实验二 同步互斥

2.1 实验目的

通过编程实现进程间同步互斥的过程，学习实现进程同步互斥的方法，体会进程管理的思想。

2.2 实验内容

使用条件变量和互斥锁，实现线程的同步互斥。

本组选择的同步互斥问题为***老和尚喝水-小和尚挑水问题***，问题描述如下：

- 某寺庙，小、老和尚若干；
- 水缸，由小和尚提水入缸(向缸中倒水)，老和尚从缸中提水饮用，水缸可容10 桶水；
- 水取自同一井中，水井径窄，每次只能容一个桶取水；
- 水桶总数为3个，每次向缸中倒水、或从缸中取水仅为1 桶，且不可同时进行。

2.3 实验设计原理

- **两类进程**
  - 小和尚进程

    在水缸有空余空间时申请桶资源，去水井中打水后倒入水缸中。

  - 老和尚进程

    在水缸中有水时预约并拿水桶取水饮用。

- **信号量设置**
  - 同步

```
//互斥锁
mutex empty_mutex, full_mutex, bucket_count_mtx; // 水缸空和满互斥量 水桶计数互斥量
//条件变量
condition_variable full_cv, room_cv; // 条件变量: 水缸中的水 水缸中的空间
condition_variable bucket_cv; // 条件变量 水桶
//整型变量
int empty_room = TANK_VOLUME; // 水缸剩余可装水数量(桶)
int full = 0; // 水缸中剩余水数量(桶)
int bucket_left = INIT_BUCKETS_NUM; // 可用水桶数量
```

- 互斥

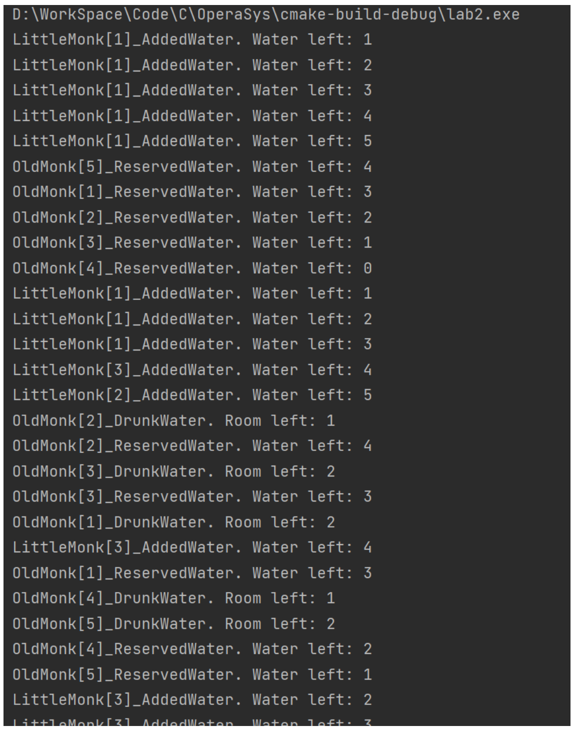
```
//互斥量
mutex well_mutex, tank_mutex;
//水井互斥量: 用于保证每次只有一个小和尚使用水井
//水缸互斥量: 用于保证每次只有一个人使用水缸
```

