Android程序分析环境搭建-macOS篇

在实际的Android软件开发过程中，可能很多开发人员有过这样的经历：

- 我有一个不错的idea，正在开发一款类似想法的软件，可是涉及到的一些功能上的具体代码细节却难以下手，我看到别人的程序中有这个功能，它们是如何实现的呢？
- 我不小心安装了一个流氓软件，软件运行时会自动下载木马程序、恶意扣费、篡改手机系统，它是如何做到这些的呢？
- 我按照网上介绍的方法来分析Android程序，可是根本就无法正确地反编译程序，或是反编译出的代码语法混乱，根本无法阅读。

这些场景都提出了一个疑问，那就是如何分析一个Android应用程序？如何掌握这些软件的架构思想？分析别人的程序在很多人看来是不能够接受的行为，在他们眼中这种行为都应被视为盗窃。其实任何技术的起源本身就是从学习开始的，用正确的态度对待程序分析技术是可以的。

如果说，开发Android程序是一种学问，那么分析Android程序更像是一门艺术。在浩瀚如海的反汇编代码中分析出程序的执行流程与架构思想是一件很了不起的事情，这需要分析人员有着扎实的编程基础与深厚的思维分析能力。分析软件的过程犹如一次艰难的旅程，这条旅程会有多长？该怎么走？会有多少崎岖险路？没有人知道，但是先行者已经为我们铺下了台阶，我们只需沿着它慢慢前行。

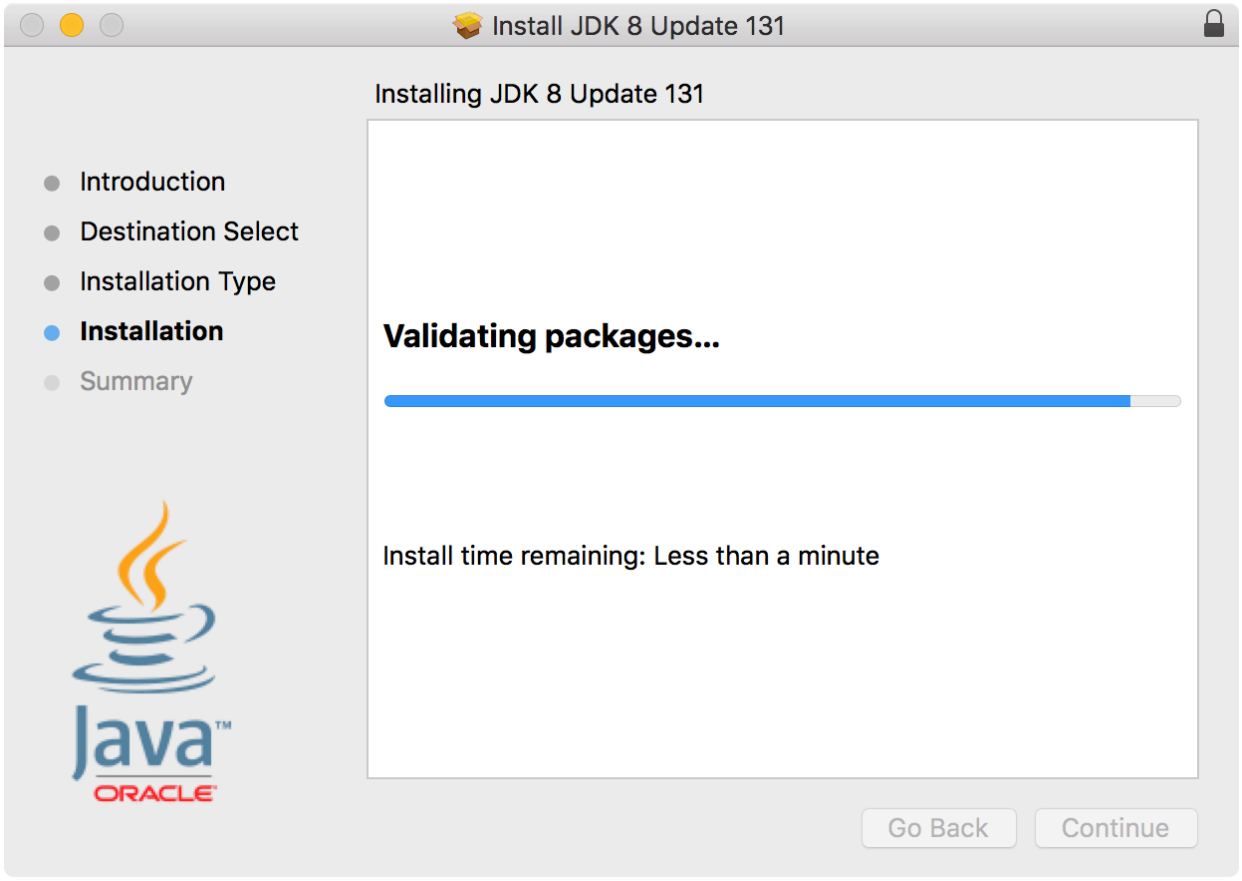
与Android软件开发使用的三大系统平台Windows、macOS、Linux一样，Android软件逆向分析的环境同样可以支持这三大平台。为了保证不同平台上工具的使用技巧与操作流程的统一性，因此在选择工具时，除一些特定的商业软件外，多以跨平台且免费开源的工具为主。而对于平台上分析环境的搭建，主要围绕Android软件开发、Android软件逆向、Android源码编译，这三个环境的配置展开。

macOS分析环境搭建

越来越多的优秀的跨平台安全工具的问世，使得在macOS系统上从事软件安全分析工作，也成为了一种非常流行的选择。非常高兴的是，Android官方默认就支持使用macOS系统来开发Android程序，以及编译Android系统源码。

