iOS面试题：Autoreleasepool所使用的数据结构是什么？AutoreleasePoolPage结构体了解么？

iOS猿_员 关注
3 2019.04.20 15:54:36 字数 913 阅读 6,243


```
每创建一个池子，会在首部创建一个 哨兵 对象,作为标记

最外层池子的顶端会有一个next指针。当链表容量满了，就会在链表的顶端，并指向下一张表。

```

Autorelease对象什么时候释放？

这个问题拿来做面试题，问过很多人，没有几个能答对的。很多答案都是"当前作用域大括号结束时释放"，显然木有正确理解Autorelease机制。

在没有手加Autorelease Pool的情况下，Autorelease对象是在当前的runloop迭代结束时释放的，而它能够释放的原因是系统在每个runloop迭代中都加入了自动释放池Push和Pop

例子：

```
__weak id reference = nil;
- (void)viewDidLoad {
    [super viewDidLoad];    NSString *str = [NSString stringWithFormat:@"sunnyxx"];
    reference = str;
}
- (void)viewWillAppear:(BOOL)animated {
    [super viewWillAppear:animated];
    NSLog(@"%@", reference);
    // Console: sunnyxx
}
- (void)viewDidAppear:(BOOL)animated {
    [super viewDidAppear:animated];
    NSLog(@"%@", reference);
    // Console: (null)
}

