第一次找Dalvik初始化源代码（失败经历）

（一）寻找源代码

//首先是找到在线源码网站androidxref.com

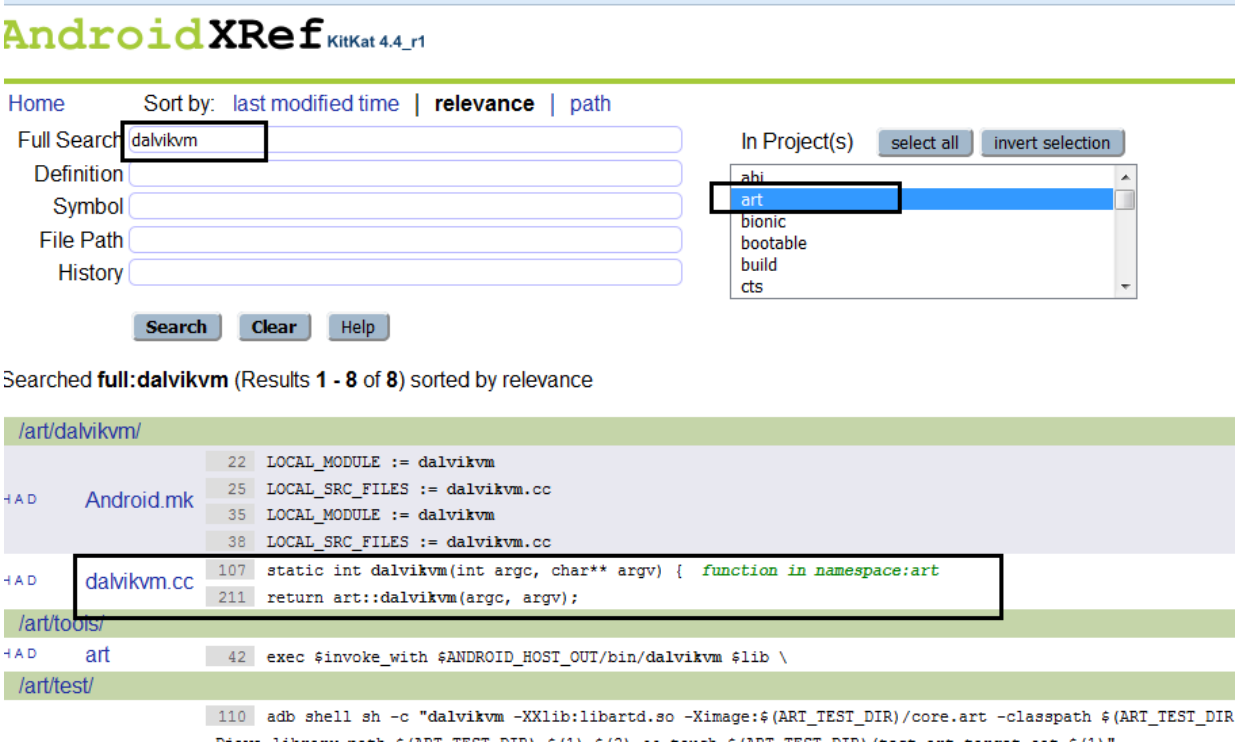
//我选择的是4.4经典版的0.0　然后点击右侧的In Project(s),选择一个分支，我已经知道了在art分支里

