

# ART虚拟机，了解一下

邓凡平

民生银行总行信息科技部  
民生科技有限公司

# 大纲

1. 研究环境介绍
2. 信息的转换和表达
3. Java方法的执行
4. 讨论

# 1

## 研究环境介绍



# 研究环境介绍——了解你的“敌人”



## 了解你的“敌人”



### Android 7.0 Nougat源码

- 源码路径：AOSP/art
- C++代码：1071个文件。
  - ✓ .cc文件共236744行（不包含注释，空行）
  - ✓ .h文件共74710行
- 汇编代码：1704个文件。
  - ✓ 代码行19955行（去除注释，空行及mterp/out）
  - ✓ 覆盖x86，arm、mips 32和64位共6个平台



# 研究环境介绍——准备好的工具

## 准备好的工具



- Vmware+Ubuntu 14.04
- 自建x86模拟器镜像
- 保留最少量的应用

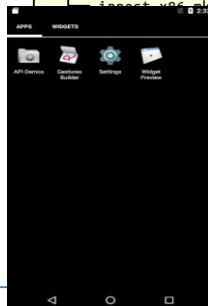
```
innost@innost:~/workspace/aosp/android-7.0$ lunch
You're building on Linux

Lunch menu... pick a combo:
 1. aosp_arm-eng
 2. aosp_arm64-eng
 3. aosp_mips-eng
 4. aosp_mips64-eng
 5. aosp_x86-eng
 6. aosp_x86_64-eng
 7. innost_arm64-userdebug
 8. innost_x86-userdebug

Which would you like? [aosp_arm-eng]
```

## Why x86 emulator?

- 模拟器启动速度快
- 相关资料多
- 汇编代码行数/总代码行数≈6.02%



```
innost@innost:~/workspace/aosp/android-7.0$ tree board
board
├── Android.mk
├── innost_arm64
│   ├── BoardConfig.mk
│   ├── device.mk
│   ├── README.txt
│   ├── system.prop
│   └── innost_x86
│       ├── BoardConfig.mk
│       ├── device.mk
│       ├── README.txt
│       └── sepolicy
│           ├── domain.te
│           ├── healthd.te
│           ├── init.te
│           ├── installd.te
│           └── zygote.te
│               └── system.prop
└── product
    ├── AndroidProducts.mk
    ├── aosp_base.mk
    ├── aosp_base_telephony.mk
    ├── core_base.mk
    ├── core_minimal.mk
    ├── core.mk
    ├── emulator.mk
    ├── full_base.mk
    ├── full_base_telephony.mk
    ├── full_x86.mk
    ├── generic_no_telephony.mk
    ├── innost_arm64_device.mk
    ├── innost_arm64.mk
    └── innost_x86.mk
```

# 2

## 信息的转换和表达

——从java源码到虚拟机内部表示



# java源码的处理流程

.java

```
class Test{  
    public int iField;  
    public Object objField;  
  
    public int myMethod() {  
        return 0;  
    }  
}
```

javac

.class

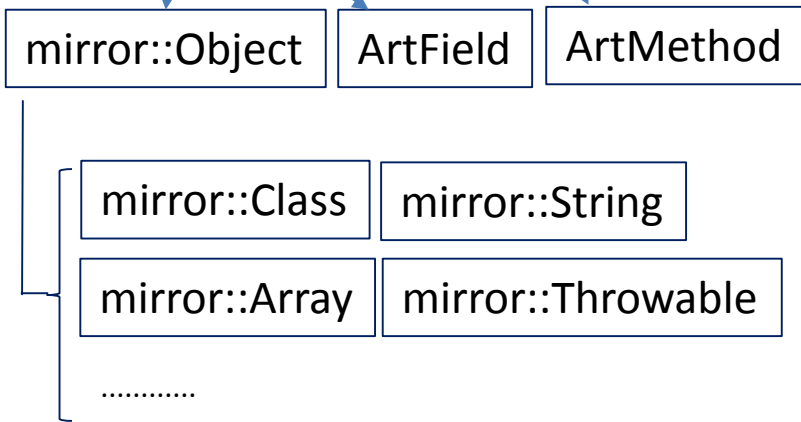
dx

.dex

dex2oat

.oat

.art





# dex文件格式 ( 1 )

.dex文件

header
string_ids
type_ids
proto_ids
field_ids
method_ids
class_defs
data
link_data

```
struct header_item {  
    public ubyte[] magic = new ubyte[8];  
    public uint checksum;  
    public ubyte[] signature = new ubyte[20];  
    public uint file_size;  
    public uint header_size;  
    public uint endian_tag;  
  
    public uint link_size; public uint link_off;  
    public uint map_off;  
  
    public uint string_ids_size; public uint string_ids_off;  
  
    public uint type_ids_size; public uint type_ids_off;  
  
    public uint proto_ids_size; public uint proto_ids_off;  
    public uint field_ids_size; public uint field_ids_off;  
  
    public uint method_ids_size; public uint method_ids_off  
  
    public uint class_defs_size; public uint class_defs_off;  
  
    public uint data_size; public uint data_off;  
}
```





# dex文件格式 ( 2 )



## Class基本信息

## Class的成员变量和成员函数

```

struct class_def {
    uint class_idx;
    uint access_flags;
    uint superclass_idx;
    uint interfaces_off;
    uint source_file_idx;
    uint annotations_off;
    uint class_data_off;
    uint static_values_off;
}

```

```

struct class_data_item{
    uleb128 static_fields_size;
    uleb128 instance_fields_size;
    uleb128 direct_methods_size;
    uleb128 virtual_methods_size;
    encoded_field[] static_fields;
    encoded_field[] instance_fields;
    encoded_method[] direct_methods;
    encoded_method[] virtual_methods;
}

```

dex指令码

## 成员变量和方法信息

## 成员方法的细节

所实现的接口类信息

```

struct type_list {
    uint size;
    type_item[] list;
}
struct type_item {
    ushort type_idx;
}

