New Technical Notes
Macintosh

Process Manager Q&As
Processes   **M.PS.ProcessMgr.Q&As**

Revised by: Developer Support Center   October 1992
Written by: Developer Support Center   October 1990

This Technical Note contains a collection of Q&As relating to a specific topic—questions you've sent the Developer Support Center (DSC) along with answers from the DSC engineers. While DSC engineers have checked the Q&A content for accuracy, the Q&A Technical Notes don't have the editing and organization of other Technical Notes. The Q&A function is to get new technical information and updates to you quickly, saving the polish for when the information migrates into reference manuals.

Q&As are now included with Technical Notes to make access to technical updates easier for you. If you have comments or suggestions about Q&A content or distribution, please let us know by sending an AppleLink to DEVFEEDBACK. Apple Partners may send technical questions about Q&A content to DEVSUPPORT for resolution.

New Q&As and Q&As revised this month are marked with a bar in the side margin.

How to tell if Macintosh Process Manager is present
Written: 1/21/91
Last reviewed: 8/1/92

What is the proper way to tell if the Process Manager is present?

___

The System 7 Process Manager is available if a Gestalt call with the selector gestaltOSAttr returns noErr and the gestaltLaunchControl bit is set in the response. This is documented on page 29-17 of Inside Macintosh Volume VI.

Code snippet to check for high-level event support in a process
Written: 11/12/90
Last reviewed: 8/1/92

How can I find out if a specific process is running on my system and if it supports high-level System 7 events?

___

You can check to see if a specific process supports high-level events with code similar to this:

```
Program Test;

USES
```

```
 Types,
 Processes;

Var
 err: OSErr;
 myPSN: ProcessSerialNumber;
 myPInfoRec: ProcessInfoRec;
 myPName: Str255;
 myProcessSignature: OSType;

BEGIN
 myPSN.highLongOfPSN := 0; { start at beginning of process list }
 myPSN.lowLongOfPSN := kNoProcess;

 myPInfoRec.processInfoLength := sizeOf(ProcessInfoRec);
 myPInfoRec.processName := @myPName;
 myPInfoRec.processAppSpec := NIL;

 myProcessSignature := 'MACS'; { looking for the Finder's signature }

 WHILE (GetNextProcess(myPSN) = noErr) DO
   IF GetProcessInformation(myPSN, myPInfoRec) = noErr
     THEN
       IF (myPInfoRec.processSignature = myProcessSignature) AND
         (BAND(myPInfoRec.processMode, modeHighLevelEventAware) <> 0)
           THEN
             Writeln(myPInfoRec.processName^, ' is high level event aware.');
END.
```

Launching a Macintosh application

Written: 4/30/91
Last reviewed: 8/1/92

Under MultiFinder or Macintosh System 7, is there a way to set a flag to terminate the current application before launching a different application? Under standard Finder, the launcher quits before the launchee starts. However, under MultiFinder or System 7, they both must stay in memory for a brief time until control returns to the launcher and it hits the ExitToShell. Because most of our programs are set to use 1 MB, we need 2 MB of memory available for a short time.

—

Two distinct cases are involved here: the System 7.0 case and the pre-System 7.0 case.

In System 7.0 there is a new way to launch applications. The function is called (appropriately enough) LaunchApplication. The function and its new extended parameter block are documented in the Process Management chapter of Inside Macintosh Volume VI.

You can determine whether your system software has the new Launch Control capabilities by using Gestalt:

```
OSErr myOSErr;
long response;

myOSErr = Gestalt(gestaltOSAttr, &response);
if (noErr != myOSErr)   /* If an OSErr occurs we must be pre-System 7.0 */
 /* Handle pre-System 7.0 case */
if BTst(response, gestaltLaunchFullFileSpec)
 /* Handle System 7.0 case */
else
```

```
 /* Handle pre-System 7.0 case */
```

The above code assumes that you are developing with interfaces and libraries based on MPW 3.2, as those interfaces and libraries support Gestalt even on system software versions that don't implement the Gestalt trap.

The portion of the Process Management chapter of Inside Macintosh Volume VI that will interest you is the description of the launchControlFlags field of the parameter block. Here's what the Process Management chapter has to say:

"When you use the LaunchApplication function, you specify the launch options in the launchControlFlags field of the launch parameter block. These are the constants you can specify in the launchControlFlags field.

```
CONST
 launchContinue = $4000;
 launchNoFileFlags = $0800;
 launchUseMinimum = $0400;
 launchDontSwitch = $0200;
 launchAllow24Bit = $0100;
 launchInhibitDaemon = $0080;
```

Set the launchContinue flag if you want your application to continue after the specified application is launched. If you do not set this flag, LaunchApplication terminates your application after launching the specified application, even if the launch fails."