2.4 实验结果及分析

运行结果:

```
//Part of the output:
LittleMonk[1]_AddedWater. Water left: 1
LittleMonk[1]_AddedWater. Water left: 2
LittleMonk[1]_AddedWater. Water left: 3
LittleMonk[1]_AddedWater. Water left: 4
LittleMonk[1]_AddedWater. Water left: 5
OldMonk[5]_ReservedWater. Water left: 4
OldMonk[1]_ReservedWater. Water left: 3
OldMonk[2]_ReservedWater. Water left: 2
OldMonk[3]_ReservedWater. Water left: 1
OldMonk[4]_ReservedWater. Water left: 0
LittleMonk[1]_AddedWater. Water left: 1
LittleMonk[1]_AddedWater. Water left: 2
LittleMonk[1]_AddedWater. Water left: 3
LittleMonk[3]_AddedWater. Water left: 4
LittleMonk[2]_AddedWater. Water left: 5
OldMonk[2]_DrunkWater. Room left: 1
OldMonk[2]_ReservedWater. Water left: 4
OldMonk[3]_DrunkWater. Room left: 2
OldMonk[3]_ReservedWater. Water left: 3
OldMonk[1]_DrunkWater. Room left: 2
LittleMonk[3]_AddedWater. Water left: 4
OldMonk[1]_ReservedWater. Water left: 3
OldMonk[4]_DrunkWater. Room left: 1
OldMonk[5]_DrunkWater. Room left: 2
OldMonk[4]_ReservedWater. Water left: 2
OldMonk[5]_ReservedWater. Water left: 1
LittleMonk[3]_AddedWater. Water left: 2
LittleMonk[3]_AddedWater. Water left: 3
```

运行截图：



```
D:\WorkSpace\Code\C\OperaSys\cmake-build-debug\lab2.exe
LittleMonk[1]_AddedWater. Water left: 1
LittleMonk[1]_AddedWater. Water left: 2
LittleMonk[1]_AddedWater. Water left: 3
LittleMonk[1]_AddedWater. Water left: 4
LittleMonk[1]_AddedWater. Water left: 5
OldMonk[5]_ReservedWater. Water left: 4
OldMonk[1]_ReservedWater. Water left: 3
OldMonk[2]_ReservedWater. Water left: 2
OldMonk[3]_ReservedWater. Water left: 1
OldMonk[4]_ReservedWater. Water left: 0
LittleMonk[1]_AddedWater. Water left: 1
LittleMonk[1]_AddedWater. Water left: 2
LittleMonk[1]_AddedWater. Water left: 3
LittleMonk[3]_AddedWater. Water left: 4
LittleMonk[2]_AddedWater. Water left: 5
OldMonk[2]_DrunkWater. Room left: 1
OldMonk[2]_ReservedWater. Water left: 4
OldMonk[3]_DrunkWater. Room left: 2
OldMonk[3]_ReservedWater. Water left: 3
OldMonk[1]_DrunkWater. Room left: 2
LittleMonk[3]_AddedWater. Water left: 4
OldMonk[1]_ReservedWater. Water left: 3
OldMonk[4]_DrunkWater. Room left: 1
OldMonk[5]_DrunkWater. Room left: 2
OldMonk[4]_ReservedWater. Water left: 2
OldMonk[5]_ReservedWater. Water left: 1
LittleMonk[3]_AddedWater. Water left: 2
LittleMonk[3]_AddedWater. Water left: 3
```

结果说明：

设置小和尚进程个数为3，老和尚进程个数为5，开始取水饮水的过程。

在输出中，小和尚进程的行为只有一个，就是添加[**Add**]水，在添加成功后会输出当前剩余水的数量（桶）；老和尚进程有两个动作，首先预约[**Reserve**]水桶中的水（为了避免死锁），此时水的数量减一，随后取桶饮水[**Drink**]，饮完水后水缸剩余空间加一。

2.5 实验总结与收获

本组实验二的结果成功的体现了进程同步互斥的过程，总体来说比较成功。

在实验二中，我组使用C++的条件变量加互斥量这个组合实现了进程间的同步互斥，对于条件变量和信号量的区别有了新的认识：信号量和条件变量其实有功能重合的地方，就是在进程间同步时对于进程的阻塞和唤醒功能，两者都能实现。

但是两者也有不同：首先，信号量同时还可以完成互斥的功能，条件变量则需要与互斥锁同时使用来保证同步时的互斥访问；另外，信号量表征了某种资源，阻塞和唤醒都根据一个整数量来进行，但是条件变量在唤醒条件上可以由编程者自己定义，更加灵活。