```

```

struct encoded_field {
    uleb128 field_idx_diff;
    uleb128 access_flags;
}
struct encoded_method {
    uleb128 method_idx_diff;
    uleb128 access_flags;
    uleb128 code_off;
}

```

```

struct code_item {
    ushort registers_size;
    ushort ins_size;
    ushort outs_size;
    ushort tries_size;
    uint debug_info_off;
    uint insns_size;
    ushort[] insns;
    ushort padding;
    try_item[] tries;
    encoded_catch_handler_list handlers;
}

```



# oat文件格式 ( 1 )

.oat文件是一个定制化的ELF文件

Section Headers:

[Nr]	Name	Type	Addr	Off	Size	ES	Flg	Lk	Inf	Al
[ 0]		NULL	00000000	000000	000000	00		0	0	0
[ 1]	.rodata	PROGBITS	713b7000	001000	273e000	00	A	0	0	4096
[ 2]	.text	PROGBITS	73af5000	273f000	18ace5f	00	AX	0	0	4096
[ 3]	.dynstr	STRTAB	753a2000	3fec000	000037	00	A	0	0	4096
[ 4]	.dynsym	DYNSYM	753a2038	3fec038	000040	10	A	3	0	4
[ 5]	.hash	HASH	753a2078	3fec078	00001c	04	A	4	0	4
[ 6]	.dynamic	DYNAMIC	753a3000	3fed000	000038	08	A	3	0	4096
[ 7]	.shstrtab	STRTAB	00000000	3fee000	000038	00		0	0	4096

Symbol table '.dynsym' contains 4 entries:

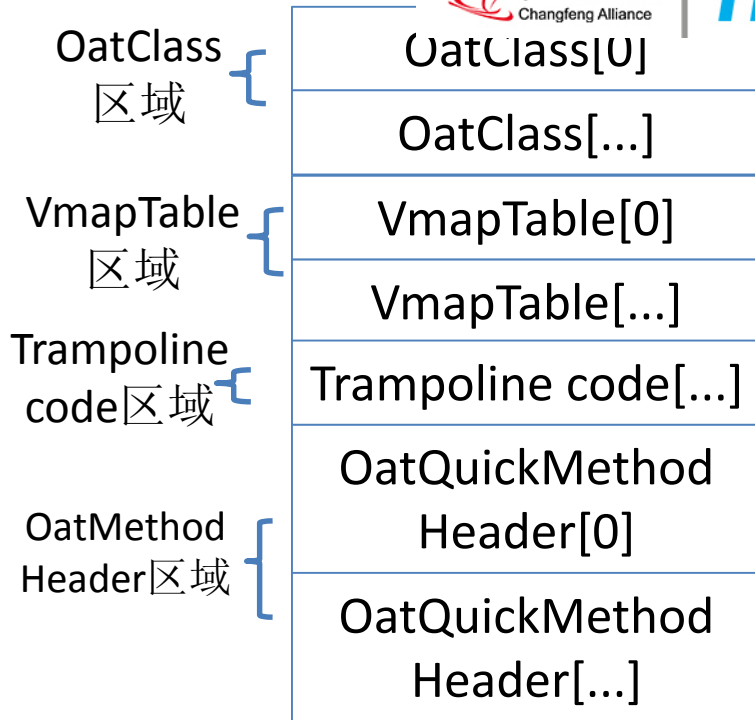
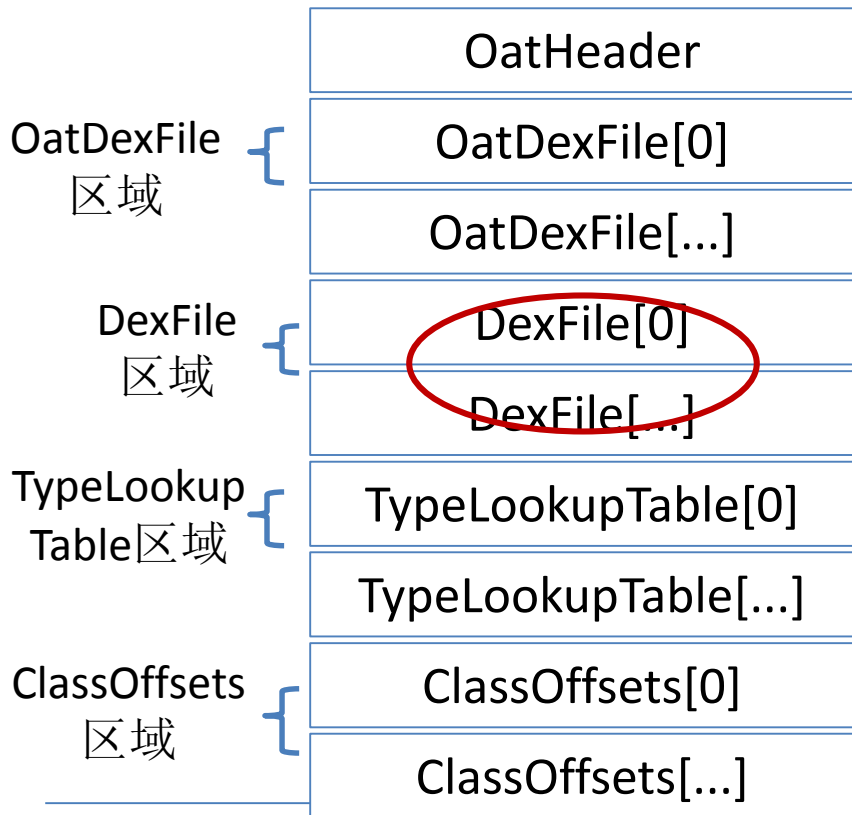
Num:	Value	Size	Type	Bind	Vis	Ndx	Name
0:	00000000	0	NOTYPE	LOCAL	DEFAULT	UND	
1:	713b7000	0x273e000	OBJECT	GLOBAL	DEFAULT	1	oatdata
2:	73af5000	0x18ace5f	OBJECT	GLOBAL	DEFAULT	2	oatexec
3:	753a1e5b	4	OBJECT	GLOBAL	DEFAULT	2	oatlastword





