前言

最近在群里看到有人提问：两个类A，B,为什么在类A的方法里可以创建B对象，调用B类对象的方法，不是说load方法是在类被加载时调用的方法吗。那么B的load方法没调用前类B不是没被加载吗，怎么能调用。

当然这里是提问者本身对load方法调用时机的误解，当时我对这块理解也不是很到位，平时对于runtime也只是一般的使用:动态绑定属性，Swizzle方法实现等，所以仅仅是引用了官方的对load方法描述来回答这个问题，“在load方法中你可以调用其他类的方法，但是并不代表该类的load方法被执行过”。

> In a custom implementation of `load` you can therefore safely message other unrelated classes from the same image, but any `load` methods implemented by those classes may not have run yet.

对于这里，回家后也一直挂在心上，load方法到底是何时被调用的，Class又是何时被加载的呢，他们之间的顺序又是什么呢?

从runtime初始化开始

一切都要从runtime初始化开始，_objc_init方法是runtime初始化的方法，可在开源代码中查看

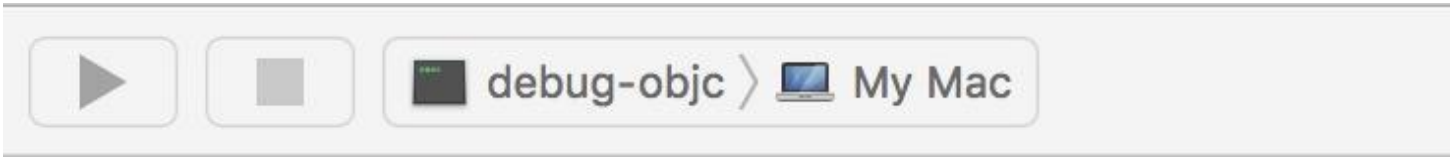
```
void _objc_init(void)
{
    static bool initialized = false;
    if (initialized) return;
    initialized = true;

    // fixme defer initialization until an objc-using image is found?
    environ_init();
    tls_init();
    static_init();
    lock_init();
    exception_init();

    _dyld_objc_notify_register(&map_images, load_images, unmap_image);
}
```

其中，最后两行注册了两个通知，分别交给map_images，load_image方法来处理。
map_images 主要是在image加载进内容后对其二进制内容进行解析，初始化里面的类的结构

等。
load_images 主要是调用call_load_methods。按照继承层次依次调用Class的+load方法然后再是Category的+load方法。

