嵌入式系统导论实验报告

姓名	学号	班级	电话	邮箱
许先涛	15352367	1516	17612034037	ku8834367@163.com

1.实验题目

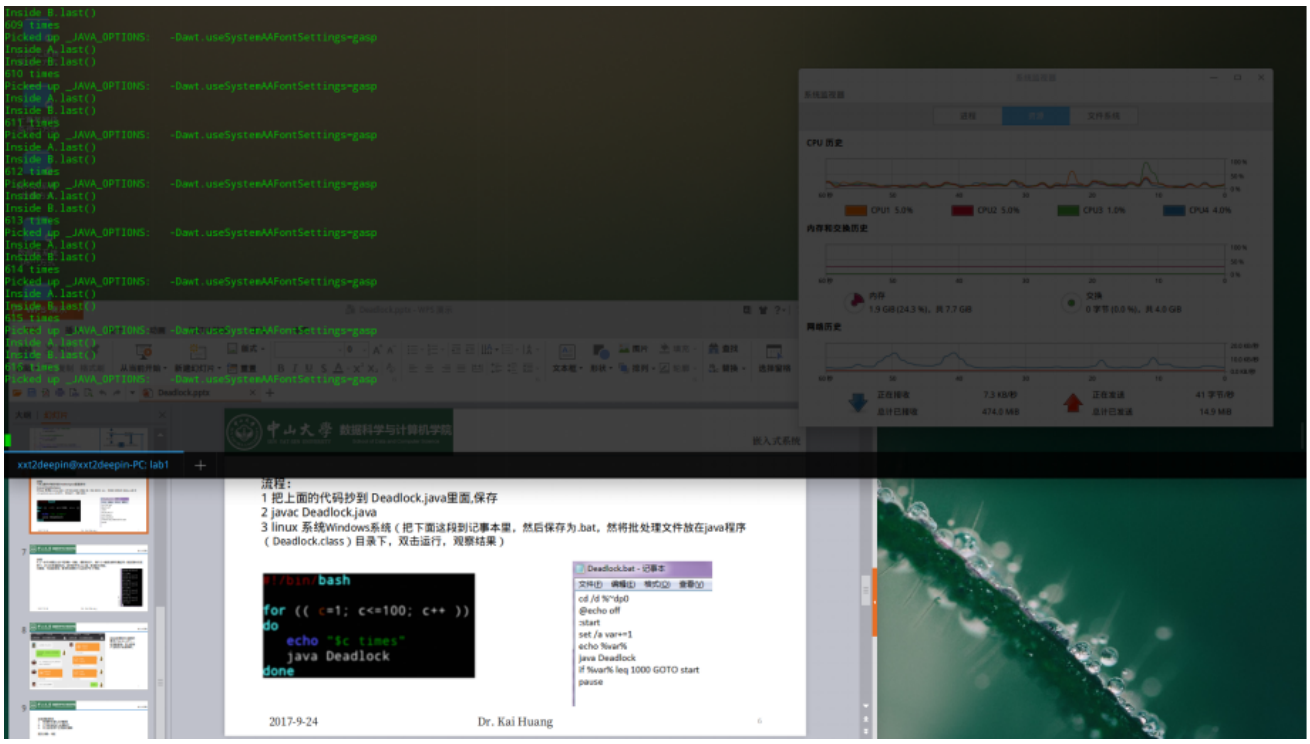
Dead Lock

2.实验结果

- 根据师兄给的代码编写相关java文件

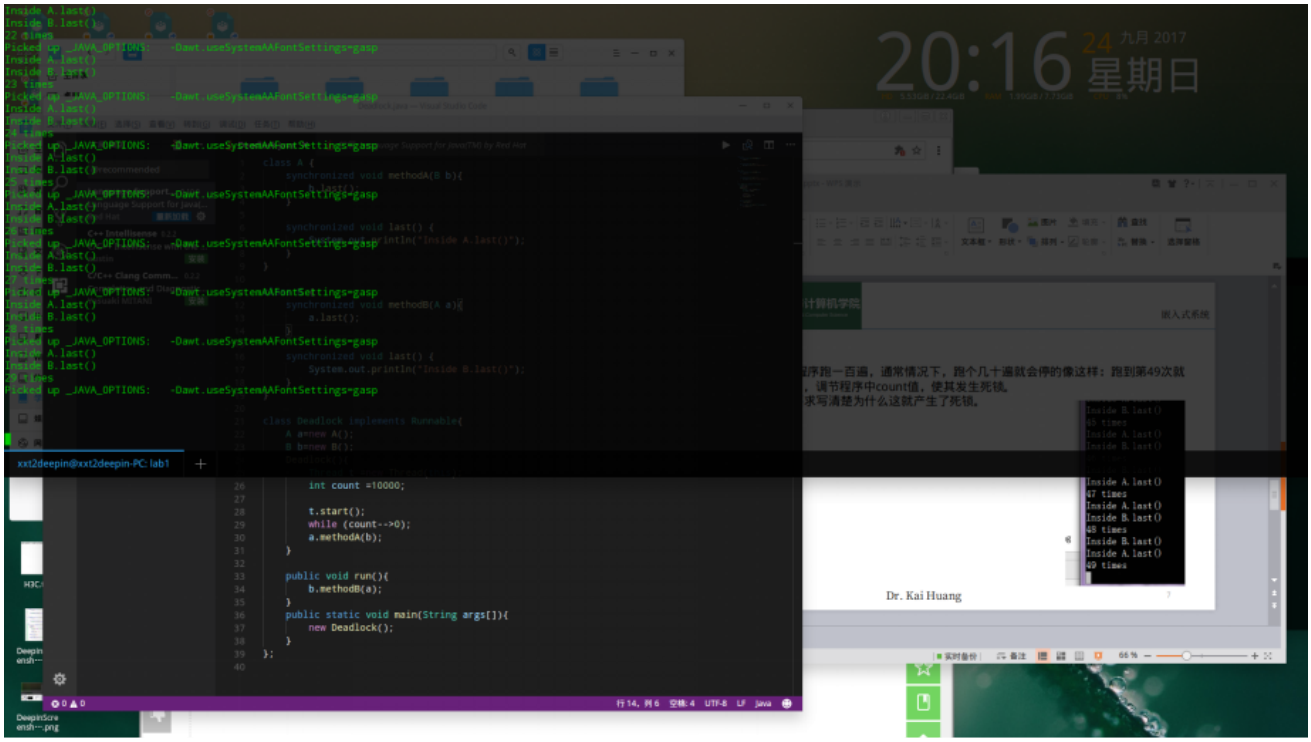
```
class A {
    synchronized void methodA(B b){
        b.last();
    }

    synchronized void last() {
        System.out.println("Inside A.last()");
    }
}

class B {
    synchronized void methodB(A a){
        a.last();
    }

    synchronized void last() {
        System.out.println("Inside B.last()");
    }
}

class Deadlock implements Runnable{
    A a=new A();
    B b=new B();
    Deadlock(){
        Thread t =new Thread(this);
        int count =20000;

        t.start();
        while (count-->0);
        a.methodA(b);
    }

    public void run(){
        b.methodB(a);
    }
