;DebuggingAMultithreadedProgram.rtfd;;¬ Next Section ;ExaminingTheValueOfAVariableOrAnObject.rtfd;;¬ Previous Section

Debuggingobjectallocationanddeallocation;¬Debugging object allocation and deallocation

arrow.eps ¬ Use enableFreedObjectCheck: inside gdb.

Or

arrow.eps ¬ Use the oh tool to see where and when objects are allocated and deallocated.

Or

arrow.eps ¬ Use the AnalyzeAllocation tool to see where and when objects are allocated and deallocated.

Object allocation and deallocation are often trouble spots. Two common problems are using an object after it has been deallocated and releasing an object too many times. Here are some strategies and tools to debug object allocation and deallocation.

A typical autorelease error:

```
objc: FREED(id): message objectForKey: sent to freed object=0xfde44
```

Debugging Autorelease Errors in gdb

If you are releasing an object too many times, invoke the NSAutoreleasePool class method **enableFreedObjectCheck:** and set a breakpoint on **_NSAutoreleaseFreedObject**.

enableFreedObjectCheck: causes all **autorelease** and **release** messages to first check to see if the receiving object is already in an autorelease pool. If it is, they won't deallocate the object. When the program hits the breakpoint, look at the stack to see what method was releasing the object.

_DebuggingObjectAllocation3.eps ¬

Using the oh Command

Another way to debug the **autorelease** and **release** errors is to use the **oh** command in conjunction with **gdb**. When you start the **oh** command, it starts recording allocation and deallocation events related to the process you specify. You set **NSZombieEnabled** so that the memory for deallocated objects is not reclaimed. (Released objects are just turned into ªzombies.º) The advantage to setting this variable is that you can ensure than an object's address is unique.

_DebuggingObjectAllocation4.eps ¬

When you receive an autorelease error perform the command:

```
% oh pid address
```

where *address* is address of the object that is being release twice. **oh** will produce a report showing you the stack frame each time that object is allocated, copied, retained, or released, like the one shown on the next page.

```
== Stacks for address 0xfa31c, in temporal order (oldest first):
(
    "+[NSMutableDictionary allocWithZone:]",
    "+[NSDictionary dictionary]",
    "-[TAController init]",
    "-[NSCustomObject nibInstantiate]",
    "-[NSIBObjectData instantiateObject:]",
    "-[NSIBObjectData nibInstantiateIn:owner:]",
    _loadNib,
    "+[NSBundle(NSNibLoading) loadNibFile:...]",
    "+[NSBundle(NSNibLoading) loadNibNamed:owner:]",
    _main,
    start
)
(
    "+[NSDictionary dictionary]",
    "-[TAController init]",
```

```
    "-[NSCustomObject nibInstantiate]",
    "-[NSIBObjectData instantiateObject:]",
    "-[NSIBObjectData nibInstantiateIn:owner:]",
    _loadNib,
    "+[NSBundle(NSNibLoading) loadNibFile:...]"
    "+[NSBundle(NSNibLoading) loadNibNamed:owner:]",
    _main,
    start
)
(
    __NSAPDataReleaseToOffset,
    "-[NSAutoreleasePool release]",
    "+[NSBundle(NSNibLoading) loadNibFile:...]",
    "+[NSBundle(NSNibLoading) loadNibNamed:owner:]",
    _main,
    start
)
(
    "-[NSConcreteMutableDictionary release]",
    __NSAPDataReleaseToOffset,
    "-[NSAutoreleasePool release]",
    "+[NSBundle(NSNibLoading) loadNibFile:...]",
    "+[NSBundle(NSNibLoading) loadNibNamed:owner:]",
    _main,
    start
)
```

Keeping Memory Allocation Statistics

Another command, **AnalyzeAllocation**, lets you look at memory allocation after your program has finished executing. To use **AnalyzeAllocation**:

1. Set this environment variable:

```
% setenv NSKeepAllocationStatistics YES
```

The **NSKeepAllocationStatistics** variable tells your program to record information about memory allocation in a file named **/tmp/alloc_stats**_*name_pid*.

2. Run a specific task in your application.The allocation statistics file becomes very large very quickly, so it is important not to run too much of your program at once with **NSKeepAllocationStatistics** turned on.
3. Turn off the environment variable:

```
% unsetenv NSKeepAllocationStatistics
```

4. Perform this command in a Terminal window:

```
% AnalyzeAllocation -v /tmp/alloc_stats_name_pid
```

Related Concept: ;DebuggingConcepts.rtfd;linkMarkername IgnoringAutoreleaseErrors;, Ignoring Autorelease Errors

Related Concept: ;DebuggingConcepts.rtfd;linkMarkername CommonAutoreleaseMistakes;, Common Autorelease Mistakes