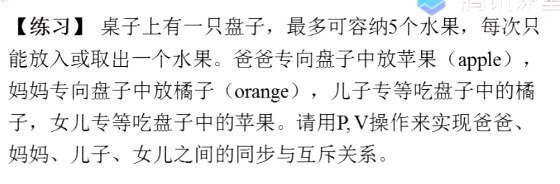
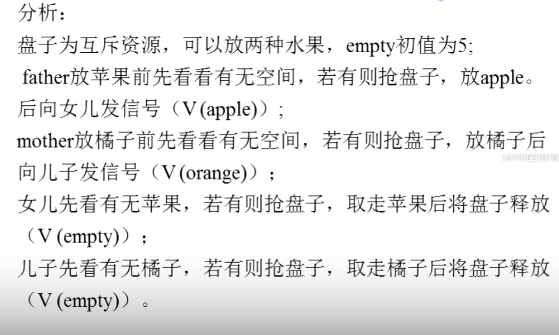
# 题目一





semaphore empty = 5, mutex = 1, apple = 1, orange = 0; // 不要漏了 mutex，这是盘子的锁，实现盘子资源的互斥

**爸爸 ：**

void father() {

while (True) {

P(empty);

P(mutex);

放苹果

V(mutex)  
 V(apple)

}

}

**妈妈 ：**

void mather() {

while (True) {

P(empty);

P(mutex);

放橘子  
 V(mutex);

V(orange);

}

}

**Son :**

V(mutex)

void Son() {

while (True) {

P(apple);

P(mutex);

吃苹果

V(mutex);

V(empty);

}

}

**Daughter :**

void Daughter() {

while (True) {

P(orange);

P(mutex);

吃橘子

V(mutex);

V(empty);

}

}

void main() {

cobegin

father(); mother(); Son(); Daughter();

coend

}

# 哲学家就餐问题

[哲学家进餐问题\_哔哩哔哩\_bilibili](https://www.bilibili.com/video/BV17Q4y1m7Pe/?spm_id_from=333.337.search-card.all.click&vd_source=cdddf6b462dc99204224a6ed9c8e2636)

假设 5 个哲学家，先看方案一 ：

For philosopher-i :

P (chopstick[left[i]]) ;

P (chopstick[right[i]]) ;

eat ;

V (chopstick[left[i]]) ;

V (chopstick[right[i]]) ;

think;

这可能造成死锁，if all the 5 philosophers take the left chopsticks simultaneously.

Consider this solution :

Control that there has 4 philosophers eating at most，and he can take the chopstick only his left and right chopsticks can be taken.

semaphore cs[5] = {1, 1, 1, 1, 1}, count = 4;

P[i]() {

do {

P(count);

P(cs[i]);

P(cs[(i+1)%5]);

eating ;

V(cs[i]);

V(cs[(i+1)%5]);  
 V(count);

thinking;

}while(1);

}

其实这个 Solution 不一定好，描述不准确，实际是通过 N < 资源 保证可以同时拿筷子的。  
这个是参考视频里写的，其实是不对的，并没有保证拿筷子的人一定同时拿起左右筷子。

semaphore cs[5] = {1, 1, 1, 1, 1}, mutex = 1;

P[i]() {

do {

P(mutex);

P(cs[i]);

P(cs[(i+1)%5]);

V(mutex);

eating ;

V(cs[i]);

V(cs[(i+1)%5]);

thinking;

}while(1);

}