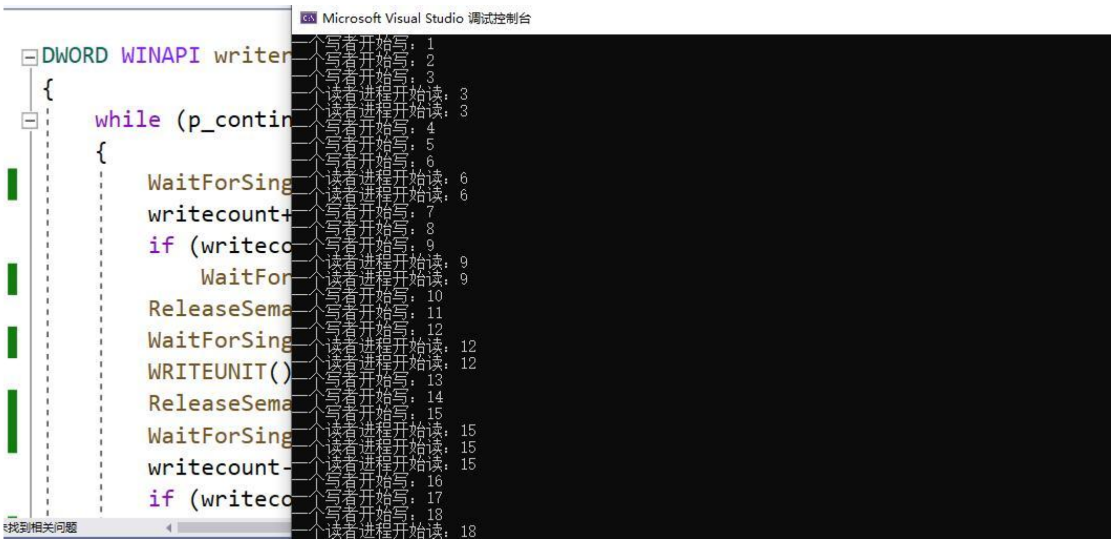
读者与写着实验代码：

```
#include<windows.h>
#include <iostream>
#include<stdio.h>

#define MAX_READER_NUM 512
#define READER_NUM 2
#define WRITER_NUM 3
#define MOD 100

using namespace std;

int readcount = 0;
int writecount = 0;
HANDLE a;////不允许在 b 上建造长队列，否则写进程将无法跳过这个队列，因此只允许一个读进程在 b 上排队
HANDLE b;//写着优先读者的第 2 个互斥量，同时也是写者让读者阻塞的信号量
HANDLE c;//保证 readcount 正常更新
HANDLE y;//保证 writecount 正常更新
HANDLE middle;//实现读者写者互斥的信号量
HANDLE input;//这是将输出作为临界资源，这样有利于防止两个进程同时输出，造成本应该换行的输出在一行输出，此外这是读者的输出，命名不规范
HANDLE output;//同 Input，这是写者的输出
DWORD WINAPI reader(LPVOID);//调用 windowsAPI 函数
DWORD WINAPI writer(LPVOID);
bool p_continue = true;
int test = 0;
int process_write_num = 0;//统计每个进程执行的次数
int process_read_num = 0;

int main()
{
	a = CreateSemaphore(NULL, 1, MAX_READER_NUM, NULL);
	b = CreateSemaphore(NULL, 1, MAX_READER_NUM, NULL);
	c = CreateSemaphore(NULL, 1, MAX_READER_NUM, NULL);
	input = CreateSemaphore(NULL, 1, MAX_READER_NUM, NULL);
	output = CreateSemaphore(NULL, 1, MAX_READER_NUM, NULL);
	middle = CreateSemaphore(NULL, 1, MAX_READER_NUM, NULL);
	y = CreateSemaphore(NULL, 1, MAX_READER_NUM, NULL);
	HANDLE hThreads[READER_NUM + WRITER_NUM];
	DWORD readerID[READER_NUM];
	DWORD writerID[WRITER_NUM];
	for (int i = 0; i < WRITER_NUM; i++)
	{
        hThreads[i] = CreateThread(NULL, 0, writer, NULL, 0, &writerID[i]);//创建写者进程
        if (hThreads[i] == NULL)
            return -1;
    }
    for (int i = 0; i < READER_NUM; i++)
    {
        hThreads[i] = CreateThread(NULL, 0, reader, NULL, 0, &readerID[i]);//创建读者线程
        if (hThreads[i] == NULL)
            return -1;
    }
    while (p_continue)//当输入空格的时候结束程序
    {
        if (getchar())
        {
            p_continue = false;
            printf("在这个过程中读者进程的数量：%d\n", process_read_num);
            printf("在这个过程中写者进程的数量：%d\n", process_write_num);
        }
    }
    return 0;
}

void READUNIT()
{
    WaitForSingleObject(input, INFINITE);//p 操作
    process_read_num++;
    cout << "一个读者进程开始读：";
    cout << test << endl;
    ReleaseSemaphore(input, 1, NULL);//v 操作
}

void WRITEUNIT()
{
    process_write_num++;
    cout << "一个写者开始写：";
    test = (test + 1) % MOD;
    cout << test << endl;
    ReleaseSemaphore(output, 1, NULL);