```

当然，我们也可以手动干预Autorelease对象的释放时机：

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    @autoreleasepool {        NSString *str = [NSString stringWithFormat:@"sunnyxx"];
    }    NSLog(@"%@", str);
// Console: (null)
}

```

Autorelease原理

AutoreleasePoolPage

ARC下，我们使用@autoreleasepool{}来使用一个AutoreleasePool，随后编译器将其改写成下面的样子：

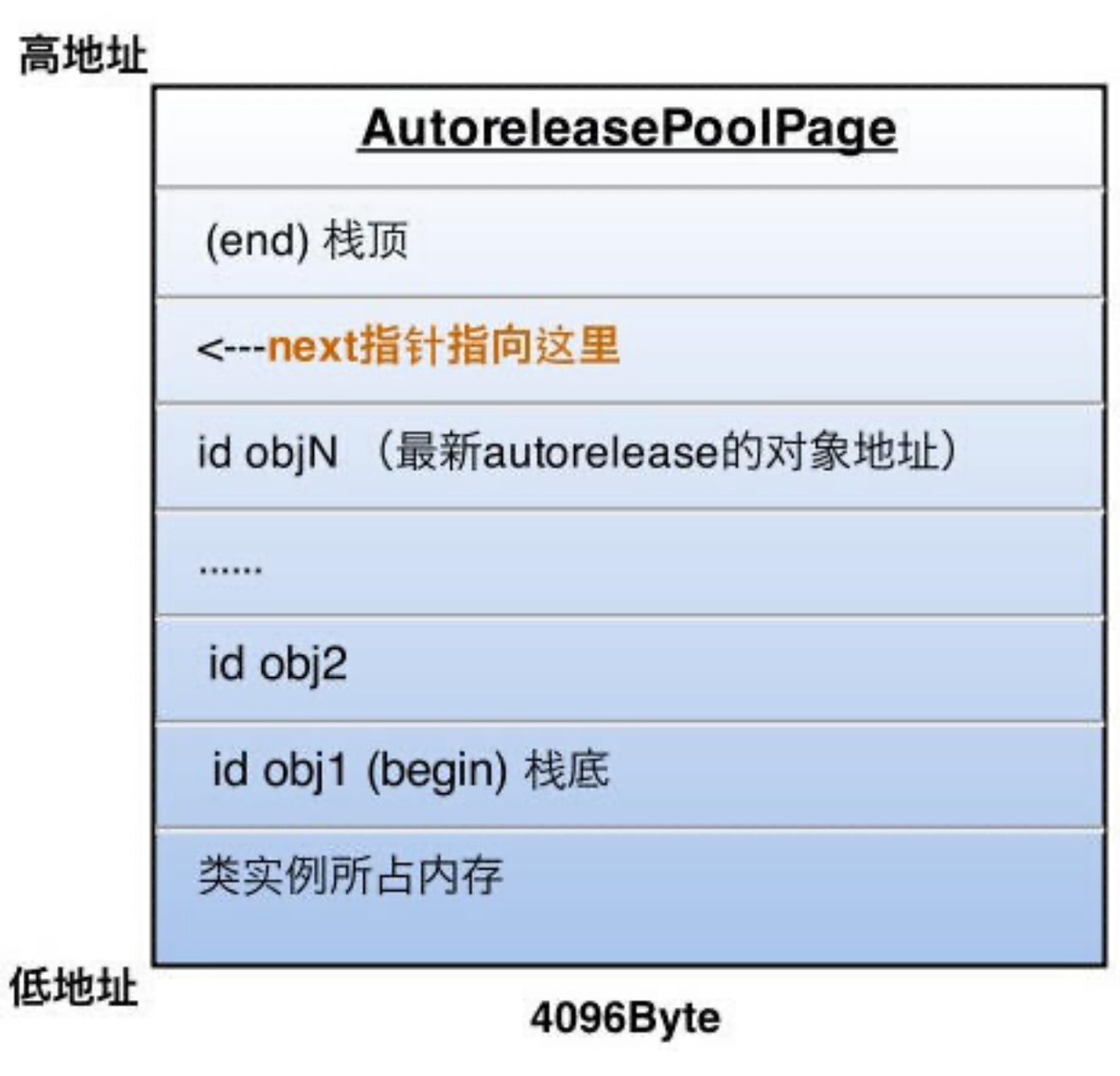
```
void *context = objc_autoreleasePoolPush();
// {}中的代码objc_autoreleasePoolPop(context);

```

而这两个函数都是对AutoreleasePoolPage的简单封装，所以自动释放机制的核心就在于这个类。

AutoreleasePoolPage是一个C++实现的类

- AutoreleasePool并没有单独的结构，而是由若干个`AutoreleasePoolPage以双向链表`的形式组合而成（分别对应结构中的parent指针和child指针）。
- AutoreleasePool是按`线程一一对应`的（结构中的thread指针指向当前线程）。
- AutoreleasePoolPage每个对象会开辟4096字节内存（也就是虚拟内存一页的大小），除了上面的实例变量所占空间，剩下的空间全部用来储存autorelease对象的`地址`。
