Tech Info Library

Newton: Types of Memory & Memory Errors (11/94)

Revised: 4/21/97
Security: Everyone

Newton: Types of Memory & Memory Errors (11/94)

==

Article Created: 31 October 1994
Article Reviewed/Updated: 22 November 1994

TOPIC ---

This article describes the different types of memory on the Newton MessagePad, how to get the most out of your existing memory, and memory related errors.

DISCUSSION --

The Newton has two kinds of memory, system memory and user storage. System memory is analogous to the RAM memory in a desktop computer, user storage to the storage space on a disk drive. The memory space reported in the Memory Preferences window is for user storage. There are shareware packages that measure system memory (heap space), but it is not displayed by the Newton system software. This system memory is further broken down into different uses, but we won't expand on what they are in this discussion. It is in the system memory that programs actually run and store the intermediate data that they create. Permanent data, as well as the packages themselves are stored in the user storage.

When an out-of-memory-restart error occurs, the Newton runs out of system memory. That is, some part of the system or a package made a request to use system memory and that request failed due to insufficient memory. The machine was unable to proceed without the memory so it did the only thing that it could, offer to reset the Newton MessagePad. In simple terms, it offered to wipe the slate clean and start again. Tapping reset in the error dialog has the same effect on system memory as opening the battery door and pressing reset.

Normally there is enough system memory to go around. For instance, many members of the PIE (Personal Interactive Electronics) division have been running MessagePad 110's since product launch and never gotten an out-of-memory-restart error. But, each time you load a package, either by downloading, or by inserting a PCMCIA card, that package is "installed". Installation integrates the package into the Newton. It uses some small amount of system memory to do this. Install enough packages, and the Newton MessagePad runs out of system memory. If this were the only cause of out-of-memory-restart errors, the problem would rarely occur. Installing a package takes up only a few hundred bytes. Installing hundreds of packages would be required to run out of memory this way.

Removing the PCMCIA card or removing the software from the card or the internal user storage removes not only the package, but also the integration information, thereby freeing up the same small amount of system memory.

Packages can do other things to make permanent use of system memory. They can create objects and then leave references to those objects around so the system can't get rid of them. Resetting the Newton removes all these objects and references until a package creates them again. It is this "garbage data" left around by packages that contributes to memory problems. There are packages that leave around 20K bytes or more of data each.

Why does this limitation exist?

On a desktop computer whenever you close or quit an application, the information for that application is eliminated and the memory space reclaimed. Only when many applications are run simultaneously does a system run out of memory . You can attempt to run one application that asks for more memory than is available, but this is a less common case. Most applications know how to live within their own memory allocation. The system only has problems when several of them are run simultaneously.

It is this issue of running simultaneously that we need to address. Not enough has been said about the Newton MessagePad architecture in the press or in the developer manuals. The Newton MessagePad is a new and different kind of platform. It has distinct advantages that are not possible with older machines (operating systems). It also has limitations, some of which are imposed by the new architecture.

Consider a Macintosh or a Windows computer. A typical user has 8 to 32 megabytes of RAM and at least 120 megabytes of disk. However, there are still many Mac Plus computers out there with 1 Meg of RAM and a 20 Meg disk. These arguments apply to them too, it's just a matter of scale. When running a program on this type of machine, memory is allocated to the program when it is loaded from the disk, perhaps a data file is loaded from the disk, and the program's function is performed. Afterward the program and the data are flushed from the RAM and the memory is reused to run the next program. When the program is not running, it's dead. It can't receive messages, it can't respond to changes in its files, it can't communicate.

Consider the size of that machine. There will be a day when 8 megabytes of RAM, 120 megabytes of disk and the power to run them will fit in the palm of your hand and cost less than $600, but that day has not arrived yet.

The Newton MessagePad architecture is targeted at machines you can hold in your hand for less than $600 today. That means that the storage of the machine must be compressed in space and cost (not to mention battery power requirements). It is this compression that led to Apple's creation of NewtonScript, an interpreted language that is far more memory efficient than C or C++.