In other words, you have to request that your launching application continue if you use LaunchApplication—the default is to do precisely what you want. Note that clearing the launchContinue flag is effectively the same as following the LaunchApplication call with an ExitToShell call. Because applications should not terminate within an Apple event handler, they must set the launchContinue flag when calling LaunchApplication from within an Apple event handler.

This leaves the pre-System 7.0 case. System 4.1 and later allows for the use of an extended Launch parameter block also, but it is somewhat different from that of System 7.0. The complete details can be found in the Macintosh Technical Note "Sub(Launching) From a High-Level Language." You'll need to use the extended version of the parameter block and specify a launch rather than a sublaunch.

Unfortunately, the launching application does in fact _not_ quit until the launchee has come up. It sounds like you will have to launch a tiny utility application that then turns around and launches your real target.

A good example of how all of this works in the System 7.0 case can be found in the ProcDoggie 1.0a6 sample code, which is in the Sample Code folder on the System 7.0 February 1991 beta CD-ROM. Among other things, it demonstrates how to launch, and how to handle or send the required AppleEvents ('oapp' or 'odoc').

Changing the 'SIZE' -1 resource multilaunch bit at run time
Written: 1/6/92

Last reviewed: 8/1/92

Can an application change its own 'SIZE' -1 resource to make it multilaunch, after the user personalizes it the application does a once-only write to itself? If we change the 'SIZE' -1 resource immediately after personalization and update the resource file, can other users on the network double-click the application before the very first user (who personalized it) quits?

___

You may certainly modify the 'SIZE' -1 resource in your application at run time. The only reason for not modifying the application file is that usually keeping a preferences file is much better for the user. Your situation leaves you no choice but to modify the app file itself. Unless you were to never store back into your app and create a preferences file for the initial configuration…

If you change your multilaunch bit, the application will need to quit before you can multilaunch it. When a single launch app is launched, it is opened with read/write exclusive privileges instead of multiuser privileges. So, no one else will be able to open the file to see that it is multilaunch until after you have quit. But, once you restart your application, it will be multilaunchable from then on.

System 7 and MultiFinder's "puppet strings"
Written: 8/30/91
Last reviewed: 8/1/92

When my application is running, it relies on the MultiFinder's "puppet strings" (which choose Open from the application's File menu and suppress the SFGetFile dialog) to open a document that was double-clicked in the Finder. Why doesn't this work under System 7? The high-level event-aware bit in my 'SIZE' resource is clear.

___

System 7 will not pull puppet strings for an applications that makes use of the System 7 Standard File routines, such as StandardGetFile and CustomGetFile, nor will it pull them if the application's high-level event-aware bit is set.

If you update an older application to take advantage of any System 7 features, be sure to also add support for the 'odoc' and other required Apple events.

Background applications and alerts
Written: 5/4/90
Last reviewed: 8/1/92

My application, running in the background, detects an error condition and puts up an alert;

but only part of the alert is visible (other windows cover most of it). When I switch back to my application, the full alert becomes visible but the "OK" is not highlighted and often the icon is not shown either. Is there something I can do to prevent this?

___

Applications running in the background should not be putting up alerts. In addition to the problems and complications you have already noted, under System 7.0 the user has the ability

to hide a background application's window set and your alert would not be seen at the time you expect it to.

There is, however, a mechanism in place that your background application can and should use: the Notification Manager. Documented in Inside Macintosh Volume VI, this manager allows a background process to notify the user that there is a alert, message, or situation pending in the background that needs to be taken care of. The user can then bring your application forward and you can present your alert as the foreground application.

Process Manager saves and restores the 68881/882 registers
Written: 5/3/89
Last reviewed: 8/1/92

Does the Process Manager save and restore the 68881/882 registers and state when it switches between applications ?

___

Yes. As far as your application is concerned, it has the FPU to itself. The only time you need to worry about saving and restoring the FPU's state is when you are doing floating point operations at interrupt time, such as in a VBL or completion routine. Watch out for calls to the Sound Manager, since it will use floating point operations internally.