线程篇（下）

七日成蝶——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

关键段-临界区

关键段-临界区

定义关键段：CRITICAL_SECTION cs;

初始化关键段：InitializeCriticalSection(&cs);

进入关键段：EnterCriticalSection(&cs);

离开关键段：LeaveCriticalSection(&cs);

销毁关键段：DeleteCriticalSection(&cs);

关键段-临界区

初始化关键段：

```
void InitializeCriticalSection(
        LPCRITICAL_SECTION   lpCriticalSection
);
```

销毁关键段：

```
void DeleteCriticalSection(
        LPCRITICAL_SECTION lpCriticalSection
);
```

关键段-临界区

进入关键段：

```
void EnterCriticalSection(
        LPCRITICAL_SECTION lpCriticalSection
);
```

离开关键段：

```
void LeaveCriticalSection(
        LPCRITICAL_SECTION lpCriticalSection
);
```

关键段-临界区

```
CRITICAL_SECTION g_cs;    //定义
int main(void)
{
    InitializeCriticalSection(&g_cs);        //初始化
    HANDLE hThread[2];
    hThread[0] = (HANDLE)_beginthreadex(NULL, 0, (_beginthreadex_proc_type)ThreadProc, NULL, 0, NULL);
    hThread[1] = (HANDLE)_beginthreadex(NULL, 0, (_beginthreadex_proc_type)ThreadProc, NULL, 0, NULL);
    Sleep(1000);
    g_bFlag = FALSE;
    WaitForMultipleObjects(2, hThread, TRUE, INFINITE);
    DeleteCriticalSection(&g_cs);
    CloseHandle(hThread[0]);
    CloseHandle(hThread[1]);
    printf("\ng_iCount1 = %d\n", g_iCount1);
    printf("\ng_iCount2 = %d\n", g_iCount2);
    return 0;
}
```

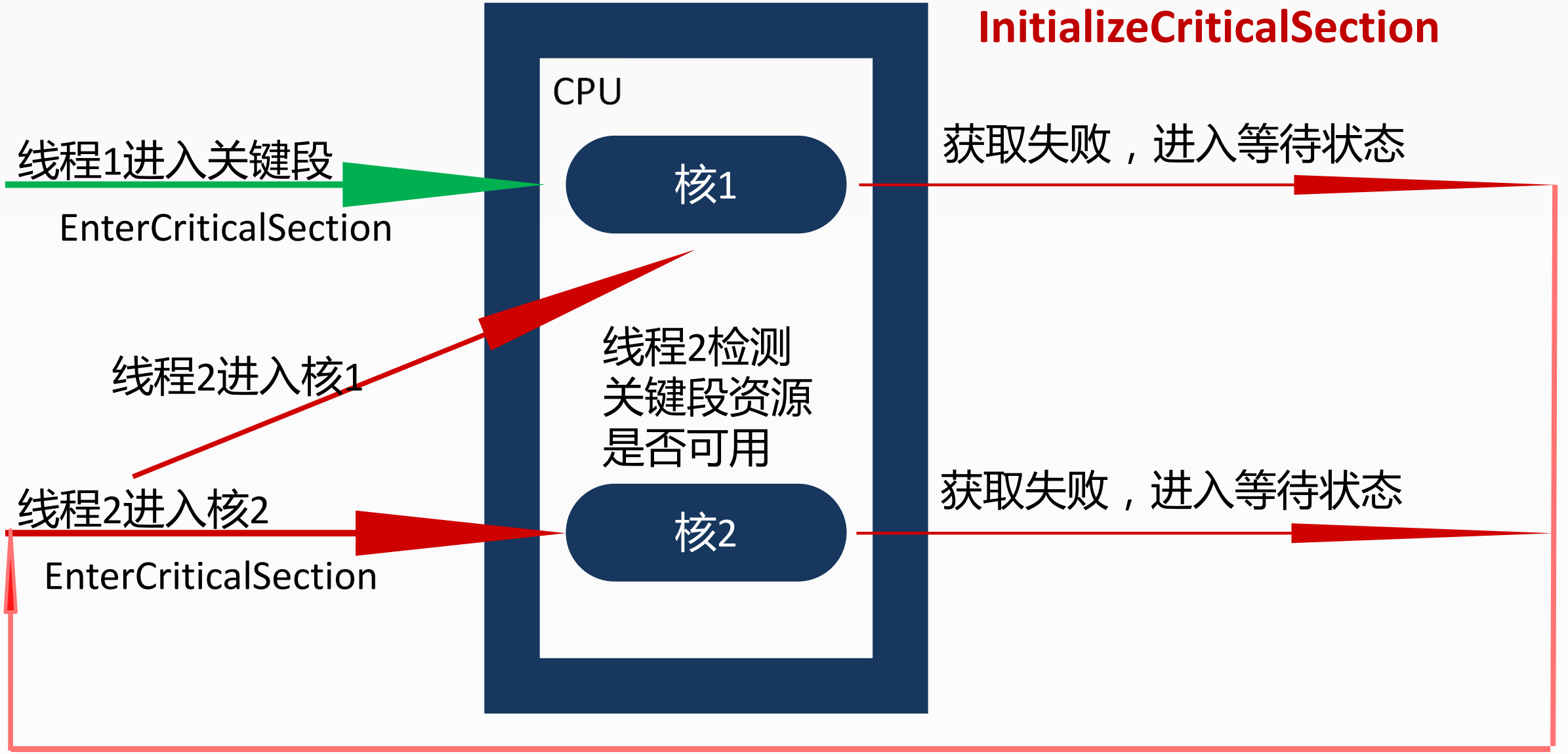
关键段-临界区

```
DWORD WINAPI ThreadProc(LPVOID *lparam)
{
        EnterCriticalSection(&g_cs);
        while (g_bFlag)
        {
                g_iCount1++;
                g_iCount2++;
        }
        LeaveCriticalSection(&g_cs);

        return 0;
}
```

