作业3

2019011008

无92 刘雪枫

1. 系统中某些一次只允许一个进程同时使用的共享资源称为临界资源；进程中访问临界资源的代码片段称为临界区。
2. 信号量是一种同步对象，在进程管理者的角度处理进程的互斥和同步问题，可以用于进程同步、管理临界资源等；信号量包括一个整数计数值和一个进程等待队列，并包含P、V两个原语。s.count>0表示还有count个资源可用，s.count=0表示无资源可用，s.count<0表示有|count|个进程在信号量的等待队列中等待。
3. 对于内核级线程，由于线程由内核实现，因此在调度问题上与之前锁讨论的进程是一样的，所以KLT存在同样的优先级反转问题。

但是对于用户级线程，由于线程全部在用户态实现，线程调度不依靠硬件中断，因此无法实现线程的抢占式调度，这样低优先级线程在不主动让出CPU所有的情况下，不会让高优先级的线程执行，因此ULT不存在优先级反转问题。

1. 由于火车在B站停留时间是A到B之间行驶时间的一半，因此只要前一辆火车行驶超过一半，后一辆火车就可以发车，从而避免火车在B站拥堵。因此，A与B间线路的前一半是临界资源。
2. 思路是使用信号量apple和orange分别代表盘子内苹果和橘子的个数，信号量empty表示盘子内空位的个数，信号量mutex用来保护临界资源盘子。但是注意到，盘子最多只有一个空位，因此逻辑保证盘子本身最多只有一个进程来访问，因此可以不需要mutex进行保护。下面提供这两种方法：
   1. 当使用mutex时，伪代码如下：

typedef int semaphore;

semaphore mutex = 1;

semaphore empty = 1;

semaphore apple = 0;

semaphore orange = 0;

void father() {

    while (1) {

        P(empty);

        P(mutex);

        lay\_apple();

        V(apple);

        V(mutex);

    }

}

void mother() {

    while (1) {

        P(empty);

        P(mutex);

        lay\_orange();

        V(orange);

        V(mutex);

    }

}

void daughter() {

    while (1) {

        P(apple);

        P(mutex);

        pick\_apple();

        V(empty);

        V(mutex);

    }

}

void son() {

    while (1) {

        P(orange);

        P(mutex);

        pick\_orange();

        V(empty);

        V(mutex);

    }

}

* 1. 当优化掉mutex时，伪代码如下：

typedef int semaphore;

semaphore empty = 1;

semaphore apple = 0;

semaphore orange = 0;

void father() {

    while (1) {

        P(empty);

        lay\_apple();

        V(apple);

    }

}

void mother() {

    while (1) {

        P(empty);

        lay\_orange();

        V(orange);

    }

}

void daughter() {

    while (1) {

        P(apple);

        pick\_apple();

        V(empty);

    }

}

void son() {

    while (1) {

        P(orange);

        pick\_orange();

        V(empty);

    }

}