

APC

1.线程是不能被“杀掉”、“挂起”、“恢复”的，线程在执行的时候自己占据着cpu，别人怎么可能控制它呢？所以说线程如果想死，一定是自己执行代码把自己杀死，不存在他杀的情况

那如果想改变一个线程的行为该如何？

可以给他提供一个函数，让他自己去调用，这个函数就是APC，即异步过程调用

2.APC队列

```
kd> dt _KAPC
nt!_KAPC
+0x000 Type
+0x002 Size
+0x004 Spare0
+0x008 Thread
+0x00c ApcListEntry
+0x014 KernelRoutine
+0x018 RundownRoutine
+0x01c NormalRoutine
+0x020 NormalContext
+0x024 SystemArgument1
+0x028 SystemArgument2
+0x02c ApcStateIndex
+0x02d ApcMode
+0x02e Inserted
```

找到你提供的APC函数，并不完全等于APC函数的地址，后面会讲。

将上图保存到下图的链表中

```
kd> dt _KTHREAD
nt!_KTHREAD
...
+0x034 ApcState : _KAPC_STATE
...
```

```
kd> dt _KAPC_STATE
nt!_KAPC_STATE
+0x000 ApcListHead //2个APC队列 用户APC和内核APC
+0x010 Process //线程所属或者所挂靠的进程
+0x014 KernelApcInProgress //内核APC是否正在执行
+0x015 KernelApcPending //是否有正在等待执行的内核APC
+0x016 UserApcPending //是否有正在等待执行的用户APC
```

当线程要结束的时候，会将APC函数填充到链表里，分为用户APC和内核APC

用户APC：APC函数地址位于用户空间，在用户空间执行。
内核APC：APC函数地址位于内核空间，在内核空间执行。

3.APC函数何时被执行

KiServiceExit函数：

这个函数是系统调用、异常或中断返回用户空间的必经之路

KiDeliverApc函数

负责执行APC函数

4. SaveApcState的意义

防止进程切换时CR3 产生改变，导致线程APC地址发生混乱，即进程发生切换时保存APCstate的备用APC队列

5. 挂靠环境下ApcState的意义

在挂靠的环境下，也是可以先线程APC队列插入APC的，那这种情况下，使用的时哪个APC队列呢

A进程的T线程挂靠B进程 A是T的所属进程 B是T的挂靠进程

ApcState B进程相关的APC函数

SavedApcState A进程相关的APC函数

在正常情况下，当前进程就是所属进程A，如果是挂靠情况下，当前进程就是挂靠进程B。

为了操作方便，_KTHREAD结构体中定义了一个指针数组ApcStatePointer长度为2。

正常情况下：

ApcStatePointer[0] 指向 ApcState

ApcStatePointer[1] 指向 SavedApcState

挂靠情况下：

ApcStatePointer[0] 指向 SavedApcState

ApcStatePointer[1] 指向 ApcState

6、ApcStatePointer 与 ApcStateIndex组合寻址

正常情况下，向ApcState队列中插入APC时：

ApcStatePointer[0] 指向 ApcState 此时ApcStateIndex的值为0

ApcStatePointer[ApcStateIndex] 指向 ApcState

挂靠情况下，向ApcState队列中插入APC时：

ApcStatePointer[1] 指向 ApcState 此时ApcStateIndex的值为1

ApcStatePointer[ApcStateIndex] 指向 ApcState

总结：

无论什么环境下，ApcStatePointer[ApcStateIndex] 指向的都是ApcState
ApcState则总是表示线程当前使用的apc状态

ApcQueueable用于表示是否可以向线程的APC队列中插入APC。

当线程正在执行退出的代码时，会将这个值设置为0，如果此时执行插入APC的代码(**KeInsertQueueApc** 后面会讲),在插入函数中会判断这个值的状态，如果为0，则插入失败。