**吃水果问题**

semaphore empty= 5, orange = 0, apple = 0, mutex = 1;

Dad() {

while (1) {

wait(empty);

wait(mutex);

将水果放入盘中;

signal(mutex);

if (放入的是桔子)  signal(orange);  

else  signal(apple);

}

}  
Son() {

while (1) {  
  
   wait(orange);  
  
   wait(mutex);  
  
   从盘中取一个桔子;  
  
   signal(mutex);  
  
   signal(empty);  
  
   享用桔子;  
  
 }  
  
}  
b  
  
 while (1) {  
  
   wait(apple);  
  
   wait(mutex);  
  
   从盘中取一个苹果;  
  
   signal(mutex);  
  
   signal(empty);  
  
   享用苹果;  
  
 }  
  
}  
  
void main(){  
  
cobegin  
  
Dad(); Son();Daughter();  
  
coend

**共享缓冲区的进程的同步问题**

Semaphore mutex = 1;

void Computing\_Process() {  
while(Computing\_Finish() == false) {  
Compute();  
P(s\_empty);  
P(mutex);  
Send\_To\_Buffer();  
V(mutex);  
V(s\_full);  
}  
}  
void Input\_Output\_Process() {  
while(Output\_Finish() == false) {  
P(s\_full);  
P(mutex);  
Get\_From\_Buffer();  
V(mutex);  
V(s\_empty);  
Output();  
}  
}

**哲学家就餐问题**

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#include <time.h>

#include <unistd.h>

#include <pthread.h>

#include <semaphore.h>

#define N 5

//信号量使用的参数

sem\_t chopsticks[N];

sem\_t r;

int philosophers[N] = {0, 1, 2, 3, 4};

//swap指令需要的参数

int islocked[N] = {0};

//互斥量使用的参数

pthread\_mutex\_t chops[N];

//延迟函数

void delay (int len) {

int i = rand() % len;

int x;

while (i > 0) {

x = rand() % len;

while (x > 0) {

x--;

}

i--;

}

}

//交换函数：目前为发现bug

void xchg(int \*x, int \*y) {

\_\_asm\_\_("xchgl %0, %1" : "=r" (\*x) : "m" (\*y));

}

//这个函数使用的解决办法是最多允许四个哲学家拿起左筷子

void philosopher (void\* arg) {

int i = \*(int \*)arg;

int left = i;

int right = (i + 1) % N;

int leftkey;

int rightkey;

while (1) {

leftkey = 1;

rightkey = 1;

printf("哲学家%d正在思考问题\n", i);

delay(50000);

printf("哲学家%d饿了\n", i);

sem\_wait(&r);

//sem\_wait(&chopsticks[left]);

//pthread\_mutex\_lock(&chopsticks[left]);

do {

xchg(&leftkey, &islocked[left]);

}while (leftkey);

printf("哲学家%d拿起了%d号筷子,现在只有一支筷子,不能进餐\n", i, left);

//sem\_wait(&chopsticks[right]);

//pthread\_mutex\_lock(&chopsticks[right]);

do {

xchg(&rightkey, &islocked[right]);

}while (rightkey);

printf("哲学家%d拿起了%d号筷子, 现在有两支筷子,开始进餐\n", i, right);

delay(50000);

//sem\_post(&chopsticks[left]);

//pthread\_mutex\_unlock(&chopsticks[left]);

islocked[left] = 0;

printf("哲学家%d放下了%d号筷子\n", i, left);

//sem\_post(&chopsticks[right]);

//pthread\_mutex\_unlock(&chopsticks[right]);

islocked[right] = 0;

printf("哲学家%d放下了%d号筷子\n", i, right);

sem\_post(&r);

}

}

//这个函数使用的解决办法是奇数号哲学家先拿左筷子再拿右筷子，而偶数号哲学家相反。

void philosopher2 (void\* arg) {

int i = \*(int \*)arg;

int left = i;

int right = (i + 1) % N;

while (1) {

printf("哲学家%d正在思考问题\n", i);

delay(50000);

printf("哲学家%d饿了\n", i);

if (i % 2 == 0) {//偶数哲学家，先右后左

sem\_wait(&chopsticks[right]);

printf("哲学家%d拿起了%d号筷子,现在只有一支筷子,不能进餐\n", i, right);

sem\_wait(&chopsticks[left]);

printf("哲学家%d拿起了%d号筷子, 现在有两支筷子,开始进餐\n", i, left);

delay(50000);

sem\_post(&chopsticks[left]);

printf("哲学家%d放下了%d号筷子\n", i, left);

sem\_post(&chopsticks[right]);

printf("哲学家%d放下了%d号筷子\n", i, right);

} else {//奇数哲学家，先左后又

sem\_wait(&chopsticks[left]);

printf("哲学家%d拿起了%d号筷子, 现在有两支筷子,开始进餐\n", i, left);

sem\_wait(&chopsticks[right]);

printf("哲学家%d拿起了%d号筷子,现在只有一支筷子,不能进餐\n", i, right);

delay(50000);

sem\_post(&chopsticks[right]);

printf("哲学家%d放下了%d号筷子\n", i, right);

sem\_post(&chopsticks[left]);

printf("哲学家%d放下了%d号筷子\n", i, left);

}

}

}

int main (int argc, char \*\*argv) {

srand(time(NULL));

pthread\_t PHD[N];

for (int i=0; i<N; i++) {

sem\_init(&chopsticks[i], 0, 1);

}

sem\_init(&r, 0, 4);

for (int i=0; i<N; i++) {

pthread\_mutex\_init(&chops[i], NULL);

}

for (int i=0; i<N; i++) {

pthread\_create(&PHD[i], NULL, (void\*)philosopher2, &philosophers[i]);

}

for (int i=0; i<N; i++) {

pthread\_join(PHD[i], NULL);

}

for (int i=0; i<N; i++) {

sem\_destroy(&chopsticks[i]);

}

sem\_destroy(&r);

for (int i=0; i<N; i++) {

pthread\_mutex\_destroy(&chops[i]);

}

return 0;

}