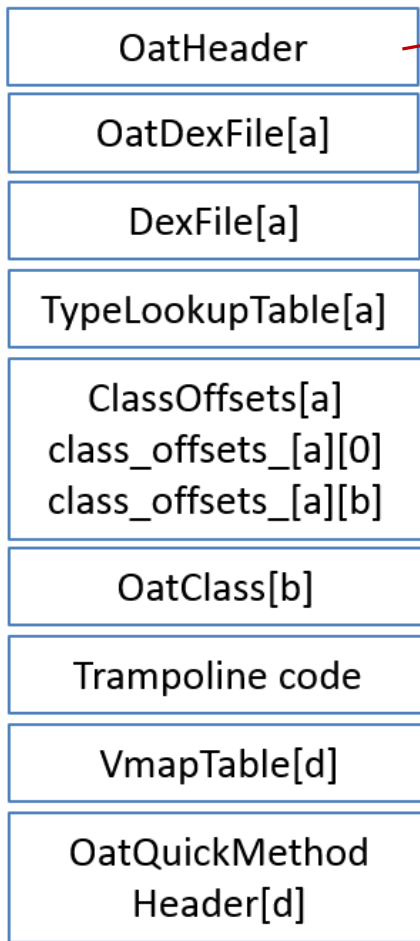
## oat文件格式 ( 2 )

### oat文件格式





## oat文件格式 ( 3 )



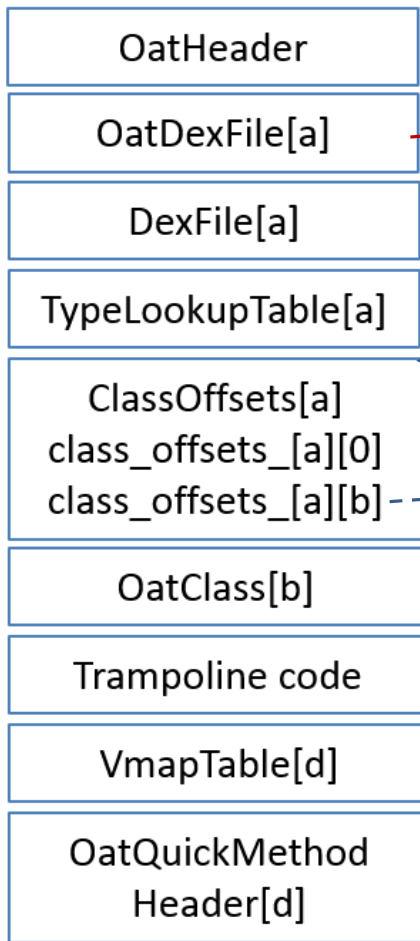
```
uint8_t magic_[4]={‘o’;‘a’;‘t’;‘\n’}  
uint8_t version_[4]={‘0’;‘7’;‘9’;‘\0’}  
uint32_t Adler32_checksum_  
InstructionSet instruction_set_  
uint32_t instruction_set_features_bitmap_  
uint32_t dex_file_count_  
uint32_t executable_offset_  
uint32_t interpreter_to_interpreter_bridge_offset_  
uint32_t interpreter_to_compiled_code_bridge_offset_  
uint32_t jni_dlsym_lookup_offset_  
uint32_t quick_generic_jni_trampoline_offset_  
uint32_t quick_int_conflict_trampoline_offset_  
uint32_t quick_resolution_trampoline_offset_  
uint32_t quick_to_interpreter_bridge_offset_  
int32_t image_patch_delta_  
uint32_t image_file_location_oat_checksum_  
uint32_t image_file_location_oat_data_begin_  
uint32_t key_value_store_size_  
uint8_t key_value_store_[0]
```

