

[Home](#) [Python](#) [Pandas](#) [C++11](#) [C++](#) [STL](#) [Multithreading](#)[Design Patterns](#) [About Us](#) [Privacy Policy](#)

thispointer.com

[Python](#) [Pandas](#) [C++11 Tutorials](#) [C++ Tutorials](#) [STL](#) [Multithreading](#)[Boost Library](#) [GDB](#) [Design Patterns](#) [java](#) [Datastructure](#) [Subscribe](#)[Home](#) » [C++ 11](#) » [c++11 Threads](#) » [Multithreading](#) » You are reading »

C++11 Multithreading – Part 1 : Three Different ways to Create Threads

👤 Varun 🕒 January 20, 2015 📄 C++ 11, c++11 Threads, Multithreading

In this article we will discuss how to create threads in C++11 using `std::thread`.

Introduction to C++11 Thread Library

Original C++ Standard supported only single thread programming. The new C++ Standard (referred to as C++11 or C++0x) was published in 2011. In C++11 a new thread library is introduced.

Compilers Required:

Linux: gcc 4.8.1 (Complete Concurrency support)

Windows: Visual Studio 2012 and MingW

How to compile on Linux: `g++ -std=c++11 sample.cpp -lpthread`

Thread Creation in C++11

In every C++ application there is one default main thread i.e. `main()` function. In C++ 11 we can create additional threads by creating objects

C++11 – Multithreading

[Part 1: Three Ways to Create Threads](#)[Part 2: Joining and Detaching Threads](#)[Part 3: Passing Arguments to Threads](#)[Part 4 : Sharing Data & Race Conditions](#)[Part 5 : Fixing Race Conditions using mutex](#)[Part 6 : Need of Event Handling](#)[Part 7: Condition Variables](#)[Part 8: `std::future` and `std::promise`](#)[Part 9: `std::async` Tutorial & Example](#)[Part 10: `std::packaged_task<>` Tutorial](#)

C++11 Thread : FAQ

[C++11 : Start thread by member function](#)[C++11 : How to put a thread to sleep](#)[C++11 : How to get a Thread ID ?](#)[C++11: Vector of Thread Objects](#)

of `std::thread` class.

Each of the `std::thread` object can be associated with a thread.

Header Required :

```
1 #include <thread>
```

What `std::thread` accepts in constructor ?

We can attach a callback with the `std::thread` object, that will be executed when this new thread starts. These callbacks can be,

- 1.) Function Pointer
- 2.) Function Objects
- 3.) Lambda functions

Thread objects can be created like this,

```
1 std::thread thObj(<CALLBACK>);
```

New Thread will start just after the creation of new object and will execute the passed callback in parallel to thread that has started it. Moreover, any thread can wait for another to exit by calling `join()` function on that thread's object.

Lets look at an example where main thread will create a separate thread. After creating this new thread, main thread will print some data on console and then wait for newly created thread to exit.

Lets implement above using three different callback mechanism,

Creating a thread using Function Pointer

```
1 #include <iostream>
2 #include <thread>
3
4 void thread_function()
5 {
6     for(int i = 0; i < 10000; i++);
7     std::cout<<"thread function Executing"<<std::endl;
8 }
9
```

[C++11 : `std::thread` as a member variable in class](#)

[C++11 : How to Stop a Thread](#)

[Std::Tuple](#)

[Lambda Functions](#)

[C++11 Utilities](#)

[Initializer_list](#) [Std::Array](#)

[Rvalue References](#)

[Smart Pointers](#)

[Multithreading](#)

[Unordered_Set](#)

[Unordered_Map](#)

[c++11 : `std::tuple` Tutorial](#)

[c++11 : `std::make_tuple` Tutorial](#)

Popular Categories

[Behavioral Design Patterns \(6\)](#) [Boost](#)

[Date Time Library \(6\)](#) [Boost](#)

[Library \(17\)](#) [C++ \(107\)](#) [C++ 11 \(59\)](#)

[c++11 Threads \(16\)](#) [C++](#)

[Interview Questions](#)

[\(30\)](#) [collections \(17\)](#)

[Dataframe \(10\)](#) [Data Science](#)

[\(10\)](#) [Debugging \(6\)](#) [Debugging](#)

[Tutorial \(6\)](#) [Design Patterns \(9\)](#)

[dictionary \(17\)](#) [Directories](#)

[\(13\)](#) [FileHandling \(18\)](#)

[Functions \(10\)](#) [gdb \(6\)](#) [gdb](#)

```

10 int main()
11 {
12
13     std::thread threadObj(thread_function);
14     for(int i = 0; i < 10000; i++);
15     std::cout<<"Display From MainThread"<<std::endl;
16     threadObj.join();
17     std::cout<<"Exit of Main function"<<std::endl;
18     return 0;
19 }

```

Creating a thread using Function Objects

```

1 #include <iostream>
2 #include <thread>
3 class DisplayThread
4 {
5 public:
6     void operator()()
7     {
8         for(int i = 0; i < 10000; i++)
9             std::cout<<"Display Thread Executing"<<std::endl;
10    }
11 };
12
13 int main()
14 {
15     std::thread threadObj( DisplayThread() );
16     for(int i = 0; i < 10000; i++)
17         std::cout<<"Display From Main Thread "<<std::endl;
18     std::cout<<"Waiting For Thread to complete"<<std::endl;
19     threadObj.join();
20     std::cout<<"Exiting from Main Thread"<<std::endl;
21     return 0;
22 }