安装JDK

在macOS系统上，Android软件开发与搭建Android源码编译环境，都需要安装JDK。如果读者选择在macOS上只开发Android软件，则安装JDK 8即可。安装方法与Windows上安装JDK一样。首先到官网下载JDK安装包，以JDK 8u131为例，下载下来是一个名为jdk-8u131-macosx-x64.dmg的镜像文件，双击该镜像挂载成功后，里面有一个JDK 8 Update 131.pkg安装文件，双击该文件即可安装。如图所示。



完装完成后，会在系统的/Library/Java/JavaVirtualMachines目录下添加一个jdk1.8.0_131.jdk的目录，需要将它的Contents/Home子目录设置为JDK的安装根目录，即设置它的完整路径为`JAVA_HOME`环境变量的值。最后，还要将`JAVA_HOME`环境变量下的bin目录添加到系统的`PATH`环境变量，方便第三方程序查找JDK的安装位置。

执行如下命令，将上述环境变量的配置信息写入“`.bash_profile`”文件，并更新环境变量，即完成了JDK的安装。

```
$ echo
JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home
>> ~/.bash_profile
$ export JAVA_HOME >> ~/.bash_profile
$ export PATH=$JAVA_HOME/bin:$PATH >> ~/.bash_profile
$ source ~/.bash_profile
```

执行如下命令确保JDK安装配置无误：

```
$ java -version
java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)
```

如果需要在macOS系统上编译Android系统源码，则可能根据编译的源码系统版本不同，需要选择安装不同版本的JDK。对于Android 7.0及以后版本的，需要安装JDK 8；Android 5.0到7.0之间的版本，则需要安装JDK 7；Android 4.4以及之前的版本，需要安装JDK 6。目前，Oracle官方只提供了JDK 7与JDK 8的安装程序。如果需要编译4.4与更早的Android系统源码，需要到Apple官网下载JDK 6。它作为Apple提供的一个补丁软件存在，地址是："https://support.apple.com/kb/DL1572?locale=zh_CN"。下载下来得到一个javaforosx.dmg文件，双击挂载后，其中有一个JavaForOSX.pkg安装文件，双击即可安装，安装方法与JDK 8安装一样。安装完成后，会在系统的`/Library/Java/JavaVirtualMachines`目录下添加一个`1.6.0.jdk`的目录，将它的Contents/Home子目录设置为JDK的安装根目录，即将其完整路径设置为`JAVA_HOME`环境变量的值即完成安装。

在笔者的机器上，编译配置过Android 4.4、Android 6.0.1、Android 7.1等多个系统的源码编译环境，因此，同时安装了JDK 6、JDK 7、JDK 8。使用时，笔者会将`JAVA_HOME`环境变量指向需要使用的Java版本，并注释掉其他两个版本的JDK路径。如下所示：

```
$ cat ~/.bash_profile
JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home
#JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.7.0_79.jdk/Contents/Home
#JAVA_HOME=/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home
export JAVA_HOME
export PATH=$JAVA_HOME/bin:$PATH
......
```

安装Android SDK

在`HomeBrew`的早期版本中，可以在macOS系统的Shell中执行`brew install android-sdk`命令，快速安装Android SDK。新版本的`HomeBrew`将其移动到了Cask中，执行如下命令即可一键安装：

```
$ brew cask install android-sdk
```

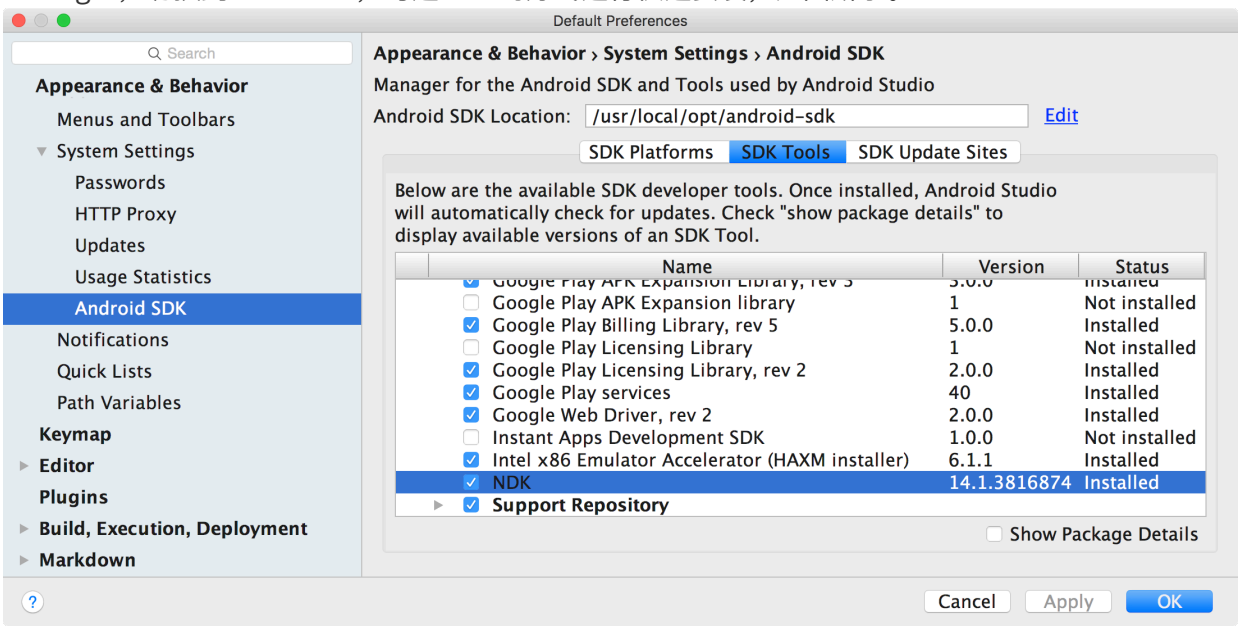
与Windows系统一样，如果打算使用`Android Studio`来开发Android程序，可以直接下载安装`Android Studio`，它包含了最新版本的Android SDK。

