实验概述

1. 适配前面章节的内容

基本保持一致，只是有些代码的位置和接口的调用方式略有变化。

2. spawn系统调用

可以同时参考 fork 和 exec 的实现，首先从 elf 数据中读取相关信息并分配内存，然后根据解析 elf 后的数据新建task，维护与当前任务的父子关系，通过*app_init_context*设置trap_ctx的内容，使新任务从用户程序的入口开始执行。

3. stride 调度算法

为每个task设置默认优先级为16，stride为0，TASK_MANAGER每次取任务不再是直接从队列中取队首任务，而是寻找stride最小的任务并移出队列，之后将该任务的stride值加上 *BIG_STRIDE* / 优先级。BIG_STRIDE简单设置为100_000，不考虑溢出。

问答作业

1. 不会，因为p2的stride再加上一个pass后会溢出，反而比p1的stride小了。

2. 只考虑有2个进程交替运行的情况，可以由2个进程推广到任意多进程。由于这两个进程的优先级都 ≥ 2，则无论哪个进程运行后，每单次 stride 增加都不会超过 $\frac{BIG\_STRIDE}{2}$。假设某次调度前，两个进程 stride 值相差为 d，且 $0 \leq d \leq \frac{BIG\_STRIDE}{2}$，那么
   - 当被选择的进程 $pass \leq d$ 时，$d = d - pass$，d 仍满足 $0 \leq d \leq \frac{BIG\_STRIDE}{2}$
   - 当被选择的进程 $pass > d$ 时，$d = pass - d$，但是 $pass \leq \frac{BIG\_STRIDE}{2}$，所以 d 仍然满足 $0 \leq d \leq \frac{BIG\_STRIDE}{2}$

   即调度前后，d 关于 $0 \leq d \leq \frac{BIG\_STRIDE}{2}$ 的假设不变。而在两个进程运行前，$d = 0$，由数学归纳法可得，两个进程的 $stride$ 之差不会超过 $\frac{BIG\_STRIDE}{2}$

3. 为了验证 *TIPS* 中的结论，这里用u8代替了u64，但核心思想一样。如果两个 stride 相差不超过 $\frac{BIG\_STRIDE}{2}$，则数值大的 stride本身也更大。 反之如果相差超过 $\frac{BIG\_STRIDE}{2}$，则数值大的 stride 本身却更小。

```
struct Stride(u8);
const BIG_STRIDE: u8 = u8::MAX;

impl PartialOrd for Stride {
    fn partial_cmp(&self, other: &Self) -> Option<Ordering> {
        let d = self.0.max(other.0) - self.0.min(other.0);
        let is_max_stride = self.0 > other.0;
        if d <= BIG_STRIDE / 2 {
            if is_max_stride {
                Some(Ordering::Greater)
            } else {
                Some(Ordering::Less)
            }
        } else if is_max_stride {
            Some(Ordering::Less)
```

```
        } else {
            Some(Ordering::Greater)
        }
    }
}

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

#[cfg(test)]
mod tests {
    use std::cmp::Ordering;
    use crate::Stride;

    #[test]
    fn test() {
        // (125 < 255) == false
        assert_eq!(Stride(125).partial_cmp(&Stride(255)),
Some(Ordering::Greater));
        // (129 < 255) == true
        assert_eq!(Stride(129).partial_cmp(&Stride(255)), Some(Ordering::Less));
    }
}
```

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。