C++ 11
std::mutex

https://youtu.be/YwZGU9OnnOM

CONTENTS

Introduction to std::mutex

How to use std::mutex

Examples

https://youtu.be/YwZGU9OnnOM

Part - 1

Introduction to std::mutex

https://youtu.be/YwZGU9OnnOM

Introduction to Mutex concepts

Mutex concepts is designed for inter-thread synchronization.,

Real World Scenario

A store has 2 Billing Counter
There is only 1 card swiping machine for card payment.
Both billing counter shares the Card Swiping Machine

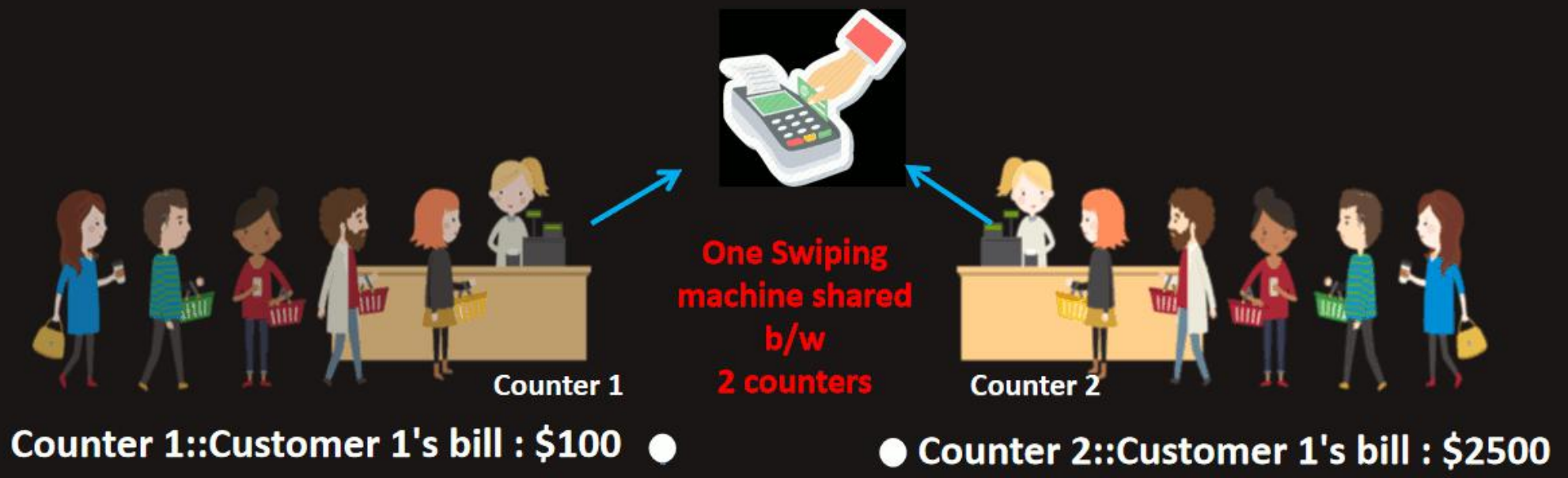
Programatic Parameters

Thread : Billing Counter
Shared resource : Card Swiping Machine
Functionality : Bill Payement

https://youtu.be/YwZGU9OnnOM

Introduction to Mutex concepts

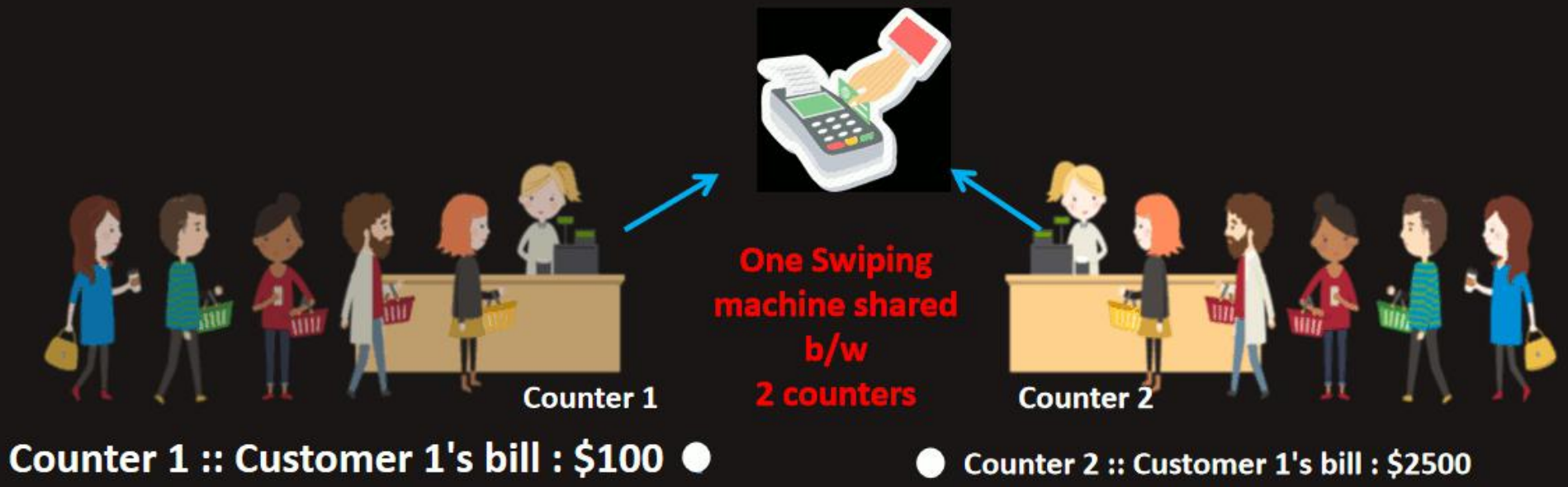
Mutex concepts is designed for inter-thread synchronization.,