安装Android NDK

与Android SDK一样，在`HomeBrew`的早期版本中，可以在macOS系统的Shell中执行`brew install android-ndk`，快速安装Android NDK。新版本的`HomeBrew`将其移动到了Cask中，执行如下命令即可一键安装：

```
$ brew cask install android-ndk
```

如果打算使用`Android Studio`来开发Android原生程序，可以打开`Android Studio`的SDK Manager，切换到SDK Tools，勾选NDK的方式进行快速安装，如图所示。



除了NDK外，原生程序的开发配置，还需要勾选Cmake与LLDB。

Android Studio集成开发环境

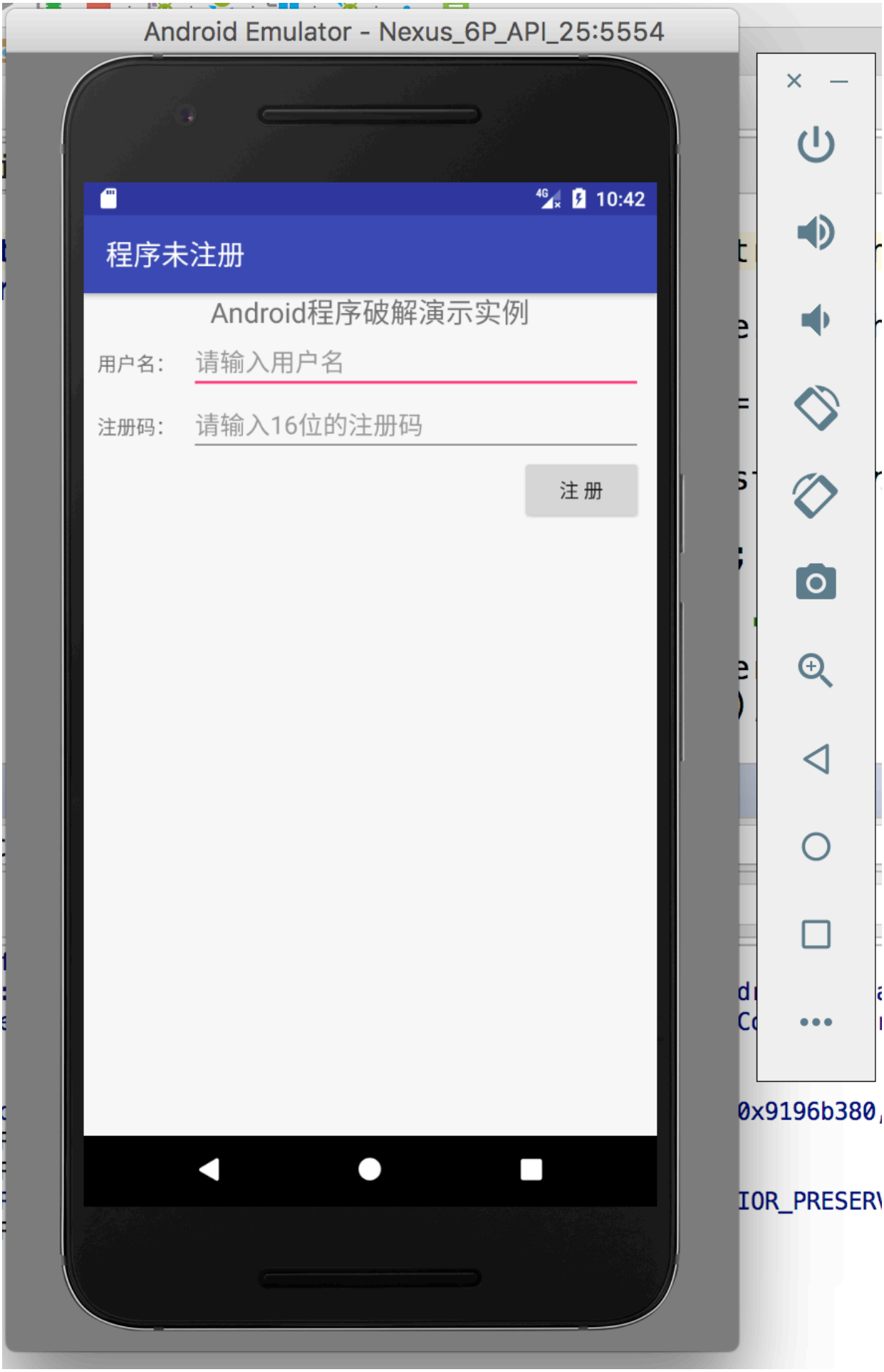
macOS平台上`Android Studio`的安装非常简单，到https://developer.android.com/studio/index.html 下载macOS版本的dmg文件，双击dmg文件将其挂载后，将`Android Studio.app`复制到/Applications目录即完成安装。

