操作系统 · Lab5

计01　容逸朗　2020010869

功能简述

- 本次实验实现了银行家算法，可以对锁和信号量进行死锁检测。
- 具体来说，在进程的 `PCB_Inner` 模块中加入了 `deadlock_detect`，`mutex_available` 和 `semaphore_available` 字段，分别代表是否检测死锁以及各种锁 / 信号量的可利用数量。
- 除此之外，在线程的 `TCB_Inner` 模块中加入了锁和信号量的 `allocation` 和 `need` 字段，以表示线程分配到 / 需要的资源量。
- 然后在 `syscall/sync.rs` 中，调用 `mutex_lock` 和 `sem_down` 前需要加入检测功能。
- 还需要在 `sync/mutex.rs` 和 `sync/semaphore.rs` 中适当更新新字段的值。
- 为了满足测例要求，还需要实现 `sys_get_time` 调用。

简答题

1. 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?

     需要回收的资源包括:

     - 当前线程的 PCB、地址空间、文件描述符表
     - 所有子线程的 TID 、用户栈、 Trap 上下文（跳板页）

   - 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?

     有两个位置引用了 TCB:

     - 回收子线程（遍历 `tasks` 数组删除）时，此时需要回收
     - 回收主线程地址空间时可能被引用，但不需要回收

     这样做的原因是为了避免重复释放导致程序出错。

2. 对比以下两种 `Mutex.unlock` 的实现:

```rust
impl Mutex for Mutex1 {
    fn unlock(&self) {
        let mut mutex_inner = self.inner.exclusive_access();
        assert!(mutex_inner.locked);
        mutex_inner.locked = false;
```

```
        if let Some(waking_task) = mutex_inner.wait_queue.pop_front()
{
            add_task(waking_task);
        }
    }
}

impl Mutex for Mutex2 {
    fn unlock(&self) {
        let mut mutex_inner = self.inner.exclusive_access();
        assert!(mutex_inner.locked);
        if let Some(waking_task) = mutex_inner.wait_queue.pop_front()
{
            add_task(waking_task);
        } else {
            mutex_inner.locked = false;
        }
    }
}
```

- 二者有什么区别？这些区别可能会导致什么问题？

  前者把任务放入调度队列时，任务不一定持有锁，因为优先级高的任务在解除锁时可能已经把锁抢占了，这会导致互斥锁机制失效。

  后者的实现方式保证加入队列中的任务必定持有锁，从而保证了系统运行的安全。

其他

- 完成本次实验用时 12 小时。

Honor Code

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：

   rCore-Tutorial-Guide 2023 春季学期 的第八节。

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。