https://youtu.be/YwZGU9OnnOM

Introduction to Mutex concepts

Mutex concepts is designed for inter-thread synchronization.,



https://youtu.be/YwZGU9OnnOM

Introduction to Mutex concepts

Mutex concepts is designed for inter-thread synchronization.

As per C++ Mutex concepts, a mutex need to satisfy all below,

- Lockable
- Default Constructible
- Destructible
- Not copyable
- Not movable

https://youtu.be/YwZGU9OnnOM

Introduction to Mutex concepts

All below standard library types satisfy C++ Mutex concepts

- std::mutex
- std::recursive_mutex
- std::timed_mutex
- std::recursive_timed_mutex
- std::shared_mutex

Introduction to std::mutex

Class in C++
STL library

C++ 11

Synchronization
Primitive

Stack Object

Normally

Exclusive &
Non-recursive
ownership

https://youtu.be/YwZGU9OnnOM

Introduction to Mutex concepts

Non-recursive ownership

```
1  #include <mutex>
2  std::mutex mtx;
3  void DoSomeThing() {
4      for( auto i = 0; i < 100; ++i ) {
5          mtx.lock();
6          // Synchronized critical code section
7      }
8      mtx.unlock();
9  }
```

https://youtu.be/YwZGU9OnnOM

Introduction to Mutex concepts

Non-recursive ownership

```
1  #include <mutex>
2  std::mutex mtx;
3  void DoSomeThing() {
4      for( auto i = 0; i < 100; ++i ) {
5          mtx.lock();
6          // Synchronized critical code section
7          mtx.unlock();
8      }
9  }
```

https://youtu.be/YwZGU9OnnOM

Part - 2

How to use std::mutex

https://youtu.be/YwZGU9OnnOM

How to use std::mutex?

Step 1

Include the header file <mutex>

```
#include <mutex>
```

This header file contains the implementations of class mutex

https://youtu.be/YwZGU9OnnOM

How to use std::mutex?

Step 2

Create an object of std::mutex in the required scope.

```
#include <mutex>
std::mutex mtx;
void DoSomeThing() {
    for( auto i = 0; i < 100; ++i ) {
        mtx.lock();
        // Synchronized critical code section
        mtx.unlock();
    }
}
```

https://youtu.be/YwZGU9OnnOM

How to use std::mutex?

Step 3

Using mutex object, call mutex::lock() API to accquire mutex.

```
#include <mutex>
std::mutex mtx;
void DoSomeThing() {
    for( auto i = 0; i < 100; ++i ) {
        mtx.lock();
        // Synchronized critical code section
        mtx.unlock();
    }
}
```

https://youtu.be/YwZGU9OnnOM

How to use std::mutex?

Step 4

Using mutex object, call mutex::unlock() API to release mutex.

```
#include <mutex>
std::mutex mtx;
void DoSomeThing() {
    for( auto i = 0; i < 100; ++i ) {
        mtx.lock();   // call lock() to Accquire mutex ownership
        // Synchronized critical code section
        mtx.unlock();
    }
}
```

