操作系统第四次作业

2019011008 无 92 刘雪枫

1. 同步机制应遵循的原则有：
   a) 空闲让进：当无进程在临界区时，任何有权使用临界区的进程都可以进入；
   b) 忙则等待：不允许两个以上的进程同时进入临界区；
   c) 有限等待：任何进入临界区的要求应在有限的时间内得到满足；
   d) 让权等待：不能进入临界区的进程应放弃占用 CPU。
2. 将自己的状态置为 HUNGRY 表明自己已经思考完毕想要进食。而如果此时左侧或右侧正在进食，则自己暂时无法进食，在对信号量进行 P 操作时就会开始等待。这样当左侧或右侧完成进食放下叉子时，就能够检测到它处于 HUNGRY 状态，从而对其信号量执行 V 操作将其唤醒，使其开始进食。
   在函数 put_forks 中，如果在对 test 两次调用之前不将 state[i]置为 THINKING，则该值为 EATING。那么，在 test 中，如果其相邻的哲学家是 HUNGRY 的且已经或即将执行 down 操作，则判断会因为自己的状态是 EATING 而为假，因此不会对该相邻的哲学家的信号量执行 up 操作，则该相邻的哲学家不会被唤醒而进入无限的等待。
3. 条件变量的 signal 用于唤醒正在进行等待的进程；而信号量的 V 操作会唤醒一个正在等待的进程，如果无进程等待则将计数加 1。两者的一个比较重要的区别是，如果没有进程正在进行等待，那么条件变量的 signal 操作就会丢失，但是信号量的 V 操作则会对计数加 1 而不会丢失。
4. 我认为，发送和接收是选择阻塞式还是非阻塞式要视应用场景而决定，不同的情况可能会有不同的需要。相比之下，我更倾向于非阻塞式发送和阻塞式接收：发送时，只需要将消息发出即可；而接收时，接收更多意味着想要得到结果，因此得不到消息时阻塞直到取得消息也许会是应用更多的情况。
5. 本题只需要设置一个初始值为 200 的信号量即可。当有游客要进入博物馆时对其进行 P 操作，离开时进行 V 操作。该算法的伪代码如下：

```
typedef int semaphore;
semaphore gate_machine = 200;

void enter_museum(void) {
    P(gate_machine);
}

void exit_museum(void) {
    V(gate_machine);
}

void visitor(void) {
    enter_museum();
    visit_museum();
    exit_museum();
}
```