dex文件经过安装时进行提前缓存，会生成.art、.oat两个文件，oat是一个android定制的elf文件，原始dex也保存在其中。8.0后，dex单独保存到.vdex文件中。art文件类似于一个内存映像，缓存常用的ArtField、ArtMethod、DexCache等内容，加载后可直接使用，避免解析耗时。

art文件格式介绍

以boot.art为例，它分为Image Section和Bitmap Section区域。每个Section在文件中的偏移量和大小由ImageSection类来描述。

主要Section介绍：

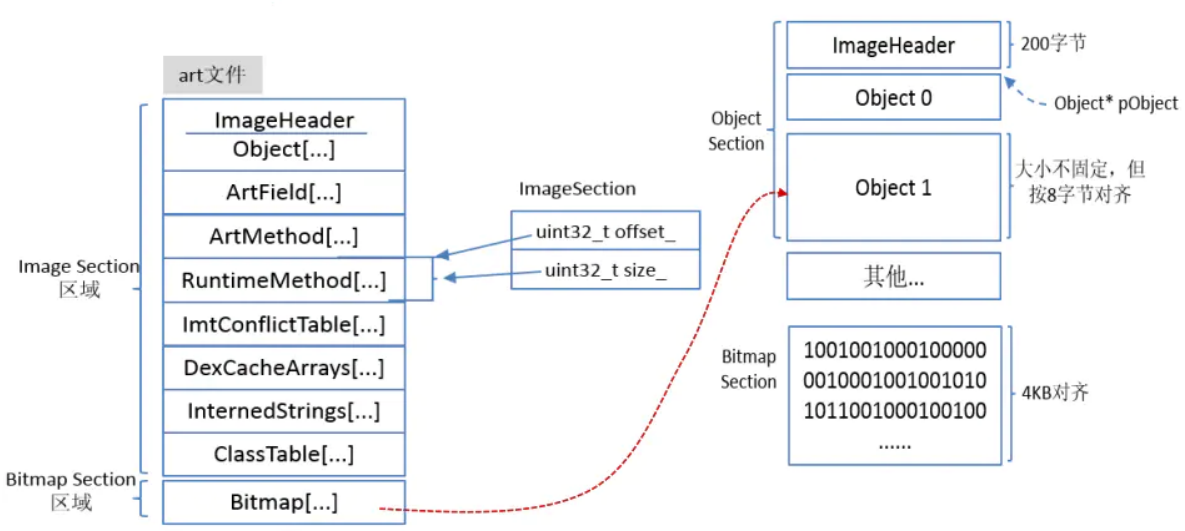
- **Object Section：**存储的一个个的mirror Object对象。需要这个Object对象时，从art文件里读出来(反序列化)即可。**Object Section前200个字节保存的是art文件头ImageHeader内容。**
- **ArtField和ArtMethod Section：**ArtField和ArtMethod对象的内容。
- **DexCacheArrays Section：**DexCache有关，通过DexCacheArraysLayout将一个DexCache对象所关联的GcRoot数组、ArtMethod*数组*、*ArtFiled*数组、GcRoot数组按顺序存储在该Section中。
- **ClassTable Section：**存储的是一个ClassTable对象的内容。

Bitmap Section：

Bitmap区域是一个位图，用于描述Object Section里各个Object的地址，以8字节对齐。如果一个比特位的值为1，则它指向Object Section中的一个Object对象。

假设Object存储的基地址是0x70000000，如果位图第N个比特位为1，那么这个比特位指向的Object对象地址为0x70000000+N*8。

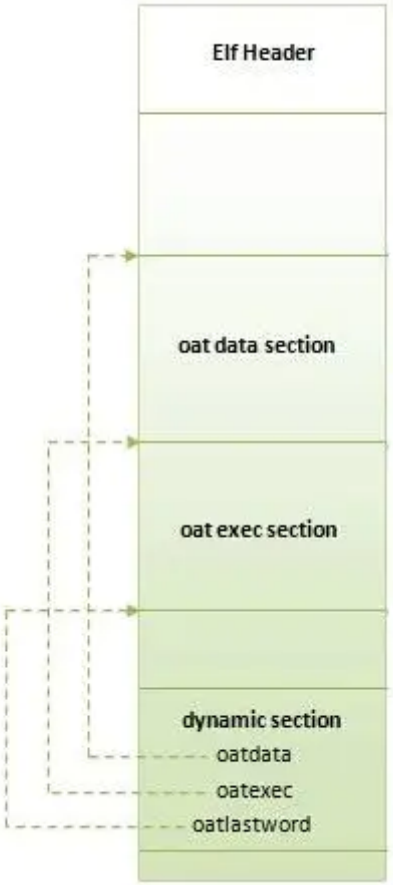
art/runtime/image.h：



oat文件格式介绍

oat文件本质上是一个ELF文件，它将OAT文件格式内嵌在ELF文件里。

在oat文件的dymanic section中，导出了三个符号oatdata、oatexec和oatlastword，分别用来描述oatdata和oatexec段加载到内存后的起止地址。

