

63 Timing in C++

我们如何计算完成某个操作或者执行某个代码所需要的时间呢？

计时对很多事情都很有用，不论你是希望某些事情在特定时间发生，还是只是评估性能或做基准测试，看你的代码运行得有多快，你需要知道应用程序实际运行的时间。

有几种方法可以实现这一点，C++11之后我们有了“chrono”，它是C++库的一部分，不需要去使用操作系统库。但在有chrono之前，如果你想要高分辨率的时间，你想要一个非常精确的计时器，那你就需要用到操作系统库了。例如在Windows中有一个叫做“QueryPerformanceCounter”的东西，我们仍然可以使用那些东西。事实上如果你想要更多地控制即使，控制CPU的计时能力，那么你可能会使用平台特定的库。不过本节只会看一看和平台无关的C++标准库方法（chrono库的一部分），它可以计算出执行代码时，代码之间经过了多长时间。

1. 独立平台的方法

记录sleep_for的用时：

```
C++

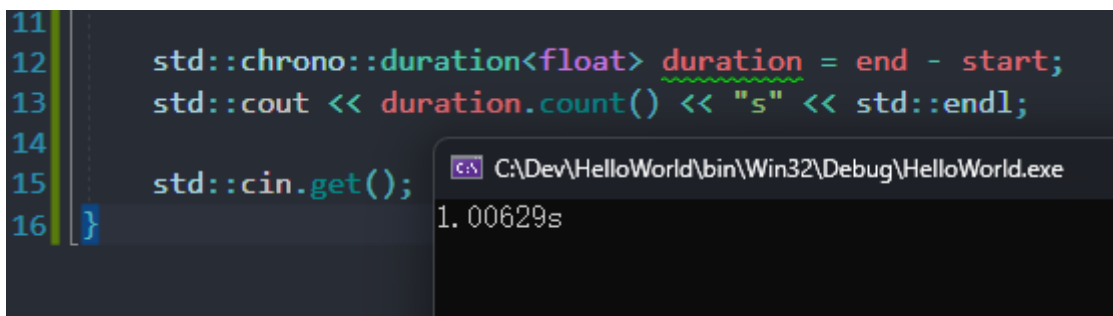
#include <iostream>
#include <chrono>
#include <thread>

int main()
{
    using namespace std::literals::chrono_literals;

    auto start = std::chrono::high_resolution_clock::now();
    std::this_thread::sleep_for(1s);
    auto end = std::chrono::high_resolution_clock::now();

    std::chrono::duration<float> duration = end - start;
    std::cout << duration.count() << "s" << std::endl;

    std::cin.get();
}
```



The screenshot shows a C++ program being executed in a debugger. The code is the same as the one in the previous block. The output window shows the result of the program: 1.00629s. The title bar of the output window reads: C:\Dev\HelloWorld\bin\Win32\Debug\HelloWorld.exe.

```
11
12     std::chrono::duration<float> duration = end - start;
13     std::cout << duration.count() << "s" << std::endl;
14
15     std::cin.get();
16 }
```

C:\Dev\HelloWorld\bin\Win32\Debug\HelloWorld.exe
1.00629s

chrono库非常好，可以高精度计时，几乎适用于所有平台，所以非常建议使用这个方法来满足你所有的计时需求，除非你在做一些特定的底层的事情。

2. 一个更聪明的方法

你想要给Function计时：

C++

```
void Funtion()
{
    for (int i = 0; i < 100; i++)
        std::cout << "Hello" << std::endl; //想要计算这些cout代码运行需要的时间
}
```

设置一个简单的结构体：

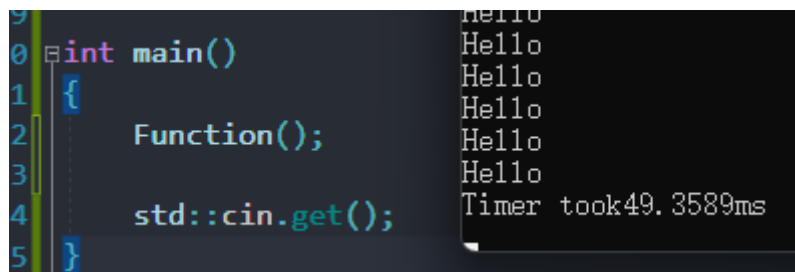
C++

```
struct Timer
{
    std::chrono::time_point<std::chrono::steady_clock> start, end;
    std::chrono::duration<float> duration;
    Timer()
    {
        start = std::chrono::high_resolution_clock::now();
    }
    ~Timer()
    {
        end = std::chrono::high_resolution_clock::now();
        duration = end - start;
        float ms = duration.count() * 1000.0f;
        std::cout << "Timer took" << ms << "ms" << std::endl;
    }
};

void Function()
{
    Timer timer; // 利用构造和析构函数来计时
    for (int i = 0; i < 100; i++)
        std::cout << "Hello" << std::endl; //想要计算这些cout代码运行需要的时间
}

int main()
{
    Function();

    std::cin.get();
}
```



```

9
0 int main()
1 {
2     Function();
3
4     std::cin.get();
5 }
Hello
Hello
Hello
Hello
Hello
Hello
Timer took 49.3589ms
```

可以将endl换为“\n”优化性能。

```
void Function()  
{  
    Timer timer;  
    for (int i = 0; i < 100; i++)  
        std::cout << "Hello\n";  
}
```

Hello
Hello
Hello
Hello
Hello
Hello
Timer took: 36.1117ms