**不同平台架构下的OpenCV编译方法**

## 1. 准备环境

(1). Opencv源码版本: 2.4.13

(2). Opencv源码路径: /home/zhaodong\_li/openCV-2.4.13/

(3). 编译平台: Ubuntu

(4). NDK: /opt/android-ndk-r16b

(5). 设置ANDROID\_NDK的环境变量

在~/.bashrc中设置ANDROID\_NDK的值：export ANDROID\_NDK=/opt/android-ndk-r16b

source .bashrc 使刚才的设置生效

(6). 安装cmake以及依赖库(Ubuntu平台已安装)

sudo apt-get install cmake

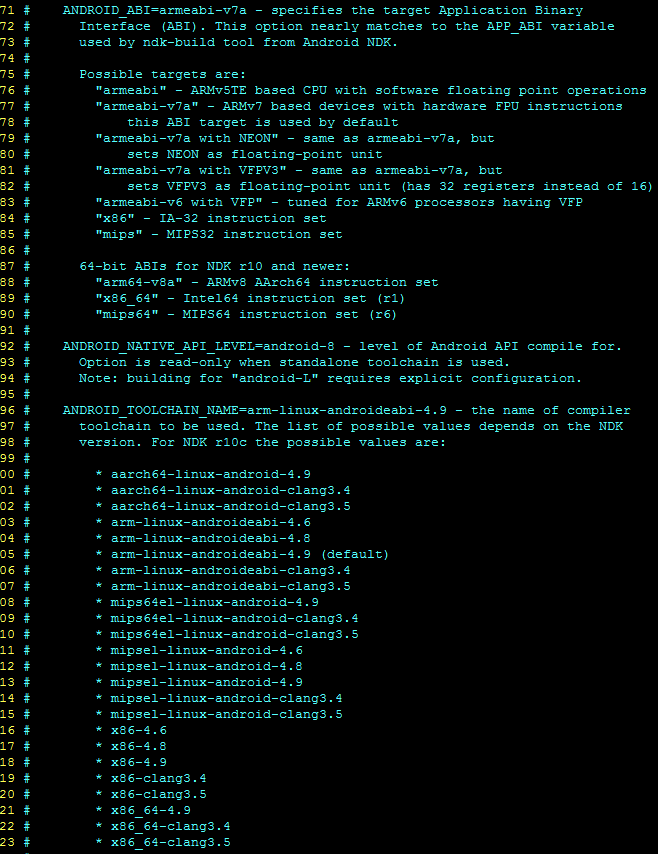
sudo apt-get install build-essential libgtk2.0-dev libavcodec-dev libavformat-dev

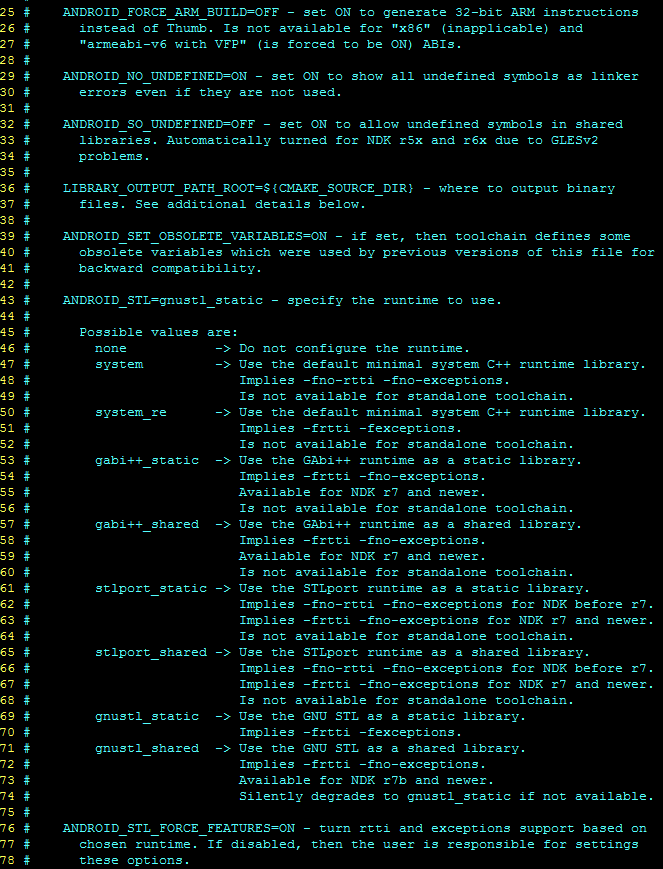
libjpeg.dev libtiff4.dev libswscale-dev libjasper-dev

