## 操作系统作业4

### 什么是进程间的互斥？什么是线程间的同步？

* 进程间的互斥：进程互斥是进程之间的间接制约关系。当并发进程竞争使用同一资源时，它们之间可能会发生冲突。为了避免这种冲突，当一个进程进入临界区使用临界资源时，另一个进程必须等待。只有当使用临界资源的进程退出临界区后，这个进程才会解除阻塞状态。
* 线程间的同步：指一个线程发出某一功能调用时，在没有得到结果之前，该调用不返回。同时其它线程为保证数据一致性，不能调用该功能。所有“多个控制流，共同操作一个共享资源”的情况，都需要同步。

### 为什么需要对进程的并发执行进行控制？

系统中的多个进程之间有可能存在某些相互影响、相互制约的关系，任意一个进程的推进都会受到另一个进程的制约和影响。所以我们加入对进程间的互斥、同步和通信来控制这些并发进程。

### 简述信号量值的含义。

信号量x的值为正，表示允许x个进程访问临时区

信号量x的值为负，其绝对值表示|x|个进程由于请求该资源而被阻塞

### 解决互斥与同步主要有些什么方法？各有什么优缺点？

* 中断禁用 优点：保证了临界区不会发生中断，有效的实现了互斥

缺点：

1. 该方法代价太高，由于禁止了中断，系统执行效率会有明显的降低。
2. 该方法不能用于多处理器结构
3. 无法处理紧急事件

* 专用机器指令 优点：

1. 适用范围广。它可适用于单处理器或共享内存的多处理器上的任意数目进程。
2. 适用简单。指令设置简单，容易验证其正确性。
3. 可支持多个临界区。

缺点：

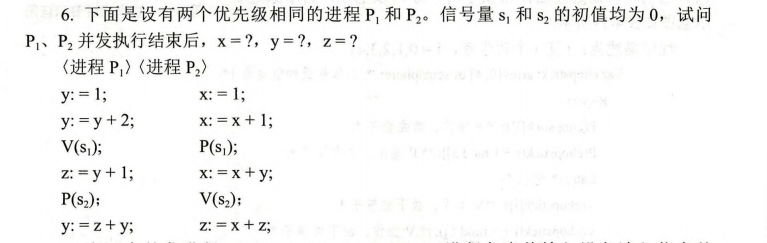
1. 导致CPU空耗。当一个进程等待进入临界区时，会不断地持续检测，消耗处理器事件。
2. 导致饥饿进程。由于由于各种原因，当有多个进程都在等待进入临界区时，在某些极端情况下有可能某个进程永远无法进入临界区，发生饿死现象