或者，执行如下命令也可以一键安装：

```
$ brew cask install android-studio
```

首次启动`Android Studio.app`，会弹出设备向导，包括设备Android SDK的目录与下载最新版本的SDK构建工具。设置完成后会进入主界面，点击Configure->SDK Manager打开SDK管理器，可以根据需要下载SDK与NDK。

如果需要创建Android模拟器，可以参考Windows平台上的操作，步骤是一样的。模拟器创建成功后，在`Android Studio`启动APK弹出设备选择对话框时，可以选择新创建的Android模拟器。如图所示，是`Android Studio`编译Crackme0201程序后，在Android模拟器上的运行效果。


Android Emulator - Nexus_6P_API_25:5554
10:42
程序未注册
Android程序破解演示实例
用户名:
请输入用户名
注册码:
请输入16位的注册码
注 册

编译Android源码

在macOS系统上编译系统Android系统源码有两种方式：使用系统直接编译与使用`Docker`容器进行编译。

首先是在系统中进行编译。Android系统的源码编译需要在文件名区分大小写的（case-sensitive）文件系统上进行，而macOS文件系统默认是不区分大小写的，因此，需要在本机中创建一个磁盘镜像存放Android系统源码。例如创建一个60g大小的区分大小写的磁盘镜像，可以执行如下命令：

```
$ hdiutil create -type SPARSE -fs 'Case-sensitive Journaled HFS+' -size 60g ~/android.dmg
```

命令执行成功后，会生成一个android.dmg或android.dmg.sparseimage文件。如果后期感觉磁盘镜像空间太小了，想重新设置为80g，可以执行如下命令：

```
$ hdiutil resize -size 80g ~/android.dmg.sparseimage
```

接下来执行如下命令将磁盘镜像挂载到/Volumes/android目录下：

```
$ hdiutil attach ~/android.dmg.sparseimage -mountpoint /Volumes/android
```

挂载成功后，就可以在/Volumes/android目录中下载与编译Android系统源码了。

下一步是安装Android源码下载与编译所需要的软件。首先是JDK的安装，根据编译的源码版本不同，需要安装不同版本的JDK，JDK的安装过程前面讲过，这里不再赘述。安装JDk完成后，下一步是安装XCode命令行工具与macOS SDK。XCode命令行工具可以执行如下命令进行安装：

```
$ xcode-select --install
```

如果本机中安装的XCode命令行工具与当前需要编译的源码支持的版本不匹配，则无法完成编译工作，例如在macOS 10.12上编译Android 4.4就会失败，原因是XCode命令行工具版本过高。一种解决方法是到Apple官网下载低版本的XCode，并安装其命令行工具，不过这里笔者不建议这样做，而是推荐以`Docker`的方式进行编译，这样既能完成源码的编译工作，又不会对系统文件结构造成污染。

macOS SDK在XCode中已经自带。但不同版本的Android系统源码，只能在特定版本的macOS SDK下才能编译完成。Android系统源码支持的系统版本可以在源码的build/core/combo/mac_version.mk文件中查看，所有支持的macOS SDK版本在`mac_sdk_versions_supported`变量中进行了预先声明。例如Android 5.1的源码，只支持10.6、10.7、10.8、10.9这四个版本的MacOS SDK，macOS 10.12的SDK就会编译失败。解决方法是到页面https://github.com/phracker/MacOSX-SDKs/releases 下载与当前编译环境相匹配的MacOS SDK，例如5.1可以下载10.9的SDK，将其解压到/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs目录下即可。

如果本机之前没有安装`HomeBrew`，需要先执行如下命令进行安装：

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

安装好`HomeBrew`后，执行如下命令安装编译时所需要的软件。

```
$ brew install gmake bison libsdl git gnupg gnupg2 repo
```

