New Technical Notes

Macintosh

Developer Support

Memory Management Overview Q&As

Memory

Revised by: Developer Support Center May 1993
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.

Setting the Current Zone for Macintosh System Extensions

Date written: 1/15/92
Last reviewed: 3/2/92

Sometimes when my system extension (INIT) starts executing, the current zone is the system zone rather than the application zone. Should I call SetZone(ApplicZone) before allocating memory in the system extension?

—

The system does not set the zone to the application zone before loading each system extension, so if a previous extension left the zone set to the system zone, it's possible that an extension could unintentionally be loaded into the system heap and have the current zone be the system zone.

To ensure that nonpermanent memory requested by a system extension is allocated in the application heap, do a SetZone(ApplicZone) before calling NewHandle or NewPtr. Any system extension that calls SetZone should restore the current zone to what it was upon entry.

Any permanent memory allocation by a system extension should be made in the system heap

with NewPtrSys or NewHandleSys. Use a 'sysz' resource if the system heap allocations will exceed 16K.

Macintosh disk drivers and memory allocation

Date written: 9/11/92
Last reviewed: 3/10/93

Our disk driver needs to allocate a large amount of space, and on some machines the amount of system heap space at boot is limited. What can we do to ensure enough system memory for our driver?

—

Drivers that require more than the default amount of heap space need to adjust the size of the system heap directly. The following code is used by Apple's current SCSI disk driver to increase the system heap at boot time. This adjustment should be as minimal in size as necessary. It works with 8K, but there are upper limits. This code should be used in the driver's install routine. It's important that the driver does this as the very first step, after performing the checksum on the driver code if you use one. This code will fail after boot time.

```
; ---ONLY PERFORM THIS CODE AT BOOT TIME.---
;
; Bit 31 of d5 is 0 if we were booted by the ROM (in
; which case we want to bump up the system heap size),
; else it's set by the Installer (leave heap alone).
;
     tst.l    d5                  ; if Installer called us...
     bmi.s    InstallDCE          ;   don't do anything else

     move.l    SysZone,a0         ; get start of system heap
     move.l    (a0),a0            ; get bkLim    add.w    #8000,a0        ; add about 8K
     move.l    TheZone,-(sp)    ; save so it doesn't get trashed
     _SetAppBase                ; (boot blocks can make a larger
                                   ; system heap if they want to)
     move.l    (sp)+,TheZone    ; fixes startup screen bug
```

A SCSI driver being loaded will be passed D5 with the high byte clear, and the low byte containing the SCSI ID. If the high byte is clear, it's a good bet that it's boot time. If your application is installing the driver, such as the formatter/installer utility, it should set the high bit of D5. When the driver's loader code gets called, it will test D5 for this bit being set. When this bit is set, you'll know not to perform the adjustment to the system heap. Or you can use the following method from the original sample SCSI driver:

```
;Are we running at boot time? (we can tell this by looking to see if
;SysEvtMask has been initted yet!)
     TST.W    SysEvtMask     ;are we running at boot time?
     BNE.S    InitExit          ;no, we're done.
```

Additionally, if more memory is necessary the driver can set the dNeedTime bit in the drvrFlags field and wait for an accRun call at SystemTask time. This would happen after the boot process was completed. At this time there may be enough system heap space to accommodate the driver's needs. MultiFinder and System 7.0 will increase the system heap if possible. Using this method, you should be able to get the room you need in the system heap; you may simply need to do it in stages.

If an INIT is to install the driver, use the 'sysz' resource to have the system adjust the heap size for you. If an application is to install the driver, ask for the system space. MultiFinder will attempt to create space for it if the requested memory is not available. In any case, if the memory is not available, tell the user about the problem.

When to use SetCurrentA5
Date written: 11/11/90
Last reviewed: 5/21/91

During a Macintosh application's life, does the value of A5 change? Why does SetCurrentA5 have to set A5 to CurrentA5? Aren't A5 and CurrentA5 the same while an application is executing non–interrupt-time code?

—

A5 is not necessarily always equal to CurrentA5. Because the Macintosh operating system and Toolbox don't need to access your application's jump table or global variables (which A5 points to), they often use A5 for other purposes, except for the parts of the Toolbox that need to perform graphics operations and use the QuickDraw globals defined by your application.

Because the operating system or Toolbox can change A5, you must make sure it's set correctly if the operating system or Toolbox ever calls your code in the form of a callback. For example, if you make an asynchronous File Manager call, your I/O completion routine must call SetCurrentA5 for your I/O completion routine. Other places where you may want to call SetCurrentA5 are in trap patches, your GrowZoneProc, custom MDEFs, WDEFs, or CDEFs, or control action procedures.

Please note that even SetCurrentA5 is not sufficient when writing code that's executed at interrupt time, such as VBL and Time Manager tasks. When an interrupt occurs that your application wants to handle, there is no guarantee that the low-memory global CurrentA5 belongs to your application. The interrupt could occur while some other application is running under MultiFinder®. In this case, you should use other approaches to setting A5. Please see Technical Note #180, MultiFinder Miscellanea, for further information.