OatHeader

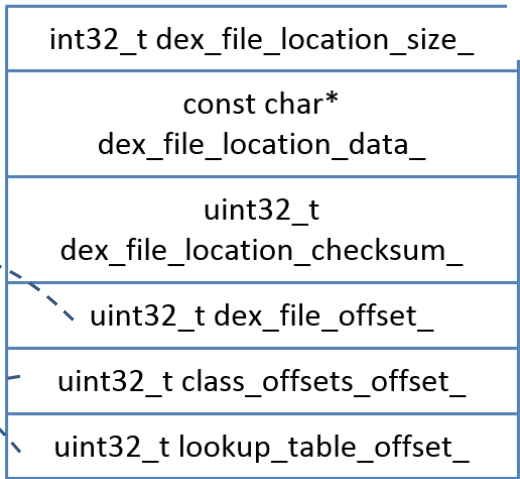




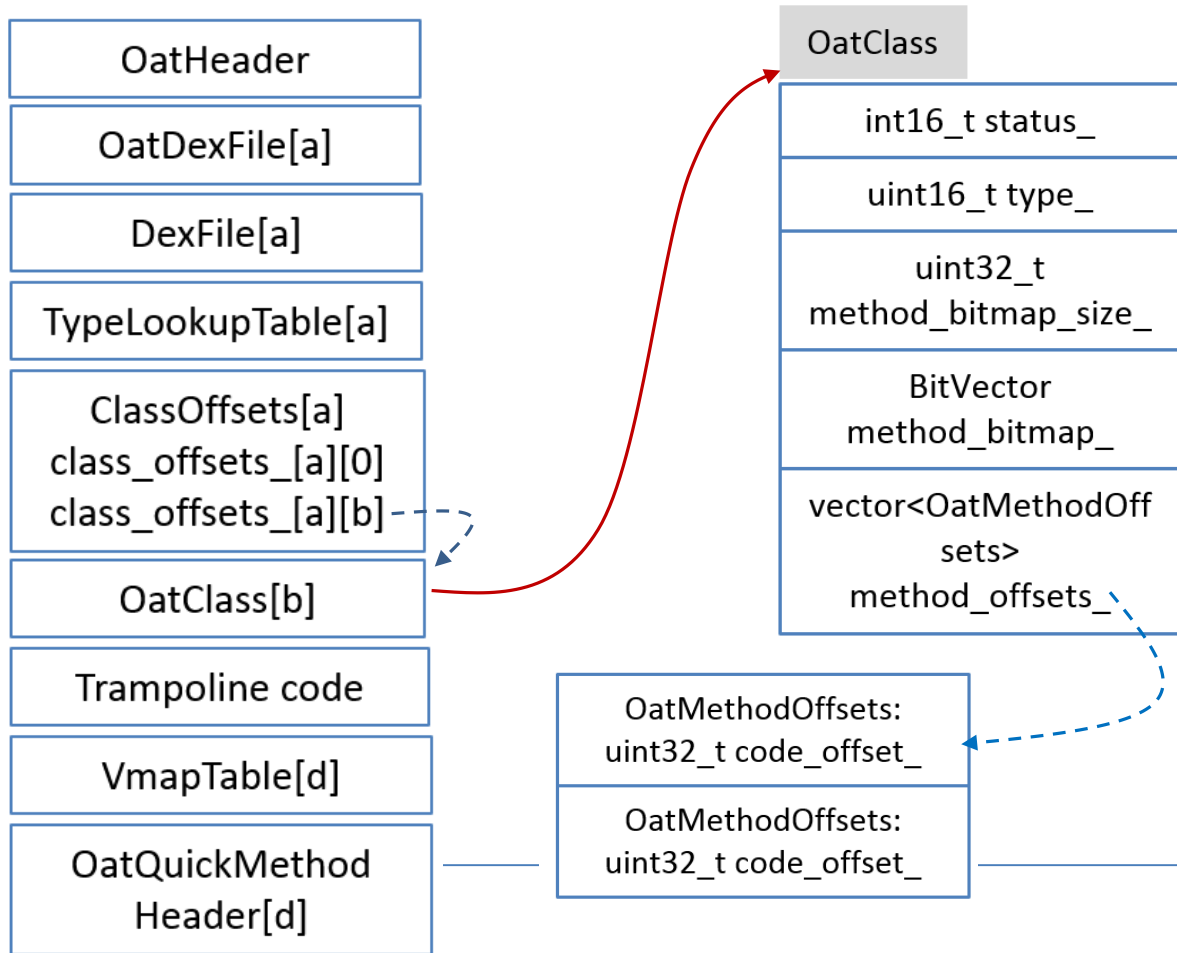
# oat文件格式 ( 3 )



## OatDexFile

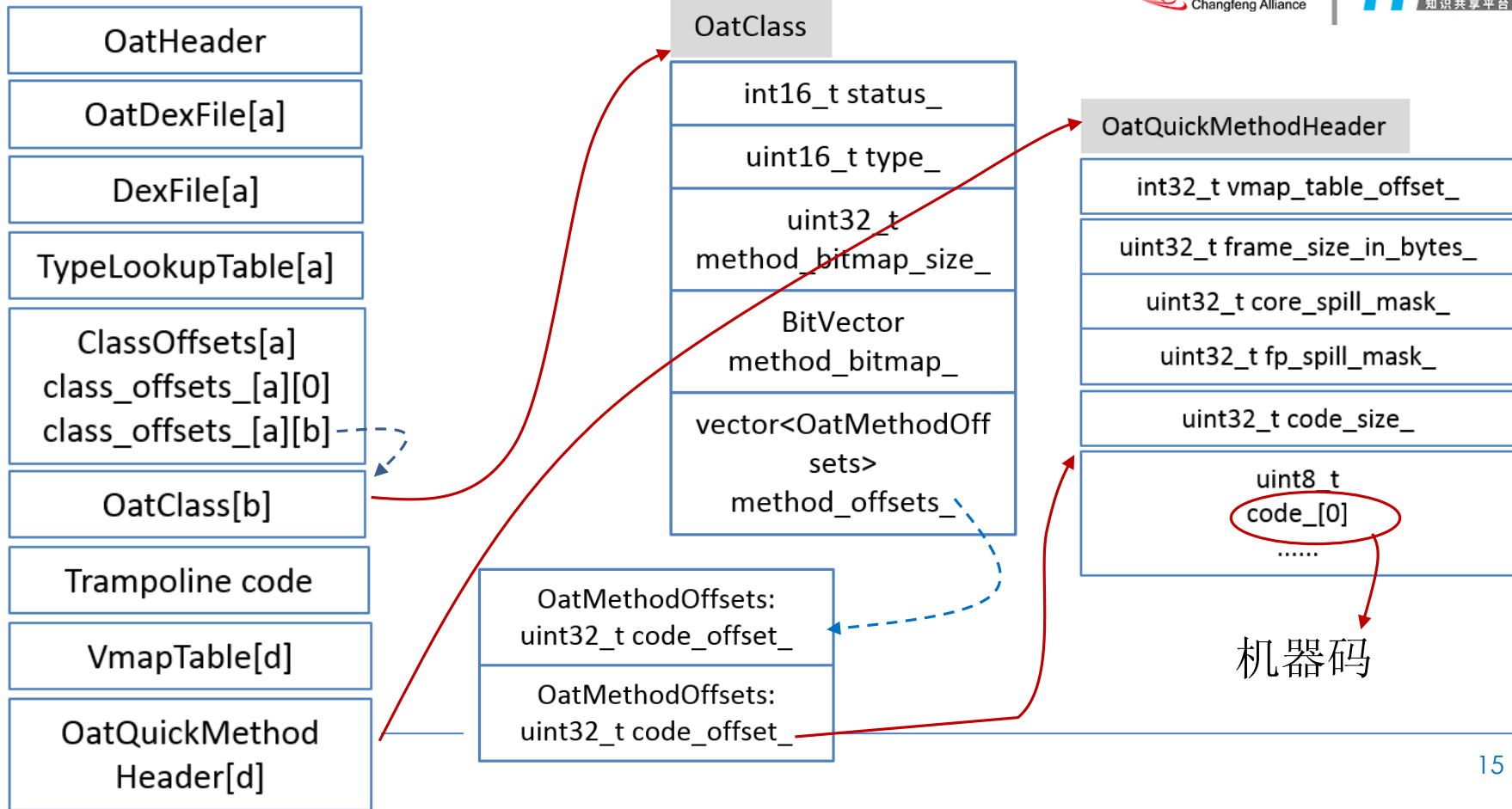


# oat文件格式 ( 3 )





# oat文件格式 ( 3 )





# Java Class及Object在ART虚拟机里的内存布



原则:

