HW3

1. 临界资源指一次只允许一个进程使用的共享资源；临界区指访问临界资源的代码片段
2. 信号量通常包括整数变量和等待队列，用于解决操作系统中的互斥和协同问题。
    - s.count>0表示有count个资源可用
    - s.count=0表示无资源可用
    - s.count<0表示有|count|个进程在s的等待队列中
3. 对于KLT，由于内核级线程由内核实现，故调度上与之前锁讨论的进程相同，存在优先级反转问题；对于ULT，线程在用户态实现，其线程调度不依赖于硬件中断，无法实现抢占式调度，故低线程不主动让出CPU，不存在优先级反转的问题
4. 停留时间为t/2，行驶时间为t，则前一辆火车行驶了t-t/2=t/2后下一辆车即可发车。假设火车速度恒定，则铁路的前半段和后半段分别均为临界资源。若B站只能从A站前往，则B站也是临界资源；若B站有其他方式前往（例如经过C站中转）则不是临界资源
5. apple表示苹果数量，orange表示橘子数量，empty表示盘子内空位数，使用mutex：

```
typedef int semaphore;
semaphore empty=1,mutex=1,apple=0,orange=0;

void father(){
     while(1){
           P(empty);     //等待盘子空
           P(metux);     //等待获取对盘子的操作
           lay_apple();
           V(mutex);    //释放对盘子的操作
           V(apple);    //通知取苹果
    };
}

void mather(){
     while(1){
           P(empty);         //等待盘子空
           P(metux);         //等待获取对盘子的操作
           lay_orange();
           V(mutex);         //释放对盘子的操作
           V(orange);        //通知取橘子
    };
}

void son(){
     while(1){
           P(orange);        //是否有桔子
           P(metux);         //等待获取对盘子的操作
           pick_orange();
           V(mutex);       //释放对盘子的操作
           V(empty);       //盘子空，可放水果
    };
}

void daugther(){
     while(1){
```

```
            P(apple);        //是否有苹果
            P(metux);         //等待获取对盘子的操作
            pick_apple();
            V(mutex);       //释放对盘子的操作
            V(empty);       //盘子空，可放水果
    };
}
```