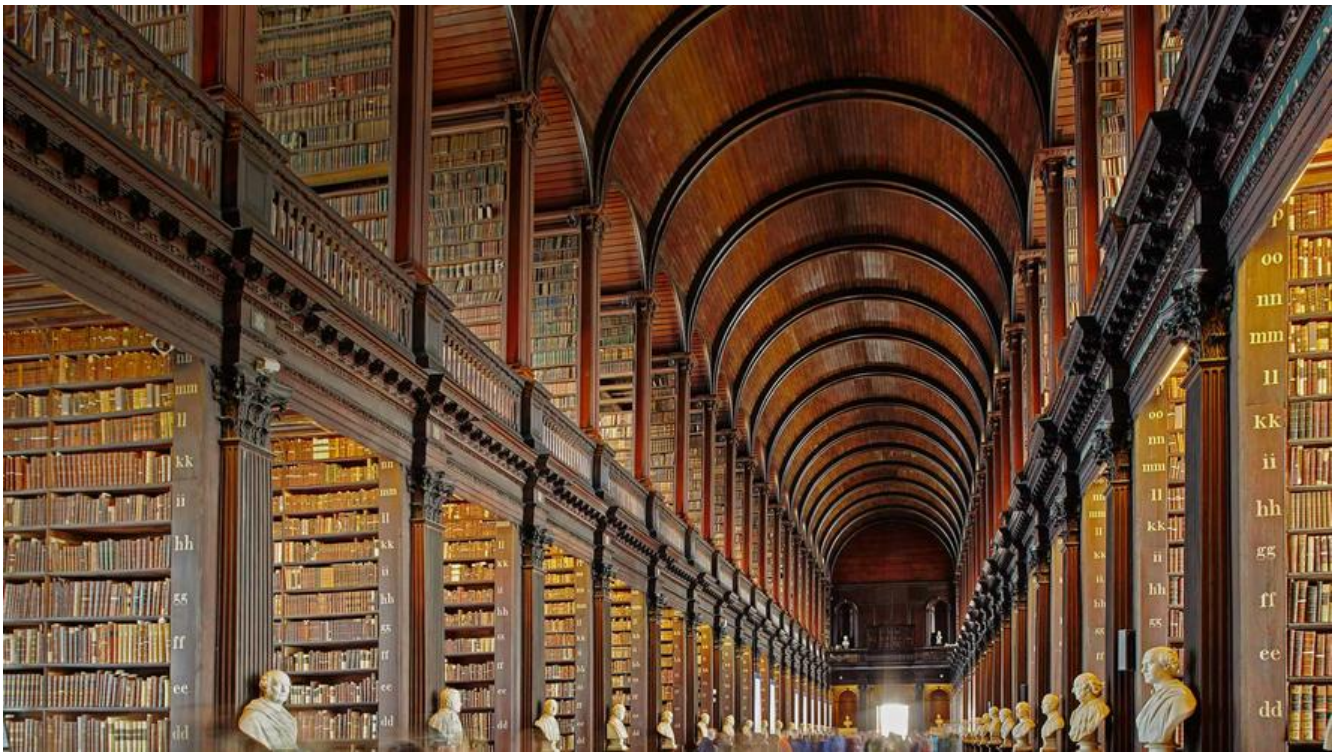
链滴

Unity 编译 mono 库

作者：bingqilin89757

原文链接：https://ld246.com/article/1537455295351

来源网站：链滴



环境说明

- mac osx 10.12.6
- unity 2017.4.6f1

步骤

- 安装相关的工具