The Newton MessagePad architecture is also targeted at being a personal assistant. As an assistant, all of its functions should be available all the time. Even more importantly, it has an object oriented database, called "soups", rather than files so that Newton functions and packages have access to all the data. Cross functional (inter-package) access to data is allowed and even encouraged.

Even when it's not "running", a Newton package can respond to changes in its soups, it can respond to incoming data from modem, network, IR, or even messages from other packages.

Newton packages can access the data in another package's soup. A fax editor package can make use of the Names soup for its address book. It can get its data from the NotePad, or from an e-mail message. A Contact Manager package can make use of the Names soup to look up contact information. It can schedule items into the To Do list, or the Calendar. Apple calls this Newton Intelligence.

There are single applications on desktop computers that integrate these sorts of cross functions. These "all in one" applications exist because desktop machines have not yet learned how to have the separate applications work together. As a result, you can't have your favorite address book program communicate with your favorite calendar program to create a contact manager function. The two programs don't talk to one another.

Newton packages need to always be active in some way. When a package is loaded into the Newton, the package tells the system about itself and its soups. It prepares itself and the system to deal with its soup data and messages it might receive. This causes the package to take up system memory even though the user has not opened it. Again, only a very small amount of memory is used up for this capability.

Not every package on the Newton, including those from Apple, has yet to exploit all of the Newton's advantages. That is why it is important to get this message out. Developers need to create packages that make use of Newton Intelligence. A package which duplicates the information in another package's soup is wasting storage space. In a sense, a package that duplicates the function of another package is also wasting space. Packages should work together, use one another's soups, and only provide the new functions that don't already exist.

Users who load packages that leave "garbage data" around will eventually see out-of-memory-restart messages. The way to minimize this problem by not loading packages that require a lot of permanent system memory. Watch out for these packages and put them on a separate card. Run these packages by themselves. When you are through running them, the Newton MessagePad should be reset. This is an extreme measure, and only applicable if you are getting a lot of out-of-memory-restart errors.

Here is another strategy. When you identify a particular package or combination of packages that causes out-of-memory-restart errors, break up that particular combination. "Break up" means don't have that combination installed simultaneously. If you install lots of packages internally, move some of them to cards that aren't normally loaded in the Newton MessagePad. If a couple of packages are suspected of causing problems, then move them to separate cards so that they are not loaded at the same time. Remove packages that have little or marginal value.

As developers learn more about the Newton MessagePad, fewer packages will waste space. Even now, the Newton performs well for the majority of customers. It offers users and developers features that electronic products have not had available before. Apple hopes more developers will take advantage of the unique Newton architecture and build products that make efficient use of existing soups and Newton functions. This will lead to a powerful integrated Newton system.

Questions & Answers

Q: How can customers identify packages that use more system memory?

A: There are various shareware packages that measure system memory (heap space). Users with access to one of these packages can measure the amount of system memory a package uses by measuring the system memory just before and just after running the package the first time. The amount that the system memory goes down is the amount used by the package. If the amount is more than one or two kilobytes then that package is using more memory than it should.

Users who don't have access to a package that measures system memory, will have to infer which packages are causing problems. When out-of-memory-restart errors are frequent, the installed combination of packages is straining the Newton. Try to break up that combination until the errors stop. It's sort of like tracking down conflicting extensions on the Macintosh or conflicting memory resident applications on the MS-DOS PC.

Q. Is removing install scripts with a soup editor such as RemoveIt, StewPot, or Souper a good or bad idea to increase system memory?

A. It's a bad idea. The install script may contain code that the package needs while running. The package authors need to be the ones to modify a package to use less system memory. It's not a good idea to try and second guess the authors of the package.

Article Change History:
22 Nov 1994 - Renamed article.
07 Nov 1994 - Titled changed to reflect article.

Support Information Services

Copyright 1994, Apple Computer, Inc

Tech Info Library Article Number:16645