesting the resource (in this case the ADC), then P2's priority will be promoted in order for P1 not to access the resource until after it's finished. Then P1 can run when P2 has lowered priority back to its normal value.

Q6-25: An interrupt service handler (ISH) and routine (ISR) form the combination that allows for responding to a hardware device (e.g. keyboard, peripheral) interrupt. The ISH, comprising the least of the "workhorse", is the minimum requirements to respond to the device and runs as part of the kernel. The ISR comprises the majority of the work, containing code to process the interrupt. Running as thread complements the RTOS's scheduling policies.

Q6-26: The dual kernel approach adds the specialized co-kernel, which handles all the interrupts in order to act on real-time processes. This is used in the POSIX standard to allow Unix system to support its real-time capabilities not found in the base OS.