请研究一下linux gettimeofday() 系统调用的用法，并测量一下gettimeofday的执行时间

**1.简介**

在C语言中可以使用函数gettimeofday()函数来得到时间。它的精度可以达到微妙

**2.函数原型**

#include<sys/time.h>

int gettimeofday(struct  timeval \*tv, struct  timezone \*tz )

**3.说明**

gettimeofday()会把目前的时间用tv 结构体返回，当地时区的信息则放到tz所指的结构中

**4.结构体**

1>timeval

struct  timeval{

       long  tv\_sec; //秒

       long  tv\_usec; //微秒

}；

2>timezone 结构定义为：

struct  timezone{

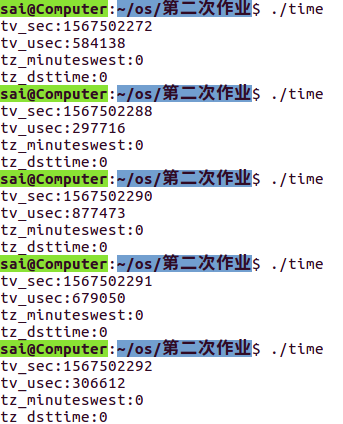
        int tz\_minuteswest; //和本初子午线时间差了多少分钟

        int tz\_dsttime; //type of DST correction

}

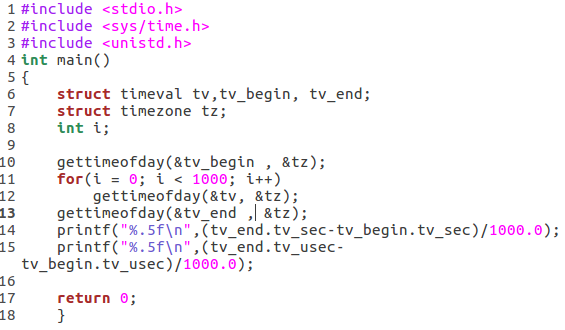
3>在gettimeofday()函数中tv或者tz都可以为空。如果为空则就不返回其对应的结构体。

4>函数执行成功后返回0，失败后返回-1，错误代码存于errno中。

**5.运行示例程序及测试结果**



**6.计算时间程序及测试结果**

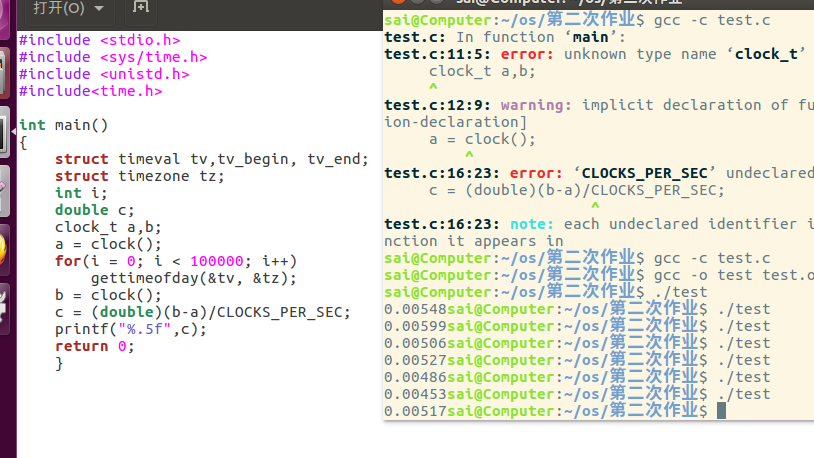


运行该程序1000次，计算首位计时差之后除以1000可得到平均运行时长。

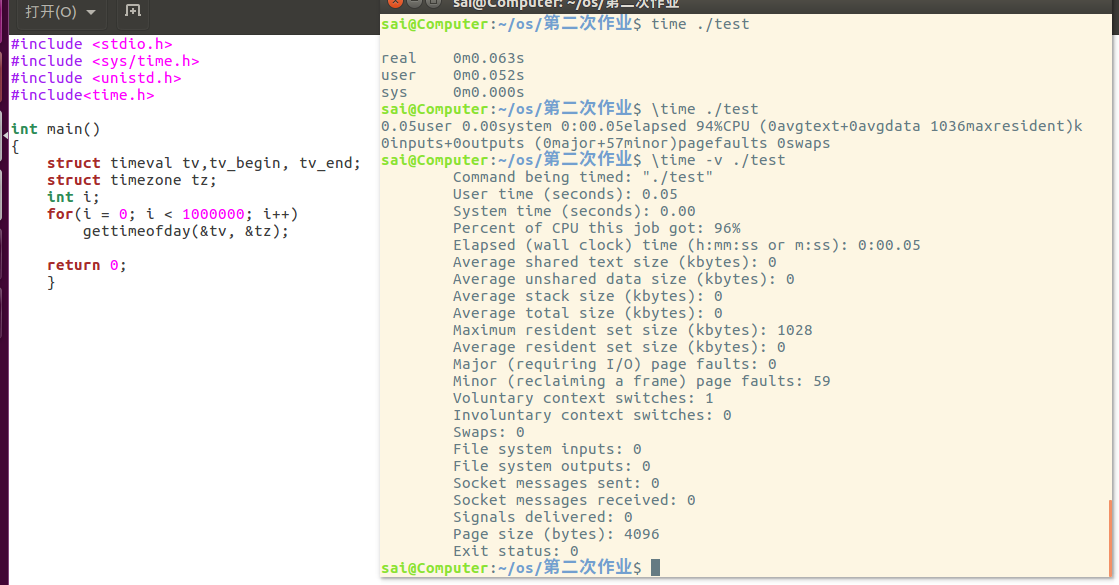
可知运行一次时长约为0.044微秒

经老师提示，发现1000次运行该程序时，重复测量运行时间的差距很大，应该是运行次数过少，在擦写cache时对整体的测量有较大的影响。于是我修改了运行次数为100000次，得到了一个较为平均的结果，为0.049微秒。

考虑到采用其他方式测量该程序的运行时间，额外采用以下调用：



使用系统调用clock（）。从此来看，clock运行本身的对程序的消耗比gettimeofday（）要小，多次重复运行趋于一个稳定值，约为0.05微秒。但是考虑到clock（）本身只能精确到秒，即使运行次数多也会存在误差。



另外，重新编写程序，使程序内容仅为运行1000000次gettimeofday，使用linux下的指令time，获得平均运行时间：对用户使用而言，为0.052微秒；对于真实时间，为0.063微秒。此处有差异为考虑到任务与任务运行之间的阻塞时间。

再使用位于user/bin下的time调用（\time），打印出了更为详细的信息。但是考虑到该部分的time测量本身就是基于gettimeofday的，所以参考价值不大。不过从cpu运行占用为96%可大体估计运行时间为0.05\*0.96=0.048微秒，与直接用gettimeofday本身测量差距不大。

最后，因为我在测试时确实没有遇到运行时间为0的情况，所以没有办法分析。但是遇到了其他现象：在测试用超过1000000次循环执行计算时间差的时候，算出来的运行时间竟然出现了负数。分析后发现，用来存储时间的变量有时会溢出？（但是还是觉得不可思议）

**7.系统环境**

Linux Computer 4.4.0-65-generic #86-Ubuntu SMP Thu Feb 23 17:49:58 UTC 2017 x86\_64 x86\_64 x86\_64 GNU/Linux