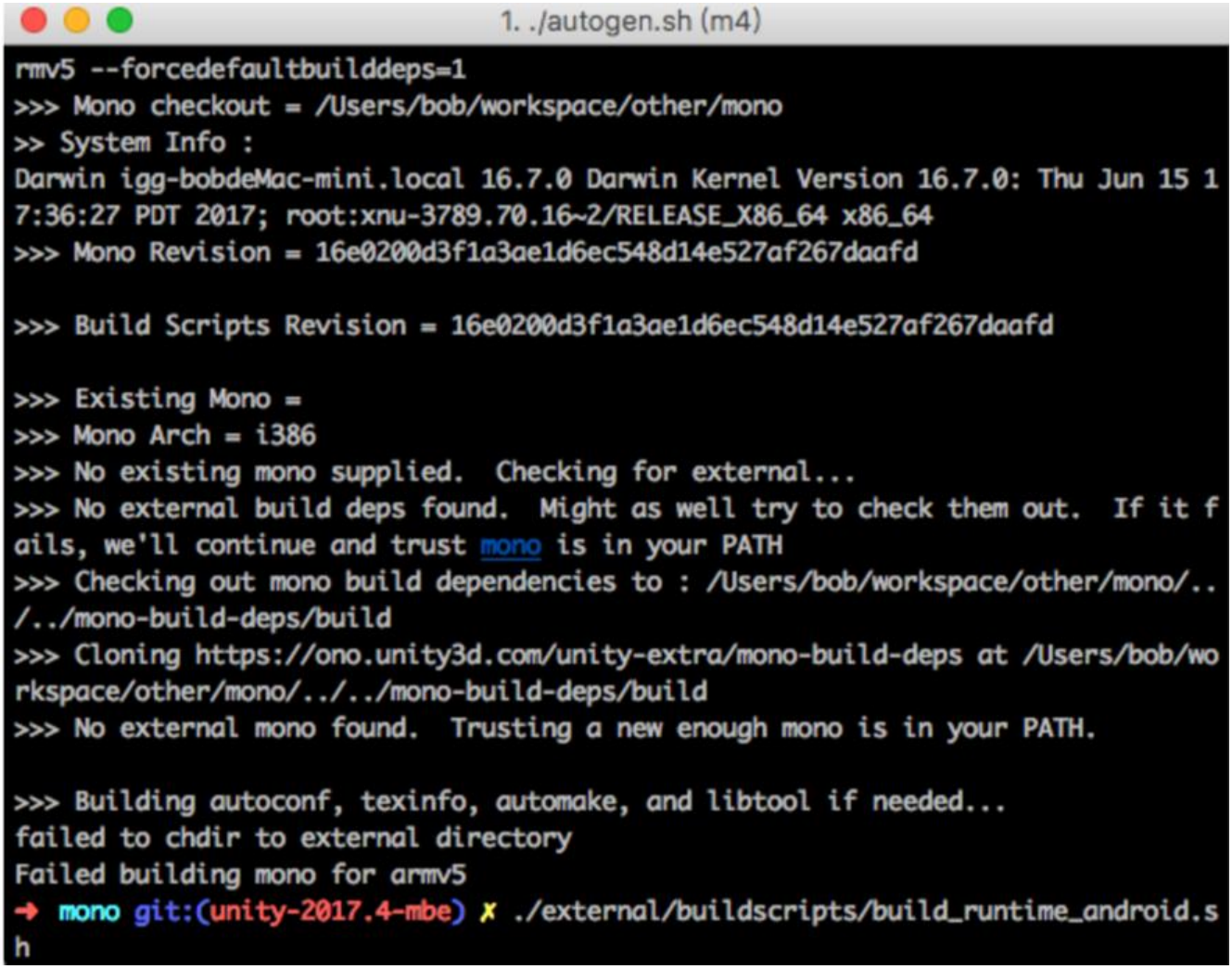
```
brew install autoconf automake libtool pkg-config
```

- 克隆目录并跳转到目录下切换unity版本分支

```
git clone https://github.com/Unity-Technologies/mono.git
cd mono
git checkout unity-2017.4
```

