北京理工大學
Beijing Institute of Technology

本科实验报告

实验名称：CPU 占用率

课程名称：	操作系统原理	实验时间：	2018/3/21
任课教师：	王耀威	实验地点：	理学楼信抗实验中心
实验教师：	苏京霞	实验类型：	■ 原理验证 □ 综合设计 □ 自主创新
学生姓名：	施念		
学号/班级：	1120161302/05011609	组　　号：	53
学　　院：	信息与电子学院	同组搭档：	
专　　业：	电子信息工程	成　　绩：	

信息与电子学院
SCHOOL OF INFORMATION AND ELECTRONICS

实验一：CPU 占用率控制

一、实验目的

1. 通过编写和调试程序以加深对 CPU 调度的理解；
2. 熟悉 Windows 任务管理器 CPU 信息的获取和使用方法；

二、实验题目

编写程序实现下面任意一题：

1. CPU 占用率为一条直线，固定在 50%;
2. CPU 占用率为一条正弦曲线。

选择题目：2.绘制占有率为正弦曲线的实验。
原因：因为电脑不是单核，所以只有当死循环的时候占有率才有可能到 50%，故选择了实验体验更好的正弦曲线。

三、实验基础知识

1. CPU 占有率

在任务管理器的一个刷新周期内，CPU 忙（执行应用程序）的时间和刷新周期总时间的比率，就是 CPU 的占用率，也就是说，任务管理器中显示的是每个刷新周期内 CPU 占用率的统计平均值。

因此可以写个程序，在一个刷新周期中，一会儿忙，一会儿闲，调节忙/闲比例，就可以控制 CPU 占有率。

2. 函数

a. GetTickcount 函数：

它返回从操作系统启动到当前所经过的毫秒数，常常用来判断某个方法执行的时间，其函数原型是 DWORD GetTickCount(void)，返回值以 32 位的双字类型 DWORD 存储。

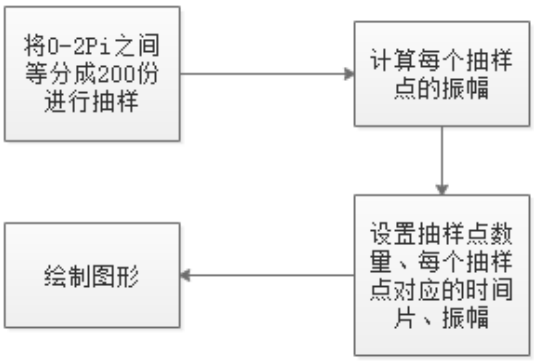
b. Sleep()函数：

windows 系统下函数名为 Sleep(),其函数原型为：#include <windows.h> （函数使用头文件），void Sleep(DWORD dwMilliseconds)（参数为毫秒）；

linux 系统下函数名为 sleep(),其函数原型为：#include <unistd.h>（函数使用头文件），unsigned int sleep(unsigned int seconds);参数为毫秒 （如果需要更精确可以用 usleep，单位为微秒）

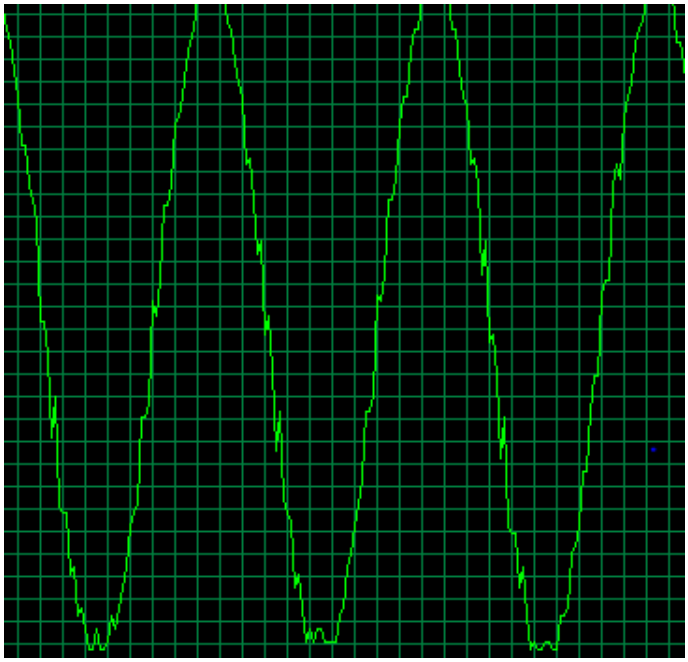
四、实验设计方法

将 0-2Pi 之间等分成 200 份进行抽样，计算每个抽样点的振幅，设置抽样点数量以及每个抽样点对应的时间片，计算每个抽样点的振幅。



五、实验结果及数据分析

1. 图像：



2. 分析

由图像可知，实验结果整体为理想的正弦曲线，但是仍有瑕疵，造成瑕疵的原因：

1. 可能是后台仍有程序在运行，这是 CPU 占有率曲线不会是一个完美的正弦曲线
2. 同时截图时使用的截图工具也会使曲线有一定的瑕疵。
3. 程序代码设置的抽样点数、时间片有待调整吗，使曲线更为光滑。

六、总结

此次实验我加深了对 CPU 的理解，同时实践操作也使我对曲线的绘制有了新的看法，不仅是正弦曲线，脉冲曲线等曲线也可以绘制出来。

在调整 CPU 占有率 50%的实验中，我通过资料了解了多核和单核的区别，同时，也了解了 windows 操作系统的特征。

除此之外，也学会了一些新的系统函数的用法，总的来说，收获良多。

七、附录

代码清单：

```
#include "Windows.h"
 #include "stdlib.h"
 #include "math.h"
 #include "tchar.h"

 const double SPLIT = 0.01;
 const int COUNT = 200;
 const double PI = 3.14159265;
 const int INTERVAL = 300;

 int _tmain(int argc, _TCHAR* argv[])
 {
     SetProcessAffinityMask(
             GetCurrentProcess(),
             0x00000001                  //cpu mask
             );

      DWORD busySpan[COUNT];    //busy 时间
      DWORD idleSpan[COUNT];    //array of idle times
      int half = INTERVAL / 2;
      double radian = 0.0;
      for(int i = 0; i < COUNT; i++)
      {
            busySpan[i] = (DWORD)(half + (sin(PI * radian) * half));
            idleSpan[i] = INTERVAL - busySpan[i];
            radian += SPLIT;
      }
      DWORD startTime = 0;
      int j = 0;
      while (true)
      {
            j = j % COUNT;
            startTime = GetTickCount();
            while ((GetTickCount() - startTime) <= busySpan[j]) ;
            Sleep(idleSpan[j]);
            j++;
      }
      return 0;
 }
```