操作系统 · Lab3

计01　容逸朗　2020010869

功能简述

- 实验主要包括 `spawn` 和 `set_priority` 的实现。

  对于 `spawn` 而言，操作和 fork+exec 是类似的：

  - 首先在 `task::task.rs` 的 TCB 中加入 spawn 方法。
  - 该方法和创建新进程的 `TaskControlBlock::new` 相近，但为了满足 fork 的定义，需要修改和 parent 相关的内容，同时要把当前进程加入 parent 的 children 列表中。
  - 在系统调用时还需要参考 exec 和 fork 的实现方式，先取得 token, cur_task 和 path，再利用前面定义的 spawn 方法生成任务，最后返回新的进程 id 即可。

  为了实现调度算法，还需要修改 `TaskManager::fetch`，使得取出的任务有最小的 stride 值，取出任务后还需要修改对应的 stride 值。

简答题

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，因为 $p2.stride + pass = 250 + 10 = 260 > 256$，超出了 8bit 可表示的最大值 255，发生溢出后值为 $4 < 255 = p1.stride$，因此还是 p2 继续执行。

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）。

  用反证法，假设某次调度后（假设是任务 x）出现 $stride_x - stride_y > \frac{BigStride}{2}$，则调度前的 stride 必有：

  $$stride'_x = stride_x - \frac{BigStride}{prior_x} > stride_x - \frac{BigStride}{2} > stride_y$$

  显然这和算法矛盾（此时应选 y 为下一个任务），故假设错误。所以严格按照算法执行下，必有 $stride_{MAX} - stride_{MIN} <= \frac{BigStride}{2}$。

- 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 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
extern crate core;
use core::cmp::Ordering;

pub const BIG_STRIDE: u64 = 0x3f3f3f3f;
struct Stride(u64);

impl PartialOrd for Stride {
    fn partial_cmp(&self, other: &Self) -> Option<Ordering> {
        let big = BIG_STRIDE / 2;
        let abs = self.0.abs_diff(other.0);
        let cmp = (self.0 as i64).wrapping_sub(other.0 as i64);
        if (abs < big && cmp > 0) || (abs >= big && cmp < 0) {
            Some(Ordering::Greater)
        } else {
            Some(Ordering::Less)
        }
    }
}

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

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 的第四部分。

   https://stackoverflow.com/q/18429021

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。