线程篇（下）

7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

互斥量mutex/互斥体/互斥锁

互斥量mutex

创建互斥量：CreateMutex

创建互斥量：CreateMutexEx

打开互斥量：OpenMutex

释放互斥量：ReleaseMutex

互斥量mutex

创建互斥量：

```
HANDLE   CreateMutex(
            LPSECURITY_ATTRIBUTES  lpMutexAttributes,
            BOOL                   bInitialOwner,
            LPCTSTR                lpName
);
```

lpTimerAttributes: 安全属性设置

bInitialOwner: 设置互斥量初始归属，TRUE：属于当前线程；FALSE：无归属

lpName: 互斥量名称

互斥量mutex

创建互斥量：

```
HANDLE  CreateMutexEx(
            LPSECURITY_ATTRIBUTES lpMutexAttributes,
            LPCTSTR               lpName,
            DWORD                 dwFlags,
            DWORD                 dwDesiredAccess
);
```

dwFlags: CREATE_MUTEX_INITIAL_OWNER(0x00000001) 当前线程为所有者

0x00000000 当前无所有者

互斥量mutex

dwDesiredAccess：

值	说明
DELETE (0x00010000L)	删除对象
READ_CONTROL (0x00020000L)	读取对象信息安全符信息
SYNCHRONIZE (0x00100000L)	对象同步
WRITE_DAC (0x00040000L)	修改DACL中的对象的安全描述符。
WRITE_OWNER (0x00080000L)	改变对象拥有者的安全描述符
MUTEX_ALL_ACCESS (0x1F0001)	互斥量对象所有权限
MUTEX_MODIFY_STATE (0x0001)	修改互斥量状态

互斥量mutex

dwDesiredAccess：

值	说明
EVENT_ALL_ACCESS (0x1F0003)	事件对象所有权限
EVENT_MODIFY_STATE (0x0002)	修改事件对象状态
TIMER_ALL_ACCESS (0x1F0003)	计时器对象所有权限
TIMER_QUERY_STATE (0x0001)	修改计时器状态
SEMAPHORE_ALL_ACCESS (0x1F0003)	信号量对象所有权限
SEMAPHORE_MODIFY_STATE (0x0002)	修改信号量状态

互斥量mutex

打开互斥量：

```
HANDLE   OpenMutex(
            DWORD       dwDesiredAccess,
            BOOL        bInheritHandle,
            LPCTSTR     lpName
);
```

互斥量mutex

释放互斥量：

```
BOOL  ReleaseMutex(
            HANDLE hMutex
);
```

```
HANDLE g_hMutex;
int main(void)
{
        g_hMutex = CreateMutex(NULL, FALSE, NULL);
        HANDLE hThread;
        hThread = (HANDLE)_beginthreadex(NULL, 0,
                (_beginthreadex_proc_type)BaoShu, NULL, 0, NULL);
        WaitForSingleObject(hThread, INFINITE);
        CloseHandle(g_hMutex);
        CloseHandle(hThread);
        return 0;
}
```

```
HANDLE g_hMutex;
int main(void)
{
        g_hMutex = CreateMutex(NULL, TRUE, NULL);
        HANDLE hThread;
        hThread = (HANDLE)_beginthreadex(NULL, 0,
                (_beginthreadex_proc_type)BaoShu, NULL, 0, NULL);
        getchar();
        ReleaseMutex(g_hMutex);
        WaitForSingleObject(hThread, INFINITE);
        CloseHandle(g_hMutex);
        CloseHandle(hThread);
        return 0;
}
```

```
DWORD WINAPI BaoShu(LPVOID *lparam)
{
        WaitForSingleObject(g_hMutex, INFINITE);
        for (int i = 0; i < 10; i++)
        {
                printf("1\n");
        }
        ReleaseMutex(g_hMutex);
        return 0;
}
```

原理说明

原理说明

互斥量内核对象

引用计数	0表示已触发
当前线程ID	0表示无占用

OpenMutex使引用计数加1

ReleaseMutex使引用计数减1

原理说明

互斥量内核对象

引用计数	0表示已触发
当前线程ID	0表示无占用

线程ID23

引用计数	1
当前线程ID	23

思考：占用互斥量的线程意外死亡？

编码实战

Thank You !