}

DWORD WINAPI writer(LPVOID lpPara)
{
    while (p_continue)
    {
        WaitForSingleObject(y, INFINITE); //保证 writecount 正常更新
        writecount++;
        if (writecount == 1)
            WaitForSingleObject(b, INFINITE); //当有一个写者时，阻塞读进程
        ReleaseSemaphore(y, 1, NULL);
        WaitForSingleObject(middle, INFINITE); //阻塞其他写者，只允许同一时刻一个写者
访问
        WRITEUNIT();
        ReleaseSemaphore(middle, 1, NULL); //释放
        WaitForSingleObject(y, INFINITE); //准备操作 writecount
        writecount--;
        if (writecount == 0)
            ReleaseSemaphore(b, 1, NULL); //此时没有写进程，读进程可读
        ReleaseSemaphore(y, 1, NULL); //writecount 正常更新
        Sleep(1000);
    }
    return 0;
}

DWORD WINAPI reader(LPVOID lpParar) //为了允许多个读进程，当至少已有一个读进程在
读时，随后的读进程无需等待，可以直接进入
{
    while (p_continue)
    {
        WaitForSingleObject(a, INFINITE);   //不允许在 b 上建造长队列，否则写进程将无
法跳过这个队列，因此只允许一个读进程在 b 上排队
        WaitForSingleObject(b, INFINITE);   //检查是否有写进程在运行
        WaitForSingleObject(c, INFINITE);    //准备操作 readcount
        readcount++;
        if (readcount == 1)
            WaitForSingleObject(middle, INFINITE); //当有读者在读时，阻塞写进程
        ReleaseSemaphore(a, 1, NULL);
        ReleaseSemaphore(b, 1, NULL);
        ReleaseSemaphore(c, 1, NULL);
        READUNIT();
        WaitForSingleObject(c, INFINITE); //准备对 readcount 操作
        readcount--;
        if (readcount == 0)
            ReleaseSemaphore(middle, 1, NULL); //写者可写
        ReleaseSemaphore(c, 1, NULL);
        Sleep(1000);
    }
    return 0;
}
```

实验结果：



```
DWORD WINAPI writer
{
    while (p_contin
    {
        WaitForSing
        writecount+
        if (writeco
            WaitFor
        ReleaseSema
        WaitForSing
        WRITEUNIT()
        ReleaseSema
        WaitForSing
        writecount-
        if (writeco
```

找到相关问题

Microsoft Visual Studio 调试控制台

```
一个写者开始写: 1
一个写者开始写: 2
一个写者开始写: 3
一个读者进程开始读: 3
一个读者进程开始读: 3
一个写者开始写: 4
一个写者开始写: 5
一个写者开始写: 6
一个读者进程开始读: 6
一个读者进程开始读: 6
一个写者开始写: 7
一个写者开始写: 8
一个写者开始写: 9
一个读者进程开始读: 9
一个读者进程开始读: 9
一个写者开始写: 10
一个写者开始写: 11
一个写者开始写: 12
一个读者进程开始读: 12
一个读者进程开始读: 12
一个写者开始写: 13
一个写者开始写: 14
一个写者开始写: 15
一个读者进程开始读: 15
一个读者进程开始读: 15
一个读者进程开始读: 15
一个写者开始写: 16
一个写者开始写: 17
一个写者开始写: 18
一个读者进程开始读: 18
```