lab3

实现功能总结

`sys_spawn` 实现

本质上是 `fork` ➕ `exec` 的实现，但是不需要复制父进程的地址空间，直接从elf中创建新进程的地址空间，但是需要修改其中的几个数据，将子进程和父进程进行绑定。

`stride` 调度算法实现

在任务管理器中，将任务添加到ready队列时，按从小到大的顺序插入到ready队列中，在fetch任务时，直接pop出第一个任务，就是当前stride最小的任务，进行执行。

简答作业

> 实际情况是轮到 p1 执行吗？为什么？

不是，因为p2执行完之后，对stride执行➕pass操作，此时stride会溢出，因此p2的stride比p1小，所以p2执行。

> 为什么？尝试简单说明（不要求严格证明）。

由于最大stride和最小stride之差一定小于等于pass步长，pass=BIG_STRIDE/priority，因为priority>=2，所以最大stride和最小stride之差一定小于BIG_STRIDE/2。

> 已知以上结论，考虑溢出的情况下，可以为 Stride 设计特别的比较器，让 BinaryHeap<Stride> 的 pop 方法能返回真正最小的 Stride。补全下列代码中的 partial_cmp 函数，假设两个 Stride 永远不会相等。

```rust
use core::cmp::Ordering;

struct Stride(u64);

impl PartialOrd for Stride {
    fn partial_cmp(&self, other: &Self) -> Option<Ordering> {
        let cmp = (self.0 - other.0) as i64;
        if cmp > 0 {
            Ordering::Greater
        }
        else {
            Ordering::Less
        }
    }
```

```
}

impl PartialEq for Stride {
    fn eq(&self, other: &Self) -> bool {
        false
    }
}
```

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。