- ✓ 方法、静态成员变量是类的属性
- ✓ 非静态成员变量归属为类的实例





# Java Class及Object在ART虚拟机里的内存布局



Java Class对象在ART虚拟机里的内存占用

sizeof(Class)
sizeof(uint_32) VTable元素个数N
IM Table ArtMethod*[64]
VTable ArtMethod*[N]
静态引用类型变量(M个) sizeof(HeapReference<Object>)*M
静态long/double型变量(O个) 8*O
静态int/float型变量(X个) 4*X
静态short/char型变量(Y个) 2*Y
静态byte/boolean型变量(Z个) 1*Z

ArtMethod

静态成员变量的存储空间

Java某Class的实例对象在ART虚拟机里的内存占用

父类的ObjectSize
非静态引用类型变量(A个) sizeof(HeapReference<Object>)*A
非静态long/double型变量(B个) 8*B
非静态int/float型变量(C个) 4*C
非静态short/char型变量(D个) 2*D
非静态byte/boolean型变量(E个)

父类非静态成员变量

非静态成员变量

ArtField:

- declaring\_class\_
- access\_flags\_
- field\_dex\_idx\_
- offset\_

方法调用:

- invoke-static/direct/
- invoke-virtual/interface .....

核心: 找到正确的ArtMethod对象

nullptr

embedded\_imtable\_

vtable\_

iftable\_

embedded\_vtable\_

methods\_

ArtMethod[klmtSize=64]

GetImtUnimplementedMethod

.....

ConcreteClass.doInterfaceWork0

ConcreteClass.doInterfaceWork1

ConcreteClass.doInterfaceWork2

ConcreteClass.doInterfaceWork3

.....

ArtMethod\*

Object 11个方法

ConcreteClass.doAbsWork0

ConcreteClass.doInterfaceWork0

ConcreteClass.doInterfaceWork3

AbsClass0.doRealWork0

ConcreteClass.doInterfaceWork2

ConcreteClass.doInterfaceWork1

ConcreteClass.doRealWork1

ConcreteChildClass.doRealWork2

LengthPrefixedArray<ArtMethod>

ConcreteChildClass.<init>

ConcreteChildClass.doRealWork2

ArtMethod:

- method\_index\_
- hotness\_count\_
- entry\_point\_from\_jni\_
- entry\_point\_from\_quick\_compiled\_code\_

