**操作系统 作业7-8**

**陈彦帆 2018K8009918002**

1. 初始available 00x12

若x=0，无法满足任何一个进程。发生死锁。

若x=1，可满足D，available变为11222

可满足A，available变为21433

可满足C，available变为32443

然后可满足B。为安全状态。

故x最小为1。

2. 不失一般性，设A需要资源的顺序为123。

对于B的6种顺序：213,231,312,321，当A和B分别各占一个资源时，会发生死锁。

而132,123不会死锁，因为A和B只有一个进程能占到1，另一个进程占不到3个资源的任何一个。

故不会发生死锁的占1/3

**作业8**

1. 产生结果的顺序如下：

y=1 x=1

y=3 x=2

z=4

x=5

y=z+3 z=9

最终x=5, z=9

若以上最后一行y=z+3先执行，则y=4+3=7，反之y=9+3=12。

2.

typedef struct Service{

    sem\_t mutex;    //init as 1

    sem\_t wait\_num; //init as 0

    int head\_id;    //init as 0

    int tail\_id;    //init as 0

} service;

int Customer\_Service(service \*sp)

{

    int service\_id;

    P(&sp->mutex);

    service\_id = sp->head\_id++;

    V(&sp->mutex);

    V(&sp->wait\_num);

    return service\_id;

}

int Teller\_Service(service \*sp)

{

    int service\_id;

    P(&sp->wait\_num);

    P(&sp->mutex);

    service\_id = sp->tail\_id++;

    V(&sp->mutex);

    return service\_id;

}

3.

#include <stack>

#include <cstdio>

#include <ctime>

#include <cstdlib>

#include <pthread.h>

#include <unistd.h>

using namespace std;

class Monitor

{

private:

    stack<long>buf;

    int count;

    pthread\_mutex\_t mutex;

    pthread\_cond\_t cond;

public:

    Monitor()

    {

        pthread\_mutex\_init(&mutex,0);

        pthread\_cond\_init(&cond,0);

        count = 15;

        printf("Init val: ");

        srand(time(NULL));

        for(int i=0;i<16;i++){

            long t = rand();

            buf.push(t);

            printf("%ld ",t);

        }

        printf("\n");

    }

    bool getTask(long result[2])

    {

        pthread\_mutex\_lock(&mutex);

        if(--count<0){

            pthread\_mutex\_unlock(&mutex);

            return 0;

        }

        while(buf.size()<2)

            pthread\_cond\_wait(&cond,&mutex);

        result[0] = buf.top();

        buf.pop();

        result[1] = buf.top();

        buf.pop();

        pthread\_mutex\_unlock(&mutex);

        return 1;

    }

    void putTask(long val)

    {

        pthread\_mutex\_lock(&mutex);

        buf.push(val);

        pthread\_cond\_signal(&cond);

        pthread\_mutex\_unlock(&mutex);

    }

    long getResult(void)

    {

        return buf.top();

    }

};

void \* calculate(void \* vargp)

{

    Monitor \* monitor = (Monitor \*)vargp;

    long buf[2] = {0};

    while(monitor->getTask(buf)){

        printf("getTask: %ld %ld\n",buf[0],buf[1]);

        usleep(1000 \* (rand()%10+1));

        long r = buf[0] + buf[1];

        monitor->putTask(r);

        printf("putTask: %ld\n",r);

    }

    return NULL;

}

int main()

{

    Monitor monitor;

    pthread\_t tid[8];

    for(int i=0;i<8;i++){

        pthread\_create(&tid[i],NULL,calculate,&monitor);

    }

    for(int i=0;i<8;i++)

        pthread\_join(tid[i],NULL);

    long r = monitor.getResult();

    printf("Result: %ld\n",r);

    return 0;

}

运行结果之一：

Init val: 1016289433 619249496 1174238286 1279589369 662835321 715775781 1709492426 651218022 1000144187 1609189160 1791190552 1706415962 1393622045 1962571893 1510514068 590284022

getTask: 590284022 1510514068

getTask: 1962571893 1393622045

getTask: 1706415962 1791190552

getTask: 1609189160 1000144187

getTask: 651218022 1709492426

getTask: 715775781 662835321

getTask: 1279589369 1174238286

getTask: 619249496 1016289433

putTask: 2100798090

getTask: 2609333347 2100798090

putTask: 2609333347

putTask: 1635538929

getTask: 2453827655 1635538929

putTask: 2453827655

putTask: 2360710448

putTask: 4710131437

getTask: 4710131437 2360710448

putTask: 3356193938

getTask: 1378611102 3356193938

putTask: 1378611102

putTask: 4089366584

putTask: 3497606514

getTask: 3497606514 4089366584

putTask: 4734805040

putTask: 7586973098

getTask: 7586973098 4734805040

putTask: 7070841885

putTask: 12321778138

getTask: 12321778138 7070841885

putTask: 19392620023

Result: 19392620023

经检验结果正确。