New Technical Notes

Macintosh

Developer Support

Giving the (Desk)Hook to INITs

Operating System

Revised by: Pete Helme October 1989
Written by: Pete Helme August 1989

This Technical Note discusses INIT evils, the foremost of which deals with clearing DeskHook and `DragHook` at INIT time.

Changes since August 1989: Added warning about clearing `DragHook`.

If you've survived the typical DTS Tirade***** and still feel the need to display a dialog box or window in an INIT, you need to be aware of a problem which exists on Macintoshes earlier than the Macintosh II (remember those?). There is a low-memory global named `DeskHook` ($A6C), which can contain a pointer to a routine responsible for painting the Macintosh desktop. If it is `NIL`, which is usually the case, the System paints the desktop with the standard pattern.

When you start displaying dialog boxes or windows that obscure the desktop in your INIT (this is really hard **not** to do, so keep reading and don't let us catch you skipping ahead to another Technical Note with pictures of human genetic experiments gone sour—you know which one I'm talking about), the System looks at `DeskHook` for a desktop updating routine. Since the Macintosh II, the System has cleared this hook prior to calling your INIT; however, on machines before the Macintosh II, this hook is not cleared before the System calls your INIT, so there is usually some junk hex lounging about in there. Since `DeskHook` is not `NIL`, when the System tries to use and perform a `JSR` to this "address," it blows big chunks.

So unless you like big chunks, the easy way to fix this problem is to clear `DeskHook` before doing any window drawing. The most logical time to do this is during your initialization:

```
PEA    thePort(A6)          ; initialize own grafPort off A6
_InitGraf
_InitFonts
_InitWindows
_TEInit
CLR.L  -(A7)
_InitDialogs
CLR.L  $A6C                 ; DeskHook
```

For you high-level types, this translates into:

```
procedure ClearDeskHook;
      inline
```

```
        $42B8, $0A6C;               { CLR.L $A6C ; DeskHook }
```

It doesn't hurt to clear it on newer machines either, even if it is already clear (you'll just have to trust me on this one), so go ahead and clear it all the time.

Note: Some INITs might actually use `DeskHook`. However, the popular ones that paint a picture on the desktop, which you might think use it, do not. They use other methods. (We know, we checked. We have the technology.) For those of you who have seen a real procedure pointer in location $A6C on earlier Macintoshes, don't worry. The system does not actually set `DeskHook` for its own use until the first application loads, so clearing it while INITs load is okay.

If there is some daring INIT out there which sets `DeskHook`, we haven't heard about it. As is the case with many low-memory globals, using `DeskHook` has never been supported.

Watch Out For This Guy Too

It should also be noted that `DragHook` ($9F6) is not cleared during INIT time on early Macintoshes either, and it will probably contain $FFFFFFFF. I guess no one in early Macintosh System Software wanted `DeskHook` to be lonely. `DragHook` can contain a pointer to a procedure that is called continuously while the mouse button is down. If you have a user interface at INIT time that ultimately calls `_TrackGoAway` for windows or `_TrackControl` for controls, look out. If the control is of the type to allow one of it's parts to be dragged with an outline, like a scroll bar's thumb, it calls `_DragTheRgn`, which checks to see if `DragHook` is `NIL`, and if it is not, it tries to perform a `JSR` to whatever address is there. `_TrackGoAway` also tries to perform a JSR to that address if it's not `NIL`. So make sure `DragHook` is clear before you attempt to use one of these routines.

In fact, if you've got a lot of spare time, like you're on the Voyager 7 project waiting to come into contact with Black Lectroids and you have an old Macintosh 512KE or Plus lying around, why not try randomly clearing out **all** low-memory globals and see what does and doesn't crash? Sure to be an ice breaker at parties.

* (Asterisk)

What am I yakking about? If you've ever written to DTS about getting help with displaying some kind of modal monster at INIT time (remember this for the later quiz kids, that's the time before this sort

of thing should normally happen), you know of what I yak.

We have this pet peeve with INITs that interrupt the boot process with a modal dialog box which asks us to enter our name then proceeds to ask us how many characters we just entered and what we had for dinner, especially when we've left the desk to go get a fix of a highly caffeinated substance. INITs were created for developers (and us) to install system patches and device drivers and make the occasional rude startup sound to annoy the person occupying the cube next to

ours. INITs were **not** developed to ask for personal (asexual) histories.

We do not mean to say that we don’t like all graphics at boot time. The ShowINIT icon mechanism that was popularized by Paul Mercer is great. In fact, we encourage it’s use and we gladly give out the ShowINIT MPW object file, with installation help, to anyone who asks

for it. This is an excellent method for a developer to inform a user whether a ROM patch or device driver has been successfully installed (show the red X through the icon on the rare occasion when things go wrong). Of course, this doesn't work if some non-social INIT makes an `_InitWindows` call, which wipes clean the entire screen (and with it all previous ShowINIT icons). You may argue that having the `_InitWindows` trap wipe out the entire screen at INIT time is a bad Macintosh OS INIT-time design, but this is one of our biggest complaints with the

whole INIT look-at-my-fancy-splash-screen-or-complete-my-insanely-great-modal-dialog-box phenomenon.

If you feel the need to notify the user of an important occurrence during boot time, initialize a notification request with the Notification Manager in your INIT code (see M.TB.Notification Manager, and yes, it is perfectly legal to use at INIT time), and the system will notify the user after the boot process, when the event mechanism starts. The now alerted user can then activate your desk

accessory, application, or whatever and you can perform whatever kind of pyrotechnics you want.

If you are going, “But, but, but…” because various Apple products are guilty of INIT evils, then you should realize that we are giving Apple engineers the same, if not more, grief to cleanse their acts as well.

It’s not that we’re telling you that you cannot put up a modal dialog box at INIT time if you feel like it’s really-absolutely-positively-it-was-your-dying-mother’s last-wish-

necessary. It's just that DTS would like to see a cleaner Macintosh interface (as I'm sure you all would), and a more uniform boot time appearance can help achieve this goal.

Further Reference:

- M.TB.Notification Manager