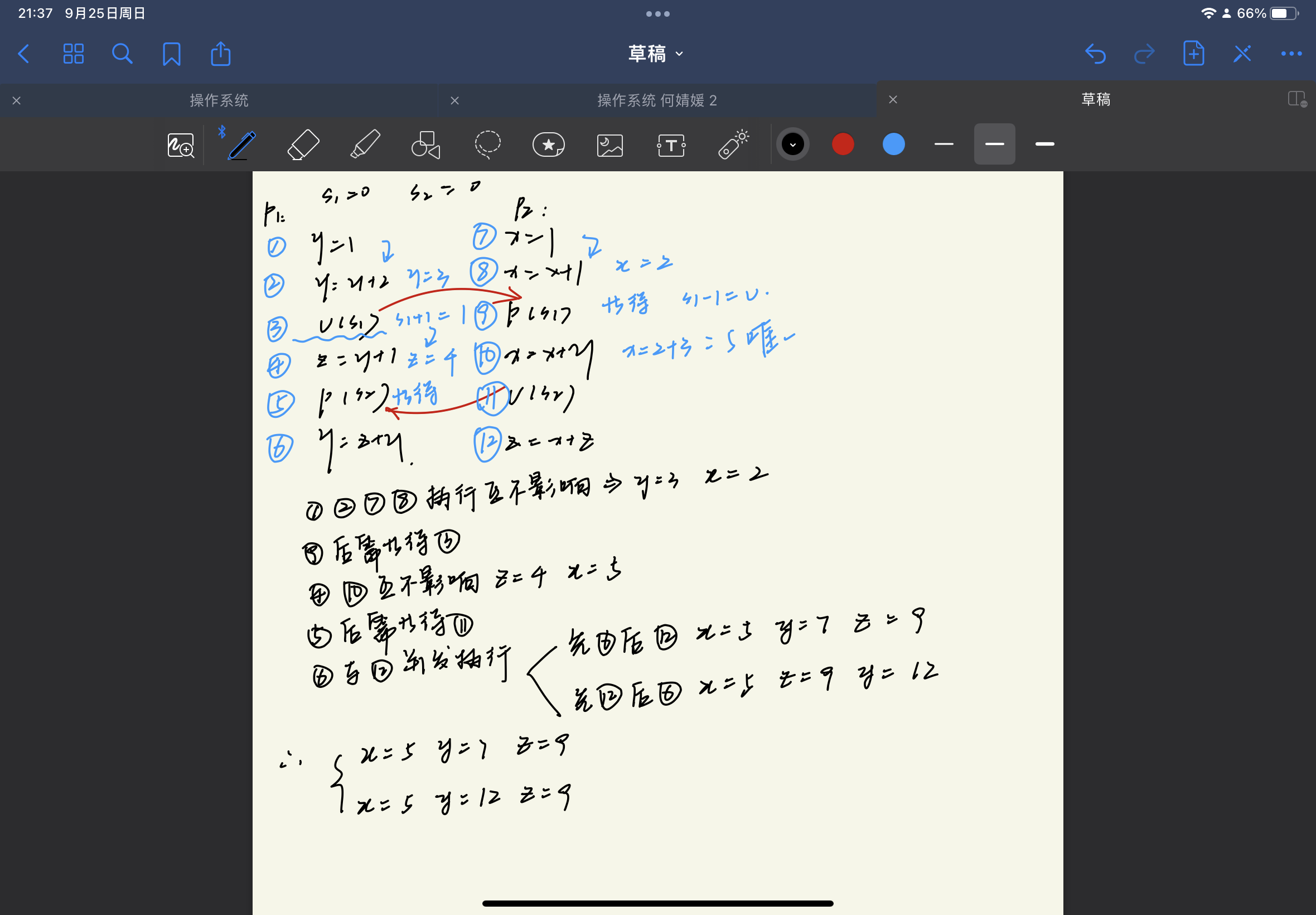
* 信号量 优点：

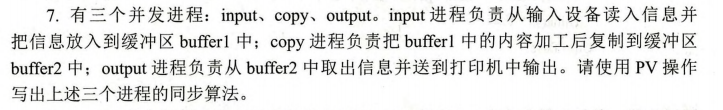
1. PV操作能够实现对临界区的管理要求
2. 实现简单
3. 允许使用它的代码休眠，持有锁的时间可相对较长。

缺点：

1. 信号量机制必须有公共内存，不能用于分布式操作系统
2. 信号量机制功能强大，但使用时对信号量的操作分散，而且难以控制
3. 不易控制和管理，不易发现和纠正







buffer1\_full = buffer2\_full = 0;   //假设一开始两个缓冲区中没有内容

buffer1\_empty = buffer2\_empty = N; //假设两个缓冲区最多可同时容纳N个数据

mutex1 = mutex2 = 1; //假设两个缓冲区的互斥量都为1

void input(){

    while(True){

        wait(buffer1\_empty);

        wait(mutex1);

        Buffer1[i] = data;

        i = (i + 1) % N;

        signal(mutex1);

        signal(buffer1\_full);

