lab5

实现功能总结

死锁检测算法，在进程开启死锁检测之后，该进程的线程在申请资源时，如果当前资源不满足申请需求，对该进程的线程池进行判断，如果除主线程外其他线程全部由于申请资源阻塞，那么则判断发生死锁，不进行资源申请，返回 `-0xDEAD` 。

简答作业

> 在我们的多线程实现中，当主线程 (即 0 号线程) 退出时，视为整个进程退出， 此时需要结束该进程管理的所有线程并回收其资源。 - 需要回收的资源有哪些？ - 其他线程的 TaskControlBlock 可能在哪些位置被引用，分别是否需要回收，为什么？

需要回收各个线程的控制块，进程的子进程，地址空间和文件描述表，不需要，因为在引用之后，如果函数无法正常退出，会调用 `drop` ，将引用的线程的任务控制块释放。

> 对比以下两种 `Mutex.unlock` 的实现，二者有什么区别？这些区别可能会导致什么问题？

第一个实现，无法保证唤醒的被阻塞的线程拥有这把锁，在将锁释放的时候，可能有其他线程被调度，访问这把锁，然后由于已经释放了锁，所以可以拥有锁，此时就有两个线程拥有了锁，发生了冲突。

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：

   >

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：

   >

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。