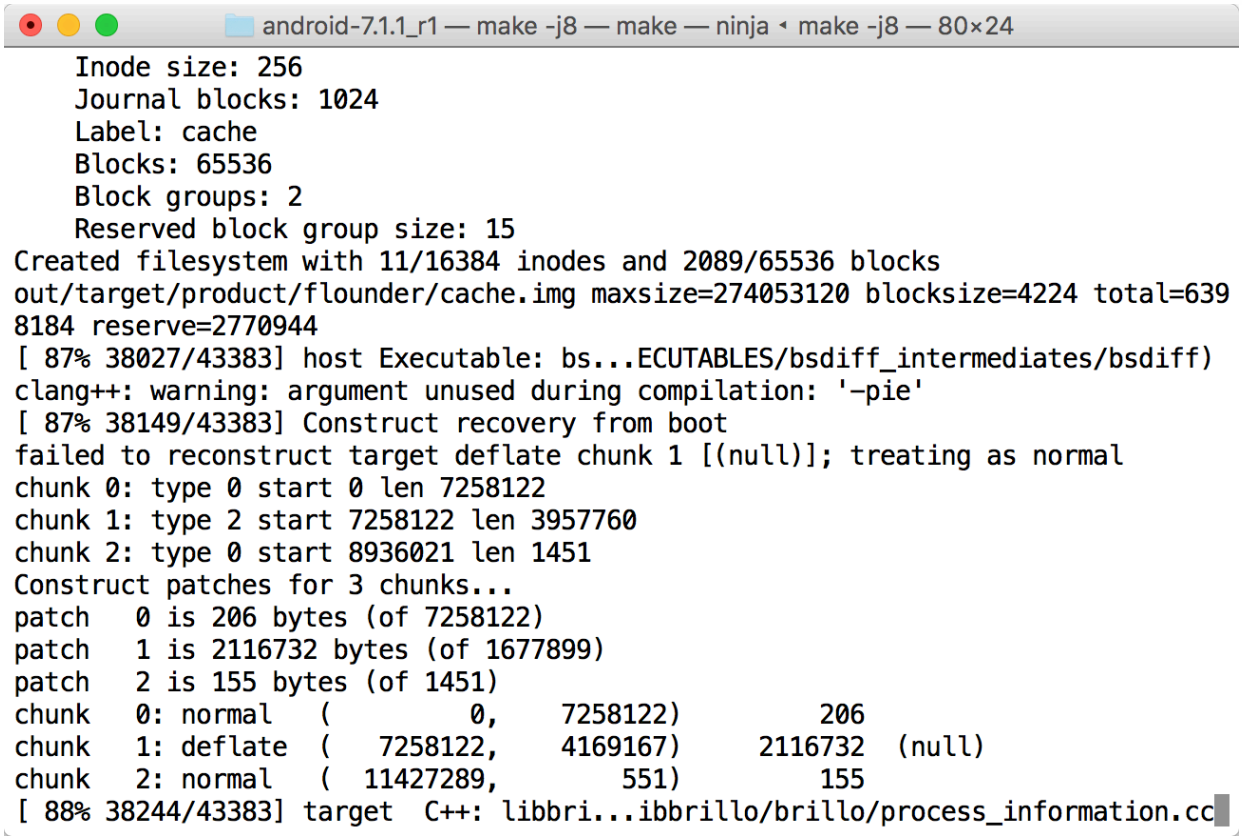
接下来在终端中切换到/Volumes/android目录，下载系统源码。以Android 7.1.1_r1为例，执行以下命令：

```
$ cd /Volumes/android
$ repo init -u https://android.googlesource.com/platform/manifest -b
android-7.1.1_r1
$ repo sync --force-sync --force-broken
```

下载安装后，执行如下命令即可编译：

```
$ export USE_CCACHE=1
$ mkdir ccache
$ export CCACHE_DIR=ccache
$ prebuilts/misc/darwin-x86/ccache/ccache -M 50G
$ sudo xcode-select -s /Applications/Xcode.app/Contents/Developer
$ export JAVA_HOME=$(/usr/libexec/java_home -v 1.8)
$ source build/envsetup.sh
$ lunch aosp_angler-userdebug
$ make clobber
$ brew uninstall curl
$ brew install curl --with-openssl
$ export PATH=$(brew --prefix curl)/bin:$PATH
$ caffeinate make -j8
```

macOS系统自带的`curl`不包含`openssl`库提供的SSL安全通信，会导致编译失败，需要使用`HomeBrew`重新安装并调整它的搜索路径。最后一行的`caffeinate`命令保证编译时系统不会进行休眠。如果以上操作没有出错，则整个编译过程就开始了，如图所示：



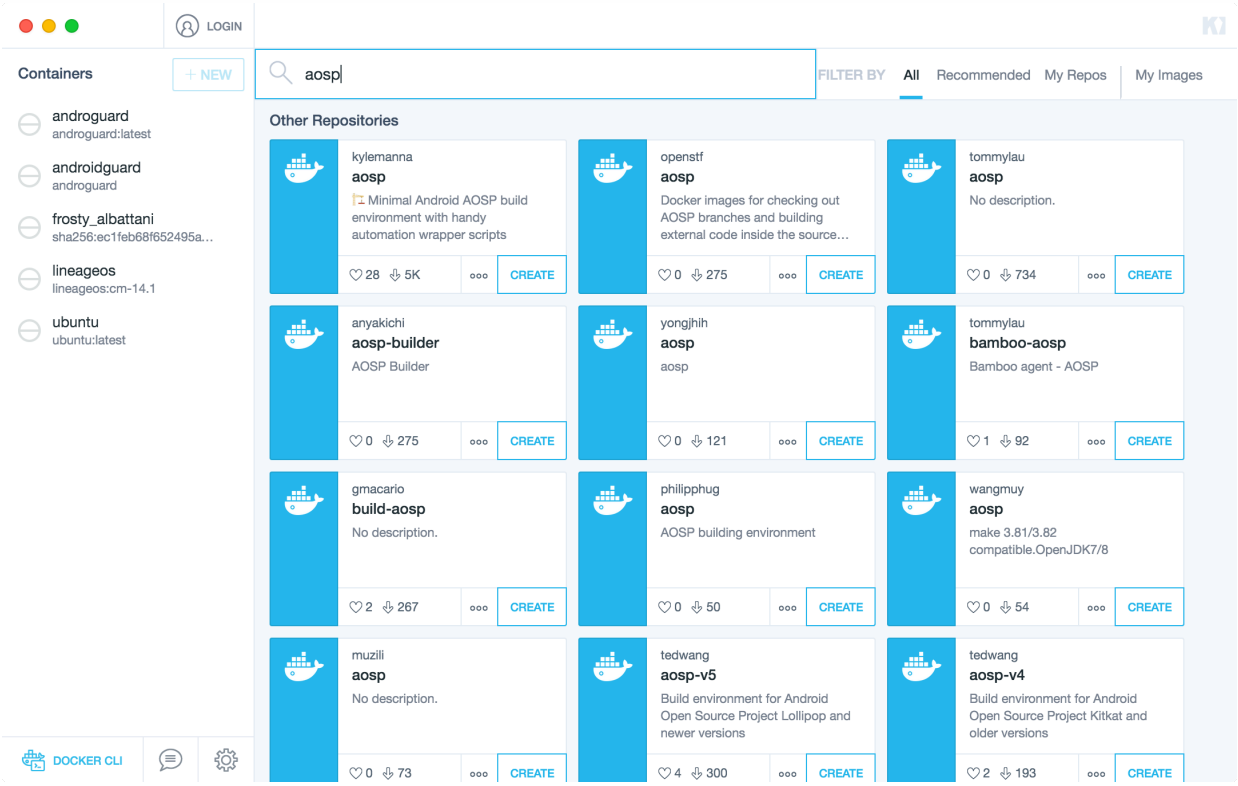
最新版本的Android源码引入了`Ninja`编译系统，在提高了源码编译速度的同时，还可以实时查看整个编译过程的进度，非常的人性化。

