


C++ 11

std::mutex

CONTENTS



**Introduction
to
std::mutex**



**How to
use
std::mutex**



Examples



Part - 1

Introduction to `std::mutex`

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

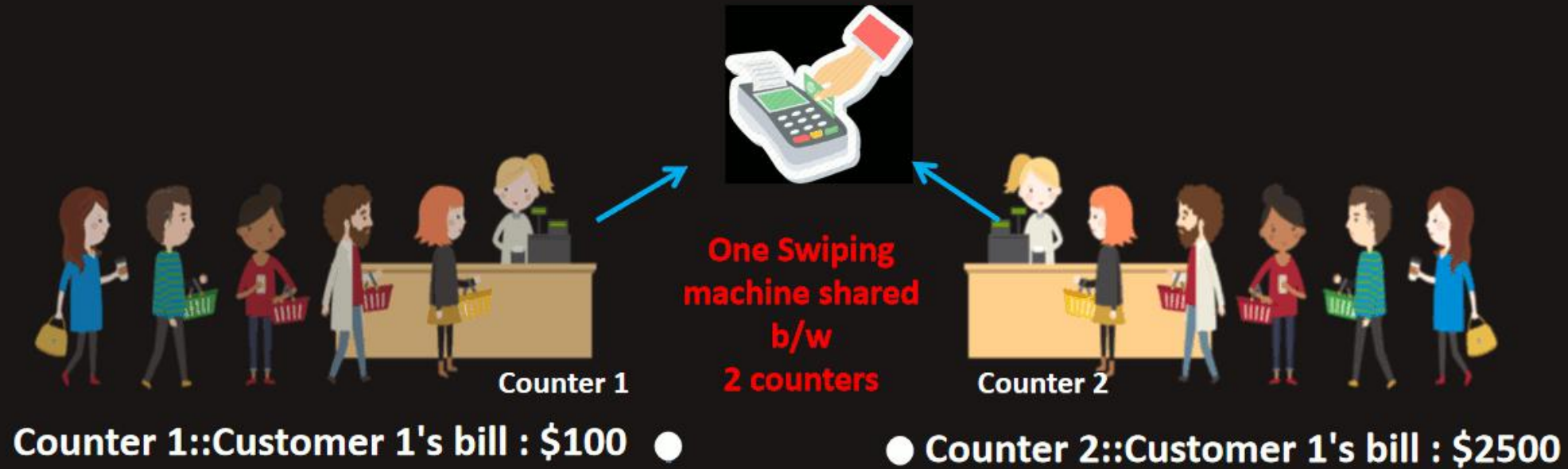
Thread : Billing Counter

Shared resource : Card Swiping Machine

Functionality : Bill Payement

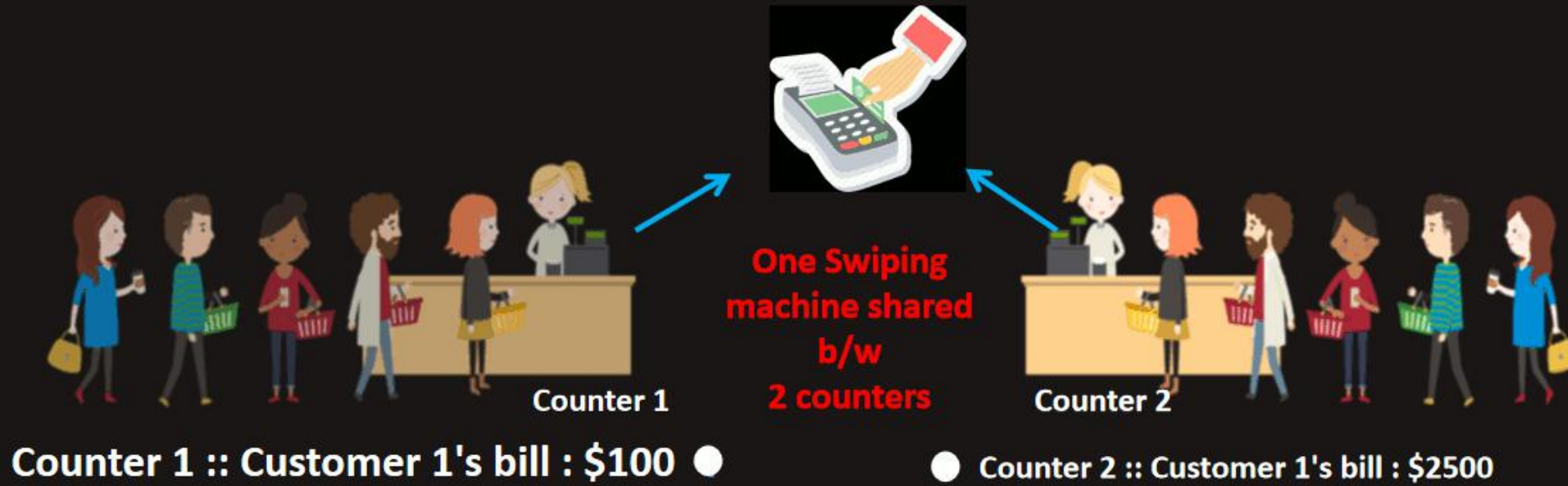
Introduction to Mutex concepts

Mutex concepts is designed for inter-thread synchronization.,



Introduction to Mutex concepts

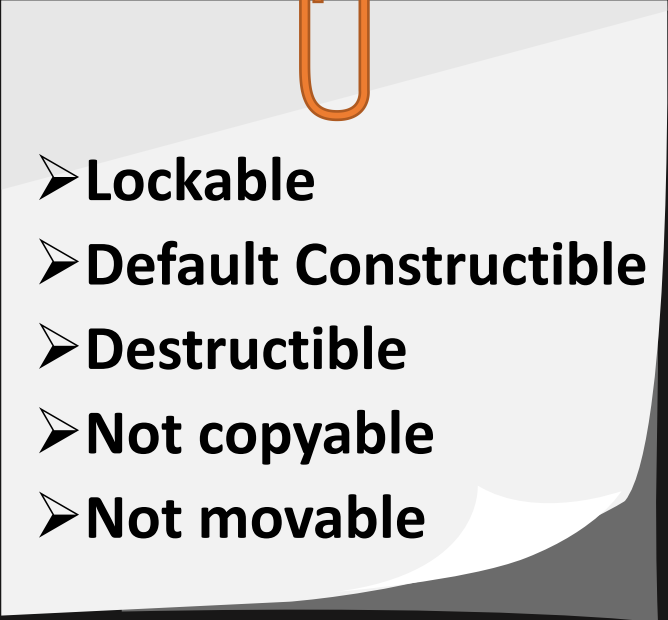
Mutex concepts is designed for inter-thread synchronization.



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



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

Synchronization
Primitive

Stack Object

Exclusive &
Non-recursive
ownership

C++ 11

Normally

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  }
```



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  }
```



Part - 2

How to use `std::mutex`

How to use `std::mutex`?

Step 1

Include the header file `<mutex>`

```
#include <mutex>
```

This header file contains the implementations of `class mutex`



How to use `std::mutex`?

Step 2

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

```
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  }
```



How to use std::mutex?

Step 3

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

```
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  }
```



How to use std::mutex?