- 初始化子模块
- 赋权并执行编译

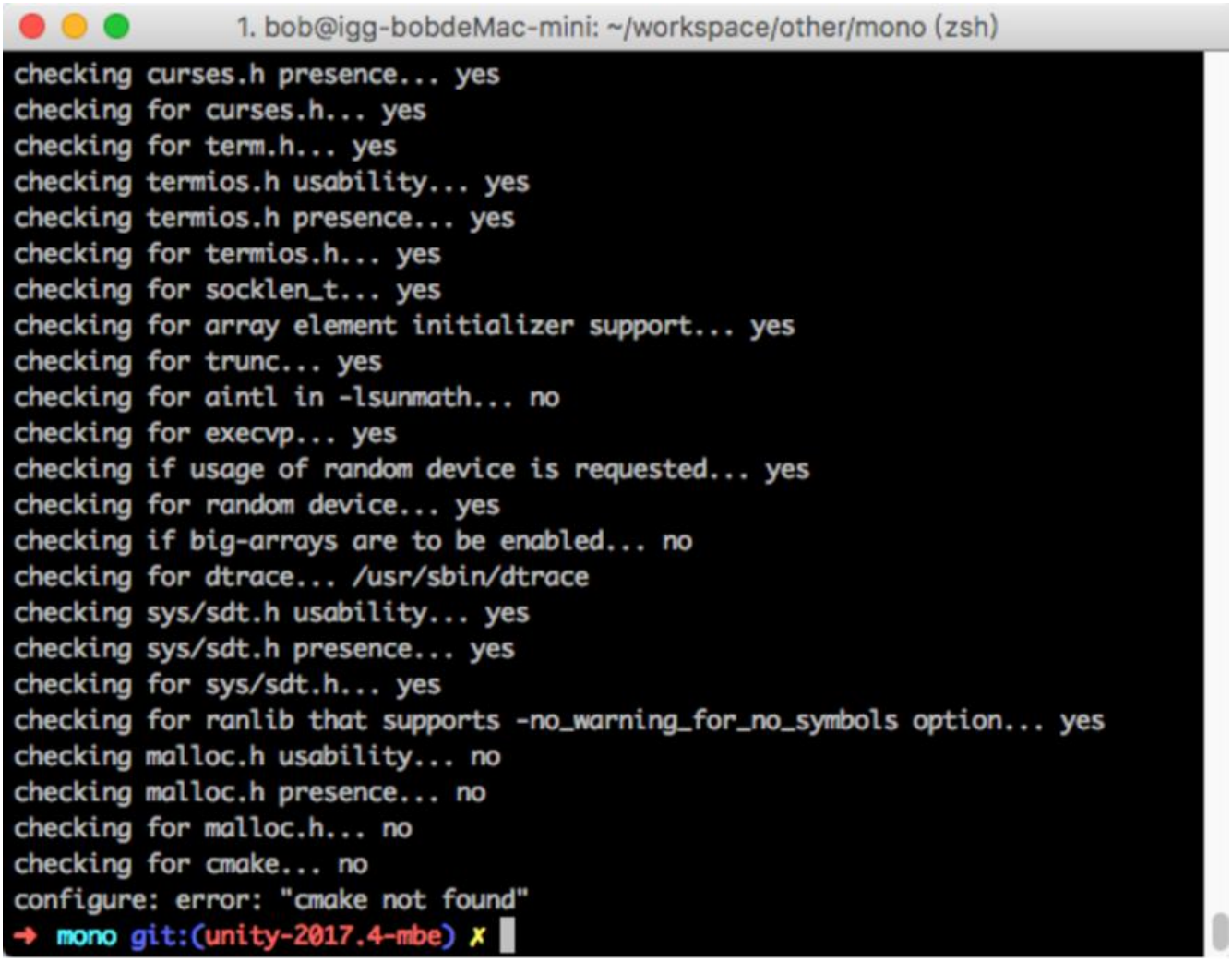
```
./external/buildscripts/build_runtime_android.sh
```