2.6 附件：程序源代码

```
/**
 * Author: LanSnowZ
 * Date: 2022.2.25
 */
#include "iostream"
#include "thread"
#include "mutex"
#include "condition_variable"
```

```
#define TANK_VOLUME 10 // 水缸容量
#define INIT_BUCKETS_NUM 3 // 桶的数量
#define LITTLE_NUM 3
#define OLD_NUM 5

using namespace std;

//同步量
mutex empty_mutex, full_mutex, bucket_count_mtx; // 水缸互斥量和水桶计数互斥量
condition_variable full_cv, room_cv; // 两个条件变量, 水缸中的水 水缸中的空间
int empty_room = TANK_VOLUME; // 水缸剩余可装水数量(桶)
int full = 0; // 水缸中剩余水数量(桶)
//互斥量
mutex well_mutex, tank_mutex; //水桶互斥量 水井互斥量
condition_variable bucket_cv; // 条件变量 水桶
int bucket_left = INIT_BUCKETS_NUM; // 可用水桶数量

void LittleMonk(int id){
    while(1){
        //检查水缸是否有空间
        unique_lock<mutex> empty_lock(empty_mutex);
        while (empty_room == 0)
            room_cv.wait(empty_lock);
        empty_room--;
        empty_lock.unlock();

        //检查并获取桶
        unique_lock<mutex> bucket_count_lock(bucket_count_mtx);
        while (bucket_left == 0)
            bucket_cv.wait(bucket_count_lock);
        bucket_left--;
        bucket_count_lock.unlock();

        //从水井中打水
        unique_lock<mutex> well_lock(well_mutex);
//          cout << "LittleMonk[" << id << "]_GotWater. Time:" << " ." << endl;
        well_lock.unlock();

        //将打来的水导入水缸中并通知
        unique_lock<mutex> tank_lock(tank_mutex);
        unique_lock<mutex> full_lock(full_mutex);
        full++;
        cout << "LittleMonk[" << id << "]_AddedWater. Water left: " << full <<
endl;
        full_lock.unlock();
        tank_lock.unlock();
        full_cv.notify_one();

        //归还桶并通知等待桶的进程
        bucket_count_lock.lock();
        bucket_left++;
        bucket_count_lock.unlock();
        bucket_cv.notify_one();
    }
}

void OldMonk(int id){
    while(1){
        //检查水缸是否有水
        unique_lock<mutex> full_lock(full_mutex);
        while (full == 0)
            full_cv.wait(full_lock);
        full--;
        cout << "OldMonk[" << id << "]_ReservedWater. Water left: " << full <<
endl;
        full_lock.unlock();

        //检查并获取桶
        unique_lock<mutex> bucket_count_lock(bucket_count_mtx);
        while (bucket_left == 0)
            bucket_cv.wait(bucket_count_lock);
        bucket_left--;
        bucket_count_lock.unlock();

        //喝水
        unique_lock<mutex> tank_lock(tank_mutex);
        unique_lock<mutex> empty_lock(empty_mutex);
        empty_room++;
        cout << "OldMonk[" << id << "]_DrunkWater. Room left: " << empty_room <<
endl;
        empty_lock.unlock();
        tank_lock.unlock();
        room_cv.notify_one();

        //归还桶并通知等待桶的进程
        bucket_count_lock.lock();
        bucket_left++;
        bucket_count_lock.unlock();
        bucket_cv.notify_one();
    }
}

int main(){
    thread little[LITTLE_NUM], old[OLD_NUM];
    for (int i = 0; i < LITTLE_NUM; i++)
        little[i] = thread(LittleMonk, i+1);
    for (int i = 0; i < OLD_NUM; i++)
        old[i] = thread(OldMonk, i+1);

    for (int i = 0; i < LITTLE_NUM; i++)
        little[i].join();
    for (int i = 0; i < OLD_NUM; i++)
        old[i].join();
    return 0;
}
```