os lab3 实验报告

计算机系 2019011312 姚建竹

代码实现

sys_spawn

类似fork和exec，首先先从应用地址空间获得一个字符串，从而可以初始化一个MemorySet，用于新建一个TaskControlBlock（同时也要设置好parent）。这里就不用像fork一样完全拷贝父节点的地址空间了。然后调用add_task将新生成的TCB add到TASK MANAGER里即可

stride

sys_set_priority

在TaskControlBlock里面新增一个设置priority的接口，然后直接调用即可。（注意要检查_prio是否合法）

算法实现

在TaskManager的fetch函数内进行更改。
遍历ready_queue来获得当前stride最小的task，然后将其stride加上BIG_STRIDE/priority，返回这个指针，并在ready_queue里面remove掉即可。
其中，BIG_STRIDE设置为0x07FFFFFF

问答题

1. 实际情况还是p2执行，因为8位无符号整型取值范围是0-255，如果p2.stride加了10，溢出了，变成了4，对比的时候还是比p1.stride小，所以还是p2执行。
2. 因为优先级全部大于等于2，所以pass一定是小于二分之一的BigStride。在不考虑溢出的情况下，假设a进程的stride小于b进程，且二者差小于二分之一BigStride，则a进程的stride + pass一定大于b，且新的差不会大于二分之一BigStride。开始时二者stride都为零，经过一次调度后满足上述情况，所以如此迭代下去，如果严格按照算法执行，那么STIDE_MAX–STRIDE_MIN<=BigStride/2成立。
3. 补全partial_cmp函数。由于进程优先级全部大于等于2，所以在不溢出的情况下最大STRIDE与最小STRIDE的差一定小于BigStride/2。假设两个Stride永远不会相等。

```
fn partial_cmp(&self, other: &Self) -> Option<Ordering> {
    if self.0 < other.0 {
        if (other.0 - self.0) > (BigStride / 2) {
            return Ording::Greater;
        } else {
            return Ording::Less;
        }
    } else if self.0 > other.0 {
        if (self.0 - other.0) > (BigStride / 2) {
            return Ording::Less;
        } else {
            return Ording::Greater;
        }
    }
}
```