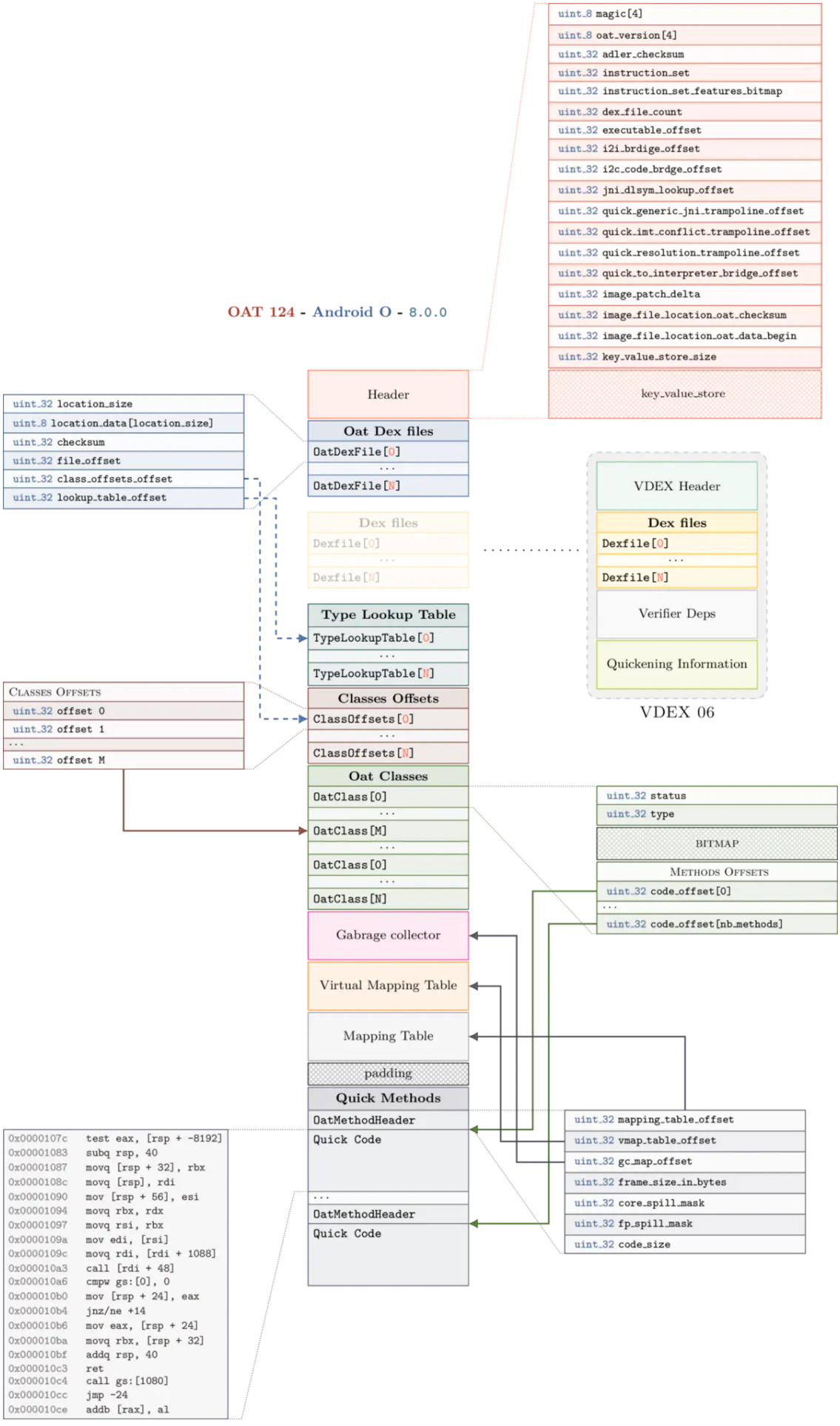


oatdata段中，包含原dex文件的完整内容(8.0后在.vdex文件)，dex文件里面的类方法所对应的本地机器指令保存在oatexec段中。

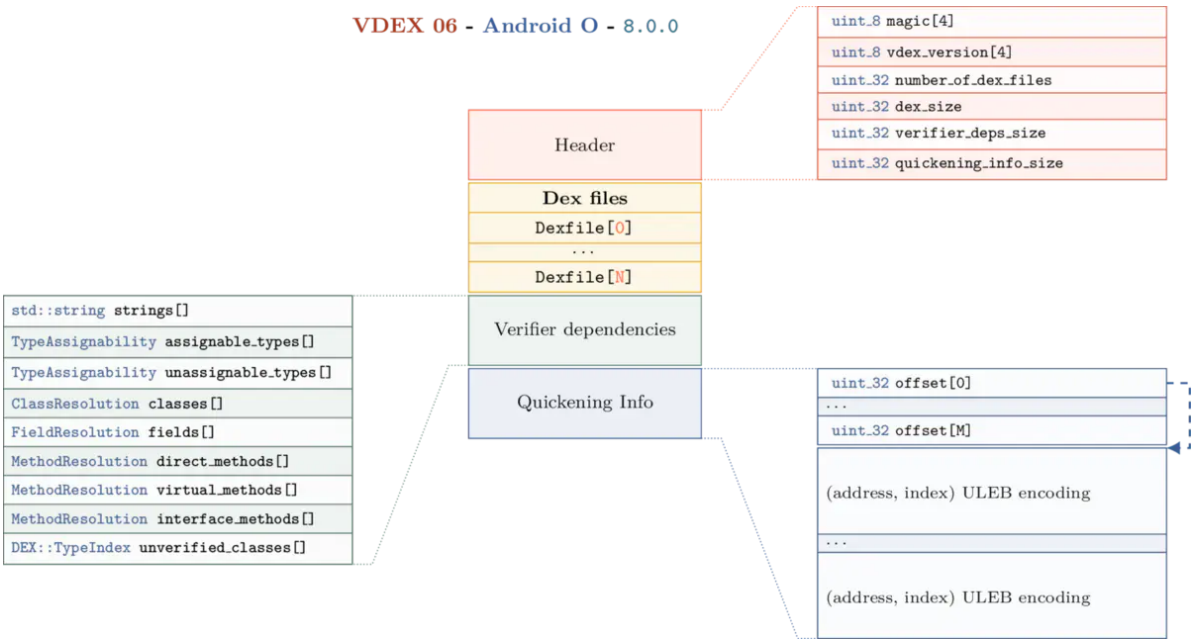
OAT主要内容介绍：

- **OatHeader**：头信息，vedx的加载地址也在这里记录。
- **OatDexFile**：包含一到多个OatDexFile，写入时借助oat_writer.cc OatWriter::OatDexFile类，而读取时转换为oat_file.h中定义的OatDexFile类实例。
- **DexFile**：包含一个到多个DexFile项(8.0后独立到vdex文件中)。
- **ClassOffsets**：数组，与dex文件一一对应。ClassOffsets[x]代表第x个dex文件，ClassOffsets[x][y]则代表第x个dex文件中的第y个类的信息。
- **OatClass**：每个类对应一个OatClass，ClassOffsets[x][y]表示第x个dex中第y个class信息，指向oatclass[y]。OatClass中method_offset*是一个数组，只有一个成员变量code_offset*指向OatQuickMethodHeader中的code_数组。
- OatMethod：