关键段-临界区

```
DWORD WINAPI ThreadProc(LPVOID *lparam)
{
        while (g_bFlag)
        {
                EnterCriticalSection(&g_cs);
                g_iCount1++;
                g_iCount2++;
                LeaveCriticalSection(&g_cs);
        }

        return 0;
}
```

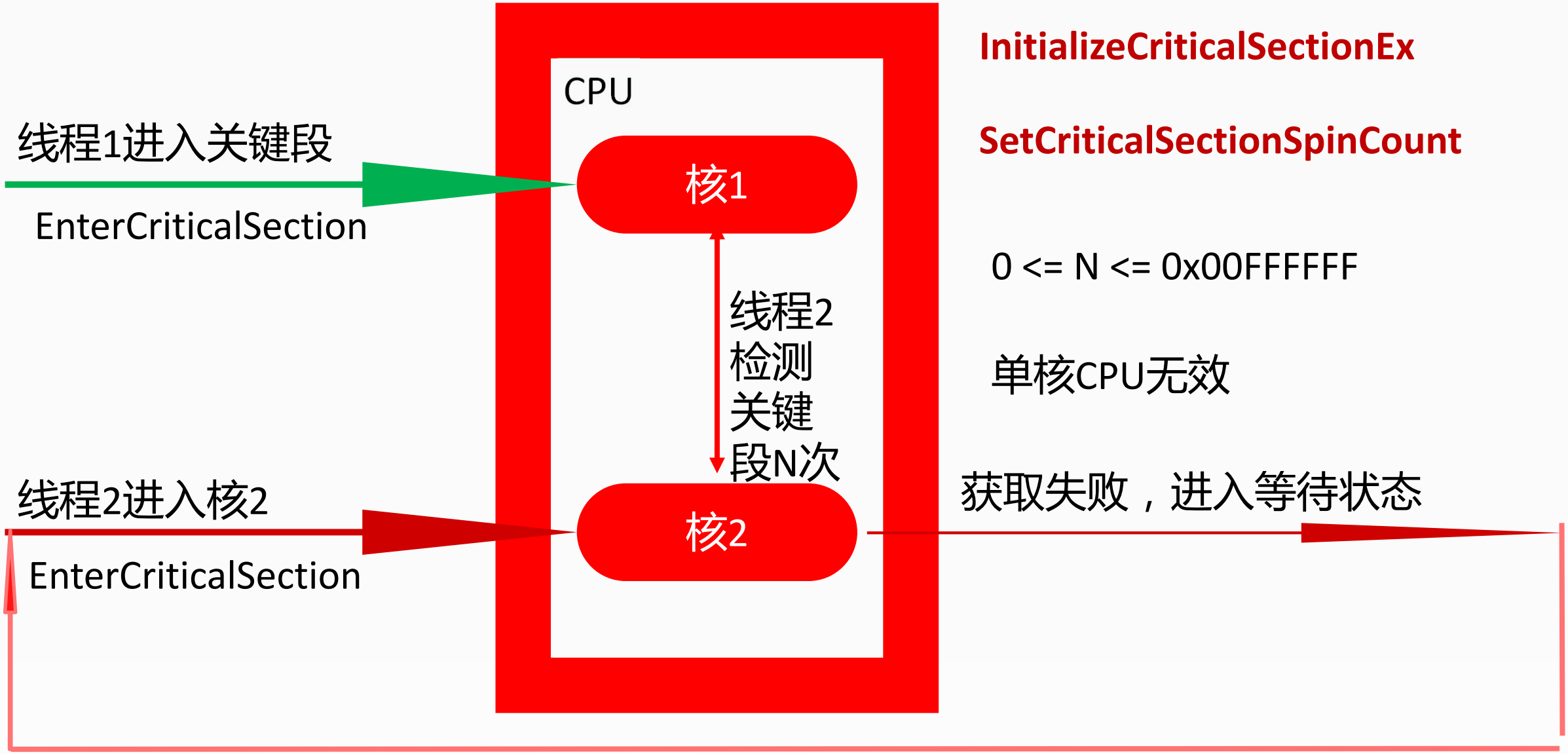
工作原理及优化


InitializeCriticalSection
CPU
线程1进入关键段
EnterCriticalSection
核1
获取失败，进入等待状态
线程2进入核1
线程2检测
关键段资源
是否可用
线程2进入核2
EnterCriticalSection
核2
获取失败，进入等待状态

