Project #2 – CPU Scheduling

Due: Thursday, April 21 (beginning of class)

This project is to write a real-time CPU scheduler that works on earliest-deadline-first (EDF) scheduling algorithm. Modify the Round Robin scheduling algorithm in the Lab #5 to accommodate deadline based scheduling for a set of threads representing periodic tasks. Use EDF scheduling algorithm, described in Section 6.6.4 on page 288 and 289.

You should create one scheduler thread, one timer thread, and multiple threads which a user may request.

- The timer thread will interrupt (send a signal to) the scheduler whenever a new period data has come.
- You need to create a non-preemptive EDF scheduler thread. This scheduler is interrupted whenever a new data has come, but it will wait until currently running thread finishes its processing. Then, it will compare the deadline of the threads which can run right now and choose a thread that has the earliest deadline.
- Each execution thread will run during the execution time of each thread and send a signal to the scheduler after finishing.

Note that all time values should be second unit. You should use mutex locks and semaphores, described in Section 5.9.4 on page 237 and 238.

Show one data set for which your scheduler is successful and one data set where it fails (misses deadline).

In case of success, you can choose these 3 threads as an example.

Input:

```
$ ./scheduling 3
Execution time for Thread 0: 10
Execution time for Thread 1: 15
Execution time for Thread 2: 5
Period for Thread 0: 30
Period for Thread 1: 40
Period for Thread 2: 50
How long do you want to execute this program (sec): 150
```

Output:

```
Thread 0 is now being executed
1
2
3
4
5
6
```

```
7
8
9
10
Thread 1 is now being executed
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
Thread 2 is now being executed
26
27
28
29
30
Thread 0 is now being executed
31
32
33
34
35
36
37
38
39
40
Thread 1 is now being executed
41
42
43
44
45
46
47
48
```

```
49
50
51
52
53
54
55
Thread 2 is now being executed
56
57
58
59
60
Thread 0 is now being executed
61
62
63
64
65
66
67
68
69
70
CPU is idling now
71
72
73
74
75
76
77
78
79
80
Thread 1 is now being executed
81
82
83
84
85
86
87
88
89
90
```

```
91
92
93
94
95
Thread 0 is now being executed
96
97
98
99
100
101
102
103
104
105
Thread 2 is now being executed
106
107
108
109
110
CPU is idling now
111
112
113
114
115
116
117
118
119
120
Thread 0 is now being executed
121
122
123
124
125
126
127
128
129
130
Thread 1 is now being executed
131
```

```
132
133
134
135
136
137
138
139
140
141
142
143
144
145
CPU is idling now
146
147
148
149
Killed
```

In case of failure, you can choose these 3 threads as an example.

Input:

```
$ ./scheduling 3
Execution time for Thread 0: 15
Execution time for Thread 1: 15
Execution time for Thread 2: 10
Period for Thread 0: 30
Period for Thread 1: 40
Period for Thread 2: 50
How long do you want to execute this program (sec): 150
```

Output:

```
These threads can't be scheduled. Program will exit.
```

Submission:

Make the followings as cmpe320p2_lastname.tar.gz and then upload it on Dropbox.

- Source code
- Any extra files needed to run your program
- Documentation
  - Description of the program
  - Algorithm
  - Outputs

You need to submit the hardcopy of your documentation and all source code on due date. Your file should include student, course, and instructor information. Here is an example.

```
/*
* Project 2: CPU Scheduling
* Programmer: your name and your partner's name
* Course: CMPE 320
* Section: 1 (9-10:50am) or 2 (11-12:50pm)
* Instructor: S. Lee
*/
```

Marking Criteria:

- Submission of required file only, with the information of student, instructor, and course on the submitted file
- Warning free compilation and linking of executable with proper name
- Source code
- Readability, suitability & maintainability of source code
- Documentation
- Demonstration