//所以直接选上，如果不知道，可以选择select all

//左侧是输入要检索的字段，输入后，再点击serch就可以查找了。结果在下方显示

//函数定义我们可以直接看到有绿色字体标识出function

//进入daivikvm.cc



（二）看到main了

//我们拉到最底下，终于看见真身了，main函数，虚拟机初始化

AndroidXRef KitKat 4.4_r1

xref: /art/dalvikvm/dalvikvm.cc

Home | History | Annotate | Line# | Navigate | Download Search only in dalvikvm.cc

```
184   if (JNI_CreateJavaVM(&vm, &env, &init_args) != JNI_OK) {
185     fprintf(stderr, "Failed to initialize runtime (check log for details)\n");
186     return EXIT_FAILURE;
187   }
188
189   int rc = InvokeMain(env, &argv[arg_idx]);
190
191 #if defined(NDEBUG)
192   // The DestroyJavaVM call will detach this thread for us. In debug builds, we don't want to
193   // detach because detaching disables the CheckSafeToLockOrUnlock checking.
194   if (vm->DetachCurrentThread() != JNI_OK) {
195     fprintf(stderr, "Warning: unable to detach main thread\n");
196     rc = EXIT_FAILURE;
197   }
198 #endif
199
200   if (vm->DestroyJavaVM() != 0) {
201     fprintf(stderr, "Warning: runtime did not shut down cleanly\n");
202     rc = EXIT_FAILURE;
203   }
204
205   return rc;
206 }
207
208 }  // namespace art
209
210 int main(int argc, char** argv) {
211   return art::dalvikvm(argc, argv);
212 }
213
```

（三）关键函数JNI_CreateJavaVM

//然后函数主体中，我们遇到了第一个关键的函数

```
  // Start the runtime. The current thread becomes the main thread.
  JavaVM* vm = NULL;
  JNIEnv* env = NULL;
  if (JNI_CreateJavaVM(&vm, &env, &init_args) != JNI_OK) {
    fprintf(stderr, "Failed to initialize runtime (check log for details)\n");
    return EXIT_FAILURE;
  }

  int rc = InvokeMain(env, &argv[arg_idx]);
```

AndroidXRef KitKat 4.4_r1

Home Sort by: last modified time | **relevance** | path

Full Search
Definition JNI_CreateJavaVM
Symbol
File Path
History

In Project(s) select all invert selection

abi
art
bionic
bootable
build
cts

Search Clear Help

Searched **defs:JNI_CreateJavaVM** (Results **1 - 3** of **3**) sorted by relevance

/libnativehelper/

HAD JniInvocation.cpp
71 "JNI_CreateJavaVM")) {
85 jint JniInvocation::JNI_CreateJavaVM(JavaVM** p_vm, JNIEnv** p_env, void* vm_args) { function in class:
114 extern "C" jint JNI_CreateJavaVM(JavaVM** p_vm, JNIEnv** p_env, void* vm_args) { function
115 return JniInvocation::GetJniInvocation().JNI_CreateJavaVM(p_vm, p_env, vm_args);

/dalvik/vm/

HAD Jni.cpp
3424 jint JNI_CreateJavaVM(JavaVM** p_vm, JNIEnv** p_env, void* vm_args) { function

/art/runtime/

HAD jni_internal.cc
2885 extern "C" jint JNI_CreateJavaVM(JavaVM** p_vm, JNIEnv** p_env, void* vm_args) { function in namespace:

```
3419 * Create a new VM instance.
3420 *
3421 * The current thread becomes the main VM thread.  We return immediately,
3422 * which effectively means the caller is executing in a native method.
3423 *
```

3424jintJNI_CreateJavaVM(JavaVM** p_vm, JNIEnv** p_env, void* vm_args)

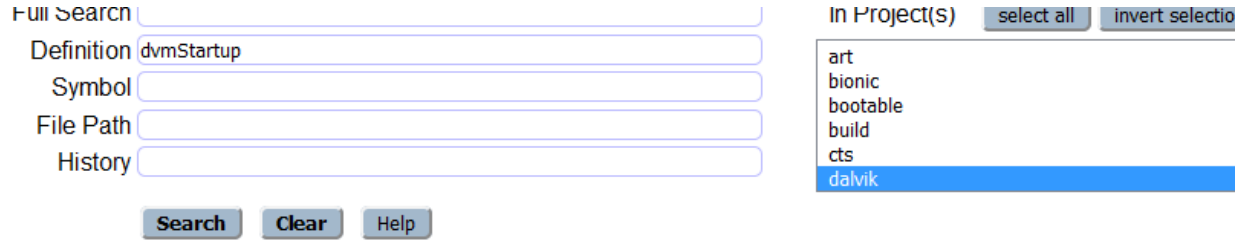
//这个参数，我也不是很了解，大概就是环境啥的，这个函数返回的status决定虚拟机是否创建成功
//我是参考这两个地方理解 的
http://blog.csdn.net/liusongliang6808/article/details/11530669
http://wenku.baidu.com/link?url=1s7aNnhI3_-1G_5wxdlIY6F1AsKf20B1uOCsiuua49S04Rxq21QIbv8vkRLUxRKPtCT3waLRuVtctkR6juMrT4mQT35PWhaBYEGw2flxcdi

