1. **十字路口小车通行问题：**

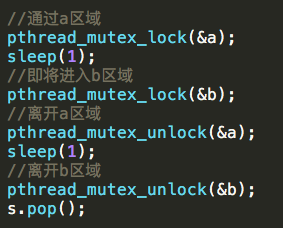
**设计思路：**

1. **对单辆车通过路口过程的模拟：**

单辆小车在通过路口的时候存在几种状态（以南方小车举例），在队列中等待，在a区域中行驶，**车头进入b区车尾还在a区**，在b区域中行驶，离开路口。

注意加粗的地方，也注意，本题目不得使用AND型信号量机制。如何对这个行驶过程进行模拟是本题目的一个重点。

我的模拟方式如图：



这里需要解释一下，我的模拟虽然形似但绝不是AND型信号量机制。同时也比较真实的还原了**车头进入b区车尾还在a区**这个过程。

小车驶入a区域，需要a区域的锁，假设小车行驶1秒后车头即将进入b区域。此时就要申请b区域的锁了（因为你马上就要开进去了，申请不到就不能开进去，否则就意味着撞车了），申请不到就只能在a区域停着等待b区域车开过去以后再进入b区域。假设这个跨越ab区域的过程是瞬间完成的（更真实的话应该使用sleep模拟这个跨越区域的过程，但我为了更易理解完成就没有去做），当车尾离开a区域，接触a的锁。假设从此刻到离开b区域需要1秒（sleep(1)），一秒以后离开b区解锁b。

这是我对这个情况尽量真实的模拟和还原，本题目不可以使用AND型信号量机制。但是汽车行驶过程中，跨越的那个时刻，是一定需要两把锁的（如果跨越过程不需要两把锁，即代码中第3、4行交换位置，先解锁a再申请b的锁，那四辆车同时驶入也不会死锁了，同时此刻难道车是在夹缝中停着等待吗？这样就不符合实际情形了）。

这是我对单辆车通行过程的模拟。

1. **对中央调度器的设计**

中央调度，实验给了一个要求，如下：

1. 来自同一个方向多个车辆到达十字路口时，车辆靠右行驶，依次顺序通过；
2. 有多个方向的车辆同时到达十字路口时，按照右边车辆优先通行规则，除非该车在十字路口等待时收到一个立即通行的信号；
3. 避免产生死锁；
4. 避免产生饥饿；

此处我将阐述我根据要求和我的理解对中央调度的设计：

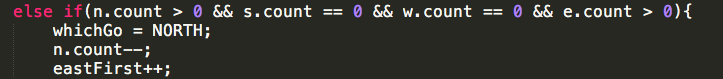
1. 来自一个方向的车来时，都在路口等待，若收到信号，则判断该车是否处于队列头，若是且没有收到等待信号，则驶入路口。若不是队列头车辆，继续等待。注意，以南方车辆为例，在我的算法中，若南方第一辆车驶离a区驶入b区，则南方队列第二辆车（若没有收到等待信号）立刻可以驶入a区，而不用等待第一辆车驶离b区离开路口才进入，这样，可以尽可能减少其他车辆等待时间，增加效率。
2. 多个方向车辆到达路口时，分多种情况考虑：
   1. 两辆车时，若来的两辆车是对向（如：东西、南北），则两辆车同时驶入路口。
   2. 两辆车时，若来的两辆车不是对向（如：东北、东南等），则以右边优先原则一辆车驶入一辆车等待。
   3. 三辆车时，必定存在一组对向和一个单（如：南北东，南北对向，东单），此时对向两车先走，单向车等待，因为对向一次可以通行两辆车。
   4. 四辆车时，存在两组对向，则一次南北向两车通行一次东西向两车通行，轮流制。

我的设计原则是，如果有对向双车，则让其先走，一次可以走两辆车，可以减少其他车辆等待时间，提高效率。若无对向双车，则让右边优先通行。

1. 我的死锁处理是，路口只有四个资源，当此处调度需要的资源+路口正在被使用的资源>=4时，本次调度sleep(3)等待3秒，并输出“发生死锁”。因为资源大于等于4时，是铁定死锁的。（跨越两个资源区域需要上两把锁，加上其他每个资源区域至少上了一把锁，上锁数量至少是5，则必定有区域死锁）。等待三秒是因为，3秒的等待，现在路口的车必定早已驶离.毕竟按照我对汽车速度的设定，在路口若无等待，则两秒可以通过。这样就避免了死锁。
2. 我的避免饥饿的方式是这样的：

屏幕快照%202016-11-11%20下午10.37.57.png

存在这样一组变量，当中央调度机制进行调度时，每让一个方向的车等待1次，则相应方向的变量++。eg：

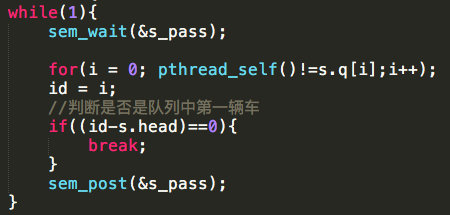


当北东来车时，北方通行（右方优先原则），东方等待，等待一次，eastfirst++。

当某个方向该变量到达5次时，即该方向车辆让行等待了5次，则下一次调度无条件让该方向车辆通行（若该方向有车），并将该变量清零。以此来保证每个方向让行等待次数有限，不陷入饥饿。

这是我对中央调度器的设计，既满足了要求，也有一些我自己的优化提高效率的想法设计。

1. **小车如何按顺序不争抢的进入路口**

****

如图，以南方小车举例。每辆小车的线程都等待s\_pass南方通过的信号，得到信号后，通过队列中存储的该方向小车的线程id顺序，知道是否该小车是队列中第一辆车，若是，驶入路口。若不是，释放该通过信号，继续等待下一次信号。

1. **遍历操作系统所有进程：**

**设计思路：**

该实验相对前一个实验来说比较简单，更多是让我们学习内核模块的编程。在了解内核模块编程方法和学会使用关于查看进程信息的一系列函数，就可以进行编程了。

设计一个指针指向所有进程的列表，通过指针遍历所有进程，printk打印信息到缓冲区。遍历到每个进程时根据进程类型进行计数，最后输出各种类型进程数即可。

编写一个shell程序，show.sh。将输出重定向到result文件中，得到printk输出到内核缓冲区的结果。