## 2. OpenCV配置移动平台(目前针对的是android平台, 如有需要可以编译ios平台)选项说明(重要)

见该文件：/home/zhaodong\_li/openCV-2.4.13/platforms/android/android.toolchain.cmake





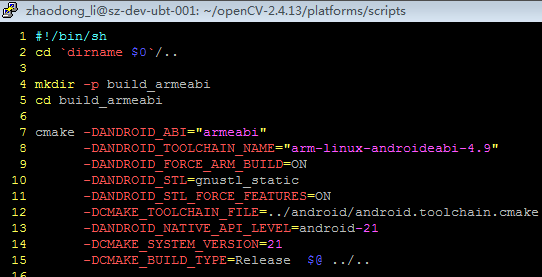


## 3. 不同平台下的OpenCV配置

### (1). armeabi架构

见该文件: /home/zhaodong\_li/openCV-2.4.13/platforms/scripts/cmake\_android\_armeabi.sh

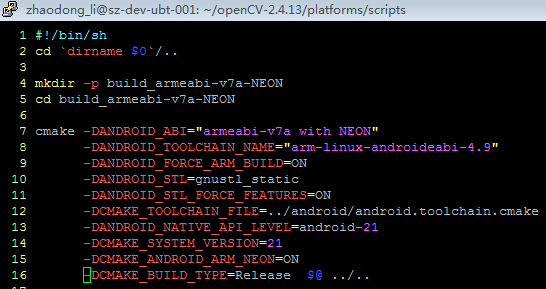
内容如下:



### (2). armeabi-v7a with NEON架构

见该文件: /home/zhaodong\_li/openCV-2.4.13/platforms/scripts/cmake\_android\_armeabi-v7a-NEON.sh

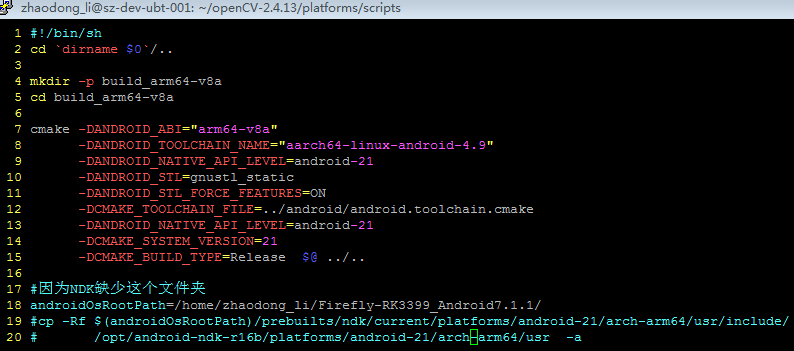
内容如下:



### (3). arm64-v8a架构

见该文件: /home/zhaodong\_li/openCV-2.4.13/platforms/scripts/cmake\_android\_arm64-v8a.sh

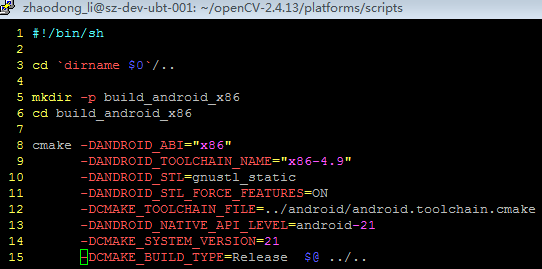
内容如下:



### (3). x86架构

见该文件: /home/zhaodong\_li/openCV-2.4.13/platforms/scripts/cmake\_android\_x86.sh

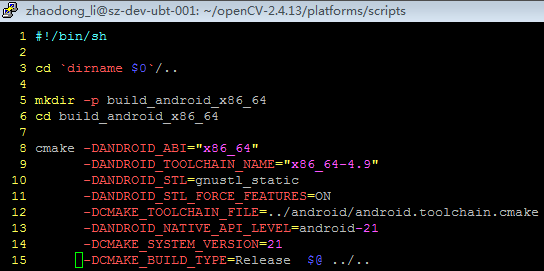
内容如下:



