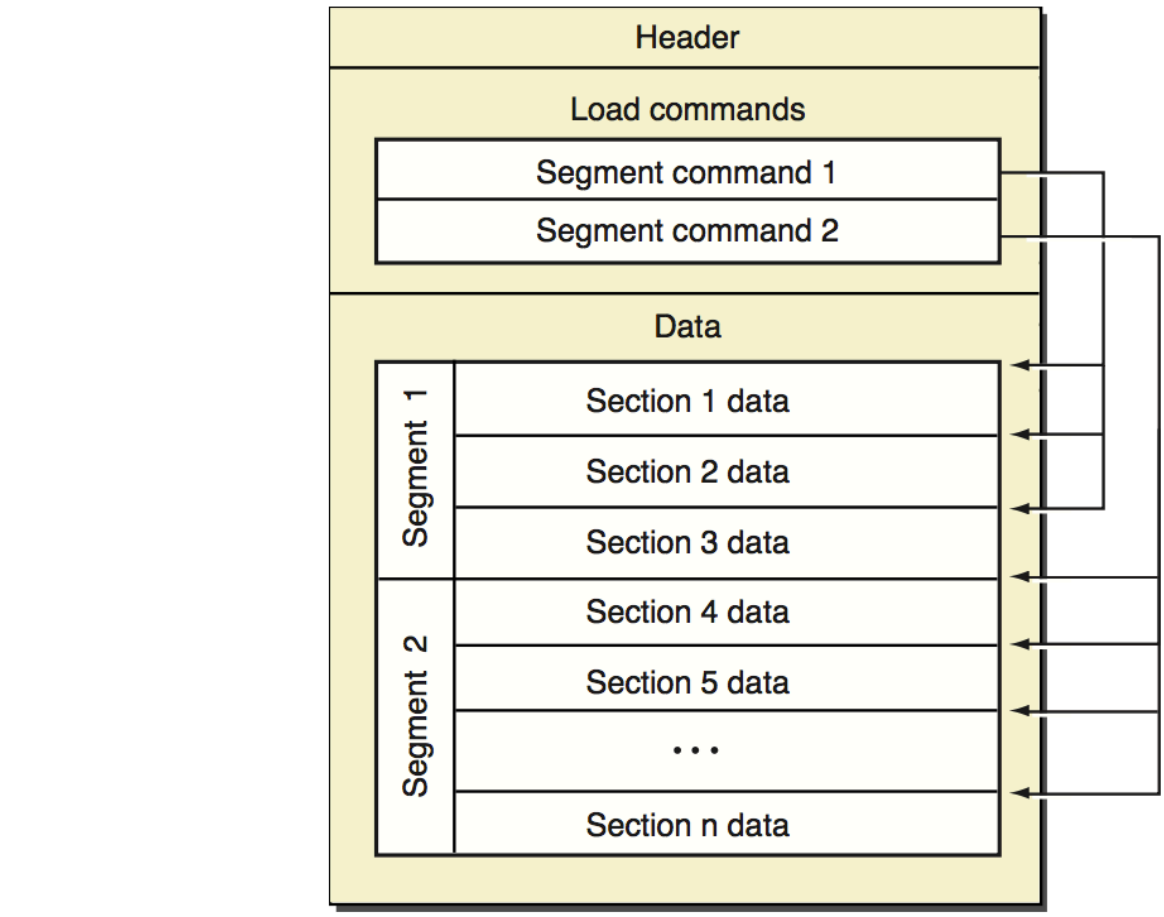
Mach-O 文件格式

基本结构

- Header，每个 Mach-O 文件的开始都是头部，包含了基本文件类型信息，目标架构和说明剩余文件的标志。
- Load Command，头部之后就是一系列长度不定的 load commands，表明了文件的布局和特征，load commands 可以表明：
    - 文件在虚拟内存中的初始化结构
    - 符号表的位置
    - 程序主线程的初始化执行状态
    - 包含被引用符号的共享库的名字
- Segment,在 load commands 之后所有的 Mach-O 文件，都会包含一个或多个 segment 的数据。每个段（segment）包含一个或多个 section。每个 区（section）包含代码或者某些特定类型的数据。每个段定义了一块虚拟内存的区域用来让动态链接器把进程的空间加载到对应的地址。段和区的布局和数量是由 load commands 和文件类型决定的。
- 在用户层链接完成的 Mach-O 文件中，最后一个segment 是 link edit。包含了link edit信息表，包括符号表，字符表等等，用来让动态加载器去链接可执行文件或者 Mach-O bundle依赖的库。



Header 结构和 Load Commands

一个 Mach-O 文件包含了一种架构的代码和数据，头部结构说明了目标架构由内核来保证，PowerPC 架构的不能在 Intel 架构的机器上运行。

可以用工具将多种架构的 Mach-O 文件合并在一个二进制文件中。（包含多种架构的文件不是 Mach-O 文件，是多个 Mach-O 文件的 archive）

segments 和 sections 通常用名字访问，Segment 惯例是两个下划线加上大写的名字（__TEXT），sections的命名是两个下划线加上小写的名字（__text），命名惯例是一个标准但是没有工具保证。

