

# 线程篇 (下)



日成蝶—Windows API 编程入门

七日做茧，一朝成蝶！



主讲：袁春旭

个人博客：<http://8413723.blog.51cto.com/>

**课程主页：**<http://edu.51cto.com/lecturer/8403723.html>

# 线程同步

# 线程同步

## 线程同步解决什么问题？

**问题一：不同线程函数的执行必须有先后顺序。**

**问题二：同一线程函数中一段代码必须作为一个单元执行。**

# 线程同步-案例一

**例：**一个线程给两个全局变量赋值，另外一个线程则负责计算赋值后变量的和，结果要在主函数中打印出来。

```
//定义全局变量a、b和c，并赋初值
int a = 10;
int b = 20;
int c = 0;
//下面这个函数用于给全局变量a和b赋值
DWORD WINAPI GetNum(LPVOID lpParameter)
{
    a = 100;
    b = 200;
    return 0;
}
```

# 线程同步-案例一

//下面这个函数用于计算全局变量a和b的和，并把结果存放在全局变量c中

```
DWORD WINAPI GetResult(LPVOID lpParameter)
```

```
{
```

```
    c = a + b;
```

```
    return 0;
```

```
}
```

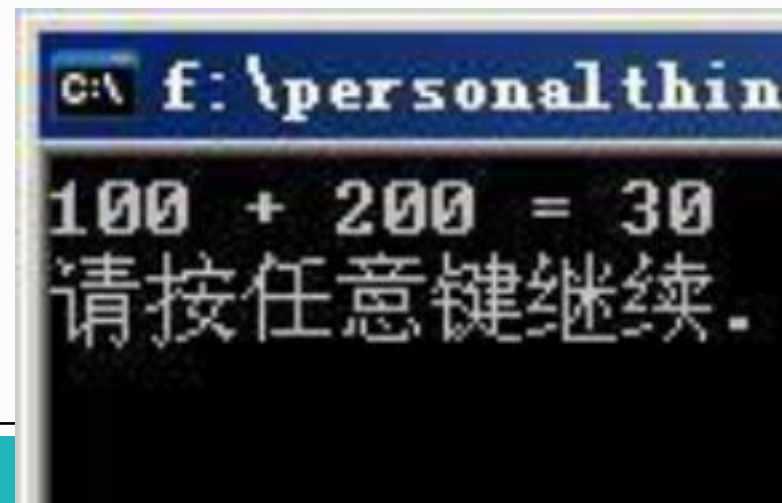
//定义两个全局的句柄，当然也可以定义成数组

```
HANDLE hThread1;
```

```
HANDLE hThread2;
```

# 线程同步-案例一

```
int main(void)
{
    hThread1 = CreateThread(NULL, NULL, GetResult, NULL, 0, NULL);
    hThread2 = CreateThread(NULL, NULL, GetNumber, NULL, 0, NULL);
    //等待这两个线程执行完毕
    WaitForSingleObject(hThread1, INFINITE);
    WaitForSingleObject(hThread2, INFINITE);
    //打印出最终结果
    printf("%d + %d = %d\n", a, b, c);
    CloseHandle(hThread1);
    CloseHandle(hThread2);
    return 0;
}
```



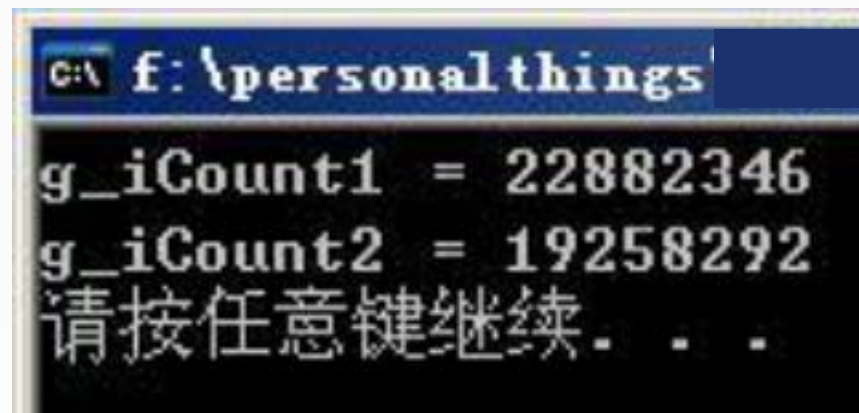
# 线程同步-案例二

```
//定义两个全局变量，用来计数
int g_iCount1 = 0;
int g_iCount2 = 0;
//定义一个全局变量，用来控制while循环
int g_iFlag = 1;
//下面函数不断让变量g_iCount1和g_iCount2自加1
DWORD WINAPI ThreadProc(LPVOID lpParameter) {
    while(g_iFlag) {
        g_iCount1++; ==> g_iCount1 = g_iCount1 + 1;
        g_iCount2++; ==> g_iCount2 = g_iCount2 + 1;
    }
    return 0;
}
```