```

Creating a thread using Lambda functions

```

1 #include <iostream>
2 #include <thread>
3 int main()
4 {
5     int x = 9;
6     std::thread threadObj([]{
7         for(int i = 0; i < 10000; i++)
8             std::cout<<"Display Thread Executing"<<std::endl;
9     });
10
11     for(int i = 0; i < 10000; i++)
12         std::cout<<"Display From Main Thread"<<std::endl;
13
14     threadObj.join();
15     std::cout<<"Exiting from Main Thread"<<std::endl;
16     return 0;
17 }

```

Differentiating between threads

[commands \(6\)](#) [gdb Tutorial \(6\)](#)
[HashSet \(8\)](#) [java \(29\)](#) [linkedlist \(5\)](#) [Linux \(8\)](#) [Linux System](#)
[Programming \(8\)](#) [List \(22\)](#)
[Method Overriding \(5\)](#)
[Multithreading \(12\)](#) [Numpy \(22\)](#) [Pandas \(44\)](#)
[Python \(165\)](#)
[Smart Pointers \(6\)](#) [std::list \(12\)](#)
[std::map \(16\)](#) [std::set \(10\)](#)
[std::string \(12\)](#) [std::thread \(5\)](#)
[std::vector \(18\)](#) [STL \(56\)](#)
[STL Algorithm \(8\)](#) [STL Interview Questions \(16\)](#) [strings \(13\)](#)
[Uncategorized \(11\)](#)
[unordered_map \(8\)](#) [unordered_set \(6\)](#)

Search

Subscribe with us

Subscribe For latest Tutorials

* indicates required

Email Address *

Subscribe

Each of the `std::thread` object has an associated ID and we can fetch using,

Member function, gives id of associated thread object i.e.

```
1 | std::thread::get_id()
```

To get the identifier for the current thread use,

```
1 | std::this_thread::get_id()
```

If `std::thread` object does not have an associated thread then `get_id()` will return a default constructed `std::thread::id` object i.e. `{} not any thread.`

`std::thread::id` is a Object, it can be compared and printed on console too. Let's look at an example,

```
1 | #include <iostream>
2 | #include <thread>
3 | void thread_function()
4 | {
5 |     std::cout<<"Inside Thread :: ID = "<<std::this_thread::get_id()
6 | }
7 | int main()
8 | {
9 |     std::thread threadObj1(thread_function);
10 |    std::thread threadObj2(thread_function);
11 |
12 |    if(threadObj1.get_id() != threadObj2.get_id())
13 |        std::cout<<"Both Threads have different IDs"<<std::endl;
14 |
15 |        std::cout<<"From Main Thread :: ID of Thread 1 = "<<threadObj1.get_id();
16 |    std::cout<<"From Main Thread :: ID of Thread 2 = "<<threadObj2.get_id();
17 |
18 |    threadObj1.join();
19 |    threadObj2.join();
20 |    return 0;
21 | }
```

Learn more about multithreading in C++11 / 14

- [Modern C++ Concurrency in Depth](#)
- [Beginning Modern C++ \(C++11/C++14\)](#)
- [C++ Concurrency in Action](#)

- [Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14](#)

Other C++11 Multi-threading Tutorials

[C++11 Multi-threading Part 2: Joining and Detaching Threads](#)

[C++11 Multi-threading Part 3: Passing Arguments to Threads](#)

[C++11 Multi-threading Part 4: Sharing Data & Race Conditions](#)

[C++11 Multi-threading Part 5: Fixing Race Conditions using mutex](#)

[C++11 Multi-threading Part 6: Need of Event Handling](#)

[C++11 Multi-threading Part 7: Using Condition Variables to do Event Handling between threads](#)

[C++11 Multi-threading Part 8: std::future and std::promise](#)

[C++11 Multithreading – Part 9: std::async Tutorial & Example](#)

If you didn't find what you were looking, then do suggest us in the comments below. We will be more than happy to add that.

[**Do Subscribe with us for more Articles / Tutorials like this,**](#)

💎 C++ 11, C++ 11 Multithreading, c++ 11 Threads, Multithreading, std::thread

2 Comments Already

Sorry, comments are closed for this post

[« Design Sorting algorithm using Strategy Design Pattern](#)

[C++11 Multithreading – Part 2: Joining and Detaching Threads »](#)

Data Analysis in Python

Dataframe in Pandas
Numpy Arrays

Terms and Conditions

Terms and Conditions
Policy

C++ / C++11 Tutorials

C++ Tutorials
C++11 Tutorial
C++ Interview Questions
Multithreading
C++17
STL Tutorials

Design Patterns

Behavioral Design
Patterns
Observer Design Pattern
State Design Pattern
Strategy Design Pattern
Structural Design
Patterns
Composite Design
Pattern
Flyweight Design Pattern
Creational Design
Patterns

Python tutorials

Strings
Lists
Tuple
Dictionary
Functions
File Handling
Lambda Functions