# Chapter 1

## 1.1 What is concurrency?

**Task switching:** illusion. One processor switch tasks many second per second.

**hardware concurrency:** these computers are capable of genuinely running more than one task in parallel provided by multicore processors.

**context switch:** the system has to perform every time it changes from one task to another, and this takes time.

## 1.2 Why use concurrency?

### 1.2.1 Separation of concerns:

Using threads in this way generally makes the logic in each thread much simpler, because the interactions between them can be limited to clearly identifiable points, rather than having to intersperse the logic of the different tasks.

### 1.2.2 Performance (increase by two ways):

**task parallelism:** divide a single task into parts and run each in parallel, reducing the total runtime.

**embarrassingly parallel (aka naturally parallel or conveniently concurrent):** Algorithms that are readily susceptible to **task parallelism.**

**data parallelism** by performing the same operation on multiple sets of data concurrently, there’s a different focus.

### 1.2.3 When not to use concurrency

the only reason not to use concurrency is when the benefit isn’t worth the cost.

* there’s an inherent overhead associated with launching a thread.
* If you have too many threads running at once, this consumes OS resources and may make the system as a whole run slower. because each thread requires a separate stack space.
* the more threads you have running, the more context switching the operating system has to do.

### 1.4.1 Hello, Concurrent World

every thread has to have an initial function, where the new thread of execution begins.

# Chapter 2

## 2.1 Basic thread management

Every C++ program has at least one thread, which is started by the C++ runtime: the thread running main().

In the same way that the program exits when it returns from main(), when the specified entry point function returns, the thread exits.

### 2.1.1 Launching a thread

It doesn’t matter what the thread is going to do or where it’s launched from, but starting a thread using the C++ Standard Library always boils down to constructing a std::thread object.

It’s therefore essential that the copy behaves equivalently to the original, or the result may not be what’s expected.

**joinable**: 检查线程是否可被 join。检查当前的线程对象是否表示了一个活动的执行线程，由默认构造函数创建的线程是不能被 join 的。另外，如果某个线程 已经执行完任务，但是没有被 join 的话，该线程依然会被认为是一个活动的执行线程，因此也是可以被 join 的。