接着是使用`Docker`编译Android系统编译。在编译源码前，需要先安装配置好`Docker`。最简单的安装方式是执行如下命令安装：

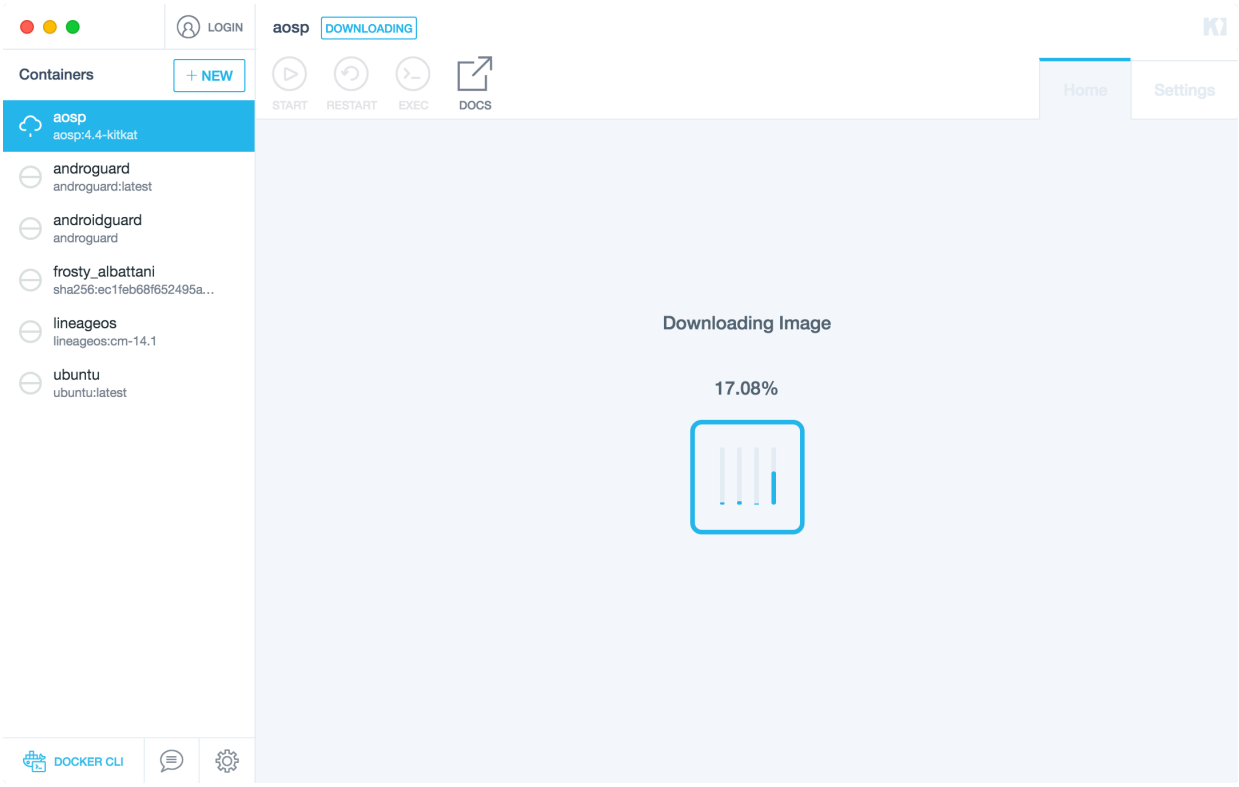
```
$ brew install docker
```

不过这种方式安装的`Docker`没有GUI管理界面，在操作上不够直观。`Docker`官方提供了`Docker ToolBox`的macOS版本，可以到官网https://www.docker.com/products/docker-toolbox 进行下载，目前macOS的稳定版本的下载链接为https://download.docker.com/mac/stable/DockerToolbox.pkg 。双击下载的DockerToolbox.pkg安装程序即可进行安装。安装完成后，在/Applications目录下会多出`Docker.app`、`Kitematic (Beta).app`、`Docker Quickstart Terminal.app`这三个软件。`Docker.app`是`Docker`的容器软件，在使用`Docker`时，必须首先启动它。启动`Docker.app`，会在系统的状态栏多出一个蓝色的小鲸鱼图标，`Docker Quickstart Terminal.app`用于在命令行下操作`Docker`，这里使用不到。启动`Kitematic (Beta).app`，会打开`Docker Hub`显示页面，在这里可以下载来自`Docker`官方与第三方提供的`Docker`镜像。