InitializeCriticalSectionAndSpinCount

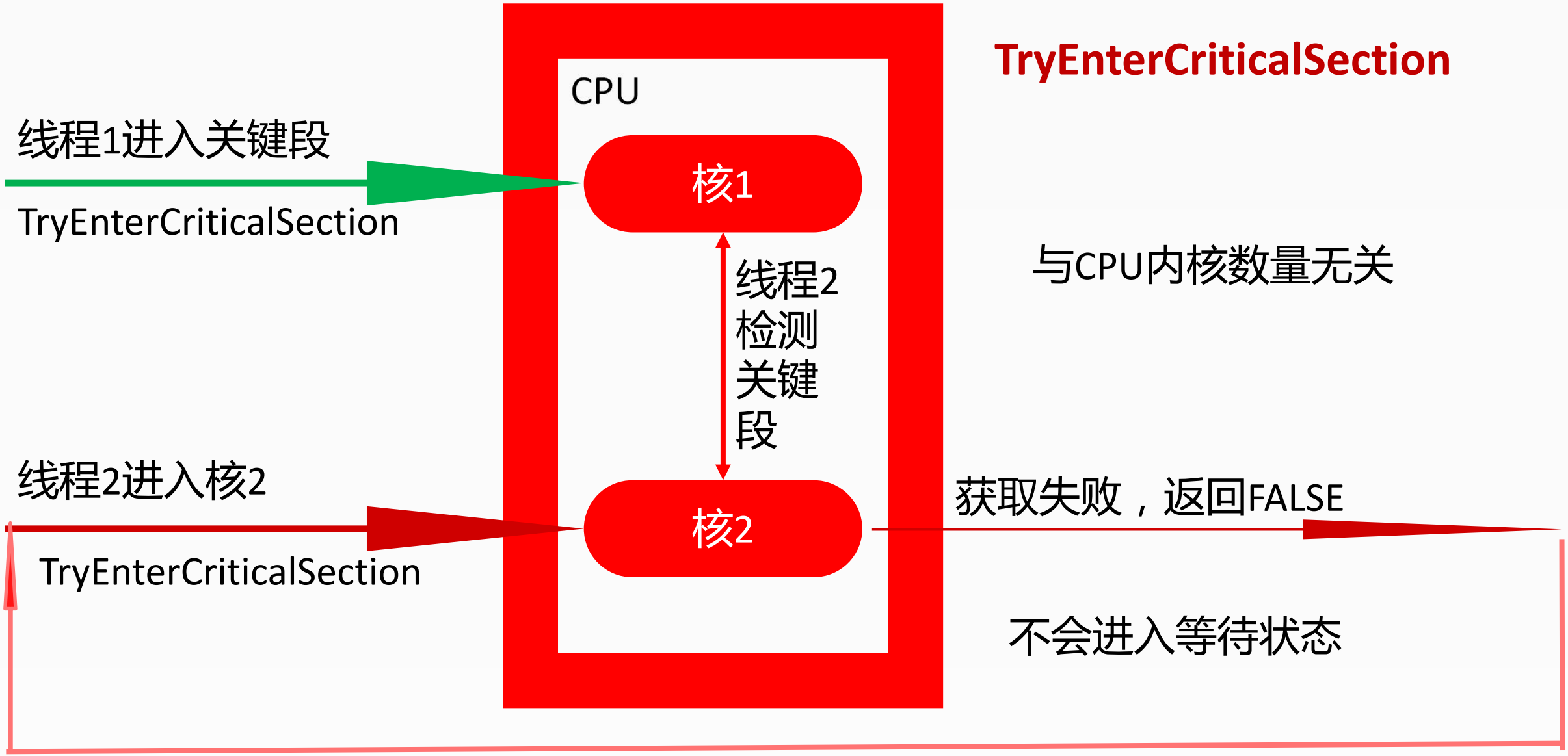
InitializeCriticalSectionEx

SetCriticalSectionSpinCount



0 <= N <= 0x00FFFFFF

单核CPU无效


TryEnterCriticalSection
CPU
线程1进入关键段
核1
TryEnterCriticalSection
线程2
检测
关键
段
与CPU内核数量无关
线程2进入核2
核2
获取失败，返回FALSE
TryEnterCriticalSection
不会进入等待状态

初始化关键段：

```
BOOL  InitializeCriticalSectionAndSpinCount(
        LPCRITICAL_SECTION lpCriticalSection,
        DWORD          dwSpinCount
);
```

初始化关键段：

```
BOOL WINAPI InitializeCriticalSectionEx(
        LPCRITICAL_SECTION lpCriticalSection,
        DWORD          dwSpinCount,
        DWORD          Flags
//CRITICAL_SECTION_NO_DEBUG_INFO
);
```

设置关键段检测次数：

```
DWORD  SetCriticalSectionSpinCount(
            LPCRITICAL_SECTION lpCriticalSection,
            DWORD              dwSpinCount
);   //返回值为设置前的检测次数
```

尝试进入关键段：

```
BOOL  TryEnterCriticalSection(
  _Inout_ LPCRITICAL_SECTION lpCriticalSection
);
```

关键段定义：CRITICAL_SECTION cs;

关键段初始化：InitializeCriticalSectionAndSpinCount(&cs, 1000);
InitializeCriticalSectionEx(&cs, 0x0000FFEE, 0);

进入关键段：TryEnterCriticalSection(&cs);

离开关键段：LeaveCriticalSection(&cs);

销毁关键段：DeleteCriticalSection(&g_cs);

可能的错误

可能的错误

关键段初始化：InitializeCriticalSection(&cs); 返回值void

内存不足时可能失败

线程首次争抢时产生事件内核对象可能失败

可能的错误

关键段初始化：

```
BOOL InitializeCriticalSectionAndSpinCount(
        LPCRITICAL_SECTION   lpCriticalSection,
        DWORD                dwSpinCount
);
```

dwSpinCount：有效范围 0x00000000~0x00FFFFFF

dwSpinCount：初始化即生成事件对象，最高位设置为1
0x80000000~0x80FFFFFF

编码实战

Thank You !