IfTable

java.lang.Class<

ConcreteClass.doInterf

ConcreteClass.doInterfaceWork1

java.lang.Class<If1>

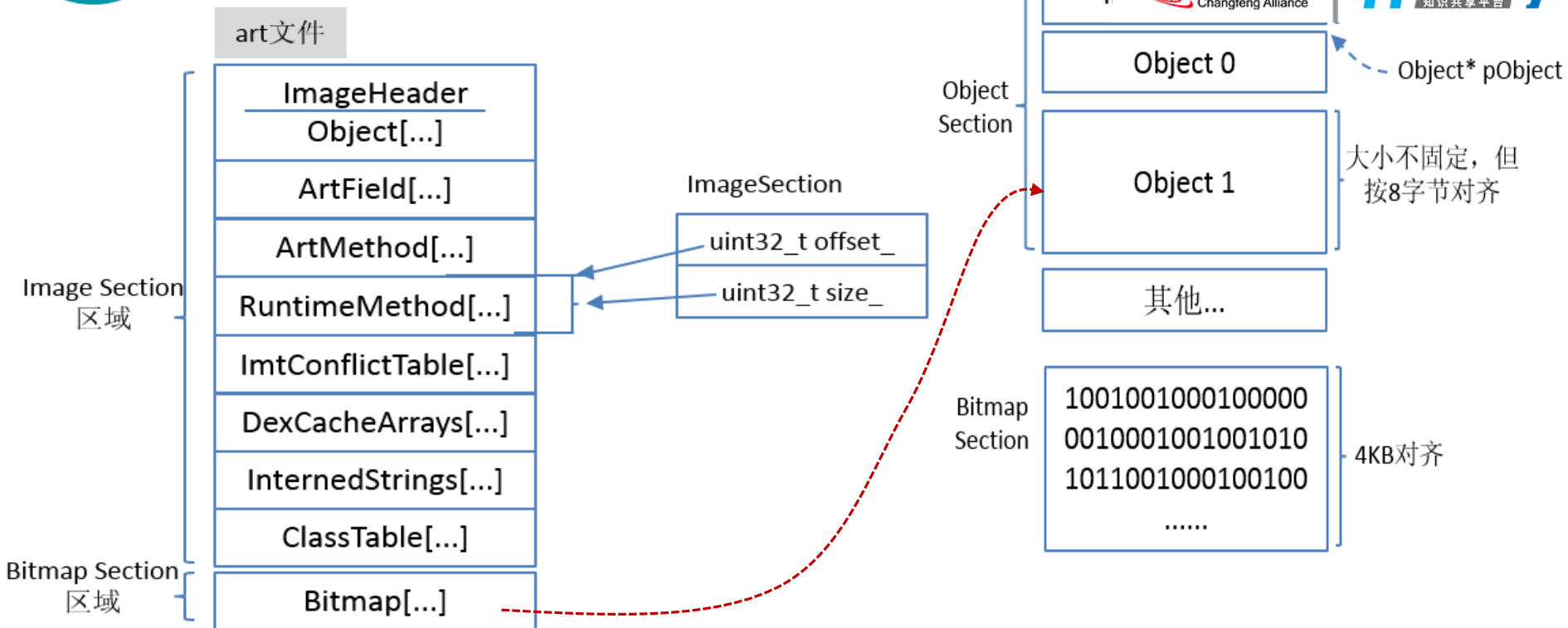
ConcreteClass.doInterfaceWork2

ConcreteClass.doInterfaceWork3



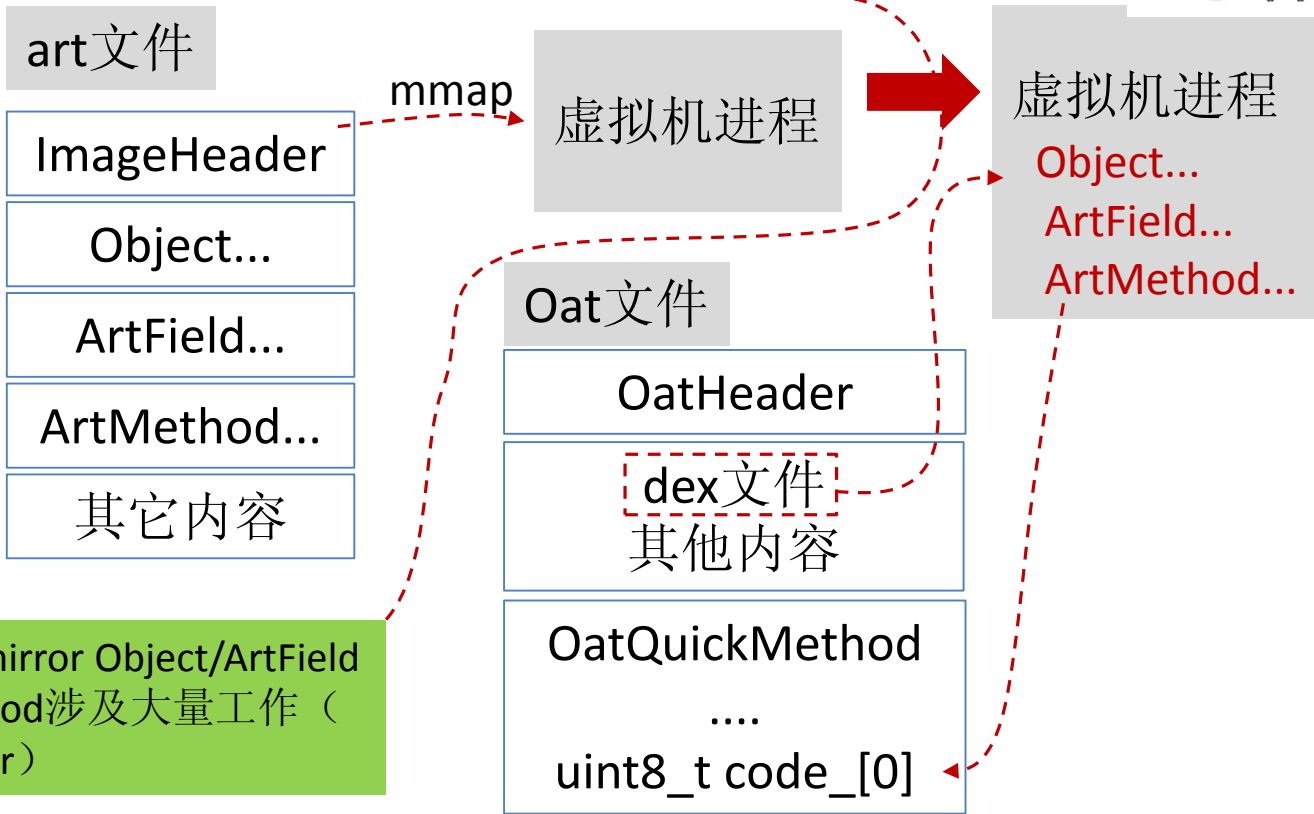


# art文件格式 (1)





# art文件格式 ( 2 ) ——art文件的作用





## art文件格式 ( 3 ) ——oatdump



adb shell oatdump --image=/system/framework/bo

```
art_file_bytes = header_bytes + object_bytes + alignment_bytes
header_bytes   =      200 ( 0% of art file bytes)
object_bytes   = 5070453 (46% of art file bytes)
art_field_bytes =  619656 ( 6% of art file bytes)
art_method_bytes = 2303328 (21% of art file bytes)
dex_cache_arrays_bytes = 2455340 (22% of art file bytes)
interned_string_bytes =  351608 ( 3% of art file bytes)
class_table_bytes =  37528 ( 0% of art file bytes)
bitmap_bytes   =   81920 ( 1% of art file bytes)
alignment_bytes = 122783 ( 1% of art file bytes)
```

object\_bytes breakdown:

```
  Android/accounts/Account$1;      8 bytes      1 instances ( 8 bytes/instance) 0% of object_bytes
  Android/animation/ArgbEvaluator;  8 bytes      1 instances ( 8 bytes/instance) 0% of object_bytes
  Android/animation/FloatEvaluator; 8 bytes      1 instances ( 8 bytes/instance) 0% of object_bytes
  Android/animation/IntEvaluator;   8 bytes      1 instances ( 8 bytes/instance) 0% of object_bytes
  Android/animation/RectEvaluator;  12 bytes     1 instances ( 12 bytes/instance) 0% of object_bytes
  Android/app/ActivityManager$MemoryInfo$1; 8 bytes     1 instances ( 8 bytes/instance) 0% of object_bytes
  Android/app/ActivityManager$RunningAppProcessInfo$1; 8 bytes     1 instances ( 8 bytes/instance) 0% of object_bytes
  Android/app/ActivityManager$TaskDescription$1; 8 bytes     1 instances ( 8 bytes/instance) 0% of object_bytes
  Android/app/ActivityManagerNative$1; 12 bytes     1 instances ( 12 bytes/instance) 0% of object_bytes
  Android/app/ApplicationErrorReport$1; 8 bytes      1 instances ( 8 bytes/instance) 0% of object_bytes
  Android/app/ApplicationLoaders; 12 bytes     1 instances ( 12 bytes/instance) 0% of object_bytes
  Android/app/IActivityManager$ContentProviderHolder$1; 8 bytes     1 instances ( 8 bytes/instance) 0% of object_bytes
  Android/app/Notification$1; 8 bytes      1 instances ( 8 bytes/instance) 0% of object_bytes
  Android/app/Notification$Action$1; 8 bytes      1 instances ( 8 bytes/instance) 0% of object_bytes
  Android/app/PendingIntent$1; 8 bytes      1 instances ( 8 bytes/instance) 0% of object_bytes
  Android/app/ResourcesManager$1; 8 bytes      1 instances ( 8 bytes/instance) 0% of object_bytes
  Android/app/ResultInfo$1; 8 bytes      1 instances ( 8 bytes/instance) 0% of object_bytes
```



