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：
   1. 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；
   2. 忙则等待：不允许两个以上的进程同时进入临界区；
   3. 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；
   4. 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。

1. 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。
2. 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。
3. 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：
4. **typedef** **int** semaphore;
5. semaphore gate\_machine = 200;
7. **void** enter\_museum(**void**) {
8. P(gate\_machine);
9. }
11. **void** exit\_museum(**void**) {
12. V(gate\_machine);
13. }
15. **void** visitor(**void**) {
16. enter\_museum();
17. visit\_museum();
18. exit\_museum();
19. }