    }

}

void copy(){ //copy会访问两个缓冲区

    while(True){

        wait(buffer1\_full);

        wait(mutex1);

        data = Buffer1[j];

        j = (j + 1) % N;

        signal(mutex1);

        signal(buffer1\_empty);

        data = process(data); //copy内部处理data

        wait(buffer2\_empty);

        wait(mutex2);

        Buffer2[m] = data;

        m = (m + 1) % N;

        signal(mutex2);

        signal(buffer2\_full);

    }

}

void output(){

    while(True){

        wait(buffer2\_full);

        wait(mutex2);

        data = Buffer2[n];

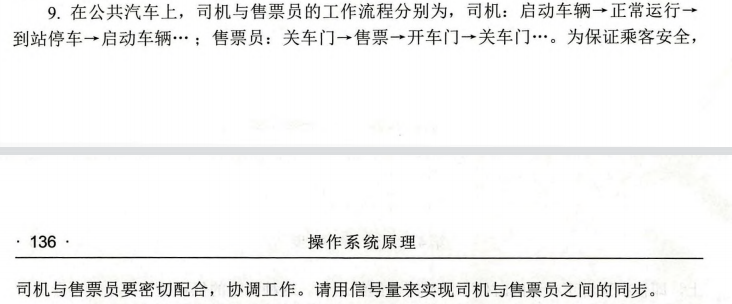
        n = (n + 1) % N;

        signal(mutex2);

        signal(buffer2\_empty);

    }

}



doorClose = 0; //初始状态车门未关闭

carStop = 0;  //初始状态汽车未行驶

void conductor(){

    while(True){

        closeDoor(); //关门

        signal(doorClose);//关门信号

        sellTicket();  //关上门后开始售票

        wait(carStop);  //等待汽车停止行驶

        openDoor();  //开门

    }

}

void driver(){

    while(True){

        wait(doorClose);

        drive()  //车门关闭后正常行驶，正常行驶结束意味着车停止

        signal(carStop) //停止信号

    }

}

1. 共享内存通信、管道通信和消息传递通信各有什么特点？各有什么优点与不足？

* 共享内存通信
  + 特点：为了传输大量数据，在存储器中划出一块共享存储区，相互通信的进程共享某些数据结构或共享存储区
  + 优点：访问共享内存区域和访问进程独有的内存区域一样快，并不需要通过系统调用或者其他需要切入内核的过程来完成，同时也避免了对数据的各种不必要性
  + 不足：
    - 共享内存的通信方式是通过将共享的内存缓冲区直接附加到进程的虚拟地址空间中来实现的．因此，这些进程之间的读写操作的同步问题操作系统无法实现。必须由各进程利用其他同步工具解决。
    - 由于内存实体存在于计算机系统中．所以只能由处于同一个计算机系统中的诸进程共享。不方便网络通信。
* 管道通信：
  + 特点：用于连接一个读进程和一个写进程，实现进程之间通信的一种共享文件，又称为pipe文件
  + 优点：简单方便
  + 不足：
    - 局限于单向通信
    - 缓冲区有限
* 消息传递通信：
  + 特点：
    - 进程之间的数据交换以消息或报文为单位
    - 直接通信方式：发送进程直接将消息发送给接收进程，接收进程可以接受来自任意发送方的消息，并且在读出消息的同时得知发送者是谁
    - 间接通信方式：消息不是直接从发送方发送到接收方，而是发送到临时消息队列（即信箱）。接收进程则从中取出发送给自己的消息。这种方式在使用上有很大的灵活性，发送发和接收方的关系可以是一对一、多对一、一对多和多对多的关系
  + 优点：在现代操作系统中应用得非常广泛，适用于所有进程之间的通信
  + 不足：
    - 直接：一旦改变进程的名称，则可能需要检查所有其他进程的定义，有关对该进程旧名称的所有引用都必须查找到，以便将其修改为新名称，显然，这样的方式不利于实现进程定义的模块化。