第9节课内容总结

_objc_init里面的初始化操作。

1. environ_init()：环境变量的初始化
2. tls_init：创建线程的析构函数
3. static_init：运行C++静态构造函数
4. runtime_init：分类表初始化，类表初始化
5. cache_t::init()：缓存的初始化
6. _imp_implementationWithBlock_init：关于macOS的相关操作。
7. didCallDyldNotifyRegister：标识对_dyld_objc_notify_register的调用已完成。

load_images

作用：执行类和分类的`load`方法。

load方法总结

1.当父类和子类都实现load函数时,父类的load方法执行顺序要优先于子类
2.当一个类未实现load方法时,不会调用父类load方法
3.类中的load方法执行顺序要优先于分类(Category)
4.load方法使用了锁,所以是线程安全的。
5.当有多个类别(Category)都实现了load方法,这几个load方法都会执行,但执行顺序不确定(其执行顺序与类别在Compile Sources中出现的顺序一致)
6.当然当有多个不同的类的时候,每个类load 执行顺序与其在Compile Sources出现的顺序一致

map_images

作用：进行类的初始化。

关键函数 read_images 流程

1: 加载所有类到类的gdb_objc_realized_classes表中。
2: 对所有类做重映射。
3: 将所有SEL都注册到namedSelectors表中。
4: 修复函数指针遗留。

5: 将所有Protocol都添加到protocol_map表中。
6: 对所有Protocol做重映射。
7: 初始化所有非懒加载的类，进行rw、ro等操作。
8：遍历已标记的懒加载的类，并做初始化操作。
9：处理所有Category，包括Class和Meta Class。
10：初始化所有未初始化的类。

非懒加载类的初始化

关键函数：`realizeClassWithoutSwift`
关键代码：

```
//给rw开辟内存空间，然后将ro的数据“拷贝”到rw里面。
rw = objc::zalloc<class_rw_t>();
rw->set_ro(ro);
rw->flags = RW_REALIZED|RW_REALIZING|isMeta;
cls->setData(rw);
```

```
//递归调用realizeClassWithoutSwift，对父类和元类进行初始化
supercls = realizeClassWithoutSwift(remapClass(cls->getSuperclass()), nil);
metacls = realizeClassWithoutSwift(remapClass(cls->ISA()), nil);
```

```
//设置父类，isa指针的初始化
cls->setSuperclass(supercls);
cls->initClassIsa(metacls);
```