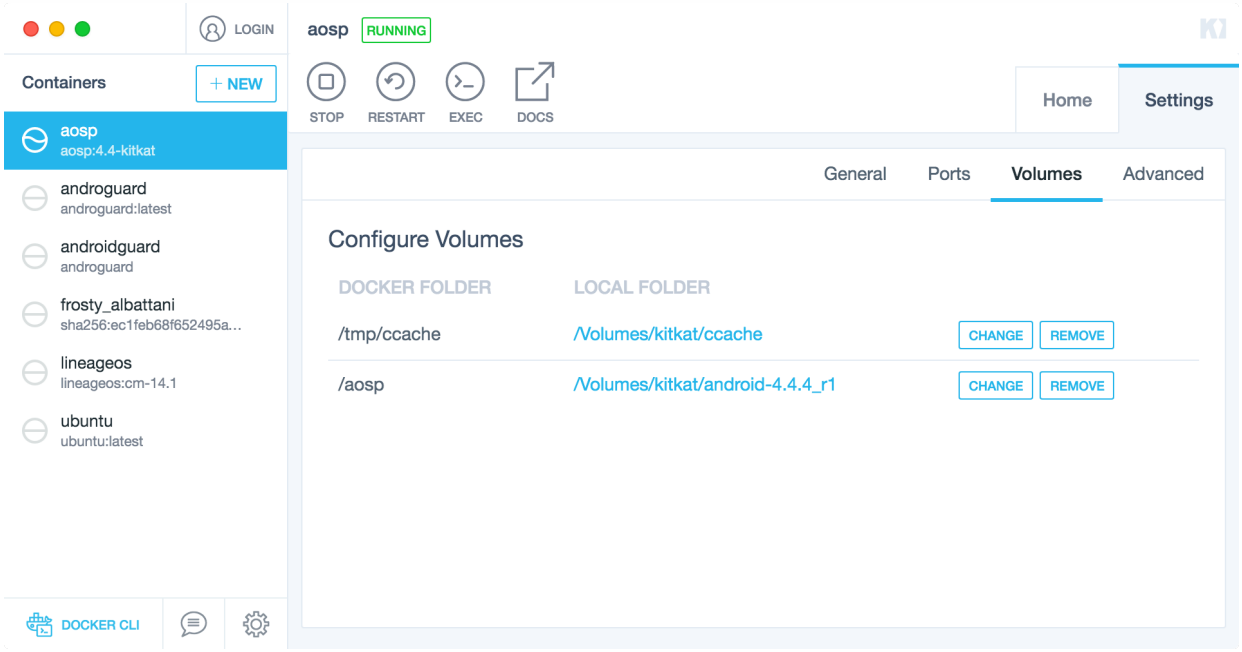
在`Kitematic (Beta).app`的主界面上，搜索“aosp”查找编译Android源码系统的环境。会列出所有的搜索结果，如图所示：



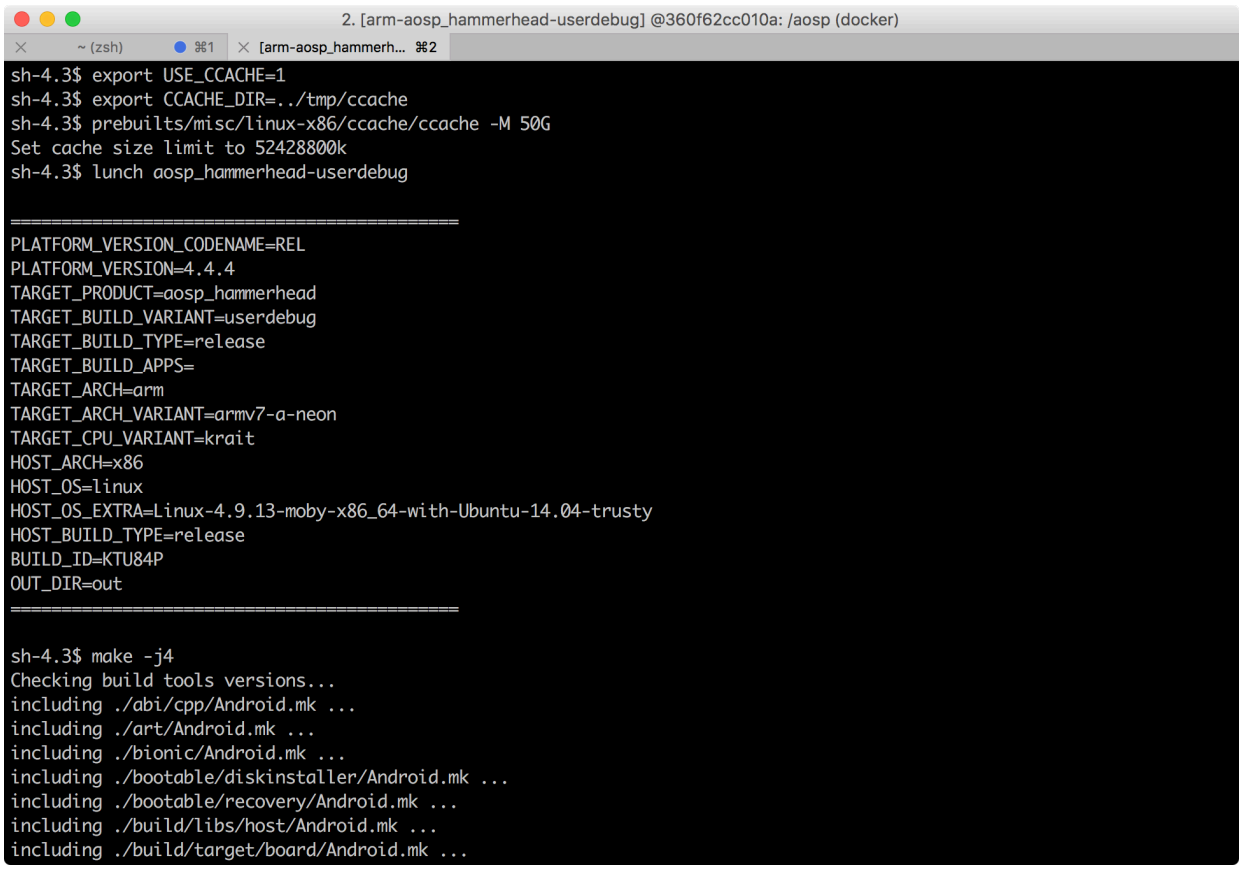
以第一个使用率最高的aosp镜像编译Android 4.4的源码为例，点击它下边的三个小圆点按钮，选择版本4.4 ，然后返回后点击Create，开始下载镜像，如图所示：