## art文件格式 ( 3 ) ——oatdump

```
0x708ac170: java.lang.String "startRecognition() failed with error code "  
  shadow$_klass_: 0x705fc268   Class: java.lang.String  
  shadow$_monitor_: 0 (0x0)  
  count: 42 (0x2a)  
  hash: -832870278 (0xce5b687a)  
0x708ac1d8: java.lang.String "android.permission.BIND_QUICK_SETTINGS_TILE"  
  shadow$_klass_: 0x705fc268   Class: java.lang.String  
  shadow$_monitor_: 0 (0x0)  
  count: 43 (0x2b)  
  hash: -1354439018 (0xaf44e696)  
0x708ac240: java.lang.String "Notification listener service not yet bound."  
  shadow$_klass_: 0x705fc268   Class: java.lang.String  
  shadow$_monitor_: 0 (0x0)  
  count: 44 (0x2c)  
  hash: -1583507273 (0xa19d98b7)  
0x708ac2a8: java.lang.String "attach() called after dream already finished"  
  shadow$_klass_: 0x705fc268   Class: java.lang.String  
  shadow$_monitor_: 0 (0x0)  
  count: 44 (0x2c)  
  hash: -1895380700 (0x8f06c924)  
0x708ac310: java.lang.String "android.service chooser.ChooserTargetService"  
  shadow$_klass_: 0x705fc268   Class: java.lang.String  
  shadow$_monitor_: 0 (0x0)  
  count: 44 (0x2c)  
  hash: 1727114772 (0x66f1ae14)
```



## art文件格式 ( 3 ) ——oatdump



```
0x7064d140: java.lang.Class "java.util.concurrent.locks.ReadWriteLock" (Sta
shadow$_klass_: 0x705fbdb8   Class: java.lang.Class
shadow$_monitor_: 0 (0x0)
accessFlags: 537396737 (0x20080601)
annotationType: null   sun.reflect.annotation.AnnotationType
classFlags: 1 (0x1)
classLoader: null   java.lang.ClassLoader
classSize: 128 (0x80)
clinitThreadId: 0 (0x0)
componentType: null   java.lang.Class
copiedMethodsOffset: 2 (0x2)
dexCache: 0x7050c110   java.lang.DexCache
dexCacheStrings: 1892852196 (0x70d2a1e4)
dexClassDefIndex: 1579 (0x62b)
dexTypeIndex: 1883 (0x75b)
iFields: 0 (0x0)
ifTable: null   java.lang.Object[]
methods: 1890465988 (0x70ae38c4)
name: 0x707ce6c0   String: "java.util.concurrent.locks.ReadWriteLock"
numReferenceInstanceFields: 0 (0x0)
numReferenceStaticFields: 0 (0x0)
objectSize: 8 (0x8)
primitiveType: 131072 (0x20000)
referenceInstanceOffsets: 0 (0x0)
sFields: 0 (0x0)
status: 10 (0xa)
superClass: 0x705fc588   Class: java.lang.Object
verifyError: null   java.lang.Object
virtualMethodsOffset: 0 (0x0)
vtable: null   java.lang.Object
```



# art文件格式 (4) ——art文件的来源



boot镜像  
|->boot.oat  
|->boot.art



```
dex2oat: option[0]=--image=/data/dalvik-cache/x86/system@framework@b
dex2oat: option[1]=--dex-file=/system/framework/core-oj.jar
dex2oat: option[2]=--dex-file=/system/framework/core-libart.jar
dex2oat: option[3]=--dex-file=/system/framework/conscrypt.jar
dex2oat: option[4]=--dex-file=/system/framework/okhttp.jar
dex2oat: option[5]=--dex-file=/system/framework/core-junit.jar
dex2oat: option[6]=--dex-file=/system/framework/bouncycastle.jar
dex2oat: option[7]=--dex-file=/system/framework/ext.jar
dex2oat: option[8]=--dex-file=/system/framework/framework.jar
dex2oat: option[9]=--dex-file=/system/framework/telephony-common.jar
dex2oat: option[10]=--dex-file=/system/framework/voip-common.jar
dex2oat: option[11]=--dex-file=/system/framework/ims-common.jar
dex2oat: option[12]=--dex-file=/system/framework/apache-xml.jar
dex2oat: option[13]=--dex-file=/system/framework/org.apache.http.legacy.boot.jar
dex2oat: option[14]=--oat-file=/data/dalvik-cache/x86/system@framework@boot.oat
dex2oat: option[15]=--instruction-set=x86
dex2oat: option[16]=--instruction-set-features=smp,ssse3,-sse4.1,-sse4.2,-avx,-avx2,-lock_add,-popcnt
dex2oat: option[17]=--base=0x70942000
dex2oat: option[18]=--runtime-arg
dex2oat: option[19]=-Xms64m
dex2oat: option[20]=--runtime-arg
dex2oat: option[21]=-Xmx64m
dex2oat: option[22]=--image-classes=/system/etc/preloaded-classes
dex2oat: option[23]=--compiled-classes=/system/etc/compiled-classes
dex2oat: option[24]=--instruction-set-variant=x86
Unexpected CPU variant for X86 using defaults: x86
dex2oat: option[25]=--instruction-set-features=default
Mismatch between dex2oat instruction set features (ISA: X86 Feature string: smp,-ssse3,-sse4.1,-sse4.2,
```

