**武汉大学国家网络安全学院**

**实验报告**

**课程名称 操作系统**

**专业年级 信安21级**

**姓名学号 赵伯俣2021302181156**

**协 作 者 无**

**实验学期 2022-2023　 学年 第二 学期**

**课堂时数 32 课外时数**

**填写时间 2023 年 5 月 2 日**

一、实验目的

采用POSIX线程、互斥锁和信号量，解决睡觉助教问题协调助教TA和学生的活动

二、实验环境

Vscode编译器进行实验代码的编译和调试功能

三、实验源代码

#include<stdio.h>

#include<string.h>

#include<pthread.h>

#include<unistd.h>

#include<semaphore.h>

sem\_t ta,student;   //无名信号量ta和student

pthread\_mutex\_t mutex;  //互斥锁数据类型

int waiting = 0;  //等待的学生数

void\* teaching(void \*value)

{

    while(true)

    {

        sem\_wait(&student);  //获得信号量

        pthread\_mutex\_lock(&mutex);  //获取互斥锁

        waiting--;

        pthread\_mutex\_unlock(&mutex);  //释放互斥锁

        printf("TA is teaching\n");

        sleep(5);

        sem\_post(&ta);   //释放信号量

        printf("TA is sleeping\n");

    }

}

void\* consulting(void \*value)

{

    while(true)

    {

        pthread\_mutex\_lock(&mutex);

        if(waiting<=3)//有椅子，可以等待

        {

            waiting++;

            pthread\_mutex\_unlock(&mutex);

            printf("student%lu is waiting for TA\n",pthread\_self());

            sem\_post(&student);

            sem\_wait(&ta);

            printf("student%lu is consulting the TA\n",pthread\_self());

            sleep(5);

            printf("consultiation is over\n");

        }

        else//没有椅子

        {

            pthread\_mutex\_unlock(&mutex);

            printf("No more chairs,student%lu has left\n",pthread\_self());

            sleep(3);

        }

    }

}

int main()

{

    pthread\_t student1,student2,student3,student4,student5,teacher;

    //初始化无名信号量

    sem\_init(&ta,0,1);

    sem\_init(&student,0,0);

    //创建互斥锁

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

    //创建线程

    pthread\_create(&student1,NULL,consulting,NULL);//学生线程

    pthread\_create(&student2,NULL,consulting,NULL);

    pthread\_create(&student3,NULL,consulting,NULL);

    pthread\_create(&student4,NULL,consulting,NULL);

    pthread\_create(&student5,NULL,consulting,NULL);

    pthread\_create(&teacher,NULL,teaching,NULL);//TA助教线程

    //线程间的阻塞

    pthread\_join(student1,NULL);//学生线程

    pthread\_join(student2,NULL);

    pthread\_join(student3,NULL);

    pthread\_join(student4,NULL);

    pthread\_join(student5,NULL);

    pthread\_join(teacher,NULL);//TA助教线程

    return 0;

}

四、实验原理

1.main函数。

在main函数中创建pthread\_t类型的变量student1~5和teacher作为线程调用的线程号。

然后使用函数sem\_init初始化建立的信号量，该函数存在三个函数调用，分别为信号量的指针，表示共享级别的标志，信号量的初始值，其中在信号量ta的初始化过程中，将ta信号量的共享级别设置为0，将信号量的初始值设置为1。在信号量student的初始化过程中将其信号量的共享级别设置为0，将其初始值设置为1，表示还没有学生前来寻求帮助。

之后通过pthread\_create函数创造线程，为5个学生和一个助教分别创造对应的线程，在创建学生线程时调用函数consulting函数，在创建助教线程时调用函数teaching。

2.teaching函数。

该函数整体在一个while循环中，在每一次的循环中都要使用函数sem\_wait来获取student信号量并且对互斥锁信号量mutex进行上锁操作。并将变量waiting减去1，代表取出一个等待区的学生解决问题，修改完后将mutex重新上锁并输出助教正在帮助学生，并让程序休眠一段时间，代表助教正在帮助学生并且方便观察程序输出的结果，然后使用函数sem\_post释放信号量ta，并且输出助教正在睡觉。