  包含一个到多个OatQuickMethodHeader元素。OatQuickMethodHeader中的code_数组指向机器码。


OAT 124 - Android O - 8.0.0
uint_8 magic[4]
uint_8 oat_version[4]
uint_32 adler_checksum
uint_32 instruction_set
uint_32 instruction_set_features_bitmap
uint_32 dex_file_count
uint_32 executable_offset
uint_32 i2i_brdige_offset
uint_32 i2c_code_brdge_offset
uint_32 jni_dlsym_lookup_offset
uint_32 quick_generic_jni_trampoline_offset
uint_32 quick_imt_conflict_trampoline_offset
uint_32 quick_resolution_trampoline_offset
uint_32 quick_to_interpreter_bridge_offset
uint_32 image_patch_delta
uint_32 image_file_location_oat_checksum
uint_32 image_file_location_oat_data_begin
uint_32 key_value_store_size
key_value_store
Header
uint_32 location_size
uint_8 location_data[location_size]
uint_32 checksum
uint_32 file_offset
uint_32 class_offsets_offset
uint_32 lookup_table_offset
Oat Dex files
OatDexFile[0]
OatDexFile[N]
Dex files
Dexfile[0]
Dexfile[N]
VDEX Header
Dex files
Dexfile[0]
Dexfile[N]
Verifier Deps
Quickening Information
VDEX 06
Type Lookup Table
TypeLookupTable[0]
TypeLookupTable[N]
Classes Offsets
uint_32 offset 0
uint_32 offset 1
uint_32 offset M
Classes Offsets
ClassOffsets[0]
ClassOffsets[N]
Oat Classes
OatClass[0]
OatClass[M]
OatClass[0]
OatClass[N]
uint_32 status
uint_32 type
BITMAP
Methods Offsets
uint_32 code_offset[0]
uint_32 code_offset[nb_methods]
Gabrage collector
Virtual Mapping Table
Mapping Table
padding
Quick Methods
OatMethodHeader
Quick Code
OatMethodHeader
Quick Code
uint_32 mapping_table_offset
uint_32 vmap_table_offset
uint_32 gc_map_offset
uint_32 frame_size_in_bytes
uint_32 core_spill_mask
uint_32 fp_spill_mask
uint_32 code_size
0x0000107c test eax, [rsp + -8192]
0x00001083 subq rsp, 40
0x00001087 movq [rsp + 32], rbx
0x0000108c movq [rsp], rdi
0x00001090 mov [rsp + 56], esi
0x00001094 movq rbx, rdx
0x00001097 movq rsi, rbx
0x0000109a mov edi, [rsi]
0x0000109c movq rdi, [rdi + 1088]
0x000010a3 call [rdi + 48]
0x000010a6 cmpw gs:[0], 0
0x000010b0 mov [rsp + 24], eax
0x000010b4 jnz/ne +14
0x000010b6 mov eax, [rsp + 24]
0x000010ba movq rbx, [rsp + 32]
0x000010bf addq rsp, 40
0x000010c3 ret
0x000010c4 call gs:[1080]
0x000010cc jmp -24
0x000010ce addb [rax], al

vdex格式：



art、oat、vdex三个文件的关系

boot.art、boot.oat、boot.vdex三者是一体的，相互依赖。

- ImageHeader中有成员变量关联到oat文件。oat_file_begin*指向oat文件加载到内存的地址，oat_data_begin*指向符号oatdata的值，oat_data_end_指向符号oatlastword的值。
- art文件里的ArtMethod对象的entry_point_from_quick_compiled_code*指向位于oat文件对应的code*数组。

zygote启动创建Heap的时候，会加载boot.art，然后加载boot.oat，再然后加载boot.vdex。
调用流程如下：

```
Heap::Heap()
    space::ImageSpace::LoadBootImage()
        ImageSpace::CreateBootImage()
            ImageSpaceLoader::Load()
                ImageSpaceLoader::Init()
                    LoadImageFile()//加载art文件
                        MemMap::MapFileAtAddress(..., image_filename);
                    OpenOatFile()
                        OatFile::Open()
                            OatFileBase::OpenOatFile<ElfOatFile>(..., vdex_fd)//
加载oat文件
                                LoadVdex()
                                    VdexFile::OpenAtAddress()//加载vdex文件
                                        OpenAllDexFiles()//加载dex文件
```

提取dex

dextra
vdexExtractor
compact_dex_converter
Android 9（Pie）推出了一种新型Dex文件，即Compact Dex（Cdex）。Cdex是一种ART内部文件格式，它压缩各种Dex数据结构（例如方法头）并对多索引文件中的常见数据blob（例如字符串）进行重

复数据删除。来自输入应用程序的Dex文件的重复数据删除数据存储在Vdex容器的共享部分中。
由于Vdex容器存储的是Cdex文件而不是标准的Dex，因此需要借助compact_dex_converter工具来实现提取dex。

安装提取工具步骤(ubuntu)：

1. git clone https://github.com/anestisb/vdexExtractor.git
2. ./make.sh
3. 下载compact_dex_converter解压到vdexExtractor/bin下
4. 下载解压dextra即可直接使用

提取：(工具并不完美，提取dex后有些不能正常jadx反编译)

1. android5、6、7:./dextra.ELF64 -dextract boot-framework.oat
2. android8：./bin/vdexExtractor -i mydex/8/services.vdex -o mydex/out8/，"failed to unquicken Dex file"则加上--no-unquicken
3. android9：./bin/vdexExtractor -i mydex/9/services.vdex -o mydex/out9/
   ./bin/compact_dex_converters -w mydex/out9/ mydex/out9/services_classes.cdex
   将生成的cdex.new改名为xxx.dex即可使用jadx反编译。(如果无法反编译，升级jadx或者修改dex头版本信息039-->035)