    public static void main(String args[]){
        new Deadlock();
    }
};
```

- 利用Linux控制台重复执行代码bing得到死锁的情况图(修改循环次数c为1000)



(count=20000,死锁发生在第616次调用)

- 修改count值得到死锁的情况图



(count=10000,死锁发生在第29次调用)

- 产生死锁的四个必要条件：
  ①互斥条件：资源不能被共享，只能由一个进程使用。
  ②请求与保持条件：已经得到资源的进程可以再次申请新的资源。
  ③非剥夺条件：已经分配的资源不能从相应的进程中被强制地剥夺。
  ④循环等待条件：系统中若干进程组成环路，该环路中每个进程都在等待相邻进程正占用的

资源。

- 对于死锁的解释：
  主要产生死锁的部分是t.start(); while(count-->0);a.method(b)；这三行。由于线程t的启动>带动了函数run导致b.method(a)运行，于是就有a.last()；在这段时间中(差不多count为0之后)，又>有a.method(b)抢先运行，这时候a.last(0就会与a.method(b)冲突，所以a.last()就要等>a.method(b)运行结束才能进行；但是a.last()又是b的函数b.method(a)与b.last()冲突，导致>b.last()也无法运行，所以a.method(b)无法完成，就造成了死锁。

3.实验心得

这次实验其实没有什么难度，主要按照TA给的代码完成，然后为了让4个类函数能够形成死锁而调整count的值count较小的时候(count=0)是先b.last()之后再a.last(),count较大时(count=20000)则反过来，所以要构成死锁需要把count值锁定在这二者之间，于是调整为10000的时候就能够在次数较少的情况下出现死锁了。