实验三 银行家算法

姓名：黄彦　　学号:71112113　　提交日期：2013.3.22

【实验内容】

1.
在 Windows 操作系统上,

利用 Win32 API 编写多线程应用程序实现银行家

算法。

2. 创建 n 个线程来申请或释放资源,
只有保证系统安全,
才会批准资源申请。

3. 通过 Win32 API 提供的信号量机制,实现共享数据的并发访问。

【实验目的】

通过实验,
加深对多实例资源分配系统中死锁避免方法——银行家算法的理解,掌握 Windows 环境下银行家算法的实现方法,同时巩固利用 Windows API
进行共享数据互斥访问和多线程编程的方法。

【实验过程】

因为电脑没有 windows 操作系统，所以在 linux 环境下进行了实现。
实验代码：

```
#include "stdio.h"
#include "pthread.h"
#include "semaphore.h"

#define PTHNUM 5
#define RESNUM 3

#define NOTGET 0
#define GET 1
#define UNSAFE 0
#define SAFE 1

// int available[RESNUM]={10,5,7};
int available[RESNUM]={3,3,2};
int max[PTHNUM][RESNUM]={7,5,3,3,2,2,9,0,2,2,2,2,4,3,3};
int allocation[PTHNUM]
```

```
[RESNUM]={0,1,0,2,0,0,3,0,2,2,1,1,0,0,2};
// int allocation[PTHNUM][RESNUM]={0};
int need[PTHNUM][RESNUM];
int finish[PTHNUM]={NOTGET};                    //NOTGET or GET

sem_t recourse[PTHNUM];
sem_t mutex;

void request_thread(int* value);   //thread

int safe_state_test(int value)
{   int all_sum=0;
    if(finish[value]==NOTGET)
    { int i,j;
      for(j=0;j<RESNUM;j++)
      for(i=0;i<PTHNUM;i++)
      {      all_sum += allocation[i][j];
             if(all_sum + available[j] < need[value][j])
             return UNSAFE;
             all_sum=0;
      }
      for(i=0;i<RESNUM;i++)
      {
             available[i] += allocation[value][i];
             allocation[value][i] = 0;
             need[value][i] = max[value][i];
      }
      finish[value]=GET;
    }
      return SAFE;
}

void main()
{
    int i,j;
    for(j=0;j<PTHNUM;j++)
    for(i=0;i<RESNUM;i++)
      need[j][i] = max[j][i] - allocation[j][i];
    pthread_t ptid[PTHNUM];
    pthread_attr_t attr[PTHNUM];

    sem_init(&recourse[0],0,10);
    sem_init(&recourse[1],0,5);
    sem_init(&recourse[2],0,7);
```

```
    sem_init(&mutex,0,1);

    for(i=0;i<PTHNUM;i++)
       pthread_attr_init(&attr[i]);
    int thread_count[PTHNUM];
    for(i=0;i<PTHNUM;i++)
    { thread_count[i]=i;
       pthread_create(&ptid[i],&attr[i],request_thread,&thread_coun
t[i]);
    }

    pthread_join(ptid[0],NULL);
    pthread_join(ptid[1],NULL);
    pthread_join(ptid[2],NULL);
    pthread_join(ptid[3],NULL);
    pthread_join(ptid[4],NULL);

}

void request_thread(int* value)
{
    while(1)
    {
    sleep(1);
    finish[*value]=NOTGET;
    sem_wait(&mutex);
            // printf("the thread%d\n",*value);
    if(safe_state_test(*value))
       {    int i;
            for(i=0;i<max[*value][0];i++)
                sem_wait(&recourse[0]);

            for(i=0;i<max[*value][1];i++)
                sem_wait(&recourse[1]);

            for(i=0;i<max[*value][2];i++)
                sem_wait(&recourse[2]);

            printf("this is thread%d\n",*value);
            for(i=0;i<max[*value][0];i++)
                sem_post(&recourse[0]);

            for(i=0;i<max[*value][1];i++)
                sem_post(&recourse[1]);
```

```
            for(i=0;i<max[*value][2];i++)
                sem_post(&recourse[2]);
        }
    sem_post(&mutex);
    }
}
```

gcc banker.c -o banker -l pthread 进行编译。

【心得体会】
根据老师上课内容进行了实践，算法主要内容是安全状态的测试。在安全状态下才允许分配资源。而分配资源选择使用多个信号量循环的方式，不知道有没有其他更好的办法。