线程篇（上）

七日成蝶——Windows API 编程入门

七日做茧，一朝成蝶！

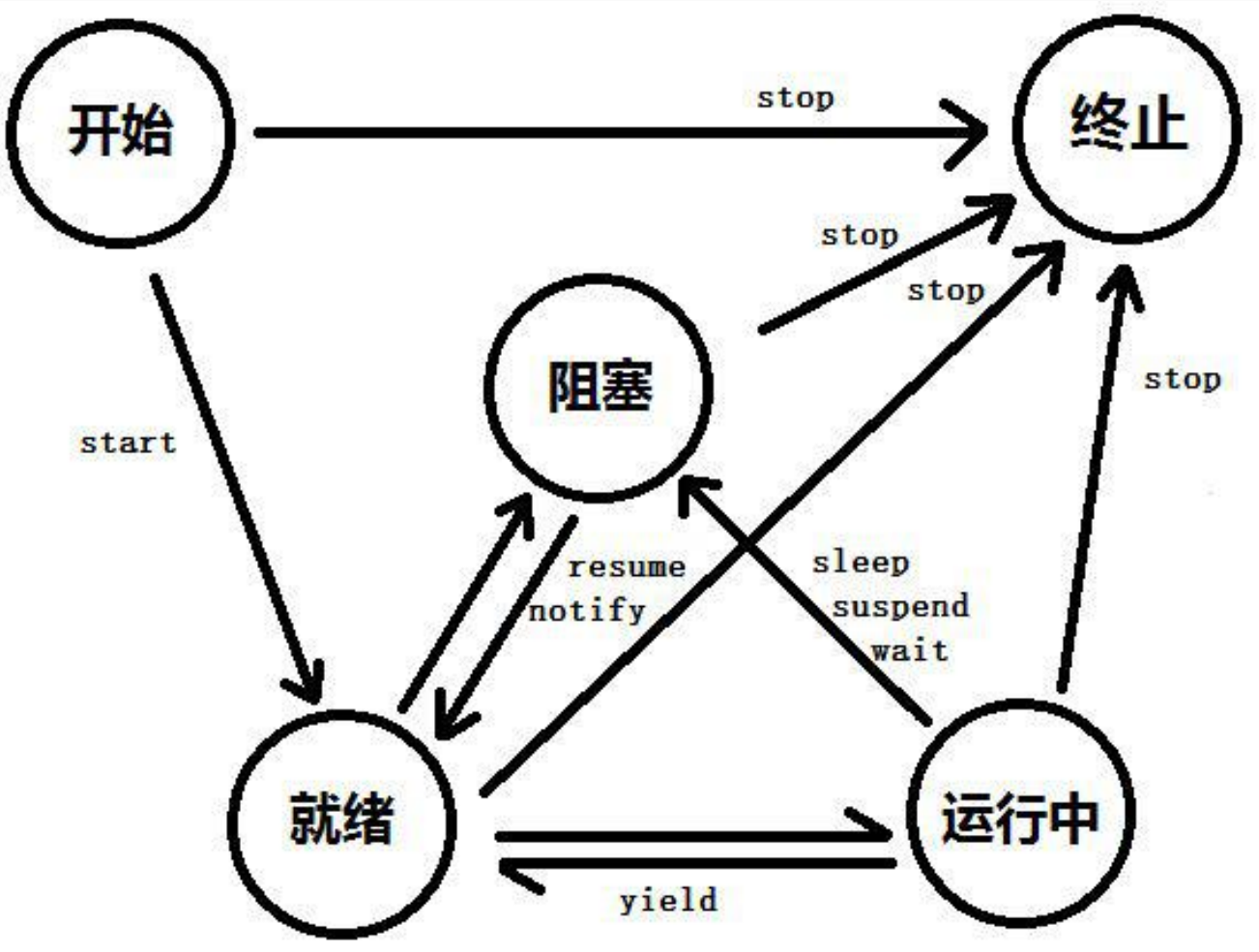
主讲：袁春旭

个人博客：http://8413723.blog.51cto.com/

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

线程基本状态

线程基本状态


开始
stop
终止
stop
stop
阻塞
stop
start
resume
notify
sleep
suspend
wait
就绪
运行中
yield

线程控制

线程控制-挂起

```
DWORD SuspendThread(

    HANDLE hThread    //线程句柄

 )
```

线程最多挂起MAXIMUM_SUSPEND_COUNT次，即127次，见winnt.h

思考：线程挂起的风险　可能锁定堆

线程控制-恢复

```
DWORD ResumeThread(

    HANDLE hThread    //线程句柄

 )
```

思考：线程恢复后进入哪个状态，是否可以立即执行？

注意：挂起几次就需要恢复几次

线程控制-睡眠

```
VOID Sleep(

    DWORD dwMilliseconds    //睡眠时间，单位：毫秒

 )
```

思考：

1. **Sleep函数的本质是什么？**
2. **Sleep函数可以根据参数时间准时醒来吗？**
3. **Sleep函数参数极值的意义：0 ？ INFINITE**

线程控制-切换

```
BOOL SwitchToThread( )
```

返回值：切换失败返回FALSE，否则返回TRUE

概念：饥饿线程

思考：

1. **Sleep函数传入参数0与SwitchToThread的区别**

线程执行时间

线程执行时间

```
DWORD GetTickCount(VOID)
ULONGLONG GetTickCount64(VOID)
```

两个函数的区别：GetTickCount 约49天溢出

返回值：从操作系统启动到现在的毫秒数

用法：两次记时取时差

线程执行时间

```
BOOL GetThreadTimes(
        HANDLE hThread,                 //线程句柄
        PFILETIME pftCreationTime,  //创建时间
        PFILETIME pftExitTime,          //退出时间
        PFILETIME pftKernelTime,      //内核模式下运行时间
        PFILETIME  pftUserTime)       //用户模式下运行时间
```

计量单位：100ns

用法：两次记时取时差

进程执行时间

```
BOOL GetProcessTimes(
        HANDLE hProcesss,                //进程句柄
        PFILETIME pftCreationTime,  //创建时间
        PFILETIME pftExitTime,          //退出时间
        PFILETIME pftKernelTime,      //内核模式下运行时间
        PFILETIME  pftUserTime)       //用户模式下运行时间
```

