实验概述

1. 死锁检测

首先新建一个结构体 DeadLockDetector 其中包含 available, work, need, allocation等数据结构。在每个 PCB 中 为 mutex 和 semaphore 分别构造一个检测器。

每当新建一个 mutex 或 semaphore时，就向检测器里添加一种资源，need，allocation等全部初始化为0， work 和 available 初始化为1(mutex) 或 res_count(semaphore)。当开启死锁检测且调用 sys_mutex_lock 或 sys_semaphore_down 时，检测器就按文档算法进行检查，如果系统处于安全状态，则为当前线程申请资源，若work资源不够，则多出来的放进 need 中。

问答作业

1. 需要回收申请的动态内存，用户栈，内核栈，trap空间，打开的文件列表等。其他线程的 TaskControlBlock 可能在 ProcessControlBlockInner 的 tasks 属性中，也可能在 TaskManager 的 ready_queue 中，或者 MutexBlocking、Semaphore等结构体中。tasks 中的引用会随着 Vec 的清空而消失，ready_queue 中相关的引用会逐个被删除，MutexBlocking 会随着 ProcessControlBlock 的释放而释放，进而释放包含的引用。

2. 两种实现的区别在于修改标志位和释放阻塞线程的相对顺序不同，第一种实现可能会导致饥饿，即先申请 lock 的线程一直无法获得锁。第二种实现则可以保证先申请 lock 的线程优先获得锁。

荣誉准则

1. 在完成本次实验的过程（含此前学习的过程）中，我曾分别与 **以下各位** 就（与本次实验相关的）以下方面做过交流，还在代码中对应的位置以注释形式记录了具体的交流对象及内容：

   > 无

2. 此外，我也参考了 **以下资料** ，还在代码中对应的位置以注释形式记录了具体的参考来源及内容：

   > 无

3. 我独立完成了本次实验除以上方面之外的所有工作，包括代码与文档。 我清楚地知道，从以上方面获得的信息在一定程度上降低了实验难度，可能会影响起评分。

4. 我从未使用过他人的代码，不管是原封不动地复制，还是经过了某些等价转换。 我未曾也不会向他人（含此后各届同学）复制或公开我的实验代码，我有义务妥善保管好它们。 我提交至本实验的评测系统的代码，均无意于破坏或妨碍任何计算机系统的正常运转。 我清楚地知道，以上情况均为本课程纪律所禁止，若违反，对应的实验成绩将按“-100”分计。

(optional)

除了死锁检测外，只实现了sys_get_time，其他系统调用似乎没有在要求范围内。