Operating Systems Lab
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: 320232
Date: 2016-09-15
Deadline: 2016-09-21

Problem Sheet #3

Problem 3.1: *multi-threaded coin flipping* (7 points)

You have 20 coins on the table lying in a row. P persons flip all coins on the table N times. Write a program that emulates this by using threads, one thread emulating one person. By default, there are $P = 100$ persons and each person flips each coin $N = 10000$ times. Provide command line options that allow to control the number of persons and the number of coin flips per person.

Use a global array of characters to represent the coins, where an `X` stands for one side, while a `O` represents the other side. Print the coins on the standard output before the coin flipping starts and print the coins when all persons have finished flipping coins. In addition, print the time it took for all persons to flip the coins.

You must ensure that no coin is flipped by another person, while one person has his turn. Implement two strategies:

1. In the first strategy, you use a global lock to protect the coins, i.e., a person first obtains a lock covering all coins and then the person flips all coins and finally, when done flipping coins, the person releases the lock.
2. In the second strategy, there is a lock for each coin, i.e., a person obtains a lock for a coin, flips the coin, and releases the lock immediately after each coin flip.

Let the program measure the time it took all persons to flip all coins and write the results to the standard output.

An example execution might look as follows:

```
$ ./coins
coins: OOOOOOOOOOXXXXXXXXXX (start - global lock)
coins: OOOOOOOOOOXXXXXXXXXX (end - global lock)
100 threads x 10000 flips:    100.800 ms

coins: OOOOOOOOOOXXXXXXXXXX (start - coin lock)
coins: OOOOOOOOOOXXXXXXXXXX (end - coin lock)
100 threads x 10000 flips:    468.660 ms
```

Problem 3.2: *coin flipping performance* (3 points)

Execute your coin flipping program with different parameters to fill in the following tables:

threads	flips	global [ms]	local [ms]
1	10000		
10	10000		
100	10000		
1000	10000		

threads	flips	global [ms]	local [ms]
100	1		
100	10		
100	100		
100	1000		
100	10000		

threads	**flips**	**global [ms]**	**local [ms]**
1	10000		
10	1000		
100	100		
1000	10		
10000	1		

Discuss the results you have obtained. Pay attention to the number of cores you have available and what the variation of the parameters mean in terms of the total number of coin flips.