Macintosh GrowZone routine is not reentrant
Date written: 12/10/90
Last reviewed: 2/20/91

My Macintosh application receives information from the serial port, keeping as much information in memory as possible. However, if I do run out of memory, my GrowZone routine is allowed to delete older information. Does the operating system allow the GrowZone routine to be called recursively? That is, if in handling a GrowZone call, my code does something that requires memory, such as bringing in a font that isn't there, can I call the GrowZone routine again or would doing so cause an error? Am I allowed to update things on the screen, such as windows and text in windows, while in my GrowZone routine?

—

Neither the GrowZone routine nor the Macintosh system is reentrant. There is very little that you can safely do in a GrowZone routine. The system expects that the typical extent of a

GrowZone routine is to check a table or some other predesignated criteria and then to release or not to release any memory blocks it can. Any sort of user interface is well beyond the reasonable scope of this level of routine.

Keep a list of handles to blocks of text that you're willing to purge, so that your GrowZone routine can look through them when called and then release one or more blocks. Any more than that is really stretching GrowZone too far.

By the way, SetGrowZone should not be assumed to return a result. *Inside Macintosh* is incorrect in saying that D0 contains a result code on exit.

Increasing heap space for Macintosh DAs
Date Written: 3/12/91
Last reviewed: 8/1/92

I am writing a Macintosh desk accessory (DA) that opens large PICT files and makes off-screen pixel maps for them. Is there a way for a DA to ask for more space when it opens up, or should we be modern about this and make it an application instead of a DA?

___
A DA, like any application, can ask for available MultiFinder temporary memory, as documented in *Programmer's Guide to MultiFinder* (APDA #M7044) for running under System 6.0.x and in *Inside Macintosh* Volume VI for running under 7.0. You can also set the current heap to the system heap so that memory allocation will take place there until you set it back to the current heap. If you do this, you open up the can of worms of "how much space is left in the system heap?" Under Finder there's a fixed amount of space, set by the boot blocks. Under MultiFinder (and System 7.0) the system heap adjusts itself to keep about 16K free, but if you allocate all 16K you will not know when it will resize larger or when the system might crash because all of its heap space has been used up.

DAs, however, were never intended to use large amounts of memory. For all but the most trivial (and small) functionality, it is better to implement as an application than as a DA, especially under System 7.0 where an application has all the advantages of a DA with none of the restrictions.

Sharing data between Macintosh applications
Date Written: 8/23/91
Last reviewed: 8/1/92

What are the preferred methods for sharing data between applications? I want to use the Program-to-Program Communications (PPC) stuff or Apple events to dynamically link libraries, but the size of the data blocks to be passed between the applications makes copying them around unacceptable. The best method would be to somehow share the data and remain compatible with future systems (presumably with memory protection). How is the system heap on handling large chunks of memory?

___

As you point out, there could be some memory protection someday, but at this time there's no indication of how shared memory will be allowed under these tighter rules.For the time being, if you are willing to pay the consequences later, then the following might help:

The 4 bytes of an address for a location in memory could be passed as data in a PPC data transfer, or Apple events could carry the information. Secondly, the Gestalt mechanism could be used to pass an address between processes or tasks. For your problem this probably is better then the other forms of IPC.

The best thing to do is allocate a shared handle in the system heap. HLock the block only when necessary. Remember, the system heap grows dynamically under System 7.0.

Preloading and locking Macintosh code segments
Date Written: 6/8/92
Last reviewed: 9/15/92

What is the correct way to preload and lock all of a Macintosh program's segments?

—

The best way to preload and lock segments is to use the Resource Manager. Just set the Preload and Locked bits on the segment resources. This will cause all these segments to be loaded contiguously, low in the heap. To avoid fragmentation, you should leave these segments loaded all the time; never call UnloadSeg for any of them. The resources get loaded in early, and their jump table entries get converted the first time they are used. This is probably preferable to just calling functions from all the segments to get them loaded in. If you are concerned about LoadSeg being called at run time, either for speed reasons or because you want to avoid calling it at interrupt time, then you can mark them locked and preloaded (for heap management reasons), and call routines in each of the segments to convert their jump table entries to the "loaded" state. You can mark the segments by adding the options "-ra =resPreload,resLocked" to your link command. If you only wanted to preload a few segments, you should generate multiple -ra commands, each for a different segments, such as "-ra STDIO=resPreload,resLocked."

An alternative is to put everything into one big segment using -model far. In this way you'll get good heap management and the jump table will all be initialized at once. However, it's somewhat less flexible and you will incur a small speed penalty if you use -model far; the choice is up to you.

Macintosh system heap limitations
Date Written: 7/13/92
Last reviewed: 11/30/92

In System 7, the memory allocated for INITs by the 'sysz' resource mechanism seems to be limited to about 16 MB on our 32 MB Macintosh IIfx. For 'sysz' values up to 15 MB it works great, but it seems the system heap can't grow beyond 16 MB. Is there some reason for this?

—

The system heap size at startup is limited to approximately half the size of total RAM. This is because the early startup code places the stack and some globals in the middle of RAM so that the system heap can grow up from below while BufPtr is lowered from above. This is basically the situation until an application is launched. Things are eventually rearranged so that the system heap will have more room to grow, but this doesn't happen until the Process Manager is launched, after INIT time. This limitation would mean that you could size your

heap until it reached nearly (but not quite) half the size of RAM. We suggest that you attempt to allocate some of your RAM later, after the Process Manager starts up; at that point, the system heap should be somewhat less limited.