- 上面的id *next指针作为游标指向栈顶最新add进来的autorelease对象的`下一个位置`。
- 一个AutoreleasePoolPage的空间被占满时，会新建一个AutoreleasePoolPage对象，连接链表，后来的autorelease对象在新的page加入。

所以，若当前线程中只有一个AutoreleasePoolPage对象，并记录了很多autorelease对象地址时内存如下图：

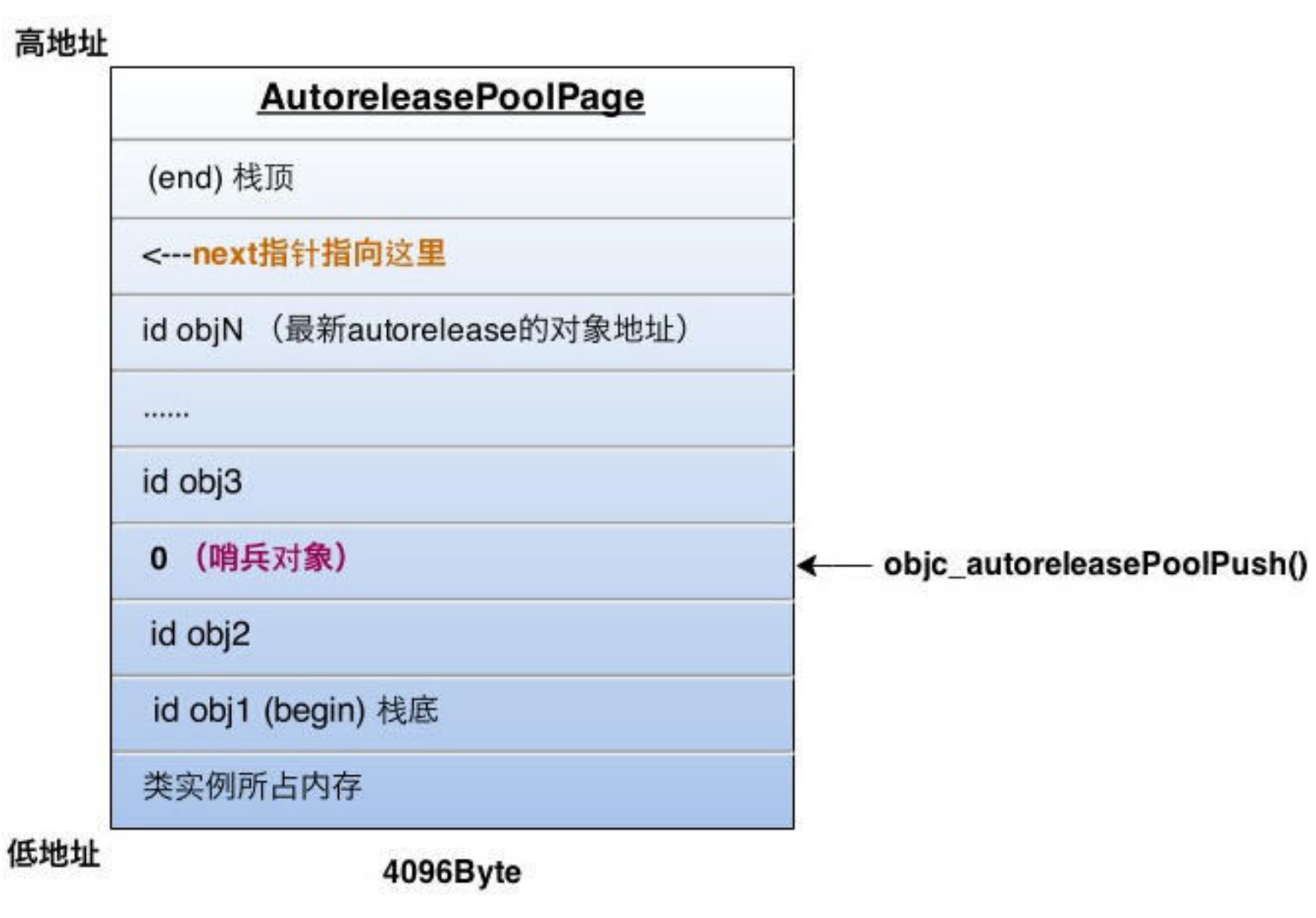


图中的情况，这一页再加入一个autorelease对象就要满了（也就是next指针马上指向栈顶），这时就要执行上面说的操作，建立下一页page对象，与这一页链表连接完成后，新page的next指针被初始化在栈底（begin的位置），然后继续向栈顶添加新对象。

所以，向一个对象发送- autorelease消息，就是将这个对象加入到当前AutoreleasePoolPage的栈顶next指针指向的位置

释放时刻

每当进行一次objc_autoreleasePoolPush调用时，runtime向当前的AutoreleasePoolPage中add进一个哨兵对象，值为0（也就是个nil），那么这一个page就变成了下面的样子：



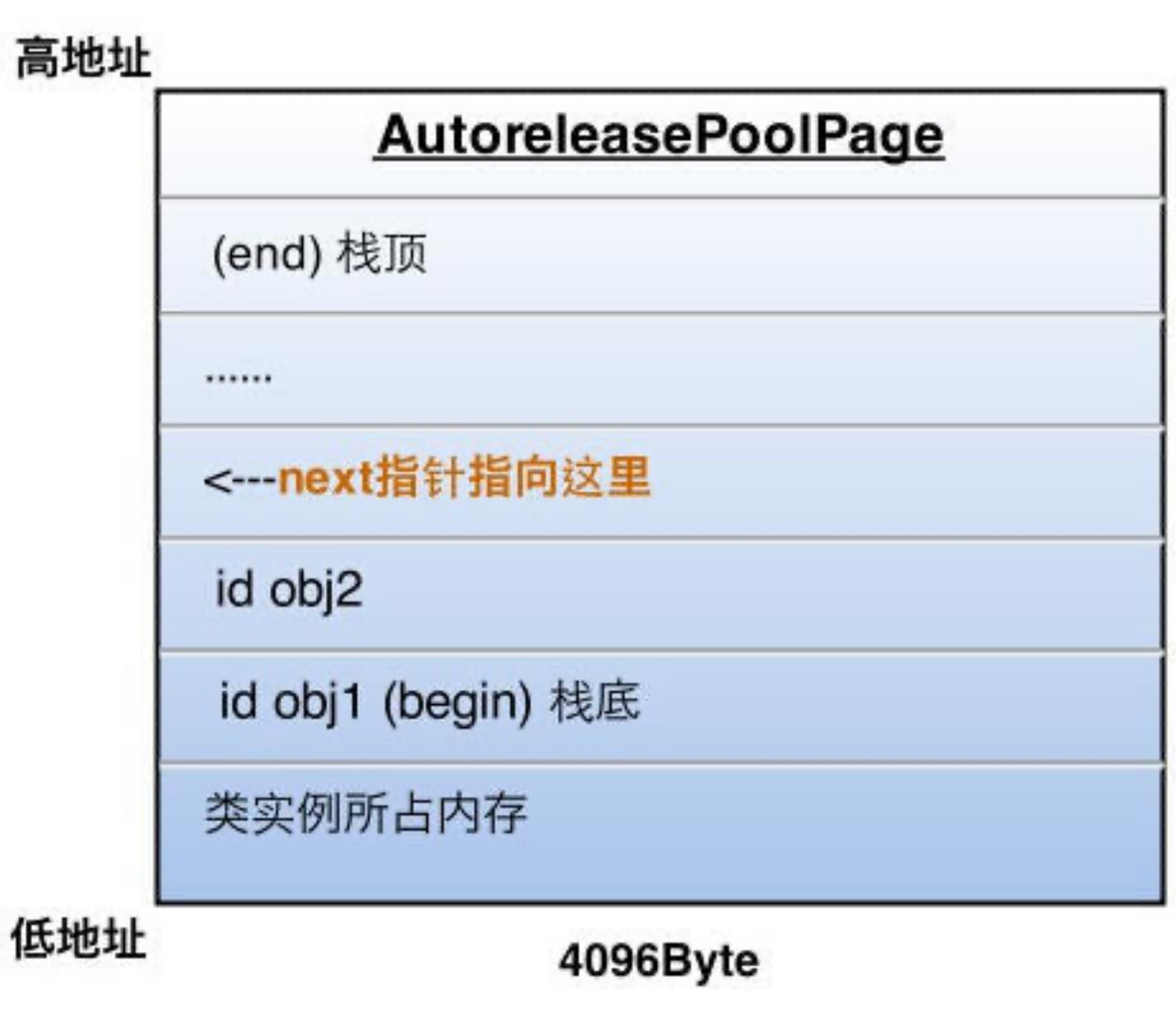
objc_autoreleasePoolPush的返回值正是这个哨兵对象的地址，被objc_autoreleasePoolPop(哨兵对象)作为入参，于是：

1.根据传入的哨兵对象地址找到哨兵对象所处的page

