目录

[2.5经典进程的同步问题 2](#_Toc463898711)

[2.5.1生产者——消费者关系 2](#_Toc463898712)

[1利用记录型信号量解决生产者——消费者问题（P60） 2](#_Toc463898713)

[2利用AND信号量解决生产者——消费者问题(P61) 3](#_Toc463898714)

[3利用管程解决生产者——消费者问题(P62) 3](#_Toc463898715)

[2.5.2哲学家进餐问题 4](#_Toc463898716)

[1利用记录型信号量解决哲学家进餐问题(P64) 4](#_Toc463898717)

[2利用AND信号量机制解决哲学家进餐问题(P64) 5](#_Toc463898718)

[2.5.3读者——写者问题 5](#_Toc463898719)

[1利用记录型信号量解决读者——写者问题(P65) 5](#_Toc463898720)

[2利用信号量机制解决读者——写者问题(P66) 6](#_Toc463898721)

[2.6进程通信 7](#_Toc463898722)

[2.6.3直接消息传递系统实例 7](#_Toc463898723)

[1消息缓冲队列通信机制中的数据结构(P73) 7](#_Toc463898724)

[2发送原语(P74) 7](#_Toc463898725)

[3接收原语(P74) 8](#_Toc463898726)

# 2.5经典进程的同步问题

## 2.5.1生产者——消费者关系

### 1利用记录型信号量解决生产者——消费者问题（P60）

int in=0,out=0;

item buffer[n];

semaphore mutex=1,empty=n,full=0;

void proceducer(){

do{

producer an item nextp;

...

wait(empty);

wait(mutex);

buffer[in]=nextp;

in:=(in+1)%n;

signal(mutex);

signal(full);

}while(TRUE);

}

void consumer(){

do{

wait(full);

wait(mutex);

nextc=buffer[out];

out=(out+1)%n;

signal(mutex);

signal(empty);

consumer the item in nextc;

...

}while(TRUE);

}

void main(){

cobegin

proceducer();consumer();

coend

}

### 2利用AND信号量解决生产者——消费者问题(P61)

int in=0,out=0;

item buffer[n];

semaphore mutex=1,empty=n,full=0;

void proceducer(){

do{

producer an item nextp;

...

Swait(empty,mutex);

buffer[in]=nextp;

in:=(in+1)%n;

Ssignal(mutex,full);

}while(TRUE);

}

void consumer(){

do{

Swait(full,mutex);

nextc=buffer[out];

out=(out+1)%n;

Ssignal(mutex,empty);

consumer the item in nextc;

...

}while(TRUE);

}

### 3利用管程解决生产者——消费者问题(P62)

#### （1）

Monitor proceducerconsumer{

item buffer[N];

int in,out;

condition notfull,notempty;

int count;

public:

void put(item x){

if(count>=N)cwait(notfull);

buffer[in]=x;

in=(in+1)%N;

count++;

csignal(notempty);

}

void get(item x){

if(count<=0)cwait(notempty);

x=buffer[out];

out=(out+1)%N;

count--;

csignal(notfull);

}

{ in=0;out=0;count=0;}

}PC;

#### （2）(P63)

void proceducer(){

item x;

while(TRUE){

...

produce an item nextp;

PC.put(x);

}

}

void consumer(){

item x;

while(TRUE){

PC.get(x);

consume the item nextp;

...

}

}

## 2.5.2哲学家进餐问题

### 1利用记录型信号量解决哲学家进餐问题(P64)

do{

wait(chopstick[i]);

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

...

//eat

...

signal(chopstick[i]);

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

...

//think

...

}while[TTRUE];

### 2利用AND信号量机制解决哲学家进餐问题(P64)

semaphore chopstick chopstick[5]={1,1,1,1,1};

do{

...

//think

...

Swait(chopstick[i+1]%5,chopstick[i]);

...

//eat

...

Ssignal(chopstick[i+1]%5),chopstick[i]);

}while[TTRUE];

## 2.5.3读者——写者问题

## 1利用记录型信号量解决读者——写者问题(P65)

semaphore rmutex=1,wmutex=1;

int readcount=0;

void reader(){

do{

wait(rmutex);

if(readcount==0)wait(wmutex);

readcount++;

signal(rmutex);

...

perform read operation;

...

wait(rmutex);

readcount--;

if(readcount==0) signal(wmutex);

signal(rmutex);

}while(TRUE);

}

void writer(){

do{

wait(wmutex);

perform write operation;

signal(wmutex);

}while(TRUE);

}

void main(){

cobegin

reader();writer();

coend

}

## 2利用信号量机制解决读者——写者问题(P66)

int RN;

semaphore L=RN,mx=1;

void reader(){

do{

Swait(L,1,1);

Swait(mx,1,0);

...

perform read operation;

...

Ssignal(L,1);

}while(TRUE);

}

void writer(){

do{

Swait(mx,1,1;L,RN,0);

perform write operation;

Ssignal(mx,1);

}while(TRUE);

}

void main(){

cobegin

reader();writer();

coend

}

# 2.6进程通信

## 2.6.3直接消息传递系统实例

### 1消息缓冲队列通信机制中的数据结构(P73)

#### (1)

typedef struct messages\_buffer{

int sender; 发送者进程标识符

int size; 消息长度

char\*text; 消息正文

struct messages\_buffer\*next; 指向下一个消息缓冲区的指针

}

#### (2)

typedef struct processcontrol\_block{

...

struct messages\_buffer\*mq; 消息队列队首指针

semaphore mutex; 消息队列队互斥信号量

semaphore sm; 消息队列队资源信号量

...

}PCB;

### 2发送原语(P74)

void send(receiver,a){ receiver为接收进程标识符，a为发送区首址；

getbuf(a.size,i); 根据a.size申请缓冲区；

i.sender=a.sender;

i.size=a.size;

copy(i.text,a.text); 将发送区a中的信息复制到消息缓冲区i中；

i.next=0;

getid(PCBset,receiver.j); 获得接收进程内部的标识符；

wait(j.mutex);

insert(&j.mq,i); 将消息缓冲区插入消息队列；

signal(j.mutex);

signal(j.sm);

}

### 3接收原语(P74)

void receive(b){

j=internal name; j为接收进程内部的标识符；

waiit(j.sm);

wait(j.mutex);

remove(j.mq,i); 将消息队列中第一个消息移出；

signal(j.mutex);

b.sender=i.sender;

b.size=i.size;

copy(b.text,i.text); 将消息缓冲区i中信息复制到接收区b；

releasebuf(i); 释放消息缓冲区；

}