# “哲学家进餐”问题

教材中解决哲学家进餐问题的算法有可能发生死锁，为避免死锁，可以采用以下三种策略：

策略一原理：至多只允许四个哲学家同时进餐，以保证至少有一个哲学家能够进餐，最终总会释放出他所使用过的两支筷子，从而可使更多的哲学家进餐。定义信号量count，初值为4，表示只允许4个哲学家同时进餐，这样就能保证至少有一个哲学家可以就餐。

semaphore chopstick[5]={1，1，1，1，1};

semaphore count=4;

void philosopher(int i)

{while(true)

{

think();

wait(count); //请求进入房间进餐

wait(chopstick[i]); //请求左手边的筷子

wait(chopstick[(i+1)%5]); //请求右手边的筷子

eat();

signal(chopstick[(i+1)%5]); //释放右手边的筷子

signal(chopstick[i]); //释放左手边的筷子

signal(count); //退出房间释放信号量

}

}

策略二原理：仅当哲学家的左右两支筷子都可用时，才允许他拿起筷子进餐。可以利用AND 型信号量机制实现，也可以利用信号量的保护机制实现。利用信号量的保护机制实现的思想是通过记录型信号量mutex对取左侧和右侧筷子的操作进行保护，使之成为一个原子操作，这样可以防止死锁的出现。描述如下：

semaphore mutex = 1 ;

semaphore chopstick[5]={1，1，1，1，1};

void philosopher(int i)

{while(true)

{

think();

wait(mutex);

wait(chopstick[(i+1)%5]);

wait(chopstick[i]);

signal(mutex);

eat();

signal(chopstick[(i+1)%5]);

signal(chopstick[i]);

}

}

策略三原理：规定奇数号的哲学家先拿起他左边的筷子，然后再去拿他右边的筷子；而偶数号的哲学家则相反。按此规定，将是1、2号哲学家竞争1号筷子，3、4号哲学家竞争3号筷子。即五个哲学家都竞争奇数号筷子，获得后，再去竞争偶数号筷子，最后总会有一个哲学家能获得两支筷子而进餐。

semaphore chopstick[5]={1，1，1，1，1};

void philosopher(int i)

{while(true)

{

think();

if(i%2 == 0) //偶数哲学家，先右后左。

{

wait (chopstick[(i + 1)%5]) ;

wait (chopstick[i]) ;

eat();

signal (chopstick[(i + 1)%5]) ;

signal (chopstick[i]) ;

}

else //奇数哲学家，先左后右。

{

wait (chopstick[i]) ;

wait (chopstick[(i + 1)%5]) ;

eat();

signal (chopstick[i]) ;

signal (chopstick[(i + 1)%5]) ;

}

}

# 例题精析

【例1】 桌上有1空盘，允许存放1个水果。爸爸向盘中放苹果，也可以向盘中放桔子。儿子专等吃盘中的桔子，女儿专等吃盘中的苹果。规定当盘空时一次只能放1个水果供吃者取用。请用Wait()、Signal()原语实现爸爸、儿子、女儿三个并发进程的同步。

【分析】本题目考查进程的同步与互斥。这是第二种情况的“生产者—消费者”问题，既有同步又有互斥。爸爸进程需要与儿子进程、女儿进程同步，儿子进程与女儿进程需要互斥。设置4个信号量S（盘子是否为空，初值为1）、So（盘中是否有桔子，初值为0）、Sa（盘中是否有苹果，初值为0）和mutex（用于对盘子的互斥访问，初值为1）。由于只有一个盘子（相当于只有一个缓冲区），S、So和Sa就可以保证对盘子的互斥操作了，故mutex也可以省略。

【答案】

设三个信号量：

S --- 盘子是否为空，初值为1；

So --- 盘中是否有桔子，初值为0；

Sa --- 盘中是否有苹果，初值为0；

Semaphore S=1, So=0, Sa=0;

Main() {

Cobegin

Father();

Son();

Daughter();

Coend

}

Father() {

while(1) {

wait(S); 将水果放入盘中；

if (放入的是桔子) signal(So);

else signal(Sa);

}

}

Son() {

while(1) {

wait(So); 从盘中取出桔子；signal(S); 吃桔子；

}

}

Daughter() {

while(1) {

wait(Sa); 从盘中取出苹果；signal(S); 吃苹果；

}

}

【例2】桌上有1空盘，最多可以容纳两个水果，每次只能放入或取出1个水果。爸爸专向盘中放苹果，妈妈专向盘中放桔子。儿子专等吃盘中的桔子，女儿专等吃盘中的苹果。请用Wait()、Signal()原语实现爸爸、妈妈、儿子、女儿之间的同步与互斥关系。

【分析】本题目考查进程的同步与互斥。本题目可以看做是两个生产者和两个消费者共享一个大小为2的缓冲区。由于盘子每次只能放入或取出一个水果，可知爸爸和妈妈以及儿子和女儿之间是互斥的关系。爸爸放苹果，女儿吃苹果；妈妈放桔子，儿子吃桔子。因此爸爸和女子及妈妈和儿子是同步的关系。因此需设置4个信号量，其中empty表示还可以向盘中放几个水果，其初值为2；apple对应已放入盘中的苹果，orange对应已放入盘中的橘子，它们的初值均为0；mutex用来实现对盘子的互斥访问，其初值为1。

【答案】

semaphore mutex=1,empty=2,apple=0,orange=0;

Main()

{

Cobegin

Father();

Mather();

Son();

daughter();

coend

}

Father()

{ while(1)

{ p(empty);

P(mutex);

向盘中放水果；

V(mutex);

V(apple);

}

}

Mather()

{ while(1)

{ p(empty);

P(mutex);

向盘中放水果；

V(mutex);

V(orange);

}

Son()

{ while(1)

{ p(orange);

P(mutex);

从盘中取橘子；

V(mutex);

V(empty);

吃橘子；

}

}

daughter()

{ while(1)

{ p(apple);

P(mutex);

从盘中取橘子；

V(mutex);

V(empty);

吃苹果；

}

}