实验三报告

实现的功能

- 合并 Ch4 的相关实现到 Ch5
- 参照 `fn fork` 和 `fn exec` 完成了 `fn spawn`，主要区别在于没有复制父进程的地址空间
- 完成了 Stride 调度算法

问答题

1. stride 算法原理非常简单，但是有一个比较大的问题。例如两个 pass = 10 的进程，使用 8bit 无符号整形储存 stride， p1.stride = 255, p2.stride = 250，在 p2 执行一个时间片后，理论上下一次应该 p1 执行。

   - 实际情况是轮到 p1 执行吗？为什么？

     不是，p2 在执行完后 stride 会溢出，p2.stride < p1.stride，还是 p2 执行

2. 我们之前要求进程优先级 ≥ 2 其实就是为了解决这个问题。可以证明， 在不考虑溢出的情况下 ，在进程优先级全部 ≥ 2 的情况下，如果严格按照算法执行，那么 STRIDE_MAX – STRIDE_MIN ≤ BigStride / 2。

   - 为什么？尝试简单说明（不要求严格证明）。
   - 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。

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

struct Stride(u64);

impl PartialOrd for Stride {
    fn partial_cmp(&self, other: &Self) -> Option<Ordering> {
      if self.0 < other.0 {
        if other.0 - self.0 < BIG_STRIDE / 2 {
              Some(std::cmp::Ordering::Less)
          } else {
              Some(std::cmp::Ordering::Greater)
          }
        } else if self.0 > other.0 {
        if self.0 - other.0 < BIG_STRIDE / 2 {
              Some(std::cmp::Ordering::Greater)
          } else {
              Some(std::cmp::Ordering::Less)
          }
        }
    }
}

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

TIPS: 使用 8 bits 存储 stride, BigStride = 255, 则: (125 < 255) == false, (129 < 255) == true.

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。