## 线程同步-案例二

```
int main(void)
{
    HANDLE hThread[2];
    hThread[0] = CreateThread(NULL, NULL, ThreadProc, NULL, 0, NULL);
    hThread[1] = CreateThread(NULL, NULL, ThreadProc, NULL, 0, NULL);
    Sleep(100);
    //修改全局标志位使两个线程停止运行
    g_iFlag = 0;
    printf("g_iCount1 = %d\n", g_iCount1);
    printf("g_iCount2 = %d\n", g_iCount2);
    CloseHandle(hThread[0]);
    CloseHandle(hThread[1]);
    return 0;
}
```



```
c:\ f: \personalthings
g_iCount1 = 22882346
g_iCount2 = 19258292
请按任意键继续. . .
```

# 线程同步-案例二

线程一

```
g_iCount1 = g_iCount1 + 1;
```

读取g\_iCount1到临时位置A

1

A+1赋值给A

3

将A的值赋值给g\_iCount1

4

线程二

```
g_iCount1 = g_iCount1 + 1;
```

读取g\_iCount1到临时位置A

2

A+1赋值给A

5

将A的值赋值给g\_iCount1

6

# 等待函数

# 等待函数--WaitForSingleObject

```
DWORD WaitForSingleObject(  
    HANDLE hHandle,      //对象句柄  
    DWORD dwMilliseconds //等待时间，INFINITE一直等待  
);
```

**hHandle** : Event、Job、Memory resource notification、Mutex、Process、Semaphore、Thread、Waitable timer等

**WAIT\_ABANDONED 0x00000080** : 当hHandle为mutex时，如果拥有mutex的线程在结束时没有释放内核对象会返回此返回值。

**WAIT\_OBJECT\_0 0x00000000** : 指定的对象发出有信号状态

**WAIT\_TIMEOUT 0x00000102** : 等待超时

**WAIT\_FAILED 0xFFFFFFFF** : 出现错误，可通过GetLastError得到错误代码

# 等待函数--WaitForSingleObject

解决方案：在GetResult函数里面等待GetNum线程

```
//定义全局变量a、b和c，并赋初值
int a = 10;
int b = 20;
int c = 0;

//下面这个函数用于给全局变量a和b赋值
DWORD WINAPI GetNum(LPVOID lpParameter)
{
    a = 100;
    b = 200;
    return 0;
}
```

# 等待函数--WaitForSingleObject

```
//定义两个全局的句柄，也可以定义成数组  
HANDLE hThread1;  
HANDLE hThread2;  
  
//下面这个函数用于计算全局变量a和b的和，并把结果存放在全局变量c中  
DWORD WINAPI GetResult(LPVOID lpParameter)  
{  
    WaitForSingleObject(hThread2, INFINITE);  
    c = a + b;  
    return 0;  
}
```

# 等待函数--WaitForSingleObject

```
int main(void)
{
    hThread1 = CreateThread(NULL, NULL, GetResult, NULL, 0, NULL);
    hThread2 = CreateThread(NULL, NULL, GetNumber, NULL, 0, NULL);
    //等待这两个线程执行完毕
    WaitForSingleObject(hThread1, INFINITE);
    WaitForSingleObject(hThread2, INFINITE);
    //打印出最终结果
    printf("%d + %d = %d\n", a, b, c);
    CloseHandle(hThread1);
    CloseHandle(hThread2);
    return 0;
}
```



# 等待函数--WaitForMultipleObjects

```
DWORD WaitForMultipleObjects(  
    //等待线程数量(最多MAXIMUM_WAIT_OBJECTS个, windows定义为64)  
    DWORD dwCount,  
    CONST HANDLE* phObjects,    //线程句柄数组指针  
    BOOL fWaitAll,              //是否等待全部  
    DWORD dwMilliseconds);      //等待时间INFINITE
```

当参数fWaitAll为TRUE时, 若所有线程均变为已通知状态则函数返回值为WAIT\_OBJECT\_0。

当参数fWaitAll为FALSE时, 返回的值为线程内核对象数组的索引值。



# 等待函数--WaitForMultipleObjects

```
//定义两个全局的句柄，当然也可以定义成数组
HANDLE hThread[2];
int main(void)
{
    //创建两个线程
    hThread[0] = CreateThread(NULL, NULL, GetResult, NULL, 0, NULL);
    hThread[1] = CreateThread(NULL, NULL, GetNum, NULL, 0, NULL);
    //等待这两个线程执行完毕
    WaitForMultipleObjects(2, hThread, TRUE, INFINITE);
    //关闭句柄
    CloseHandle(hThread1);
    CloseHandle(hThread2);
    return 0;
}
```

# 编码实战



# Thank You !