/**
*18648 Fall 2015 Lab2
* Group 10
* -> Mengye Gong
* -> Nishant Parekh
* -> Ziyuan Song
*Description : Write up for lab 2

1. What is the difference between concurrent execution and parallel execution?

Answer: Concurrent execution means more than one tasks can start, run, and finish within the same time period.

Parallel execution is one way to accomplish concurrency. It means multiple CPUs work on more than one tasks at the same time. But it's not the only way to implement concurrency. For example, task switching can also implement concurrency.

2. Imagine Dexter needs to instrument his kernel to collect some data points. The points are generated within the kernel at some memory address and need to be visualized in a userspace application. Suppose each point appears periodically at one memory address in the kernel, for example corresponding to a memory-mapped register of a device. If a point is not read before the next one is ready, then the point is lost.

Suppose that on Dexter's platform

- **the computation overhead of reading the point from the initial memory address is negligible**
- **it takes 100μs to complete a round-trip between user-space and kernel space**
- **it takes 10ns to copy one data point value from kernel memory to user memory.**

(a) (2 points) As a trained syscall writer, Dexter creates a syscall to retrieve the one data value and call the syscall in a tight while loop from a userspace application. Suppose the data points are generated at a rate of 1,000 points per second. What fraction of points, if any, are lost due to the overhead delay?

Answer: The deadline is 1ms. Computation time equals 100us + 10ns. So Deadline is larger than Computation time, there is no lost.

(b) (2 points) Dexter is not satisfied with a slow sampling rate, so from now on, suppose the data points are generated at a rate of 100,000 points per second. Assuming the same implementation approach, what fraction of points, if any, are lost due to the overhead delay under the faster sampling rate?

Answer: The deadline is 10us. Computation time is 100us + 10ns = 100.01us. So during the computation time, there will be 9 data points lost. The fraction is 9/10

(c) (2 points) Dexter abandons the naive approach and changes his implementation to amortize the delay of kernel-user crossing over many points by buffering the points in kernel memory. For a buffer size of 1000 points, what fraction of points, if any, are lost due to the overhead delay?

Answer: Deadline is 10us. After the buffer is full, which means there are 1000 points in the buffer, the time needed to copy data to user memory is 10ns * 1000 + 100us = 110us. In this period, 11 points is lost. So the fraction is 11/(1000 + 11) = 11/1011

(d) (3 points) Dexter is not satisfied with losing any points at all. How can he improve his implementation to achieve this?

Answer: Dexter can use two buffers in the platform. When one of the buffer is full, all points could be read by syscall. During this period of reading, another buffer will be filled. Only to make sure that the other buffer cannot be filled fully during this delay time, there will be no lose of points.