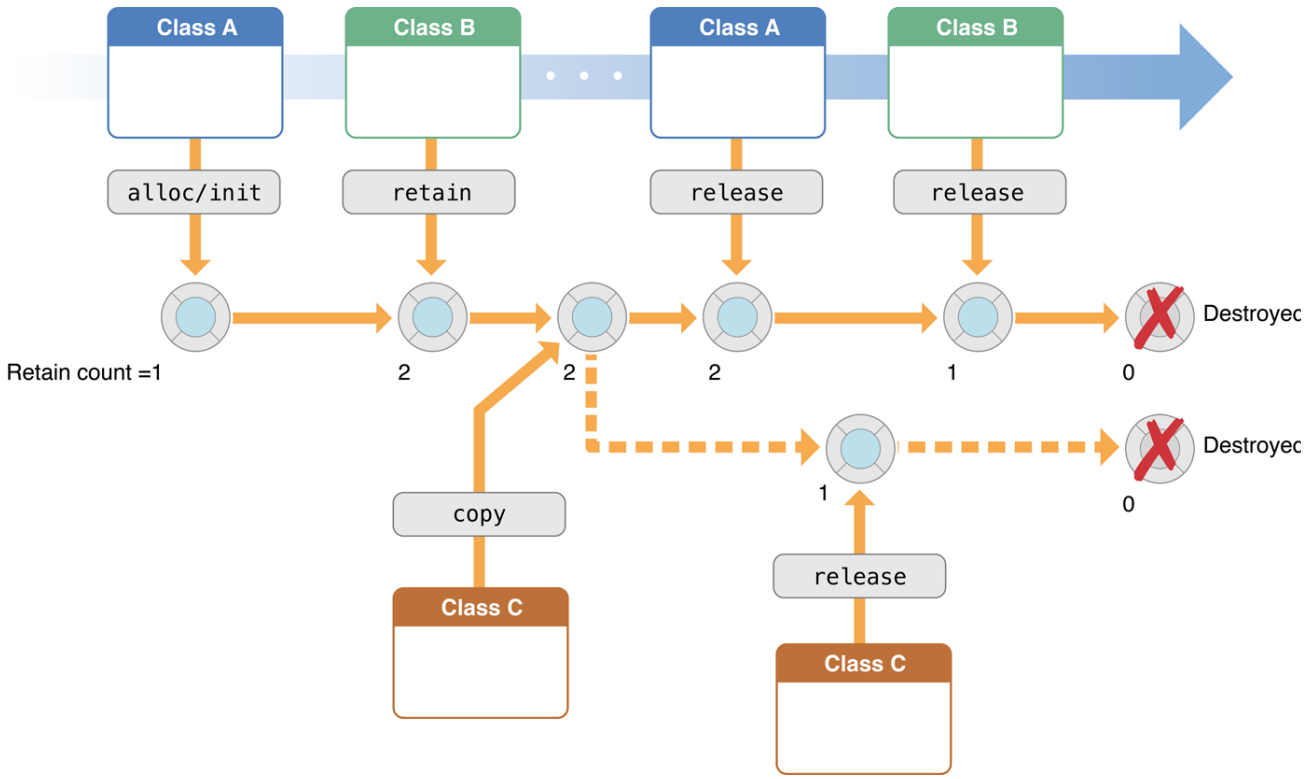
About Memory Management

Application memory management is the process of allocating memory during your program's runtime, using it, and freeing it when you are done with it. A well-written program uses as little memory as possible. In Objective-C, it can also be seen as a way of distributing ownership of limited memory resources among many pieces of data and code. When you have finished working through this guide, you will have the knowledge you need to manage your application's memory by explicitly managing the life cycle of objects and freeing them when they are no longer needed.

Although memory management is typically considered at the level of an individual object, your goal is actually to manage object graphs. You want to make sure that you have no more objects in memory than you actually need.



At a Glance

Objective-C provides two methods of application memory management.

1. In the method described in this guide, referred to as "manual retain-release" or *MRR*, you explicitly manage memory by keeping track of objects you own. This is implemented using a model, known as reference counting, that the Foundation class `NSObject` provides in conjunction with the runtime environment.
2. In Automatic Reference Counting, or *ARC*, the system uses the same reference counting system as MRR, but it inserts the appropriate memory management method calls for you at compile-time. You are strongly encouraged to use ARC for new projects. If you use ARC, there is typically no need to understand the underlying implementation described in this document, although it may in some situations be helpful. For more about ARC, see *Transitioning to ARC Release Notes*.

Good Practices Prevent Memory-Related Problems

There are two main kinds of problem that result from incorrect memory management:

- Freeing or overwriting data that is still in use

This causes memory corruption, and typically results in your application crashing, or worse, corrupted user data.

- Not freeing data that is no longer in use causes memory leaks

  A memory leak is where allocated memory is not freed, even though it is never used again. Leaks cause your application to use ever-increasing amounts of memory, which in turn may result in poor system performance or your application being terminated.

Thinking about memory management from the perspective of reference counting, however, is frequently counterproductive, because you tend to consider memory management in terms of the implementation details rather than in terms of your actual goals. Instead, you should think of memory management from the perspective of object ownership and object graphs.

Cocoa uses a straightforward naming convention to indicate when you own an object returned by a method.

See Memory Management Policy.

Although the basic policy is straightforward, there are some practical steps you can take to make managing memory easier, and to help to ensure your program remains reliable and robust while at the same time minimizing its resource requirements.

See Practical Memory Management.

Autorelease pool blocks provide a mechanism whereby you can send an object a “deferred” `release` message. This is useful in situations where you want to relinquish ownership of an object, but want to avoid the possibility of it being deallocated immediately (such as when you return an object from a method). There are occasions when you might use your own autorelease pool blocks.

See Using Autorelease Pool Blocks.

Use Analysis Tools to Debug Memory Problems

To identify problems with your code at compile time, you can use the Clang Static Analyzer that is built into Xcode.

If memory management problems do nevertheless arise, there are other tools and techniques you can use to identify and diagnose the issues.

- Many of the tools and techniques are described in Technical Note TN2239, *iOS Debugging Magic*, in particular the use of `NSZombie` to help find over-released object.
- You can use Instruments to track reference counting events and look for memory leaks. See Collecting Data on Your App.