A/UX: The Event Manager and Mouse Events

Revised: 9/4/92
Security: Everyone

A/UX: The Event Manager and Mouse Events

==========

Article Created: 28 August 1990

Article Change History

08/31/92 - REVIEWED
- For technical accuracy

TOPIC ----------

I'd like to share a general observation, concerning A/UX and multitasking support while running the Finder.

While it is true that A/UX provides true pre-emptive multi-tasking for processes on the machine, it is also true that when the mouse button is held down, all activity appears to cease. The customer observation here is that the computer halts everything to deal with the mouse being down, and nothing seems to happen during this time.

What little I know about Macintosh development and priority queues tells me that StillDown events are taking precedence over screen update events. Therefore, the machine seems to stop. Although, in reality, it continues to process other tasks. Typical workstations have different priorities for user events, and, therefore, tasks continue to run and change their displays despite the actions of the user. Potentially, this could affect the responsiveness of the machine to the user, because user actions are not assigned any special priorities.

Is this is correct?

DISCUSSION ----------

Your observations are correct. The Event Manager always returns the highest priority event waiting in the queue. When using A/UX, even though it looks like processing has stopped when you hold down the mouse button, it still is continuing to give processor time to other processes. This is handled by the A/UX scheduler process and falls outside the realm of the Finder shell.

The Event Manager priorities are defined as follows:

1) Activate (window becoming inactive is higher than window becoming active).

2) Mouse down, mouse up, key down, key up, disk inserted, network, device driver, application defined (in that order).

3) Auto key.

4) Update (in front-to-back order of the windows).

5) Null.

To demonstrate this, you could use the small shell script that follows:

1) Start up this process.

2) Let it count up to 10.

3) Press and hold the mouse button. The screen stops updating. The log file will continue to be updated (you will continue to hear the drive).

4) While still holding the mouse button down, press Control-C. Screen output from the program should be halted from the time you first pressed the mouse button down, somewhere around 10.

5) Open the log file (/tmp/junk.log). It shows that the file was being updated even though the screen was not.

C Shell Script

```
#! /bin/csh

set counter = 1

while (1)

   echo "The counter value is " $counter | tee -a /tmp/junk.log

@ counter++

end

exit 0
```


Tech Info Library Article Number:6037