`otool -h Mach-O_file` 可以打印 header

例如：

magic	cputype	cpusubtype	caps	filetype	ncmds	sizeofcmds	flags
MH_MAGIC_64	ARM64	ALL	0x00	EXECUTE	69	7232	0x00

`otool -v -l Mach-O_file` 可以打印 Load Commands

例如

```
Load command 0
      cmd LC_SEGMENT_64
  cmdsize 72
  segname __PAGEZERO
   vmaddr 0x0000000000000000
   vmsize 0x0000000100000000
  fileoff 0
 filesize 0
  maxprot ---
  initprot ---
   nsects 0
    flags (none)
Load command 1
      cmd LC_SEGMENT_64
  cmdsize 952
  segname __TEXT
   vmaddr 0x0000000100000000
   vmsize 0x000000000109c000
  fileoff 0
 filesize 17416192
  maxprot r-x
  initprot r-x
   nsects 11
    flags (none)
Load command 64
      cmd LC_LOAD_DYLIB
  cmdsize 56
     name @rpath/libswiftUIKit.dylib (offset 24)
     time stamp 2 Thu Jan  1 08:00:02 1970
  current version 800.10.13
compatibility version 1.0.0
```

Segments

Segments 在 Mach-O 文件中定义一些数据、地址和内存保护属性，在动态连接器加载程序的时候这些数据都会被映射到虚拟内存中，一个 segment 包含0或多个 section。

Segments 可以比在磁盘中的实际大小声明一块更大的内存，比如 __PAGEZERO 段在链接器为PowerPC生成可执行文件时在磁盘的大小为0，但是在虚拟内存中会有一个page的大小。应该 __PAGEZERO 没有数据，所以没有必要在可执行文件中红占用空间。

为了压缩空间在中间目标文件中只有一个segment，这个 segment 没有名字，包含的所有 section 在最终的目标文件中会被分成不同的 segment， 根据 section 中的 segname 分成不同的

segment。

为了最优的性能，segment 应该和虚拟内存的边界对齐（在PowerPC和X86处理器上是4096bytes），计算 Segment 的大小是把所有包含的所有 section 大小加上，然后对齐到虚拟内存页大小（4096bytes），所以 segment 的大小是4kb的倍数。

OS X 可执行文件一般包含的的 segment 有：

- __PAGEZERO，可执行文件的第一个 segment，空指针陷阱段，用于捕捉对空指针的引用立即 crash，虚拟内存的第一页。
- __TEXT，包含可执行代码和只读的数据，为了能够让内核直接把它从可执行文件映射到共享内存中，静态链接器把这个段的虚拟内存权限设置为不可写。当这个 segment 加载到内存中之后可以在多个进程中共享使用（主要是framework，bundle 和共享库，也可以在一个可执行文件的多进程拷贝执行中使用），由于这个 segment 是不可写的，所以不需要写回磁盘。当内核需要释放物理内存的时候，只需要简单释放一个或多个 __TEXT 页，在需要时再从磁盘中读取。
- __DATA，包含可写数据，静态链接器将这个段的虚拟内存权限设置成可读可写，因为可写的原因所以framework和共享库的__DATA段会在每个链接了这个库的进程都拷贝一份。当像__DATA段一样可写可读的段的内存页，内核会把这些内存页标记成copy-on-write。当有进场需要往页里面写数据时，会先拷贝一份作为这个进程私有的再修改。
- __OBJC，包含了 Objective-C 运行时支持库需要的数据。
- __LINKEDIT，包含动态链接器所需要的原始数据，比如符号，字符串，重定位表条目等。

Sections

`otool -s <segname> <sectname>` 可以打印 Mach-O 文件中的 section 内容

__TEXT 段的 sections

Segment and section name	Contents
__TEXT, __text	可执行的机器码
__TEXT, __cstring	c字符串常量
__TEXT, __picssymbol_-stub	Position-independent indirect symbol stubs
__TEXT, __symbol_stub	Indirect symbol stubs.
__TEXT, __const	初始化的 const 变量
__TEXT, __literal4	4byte的字面量
__TEXT, __literal8	8byte的字面量

__DATA 段的 sections

Segment and section name	Contents
__DATA, __data	初始化的可变变量
__DATA, __la_symbol_ptr	lazy 符号指针
__DATA, __nl_symbol_ptr	non-lazy 符号指针
__DATA, __dyld	动态链接器使用的占位 section
__DATA, __const	初始化的重定位 constant 变量
__DATA, __mod_init_func	模块初始化函数，C++编译器会把静态构造函数放在这
__DATA, __mod_term_func	模块终止函数
__DATA, __bss	未初始化的静态变量
__DATA, __common	未初始化的全局符号定义