Portable Systems Group

NT OS/2 Suspend/Resume Design Note

Author: *David N. Cutler*

Original Draft 1.0, February 9, 1989
Revision 1.1, March 30, 1989

This design note discusses a proposal to implement suspend and resume as part of the kernel rather than in the executive layer.

The suspension of a thread is controlled by a suspend count and a semaphore object that is built into the thread object. This semaphore has an initial value of zero and a maximum count of two (see explanation at end of this document as to why the maximum count must be two rather than one).

When an attempt is made to suspend a thread, the suspend count is incremented and a check is made to determine if the thread is already suspended (indicated by a nonzero initial suspend count). If the thread is not suspended, then a normal kernel **APC** is queued to the thread which will cause it to wait on its builtin semaphore.

A special case arises when the builtin **APC** is already queued to the target thread. This situation occurs when the target thread has been suspended and then resumed, but has never actually received the **APC** and suspended itself. Since the target thread has never actually suspended itself, the builtin semaphore count is decremented to indicate that the thread should suspend rather than resume.

The following pseudo code describes the logic of SuspendThread;

```
PROCEDURE SuspendThread (
     IN Tcb : POINTER KtThread;
     ) RETURNS integer;

VARIABLE

     OldCount : integer;

BEGIN
```

```
     Acquire dispatcher database lock;
     OldCount = Tcb.SuspendCount;
     IF Tcb.SuspendCount == 0 THEN
          IF NOT QueueApc(Tcb.SuspendAcb) THEN
               Tcb.SuspendSemaphore.Signal =
                    Tcb.SuspendSemaphore.Signal - 1;
          END IF;
     END IF;
     Tcb.SuspendCount = Tcb.SuspendCount + 1;
     Release dispatcher database lock;
     RETURN OldCount;
END SuspendThread;
```

Resuming a thread checks to determine if the thread has been suspended by examining the suspend count. If the thread has not been suspended, then no operation is performed. Otherwise the suspend count is decremented. If the resultant value is zero, then the target thread's builtin suspend semaphore is released.

The following pseudo code describes the logic of ResumeThread;

```
PROCEDURE ResumeThread (
     IN Tcb : POINTER KtThread;
     ) RETURNS integer;

VARIABLE

     OldCount : integer;

BEGIN

     Acquire dispatcher database lock;
     OldCount = Tcb.SuspendCount;
     IF Tcb.SuspendCount <> 0 THEN
          Tcb.SuspendCount = Tcb.SuspendCount - 1;
          IF Tcb.SuspendCount == 0 THEN
               Release Tcb.SuspendSemaphore;
          END IF;
     END IF;
     Release dispatcher database lock;
     RETURN OldCount;
END SuspendThread;
```

The maximum count of the builtin semaphore must be two so that the following race condition can be avoided.

1. a target thread is suspended by incrementing its suspend count to one and queuing its builtin suspend **APC**

2. before the thread can respond to the suspend **APC**, it is resumed which causes the suspend count to be decremented to zero and the builtin suspend semaphore to be incremented to one

3. the thread receives the suspend **APC**, but before it can wait on the builtin semaphore it is interrupted to deliver a special kernel **APC**

4. the special kernel **APC** code page faults and waits on the page to be brought into memory

5. the target thread is again suspended which causes its suspend count to be incremented and its builtin suspend **APC** to be queued

6. the thread is resumed before it has finished processing the special kernel **APC** which causes the suspend count to be decremented to zero and the builtin semaphore to be incremented to two

No additional nesting can occur since further attempts to queue the **APC** will fail which cause the semaphore count to be decremented. Thus the maximum count does not need to be greater than two.

Revision History:

Original Draft 1.0, February 9, 1989

Revision 1.1, March 30, 1989

1. Minor edits to conform to standard format.

[end of suspend.doc]