

New Technical Notes

Macintosh



®

Developer Support

Memory Management Overview Q&As

Memory M.ME.MemMgtOv.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.

Increasing heap space for Macintosh DAs

Written: 3/12/91

Last reviewed: 8/1/92

I am writing a Macintosh desk accessory (DA) that opens large PICT files and makes offscreen 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

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. In the time being, if you are willing to pay the consequences later, then the following might help:

The four 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 than 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.

Pre-loading and locking Macintosh code segments

Written: 6/8/92

Last reviewed: 9/15/92

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

The best way to pre-load 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 runtime, 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

Written: 7/13/92

Last reviewed: 9/15/92

Under System 7 with 32-bit, the 'sysz' resource for INITs seems to be limited to about 15-16 MB. We have a 32 MB RAM Macintosh IIfx. For 'sysz' up to 15 MB it works great. Is there some reason for this? It seems the system heap can't go beyond 16 MB. Is this true?

The system heap size at startup is limited to approximately 1/2 the size of total RAM. This is because the early startup code places the stack and some globals in the middle of RAM so the system heap can grow up from below while BufPtr is lowered from above. This is basically the situation until an application is launched. Under System 6 or 7, this doesn't happen until the Process Manager is launched, after INIT time. This limitation would mean that you could size your heap up until it reached near (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.