C/C++中计算程序运行时间

最常用的的方式：

#include <time.h>

time\_t start = clock();

time\_t end = clock();

printf("the running time is : %f\n", double(end -begin)/CLOCKS\_PER\_SEC);

clock()计算的是CPU执行耗时，注意是CPU！如果有多个核并行，最后的结果是每个CPU上运算时间的总和！想要精确到毫秒，可以使用：

double(end -begin)\*1000/CLOCKS\_PER\_SEC

一般来说，只要求精确到秒的话，time是很好使的

1. #include <stdio.h>
2. #include <time.h>
3. int main(){
4. time\_t t\_start, t\_end;
5. t\_start = time(NULL) ;
6. sleep(3000);
7. t\_end = time(NULL) ;
8. printf("time: %.0f s\n", difftime(t\_end,t\_start)) ;
9. return 0;
10. }

如果要让程序休眠3秒，Windows使用Sleep(3000)，Linux使用sleep(3)，即Windows的Sleep接口的参数的单位是毫秒，Linux的sleep接口的参数的单位是秒。

如果需要精确到毫秒，以上程序就发挥不了作用，如果在Java要达到这要求就很简单了，代码如下所示：

1. public class Time {
2. public static void main(String[] args) {
3. try {
4. long startTime = System.currentTimeMillis();
5. Thread.sleep(3000);
6. long endTime = System.currentTimeMillis();
7. System.out.println("time: " + (endTime - startTime) + " ms");
8. } catch (InterruptedException e) {
9. e.printStackTrace();
10. }
11. }
12. }

通过Google找了一些资料后，发现C语言里没有标准的接口可以获得精确到毫秒的时间，都会调用到与操作系统相关的API，下面会分别介绍在Linux和Windows系统下的多种实现方法。

**Linux系统**

1.使用gettimeofday接口：

1. #include <stdio.h>
2. #include <sys/time.h>
3. int main() {
4. struct timeval start, end;
5. gettimeofday( &start, NULL );
6. sleep(3);
7. gettimeofday( &end, NULL );
8. int timeuse = 1000000 \* ( end.tv\_sec - start.tv\_sec ) + end.tv\_usec -start.tv\_usec;
9. printf("time: %d us\n", timeuse);
10. return 0;
11. }

gettimeofday能得到微秒数，比毫秒还要更精确。

2.使用ftime接口：

1. #include <stdio.h>
2. #include <sys/timeb.h>
3. long long getSystemTime() {
4. struct timeb t;
5. ftime(&t);
6. return 1000 \* t.time + t.millitm;
7. }
8. int main() {
9. long long start=getSystemTime();
10. sleep(3);
11. long long end=getSystemTime();
12. printf("time: %lld ms\n", end-start);
13. return 0;
14. }

Windows系统

1.使用GetTickCount接口：

1. #include <</span>windows.h>
2. #include <</span>stdio.h>
3. int main() {
4. DWORD start, stop;
5. start = GetTickCount();
6. Sleep(3000);
7. stop = GetTickCount();
8. printf("time: %lld ms\n", stop - start);
9. return 0;
10. }

Windows系统下有些编译器使用printf输出64位整数参数要使用%I64d，比如VC。

2.使用QueryPerformanceX接口：

1. #include <windows.h>
2. #include <stdio.h>
3. int main(){
4. LARGE\_INTEGER li;
5. LONGLONG start, end, freq;
6. QueryPerformanceFrequency(&li);
7. freq = li.QuadPart;
8. QueryPerformanceCounter(&li);
9. start = li.QuadPart;
10. Sleep(3000);
11. QueryPerformanceCounter(&li);
12. end = li.QuadPart;
13. int useTime =(int)((end - start) \* 1000 / freq);
14. printf("time: %d ms\n", useTime);
15. return 0;
16. }

3.使用GetSystemTime接口：

1. #include <</span>windows.h>
2. #include <</span>stdio.h>
4. int main(){
5. SYSTEMTIME currentTime;
6. GetSystemTime(&currentTime);
7. printf("time: %u/%u/%u %u:%u:%u:%u %d\n",
8. currentTime.wYear,currentTime.wMonth,currentTime.wDay,
9. currentTime.wHour,currentTime.wMinute,currentTime.wSecond,
10. currentTime.wMilliseconds,currentTime.wDayOfWeek);
11. return 0;
12. }

这种方法没给出计算时间差的实现，只给出如何用GetSystemTime调用得到当前时间，计算时间差比较简单，根据年、月、日、时、分秒和毫秒计算出一个整数，再将两整数相减即可。

以冒泡排序为例计算不同算法下运算时间的差异：

测试平台：

处理器：Intel Core i5-3337U @ 1.8GHz；

主板Sony SVF1421V3CB

操作系统：Windows 10 Pro Technical Preview(Build 9926)

编译环境：Codelite X64 V6.1.1 with MinGW-4.8.1

测试内容：

对于20000长度的随机数组进行冒泡排序所需时间。

测试结果：

传统算法

改进后算法：