2.在当前page中，将晚于哨兵对象插入的所有autorelease对象都发送一次- release消息，并向回移动next指针到正确位置

3.补充2：从最新加入的对象一直向前清理，可以向前跨越若干个page，直到哨兵所在的page（在一个page中，是从高地址向低地址清理）

刚才的objc_autoreleasePoolPop执行后，最终变成了下面的样子：



嵌套的AutoreleasePool

知道了上面的原理，嵌套的AutoreleasePool就非常简单了，pop的时候总会释放到上次push的位置为止，多层的pool就是多个哨兵对象而已，就像剥洋葱一样，每次一层，互不影响。

更多：iOS面试题合集

全部评论 5 只看作者 按时间倒序 按时间正序

lucky雄
4楼 03.19 17:24

实际上都能打印，因为 taggedPointer技术

1 回复

Lsssssss
3楼 2020.05.27 17:06

- (void)viewDidAppear:(BOOL)animated {
[super viewDidAppear:animated];
NSLog(@"%@", reference);
// Console: (null)
}
这个并没有null啊

赞 回复

可可_running
2楼 2019.04.27 19:22

autoreleasePool释放时机应该是runloop每次运行结束和开始休眠之前都会释放一次~

2 回复

D了个Y
2020.05.09 08:27

休眠前会释放旧池子，并创建新池子。runloop Exit时，释放全部池子

回复

可可_running
2020.05.13 14:51

@D了个Y 对，这样就形成一个环，想法很牛～

回复

添加新评论

被以下专题收入，发现更多相似内容