# CSCI 325
# Introduction to Parallel Systems and GPU Programming

## Lecture 3
## C++ Multithreading

Dr. Talgat Turanbekuly

## Table of contents

## Concurrency Problems **Deadlock**

Two or more threads get some resources (lock) and then wait each other indefinitely to release the lock. It happens when threads acquire the lock in different order.

```
T1
synchronize(A)
{
    synchronize(B)
    {

    }
}
```

```
T2
synchronize(B)
{
    synchronize(A)
    {

    }
}
```

## Concurrency Problems **Deadlock Example**

Java Example

C++

offer your solutions

## Concurrency Problems **Deadlock Solutions**

- Try not to use block threads within each other (cycles, nests)
- Use timed waiting to avoid indefinite block
- Any other?

## Concurrency Problems **Livelock**

**THREAD 1**

```
while(y < 2)
{
    synchronized
    {
        x++;
        y = x;
    }
}
```

**THREAD 2**

```
while(y > -2)
{
    synchronized
    {
        x--;
        y = x;
    }
}
```

**possible output**

```
THREAD 1
x = 0;
x = 1;
THREAD 2
x = 0;
x = -1;
THREAD 1
x = 0;
THREAD 2
x = -1;
THREAD 1
x = 0;
x = 1;
…
```

## Concurrency Problems **Livelock Example**

C++ Livelock Example

How to figure out the error?

Offer solutions to the problem

## Concurrency Problems **Livelock Solutions**

Identify livelock:

- What if use counters for repeated situations?

Solution depends to the problem:

- Use different timed waiting for concurrent threads;

## Concurrency Problems **Starvation**

**THREAD 1**

```
while(x < 4)
{
    synchronized
(sharedObject)
    {
        x++;
    }
}
```

**THREAD 2**

```
while(x < 4)
{
    synchronized
(sharedObject)
    {
        x++;
    }
}
```

**THREAD N**

```
while(x < 4)
{
    synchronized
(sharedObject)
    {
        x++;
    }
}
```

**possible output**

```
THREAD 1
 x = 1;
 x = 2;
 x = 3;
 x = 4;
THREAD 2
 x = 1;
 x = 2;
 x = 3;
 x = 4;
THREAD N
 x = 1;
 x = 2;
 x = 3;
 x = 4;
```

## Concurrency Problems **Starvation Example**

C++ Starvation Example

## Concurrency Problems **Starvation Solutions**

Identify starvation:

- Are threads running randomly or in particular order?

Solution:

- Use scheduling

## Summary

Threads organization is important to avoid liveness problems: deadlock,  livelock and starvation.

Famous problems in concurrency: Producer-Consumer Relationship (bounded-buffer problem), dining philosophers, readers-writers problem etc. have several solutions each.

## Class work and Homework

- Read about liveness problems: deadlock, livelock and starvation
- Understand and implement reasons and solutions
- Please check if cuda is installed on your PC
- Sign up to learn.nvidia.com

## Resources

Paul J. Deitel. C++20 Fundamentals, 3rd Edition. 2024;

Harvey Deitel, and Paul J. Deitel. C++20 for Programmers: An Object's-Natural Approach, 3rd Edition, 2022;