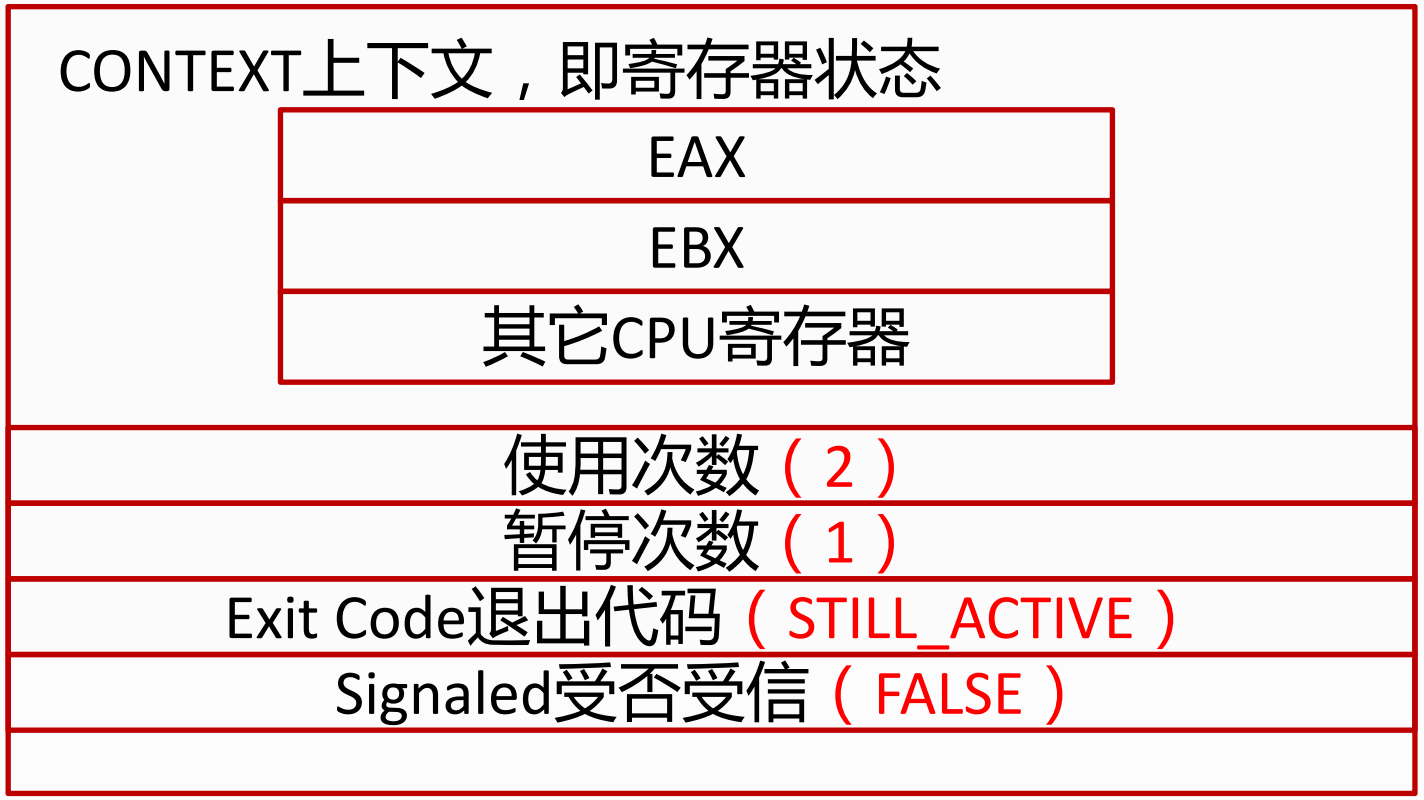
计量单位：100ns

用法：两次记时取时差

线程内核对象

线程内核对象

- 线程内核对象就是一个包含了线程状态信息的数据结构。每次对CreateThread的成功调用，系统都会在内部为其分配一个内核对象。



线程内核对象

- 线程上下文CONTEXT，反映线程上次运行的寄存器状态，确保线程间切换时可以还原现场。
- 使用计数Usage Count，初始值为2。每使用OpenThread打开一次，该值增加1，关闭一次减少1，当该值为0后，系统判定没有线程使用该对象，收回内存。
- 为什么Usage Count的初始值为2呢？

线程内核对象

对于上下文CONTEXT的操作

```
CONTEXT Context;
SuspendThread(hThread);
Context.ContextFlags = CONTEXT_CONTROL;
GetThreadContext(hThread, &Context);
Context. xxx = xxx;
...
SetThreadContext(hThread, &Context);
```

线程内核对象

引用计数示例

```
int main(void)
{
    HANDLE hCThread;
    HANDLE hOThread;
    DWORD dwThreadId;

    hCThread = CreateThread(NULL, NULL, Fun, NULL, 0, &dwThreadId);
    hOThread = OpenThread(THREAD_ALL_ACCESS, FALSE, dwThreadId);
    CloseHandle(hCThread);
    CloseHandle(hOThread);
    return 0;
}
```

线程内核对象

- 暂停次数Suspend Count，初始值为1，可以阻止新创建的线程被调度到CPU中。当内核对象初始化完毕后，该值自动减为0，此时线程是否执行取决于CREATE_SUSPENDED标志，该标志在创建线程时指定。通过SuspendThread挂起线程，它会使暂停次数加1，通过ResumeThread唤醒线程，它会使暂停次数减1。

- 退出代码Exit Code，线程函数返回值返回前的值为STILL_ACTIVE。

- 是否受信Signaled，线程运行时为FALSE，线程运行结束后为TRUE。

编码实战

51CTO学院

Thank You !

为梦想增值！

edu.51cto.com