（四）关键函数dvmStartup

//接着在上边函数主体中找到两处疑似关键点，分别进去看了下

dvmStartup(argc, argv.get(), args->ignoreUnrecognized, (JNIEnv*)pEnv);

```
3517     * up the VM will call into native code.
3518     */
3519    JNIEnvExt* pEnv = (JNIEnvExt*) dvmCreateJNIEnv(NULL);
3520
3521    /* Initialize VM. */
3522    gDvm.initializing = true;
3523    std::string status =
3524            dvmStartup(argc, argv.get(), args->ignoreUnrecognized, (JNIEnv*)pEnv);
3525    gDvm.initializing = false;
3526
3527    if (!status.empty()) {
3528        free(pEnv);
3529        free(pVM);
3530        ALOGW("CreateJavaVM failed: %s", status.c_str());
3531        return JNI_ERR;
3532    }
3533
3534    /*
```



earched **defs:dvmStartup** (Results **1 - 1** of **1**) sorted by relevance

/dalvik/vm/

AD Init.cpp 1382 std::string dvmStartup(int argc, const char* const argv[], function
1881 * An alternative to JNI_CreateJavaVM/dvmStartup that does the first bit

//于是我又搜罗的一个网址
//http://blog.csdn.net/gjsisi/article/details/38441451
//进去这个函数快速看了下，傻眼了，这么长。。。。。

```
1377  * VM initialization.  Pass in any options provided on the command line.
1378  * Do not pass in the class name or the options for the class.
1379  *
1380  * Returns 0 on success.
1381  */
1382 std::string dvmStartup(int argc, const char* const argv[],
1383         bool ignoreUnrecognized, JNIEnv* pEnv)
1384 {
1385     ScopedShutdown scopedShutdown;
1386
1387     assert(gDvm.initializing);
1388
1389     ALOGV("VM init args (%d):", argc);
1390     for (int i = 0; i < argc; i++) {
1391         ALOGV("  %d: '%s'", i, argv[i]);
1392     }
1393     setCommandLineDefaults();
1394
1395     /*
1396      * Process the option flags (if any).
1397      */
1398     int cc = processOptions(argc, argv, ignoreUnrecognized);
1399     if (cc != 0) {
1400         if (cc < 0) {
1401             dvmFprintf(stderr, "\n");
1402             usage("dalvikvm");
1403         }
1404         return "syntax error";
1405     }
1406
```

//那么，我们关心的到底是什么呢？

//此处思考一个小时，然后发现我错了。。。。
//百度Dalvik初始化
、、Dalvik虚拟机源码解读
http://www.cnblogs.com/autum/archive/2012/04/04/dalvik123.html
老罗的Android之旅：Dalvik虚拟机的启动过程分析
http://blog.csdn.net/luoshengyang/article/details/8885792

Dalvik虚拟机的入口有两个

1./Dalvikvm/Main.c

2.frameworks/base/cmds/app_process.cpp

http://blog.sina.com.cn/s/blog_4ce5cca30101qnzy.html

Android Xref KitKat 4.4_r1

xref: /frameworks/base/cmds/app_process/app_main.cpp

Home | History | Annotate | Line# | Navigate | Download Search only in

```
28     fprintf(stderr,
29         "Usage: app_process [java-options] cmd-dir start-class-name [options]\n");
30 }
31
32 class AppRuntime : public AndroidRuntime
33 {
34 public:
35     AppRuntime()
36         : mParentDir(NULL)
37         , mClassName(NULL)
38         , mClass(NULL)
39         , mArgC(0)
40         , mArgV(NULL)
41     {
42     }
43
44 #if 0
```

```
runtime.mParentDir = parentDir;

if (zygote) {
    runtime.start("com.android.internal.os.ZygoteInit",
            startSystemServer ? "start-system-server" : "");
} else if (className) {
    // Remainder of args get passed to startup class main()
    runtime.mClassName = className;
    runtime.mArgC = argc - i;
    runtime.mArgV = argv + i;
    runtime.start("com.android.internal.os.RuntimeInit",
            application ? "application" : "tool");
```