- 遇到错误,赋权并运行 autogen.sh

./autogen.sh

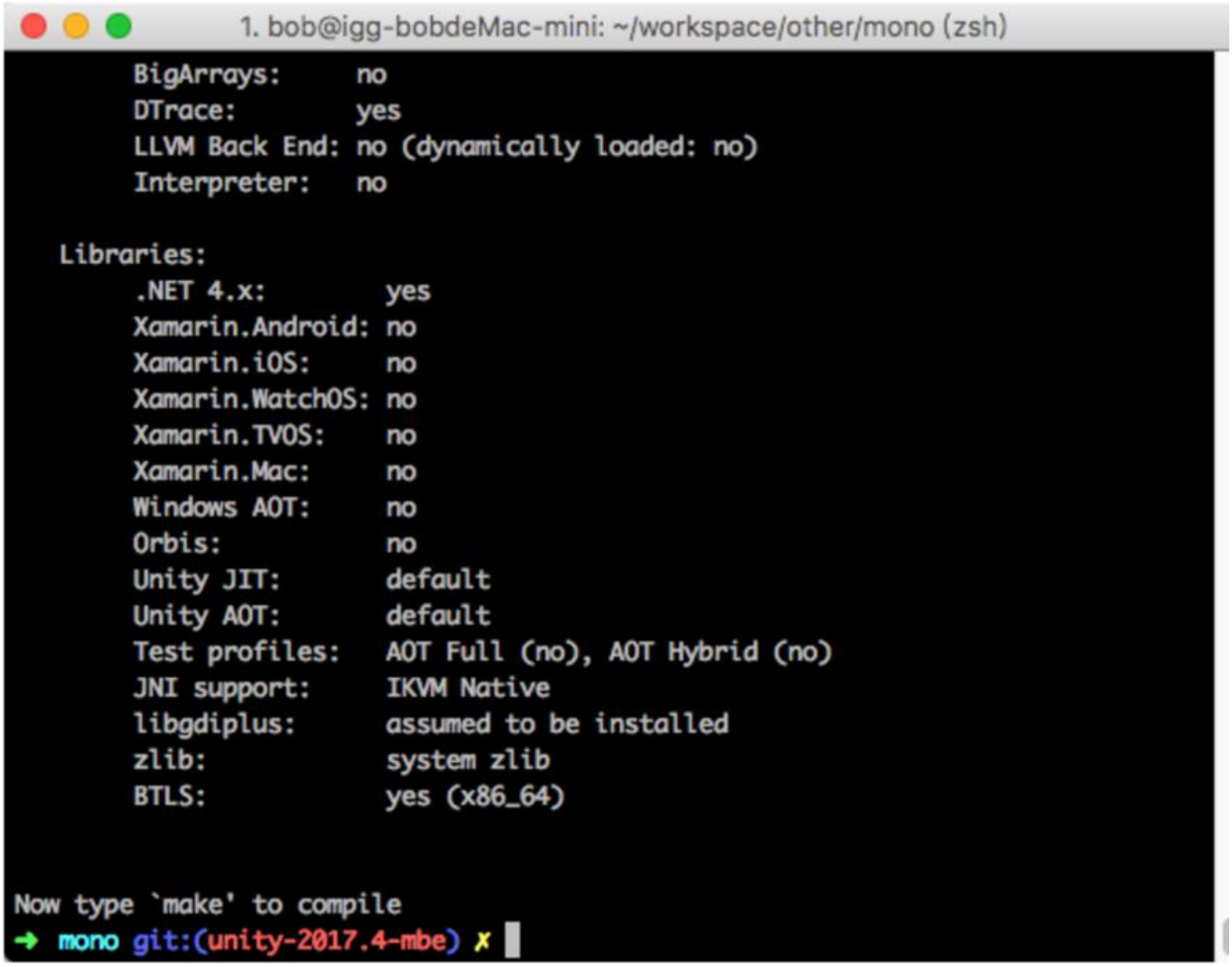
原文链接：Unity 编译 mono 库



- 遇到错误,安装cmake

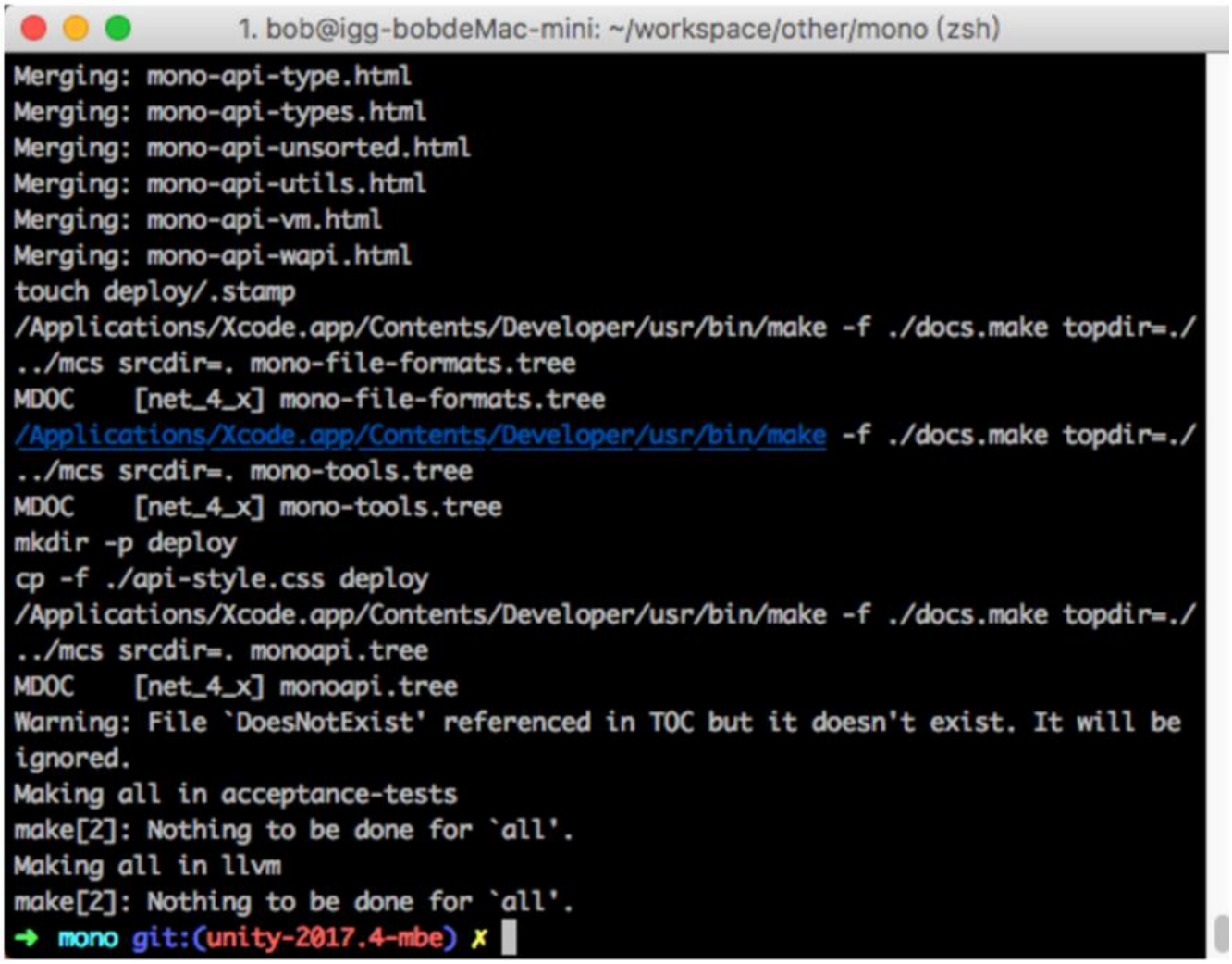
brew install cmake

- 再次运行autogen.sh,经过漫长的刷屏编译

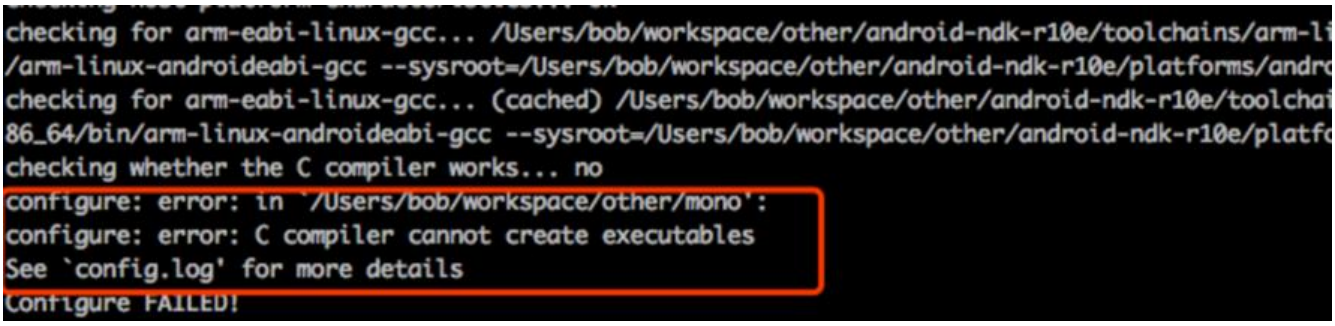


- 输入make进行编译

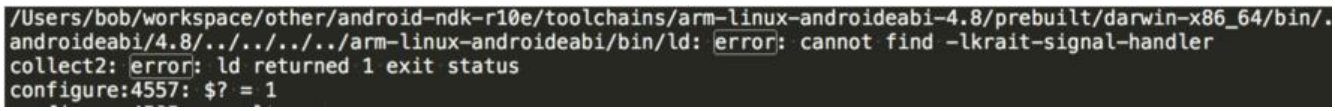
```
make
```



- 再次执行build_runtime_android.sh编译报错



- 查看日志



- 怀疑是ndk版本问题导致开始分析代码

```
 99        echo "Erasing builds folder to make sure we start with a clean slate"
100        rm -rf builds
101    fi
102
103    function clean_build_krait_patch
104    {
105        local KRAIT_PATCH_REPO="git://github.com/Unity-Technologies/krait-signal-handler.git
106        if [ ${UNITY_THISISABUILDMACHINE:+1} ]; then
107            echo "Trusting TC to have cloned krait patch repository for us"
108        elif [ -d "$KRAIT_PATCH_PATH" ]; then