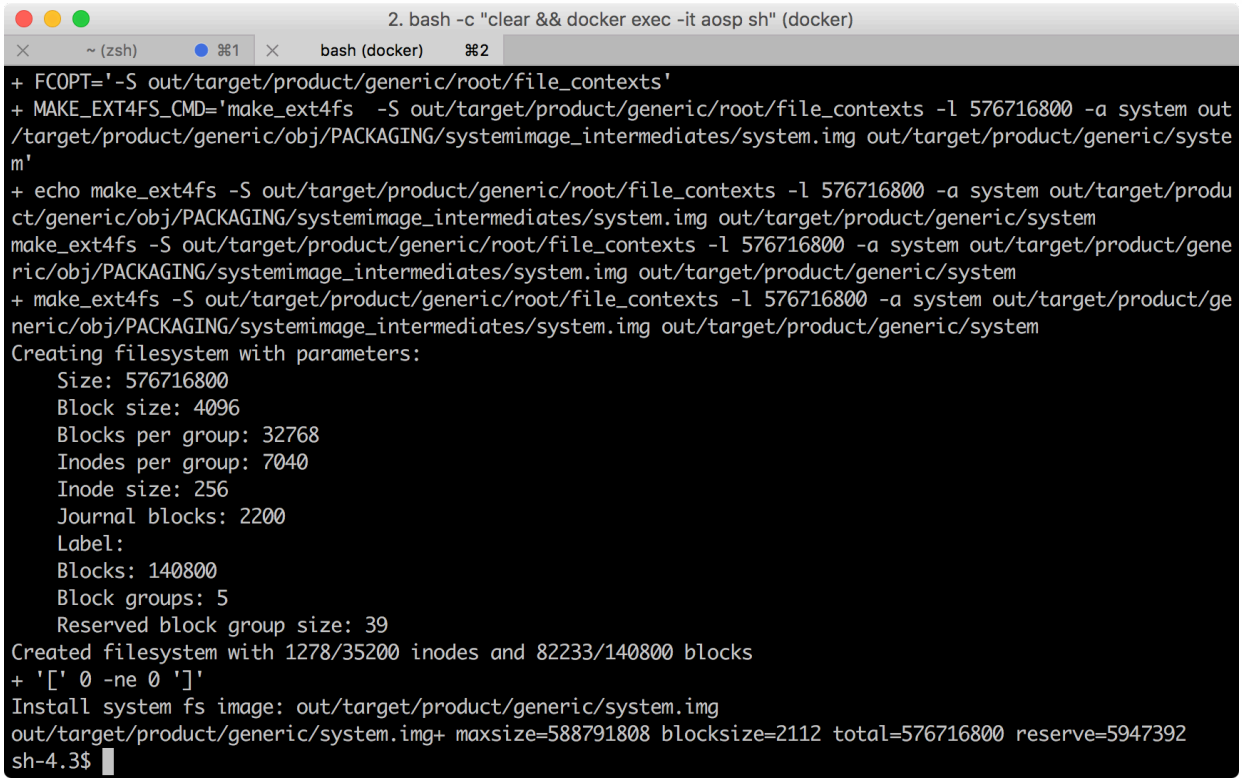
下载完成后，选中aosp镜像，点击左侧最下边的设备按钮，修改Android 4.4.4_r1的源码位置，以及ccache的缓存位置，如图所示：



配置好后，点击界面的RESTART重启镜像，然后点击EXEC打开Shell，此时，aosp镜像已经帮我们设置好了编译参数与环境变量，可以直接编译Android源码了。当然，如果不放心，可以在这里手动设置环境变量，一切操作完成后执行`make -j4`开始编译，如图所示：



根据系统的内存与配置，整个编译过程会持续不同长短的时间。编译完成后如图所示：



小结

本篇主要介绍了在macOS平台上，如何搭建安卓的开发与分析环境，以及如何在macOS系统上编译安卓系统的源码。后面，我们还会介绍如何在Linux与Windows上如何完成同样的工作。