APP镜像  
|->xxx.dex



yyy.jar  
zzz.apk



# 3

## Java方法的执行



# main函数的调用流程

虚拟机启动



找到目标方法的ArtMethod



调用ArtMethod.Invoke

```
/* start the virtual machine */  
JniInvocation jni_invocation;  
jni_invocation.Init(NULL);  
JNIEnv* env;  
if (startVm(&mJavaVM, &env, zygote) != 0) {  
    return;  
}
```

```
char* slashClassName = toSlashClassName(className);  
jclass startClass = env->FindClass(slashClassName);  
if (startClass == NULL) {  
    ALOGE("JavaVM unable to locate class '%s'\n", slashClassName);  
    /* keep going */  
} else {  
    jmethodID startMeth = env->GetStaticMethodID(startClass, "main",  
        "([Ljava/lang/String;)V");  
    if (startMeth == NULL) {  
        ALOGE("JavaVM unable to find main() in '%s'\n", className);  
        /* keep going */  
    } else {  
        env->CallStaticVoidMethod(startClass, startMeth, strArray);  
    }  
}
```

0



# ArtMethod里的入口地址

ArtMethod

```
|  
|-> ptr_sized_fields_  
    |->entry_point_from_jni_  
    |->entry_point_from_quick_compiled_code_
```



方法类型	机器码入口	jni机器码入口
jni方法	art_quick_generic_jni_trampoline或 对应机器码	art_jni_dlsym_lookup_stub
非jni方法	art_quick_to_interpreter_bridge或 对应机器码	ImtConflictTable



# ArtMethod Invoke

```
void ArtMethod::Invoke(Thread* self, uint32_t* args, uint32_t args_size, JValue* result,
                       const char* shorty) {
    .....

    if (UNLIKELY(!runtime->IsStarted() || Dbg::IsForcedInterpreterNeededForCalling(self, this
        .....
    } else {
        bool have_quick_code = GetEntryPointFromQuickCompiledCode() != nullptr;
        if (LIKELY(have_quick_code)) {
            .....
            if (!IsStatic()) {
                (*art_quick_invoke_stub)(this, args, args_size, self, result, shorty);
            } else {
                (*art_quick_invoke_static_stub)(this, args, args_size, self, result, shorty);
            }
        }
    }
}
```

```
DEFINE_FUNCTION art_quick_invoke_stub
// Save the non-volatiles.
PUSH ebp // save ebp
PUSH ebx // save ebx
PUSH esi // save esi
PUSH edi // save edi
.....
Lgpr_setup_finished:
mov 20(%ebp), %eax // move method pointer into eax
call *ART_METHOD_QUICK_CODE_OFFSET_32(%eax) // call the method
mov %ebp, %esp // restore stack pointer
CFI_DEF_CFA_REGISTER(esp)
POP edi // pop edi
POP esi // pop esi
```

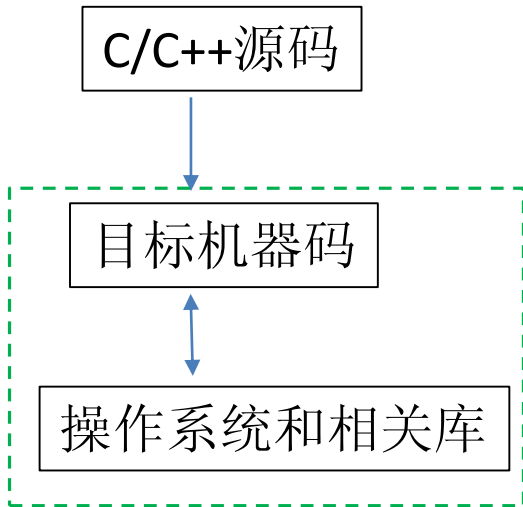
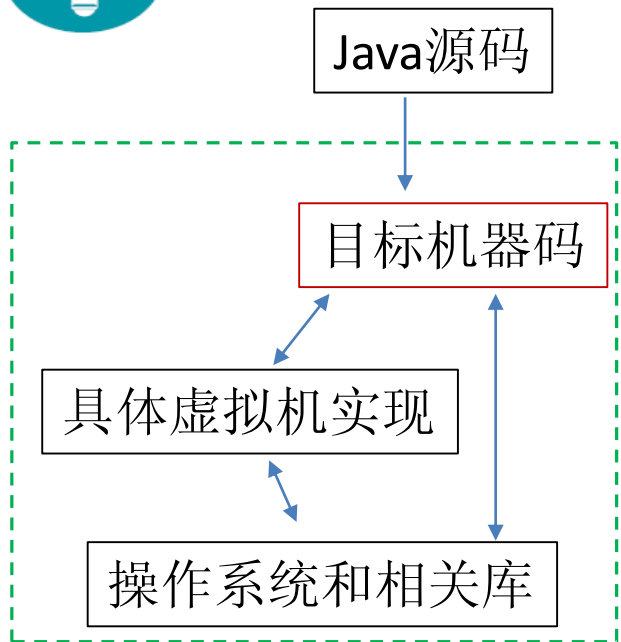
# 4

## 讨论

1. Java VS C++，谁更快
2. ART虚拟机研究路线推荐



# Java VS C++ , 谁更快



- ❑ java进程的内容=虚拟机的内容+自身的内容
- ❑ java的目标机器码强依赖于虚拟机，并受虚拟机监管（ManagedCode）
- ❑ 某些情况下将转入虚拟机内执行（SlowPathCode）



# ART虚拟机研究路线推荐

## 基础知识

- .dex文件格式
- ELF文件格式
- C++11



## dex2oat

- compiler相关知识
- ART基础模块
- 信息的转换和表达
- .oat/.art文件格式



## 执行相关

- 栈的管理
- 解释执行
- JIT等
- JNI
- 线程同步，锁等



## 内存管理

- .....

谢谢！