ch4实验总结

进程创建

- 实现 spawn 系统调用，作为 fork + exec 的替代方案
- 直接从ELF格式数据创建新进程，而不复制父进程空间
- 设置父子进程关系以支持 wait/waitpid
- 实现了更高效的进程创建机制

Stride调度算法

- 基于优先级的进程调度算法
- 每个进程维护 stride 和 priority 属性
- stride 表示进程运行长度，pass = BIG_STRIDE / priority
- 每次选择 stride 最小的进程运行，并更新其 stride 值
- 进程优先级必须 ≥ 2，初始优先级为 16

通过这些实现，系统支持了更灵活的进程创建方式和更公平的进程调度机制。

Stride算法深入分析

溢出问题分析

- 在8bit无符号整形情况下（p1.stride = 255, p2.stride = 250）
- p2执行后 stride 变为 260（250 + 10）但由于8bit溢出变成4
- 4 < 255，导致p2会被再次选中，违背了调度公平性

优先级限制原理

1. **为什么要求优先级 ≥ 2**：
    - BIG_STRIDE / priority ≤ BIG_STRIDE/2
    - 确保任意两个进程的stride差值不超过BIG_STRIDE/2
2. **原理说明**：
    - 每次调度间隔 = BIG_STRIDE / priority
    - 当优先级都≥2时，最大间隔不超过BIG_STRIDE/2
    - 这样即使发生溢出，仍能通过特殊比较方法确定真实大小

溢出处理方案

```
if (self.0).wrapping_sub(other.0) > BIG_STRIDE/2 {
    Some(Ordering::Greater)
} else if (other.0).wrapping_sub(self.0) > BIG_STRIDE/2 {
    Some(Ordering::Less)
} else {
    Some(self.0.cmp(&other.0))
}
```

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。