3.consulting函数。

consulting函数与teaching函数相同也是处在一个整体的无限循环中，在每一次循环中都将互斥信号量mutex进行上锁，然后判断等待区的学生数量，如果等待区学生数量超过3，则新来的学生会将mutex解锁后直接离开，并且输出当前学生线程的线程号，表示该学生离开。

反之，如果等待区的学生数量小于3，则将等待区的学生数量+1，将互斥锁mutex解锁之后输出该学生正在等待助教，将student信号量释放以便于助教线程进行访问，在ta助教线程对该学生访问后输出该学生正在求助TA并且使得程序休眠一段时间，代表帮助过程同时使得程序的输出结果更加方便观察。最后输出帮助完毕。

4. pthread.h库函数的应用。

在本次实验中使用pthread库函数中的函数完成了多线程应用程序的设计，以下对于使用到的函数进行说明。

（1）pthread\_creat函数。该函数的原型为函数原型：int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*start\_routine)(void \*), void \*arg)

其中thread变量代表线程的句柄，在使用之前需要先定义为pthread\_t类型的变量，将变量的地址传递进入函数中去进而得到系统创建好的线程句柄。

其中attr变量代表线程的属性，在本次实验中使用默认的属性所以直接传递为NULL。

start\_routine变量代表线程中所执行的函数，将其所要运行的函数名传递入函数即可。

arg变量代表线程的参数，即为主线程传递给子线程的参数，在本次实验中无需传递数据，因此将该变量定义为NULL。

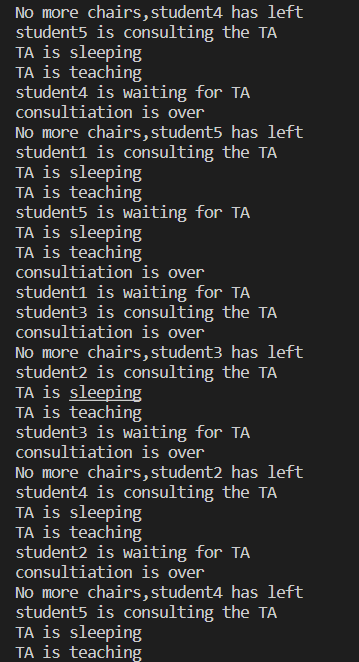
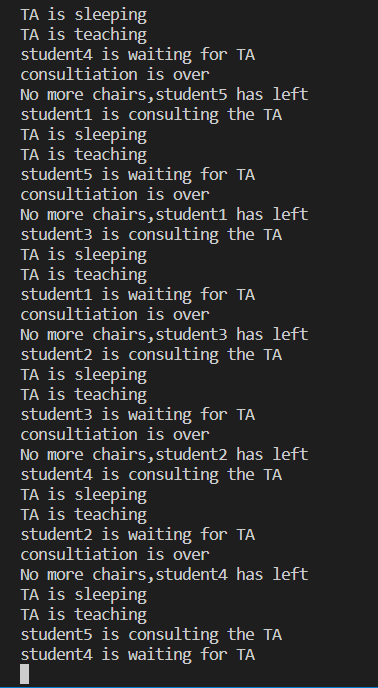
（2）pthread\_join函数。该函数的原型为：

int pthread\_join(pthread\_t thread,void \*\*retval);

其中的参数thread代表着等待线程的线程ID，retval代表该线程的退出状态，在本次实验中退出状态设置为NULL，该函数的功能为阻塞等待线程退出并可以获得线程的退出状态。

五、实验结果

将该程序运行之后的结果如下图所示。能够正确得显示各个线程之间的运行。

六、实验中出现的bug及其修改方式

1.实验结果产生速度过快不便于观察。解决方法：在如下所示的多个位置加入sleep函数使得函数的输出速度降低方便观察，同时也能满足实际情况。