109            echo "Krait patch repository already cloned"
110        else
111            git clone --branch "master" "$KRAIT_PATCH_REPO" "$KRAIT_PATCH_PATH"
112        fi
113        (cd "$KRAIT_PATCH_PATH" && ./build.pl)
114    }
```

定位到实际上是build.pl在搞鬼,mono这里编译需要的ndk版本是10e,但是krait-signal-handler要求ndk版本是13b,注释掉krait-signal-handler工程中的PrepareAndroidSDK.pm中PrepareNDK代码

PrepareAndroidSDK.pm — android_krait_signal_handler

```
236    if ($sdk)
237    {
238        PrepareSDK($sdk);
239    }
240    print "\n";
241}
242
243if ($ndk)
244{
245    print "Installing NDK '$ndk':\n";
246    if (!$ENV{$NDK_ROOT_ENV})
247    {
248        $ENV{$NDK_ROOT_ENV} = catfile($HOME, "android-ndk_auto-" . $ndk);
249        print "\t\$$NDK_ROOT_ENV not set; using $ENV{$NDK_ROOT_ENV} instead\n";
250    }
251
252    # PrepareNDK($ndk);
253    print "\n";
254}
255
```

- 再次执行build_runtime_android.sh 编译成功
- 解决编译的so包大小问题

经过对比发现,编译出来的so很大,查了资料发现是debug版本,需要修改两个编译的sh

build_runtime_android.sh

修改

```
CFLAGS="\
-DANDROID -DPLATFORM_ANDROID -DLINUX -D__linux__ \
-DHAVE_USR_INCLUDE_MALLOC_H -DPAGE_SIZE=0x1000 \
-D_POSIX_PATH_MAX=256 -DS_IWRITE=S_IWUSR \
-DHAVE_PTHREAD_MUTEX_TIMEDLOCK \
-fpic -O2 -funwind-tables \
-ffunction-sections -fdata-sections"
```

为了加快打包速度,注释掉

```
# clean_build "$CCFLAGS_ARMv5_CPU" "$LDFLAGS_ARMv5" "$OUTDIR/armv5"
# clean_build "$CCFLAGS_ARMv6_VFP" "$LDFLAGS_ARMv5" "$OUTDIR/armv6_vfp"
```

build_runtime_android_x86.sh

修改

```
CFLAGS="\
-DANDROID -DPLATFORM_ANDROID -DLINUX -D__linux__ \
-DHAVE_USR_INCLUDE_MALLOC_H -DPAGE_SIZE=0x1000 \
-D_POSIX_PATH_MAX=256 -DS_IWRITE=S_IWUSR \
-DHAVE_PTHREAD_MUTEX_TIMEDLOCK \
-fpic -O2 \
-ffunction-sections -fdata-sections"
```

- 重新编译即可,编译生成库文件在builds/embedruntimes/android目录下