Lab 3 – Real-time Scheduling over Linux

- 6% of your final grade
- Individual lab, not a team lab, no collaboration is allowed!
- Check out our Academic Integrity policy here. If two or more students' lab assignments are very similar, all will receive a minimum penalty of a zero on the given lab assignment.

Lab 3 consists of three sections.

Section 1 (30%): In this section, you are required to realize the rate monotonic scheduling (RMS) scheduling algorithm.

Please read this sample code in detail. Try to use the following example as input to see how sample code will work:

	Execution	Period
P1	10	25
P2	15	60

The sample code does not consider preemptive scenarios. Your task is to modify the sample code so that it could schedule well while preemptive scenarios occur. You can use the following example to test if your modified code could handle preemptive situation:

	Execution	Period
P1	20	50
P2	35	100

Section 2 (30%): In this section, you are required to realize the earliest deadline first (EDF) scheduling algorithm.

Based on the same sample code, think about how to modify the judgement so as to realize EDF algorithm. You can try the following sample to see if your modified version works well:

	Execution	Period
P1	10	20
P2	25	50

Please double check with the following example to see if your algorithm could handle preemptive scenario:

	Execution	Period
P1	25	50
P2	35	80

Section 3 (40%): Priority inversion and inheritance

1. In this section, you should create at least three threads to realize priority inheritance. The design could be as following

```
void function()
{
  //do some work here.
}

void funtion1()
{
  //function 1 for thread 1.
  //do some work here
}

void function2()
{
//function 2 for thread 2.
//do some work here and call function()
}

void function3()
{
//function 3 for thread 3.
//do some work here
}

void main()
{
   //create mutex and initialize it.
//create thread 1, thread 2 and thread 3
//set different priority for each thread
//set handler for thread 1, thread 2 and thread 3 respectively. For instance, function 1 for thread 1, function 2 for thread 2 and function 3 for thread 3
}
```

2. For “do some work” in design chart, you may choose to print some messages to terminal. For example: “Thread 1 in”, “Thread 1 starts”, “Thread 1 ends” and “Thread 1 out” are used to simply describe the work each thread will do.

3. Please think carefully on what priority inheritance is e.g., advent order of each thread. For example, if thread 1 has the lowest priority and thread 3 has the highest priority. The definition of priority inheritance is that thread 1 should be earlier than thread 3 and thread 2 is later than thread 3.

4. The following library, data type and function may be useful to write your code.

1. Library and header files: “stdio.h”, “stdlib.h”, “unistd.h”, “pthread.h” and “sched.h”
2. Data types:
    1. pthread_t: thread variable
    2. pthread_mutex_t: mutex variabl
    3. pthread_mutexattr_t: mutex attribute variable
    4. sched_param: structure variable for buffering thread priority
3. Functions:
    1. pthread_mutexattr_setprotocol(): set protocol for mutex attribute variable
    2. pthread_mutex_init(): initialize mutex variable
    3. pthread_mutex_lock(): lock mutex variable
    4. pthread_mutex_unlock():unlock mutex variable
    5. pthread_create(): create a thread
    6. pthread_setschedparam(): set priority, policy for a thread
    7. pthread_getschedparam(): get priority, policy of a thread
    8. pthread_join(): wait for all threads to finish before main() can exit.