Step 4

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

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



Part - 3

Examples

Example

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

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

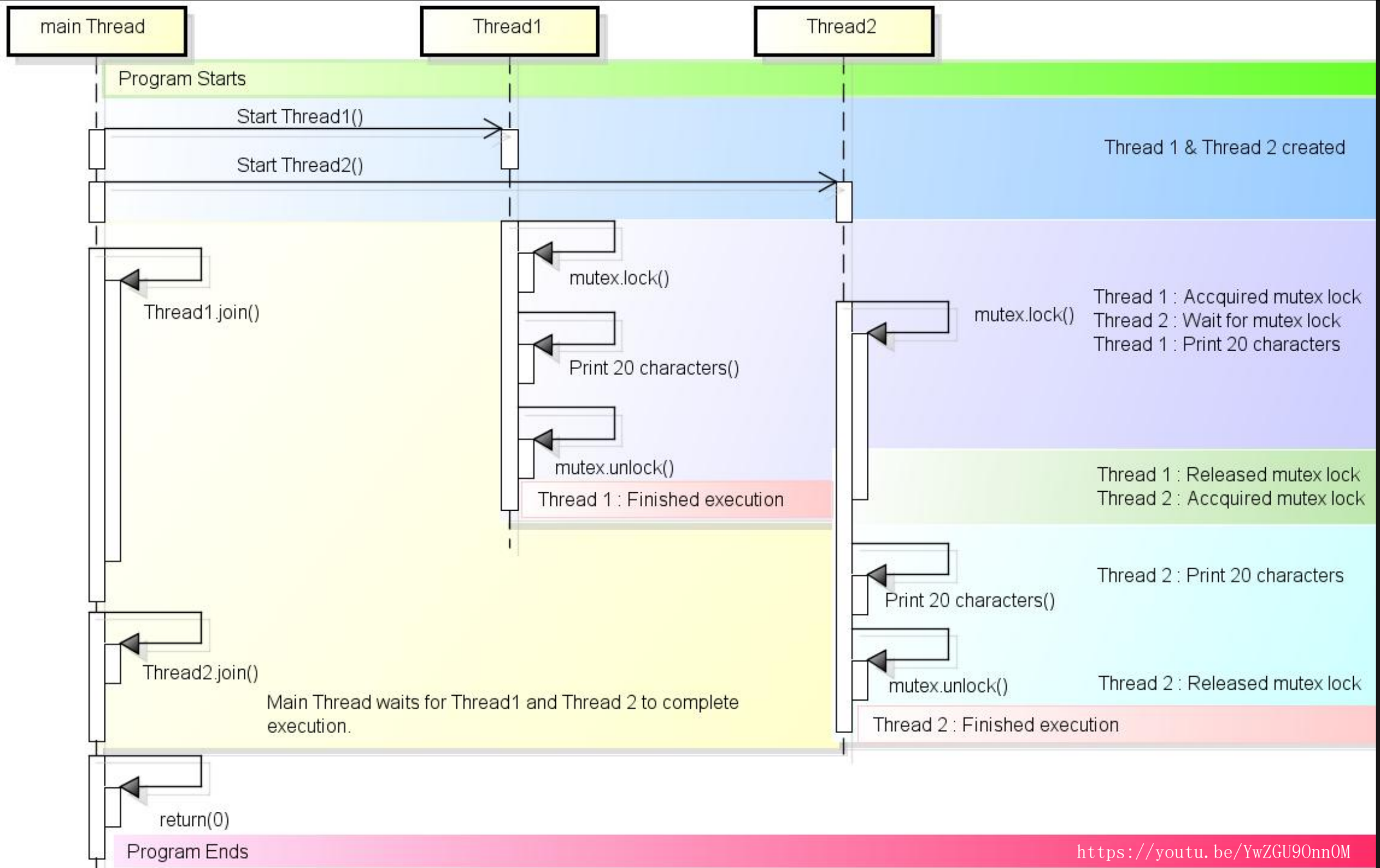
[illegible]

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

```

1  #include <iostream>
2  #include <thread>
3  #include <mutex> // step 1 : include std::mutex header file
4
5  std::mutex mtx; // step 2 : create mutex object in Stack for thread synchronization
6
7  auto PrintFunction( int nCount, char szCharacterToPrint) {
8      mtx.lock(); // step 3 : Acquires mutex lock and hence exclusive access to critical code section
9
10     // START of CRITICAL CODE SECTION
11     for ( auto i = 0; i < nCount; ++i ) {
12         std::cout << szCharacterToPrint;
13     }
14     std::cout << '\n';
15     // END of CRITICAL CODE SECTION
16
17     mtx.unlock(); // Step 4: RELEASE mutex ownership.
18 }
19
20 int main () {
21     std::thread Thread1( PrintFunction, 20, '*' );
22     std::thread Thread2( PrintFunction, 20, '$' );
23
24     Thread1.join();
25     Thread2.join();
26
27     return 0;
28 }

```



THANK YOU



Subscribe to our
& **You**  **Tube** Channel

<https://youtu.be/YwZGU90nn0M>