https://youtu.be/YwZGU9OnnOM

Part - 3

Examples

https://youtu.be/YwZGU9OnnOM

Example

Purpose : Print 2 different characters, 20 times continuously from 2 threads

Condition : Characters SHOULD NOT GET MIXED UP. Order of lines may vary

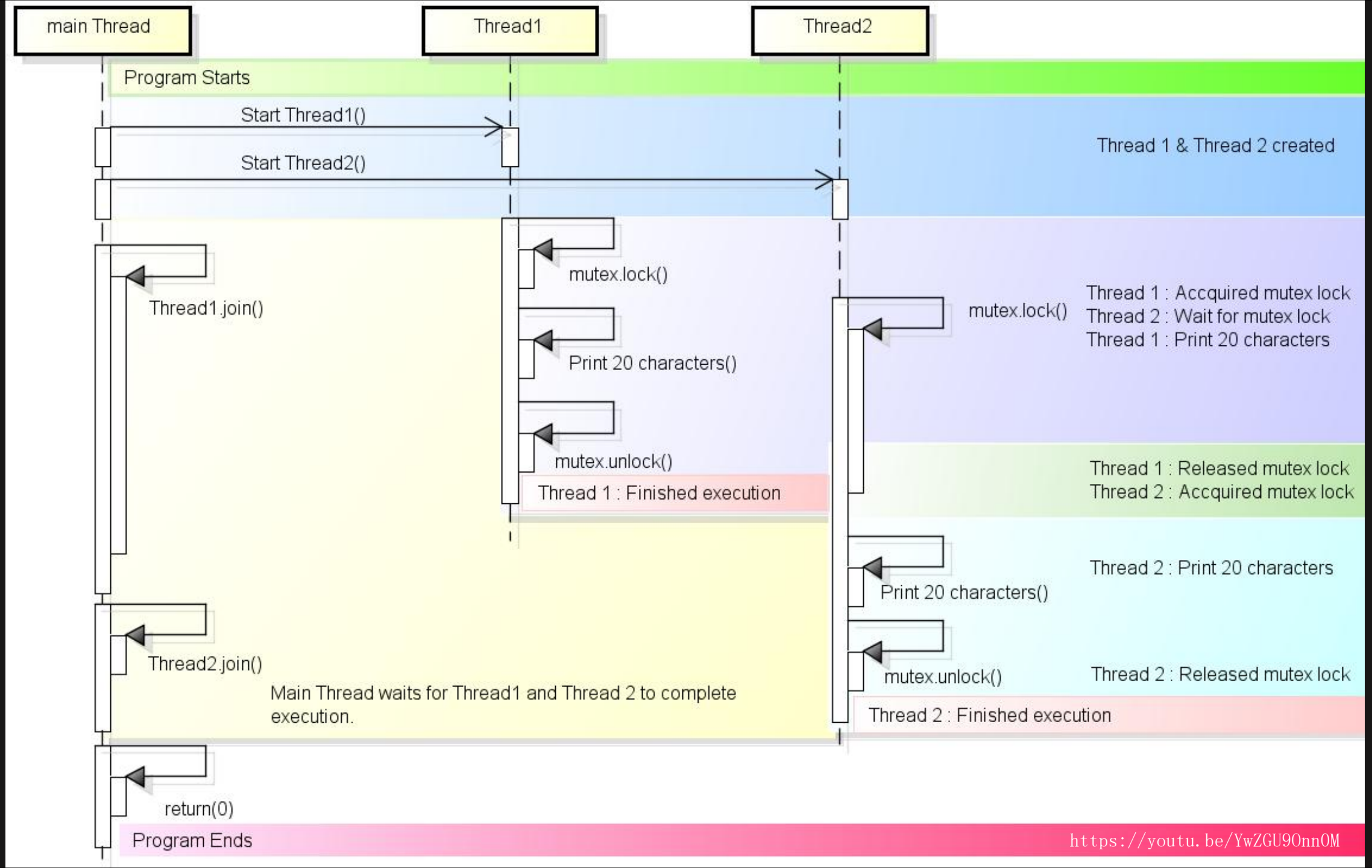
Expected : **********
&&&&&&&&&&&&&&&&&&&&

Wrong : &*&***&*&*&&**

https://youtu.be/YwZGU9OnnOM

```cpp
#include <iostream>
#include <thread>
#include <mutex> // step 1 : include std::mutex header file

std::mutex mtx;  // step 2 : create mutex object in Stack for thread synchronization

auto PrintFunction( int nCount, char szCharacterToPrint) {
    mtx.lock();  // step 3 : Accquires mutex lock and hence exclusive access to critical code section

    // START of CRITICAL CODE SECTION
    for ( auto i = 0; i < nCount; ++i ) {
        std::cout << szCharacterToPrint;
    }
    std::cout << '\n';
    // END of CRITICAL CODE SECTION

    mtx.unlock(); // Step 4: RELEASE mutex ownership.
}

int main () {
    std::thread Thread1( PrintFunction, 20, '*' );
    std::thread Thread2( PrintFunction, 20, '$' );

    Thread1.join();
    Thread2.join();

    return 0;
}
```

https://youtu.be/YwZGU9OnnOM


main Thread
Thread1
Thread2
Program Starts
Start Thread1()
Start Thread2()
Thread 1 & Thread 2 created
Thread1.join()
mutex.lock()
mutex.lock()
Thread 1 : Accquired mutex lock
Thread 2 : Wait for mutex lock
Thread 1 : Print 20 characters
Print 20 characters()
mutex.unlock()
Thread 1 : Finished execution
Thread 1 : Released mutex lock
Thread 2 : Accquired mutex lock
Thread 2 : Print 20 characters
Print 20 characters()
Thread2.join()
mutex.unlock()
Thread 2 : Released mutex lock
Main Thread waits for Thread1 and Thread 2 to complete execution.
Thread 2 : Finished execution
return(0)
Program Ends
https://youtu.be/YwZGU9OnnOM

THANK YOU

Subscribe to our

& YouTube Channel

https://youtu.be/YwZGU9OnnOM