Multi-Threaded Programming: C++11

The simplest C++11 Multithread

#include <iostream>

#include <thread>

void foo() { std::cout << "foo()\n"; }

void bar() { std::cout << "bar()\n"; }

int main()

{

std::thread t([]{

foo();

bar();

});

return 0;

}

Once our thread started, we should let the code know if we want to wait for it to finish by **joining** with it or leave it to run on its own by **detaching** it. Our program may be terminated before the **std::thread** object is destroyed if we don't do anything explicitly.

So, we need to ensure that the thread is joined or detached. If we choose not to wait the thread finish, then we need to ensure that the data accessed by the thread is valid until the thread has finished with it. Otherwise, as shown in the example below, we may encounter a situation where the thread function holds pointers/references to local variables and the thread hasn't finished when the function exits.