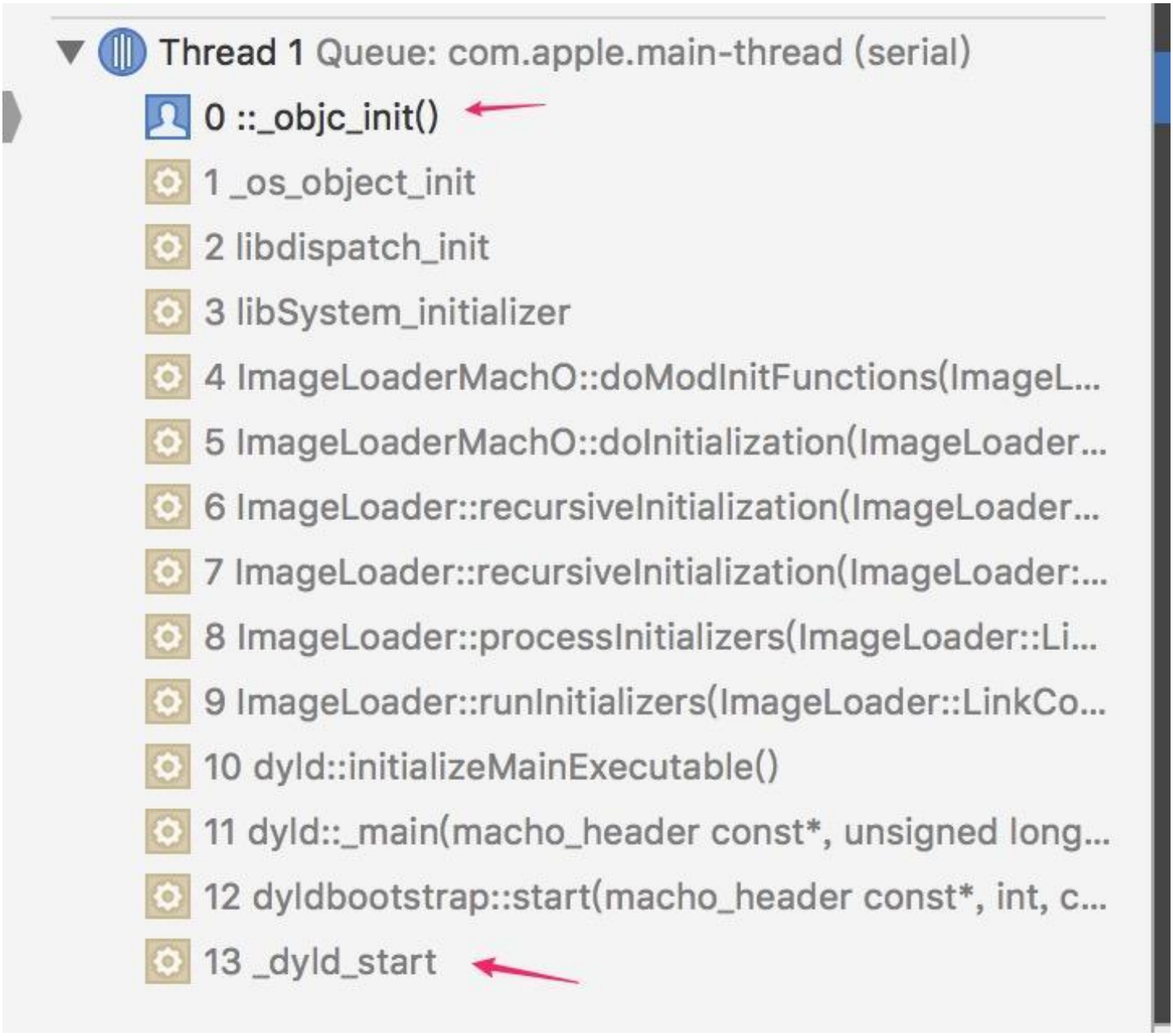
在demo里map_images和load_image方法中都打下断点

这里就只列举map_images方法断点截图
运行后名为"debug-objc"的Target



发现:项目中的对debug-objc镜像文件(tagert编译运行后的可执行文件)来说，系统在初始化runtime后，main函数启动前对其进行bind，此时会调用map_images方法对其进行类结构初始化等处理，然后再调用在load_image方法调用所有的load方法(下载demo运行试试)，所以在runtime调用项目中所有的load方法时，我们所有的类的结构已经加载进内存并初始化了，此时在load方法中可以创建任何类的实例并给他们发送消息。

我们再在_objc_init 里打个断点:



发现调用栈是由_dyld_start开始的，这里引申到dyld(Apple动态链接器)，在程序启动后当系统做好了一些初始化的准备，会将后面的事甩锅给dyld负责，dyld会将程序依赖动态库加载进内存初始化，但他干不来所有事，所以他就初始化了runtime小弟来帮他将所有即将使用的库的二进制数据进行解析并初始化里面类的结构，关于dyld更多介绍可见dyld的。

总结:

Runtime 调用项目中所有的load方法时，所有的类的结构已经初始化了,此时在load方法中可以使用任何类创建实例并给他们发送消息。

#Tips:

1.由于load方法在main函数前被调用，所以在load方法中尽量避免费时的操作，因为这些操作都将导致程序启动时间的延长。

2.Runtime中call_load_methods里是通过load方法的地址直接调用的load方法，而不是通过消息

机制来调用的。可见源码:

```
static void call_class_loads(void)
{
//调用所有类的load方法
....
for (i = 0; i < used; i++) {
Class cls = classes[i].cls;
load_method_t load_method = (load_method_t)classes[i].method;
...
(*load_method)(cls, SEL_load);
}
...
}
```

结尾

客官，点个赞再走呗，哦对了，记得带(guan)上(zhu)门(wo)