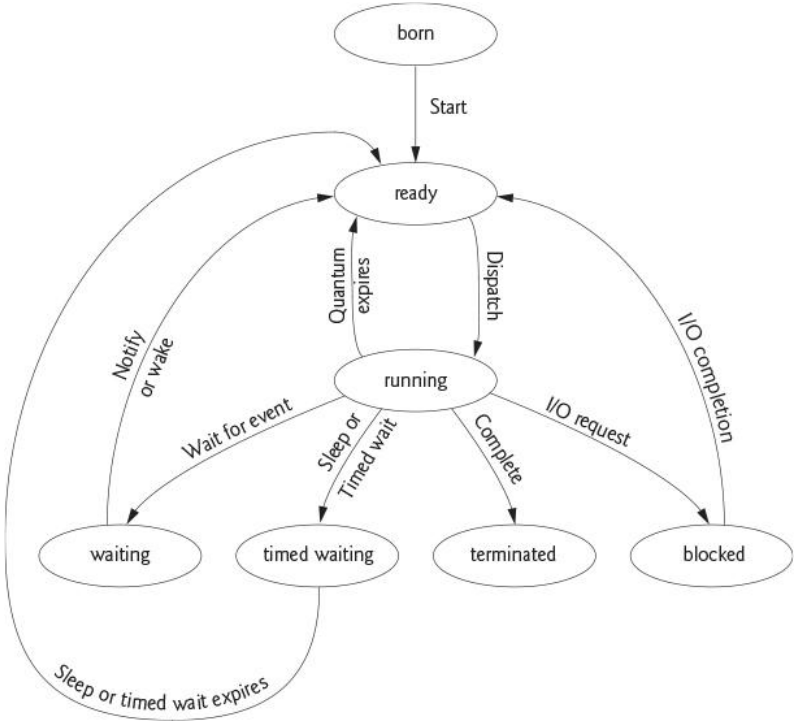
CSCI 325
Parallel Systems and GPU Programming

Lecture 2
C++ Multithreading

Dr. Talgat Turanbekuly

Table of contents

C++ thread states



Born - thread begins its life cycle and remains until the program starts.

Ready - OS allocates processor time (quantum/timeslice) and sends the thread to **Running** state. Then the thread returns to the **Ready** state. Another thread is assigned to the processor.

Thread scheduling - allocating threads per time.

Waiting - thread A is paused until thread B completes and notifies thread A or specific time is passed.

Timed waiting - thread is in the state until time interval is passed or specific task is completed.

Blocked - the thread is waiting until I/O task is completed.

Terminated - when the thread completes its task.

Harvey Deitel, Paul Deitel and David Choffnes, “Chapter 4, Thread Concepts.” Operating Systems, 3/e, p. 153. Upper Saddle River, NJ: Prentice Hall, 2004.

C++ multithreading

Example 1

Thread Synchronization

It is possible to run parallel threads and execute several tasks.

What if several threads try to change value of the same object?

Data Race

Thread-1 gets value of an object in order to change.

Thread-2 changes the value before thread-1 saves the object.

In this case thread-1 works with old value of the object.

Data Race

A ‘race condition’ may arise when two or more threads concurrently access the same memory location and at least one of the threads updates the location. It may lead to wrong results.

Thread Synchronization

problem solution:
Ensure only one thread at a time can access the object -
acquire the lock on the object.

One thread

1) gets value of the object
2) changes the value
3) saves new updated value to the object.

These three steps form **atomic operation** and cannot be divided.

Thread Synchronization

In order to manage multiple threads trying access one object:

Only one thread is given access to the object

and

the rest threads are waiting.

C++ multithreading

Example 2

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**

C++ Deadlock example

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

There are several ways to declare and operate threads in C++;

Thread synchronization allows only one thread to work with data;

Thread synchronization prevents unpredictable results;

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.

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;

Anthony Williams. C++ Concurrency in Action, Second Edition, 2019;

Class work and Homework 2 and 3

- 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

CUDA, GPU, HPC internships, MSc, PhD, jobs

https://scholarshipdb.net/scholarships?q=cuda&listed=Last-24-hours

https://scholarshipdb.net/scholarships?q=gpu&listed=Last-24-hours

https://scholarshipdb.net/scholarships?q=hpc&listed=Last-24-hours