### (4). x86\_64架构

见该文件: /home/zhaodong\_li/openCV-2.4.13/platforms/scripts/cmake\_android\_x86.sh

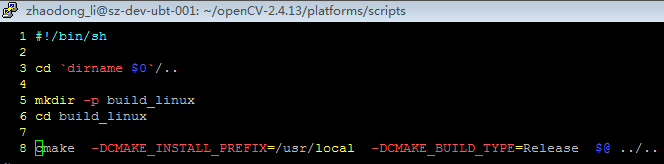
内容如下:



### (5). Linux发行版---Ubuntu

见该文件: /home/zhaodong\_li/openCV-2.4.13/platforms/scripts/cmake\_linux-ubuntu.sh

内容如下:



参考链接:

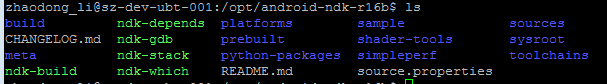
<https://blog.csdn.net/solomon1558/article/details/51967280>

<https://github.com/jayrambhia/Install-OpenCV>

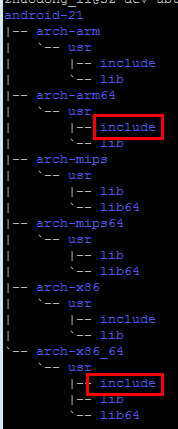
这两篇文章都使用的是默认配置, 没有涉及如何可配置的编译Ubuntu下的opencv库，比如“--ffast-math”默认是没有被配置的，该选项可以提供opencv的运算速度。如何客制化配置，下文会讲！

### (6). 以上配置选项的说明

1. 讲配置之前，先看下NDK的几个重要目录结构



查看platforms的目录结构 查看platforms/android-21的目录结构,其余结构都与其相同

 图1  图2

查看toolchains的目录结构 查看toolchains/ arm-linux-androideabi-4.9的目录结构

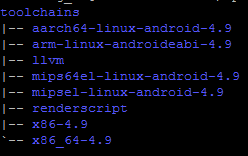
 

图3 图4

(1). 图2中红框处的include文件夹在android-ndk-r16b中不存在，两个红框处的include文件夹来自android源码自带的ndk下，任何最新的android 源码下面都可以找到这个include文件夹.

(2). toolchains/ arm-linux-androideabi-4.9/ prebuilt/linux-x86\_64/bin 目录下面的内容就是32位arm架构下的交叉编译工具, 见图中绿色部分；aarch64-linux-android-4.9是64位arm架构下的交叉编译工具, 其他架构类似。

2. 配置选项含义

(1). ANDROID\_ABI --- 指cpu的架构，主流的架构包含armeabi, armeabi-v7a, armeabi-v7a with NEON, arm64-v8a, x86, x86\_64,手机中很少见到使用x86和x86\_64平台，但是在MID设备中很常见。

(2). ANDROID\_TOOLCHAIN\_NAME --- 工具链的名字，对应的是android-ndk-r16b中图2的名字

(3). ANDROID\_FORCE\_ARM\_BUILD --- 使用32位的arm指令集来代替Tumb指令

(4). ANDROID\_STL --- runtime时使用的库，默认是gnustl\_static, 也可以改为gnustl\_shared

(5). ANDROID\_STL\_FORCE\_FEATURES --- 使c++支持 rtti和exception

(6). CMAKE\_TOOLCHAIN\_FILE --- cmake工具链的路径, 这里指定的是相对路径, 不同的opencv版本该路径可能不同

(7). ANDROID\_NATIVE\_API\_LEVEL --- android api的级别

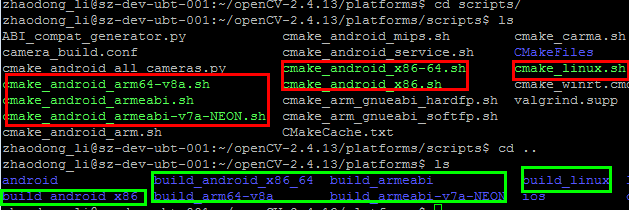
(8). CMAKE\_SYSTEM\_VERSION--- 为不同版本的主机操作系统启用的构建目标

(8). CMAKE\_BUILD\_TYPE ---- 编译出的文件是Release版本还是Debug版本

## 4. 不同平台下的OpenCV的客制化配置

上面的配置选项只是对opencv的一个初步配置，由于opencv有非常多的配置选项，要想按需定制opencv的功能，需要精细化的配置。配置方法如下:

1. 如下图所示，红框处的脚本生成了opencv相应平台的工程源码，对应平台的工程源码见绿色框.



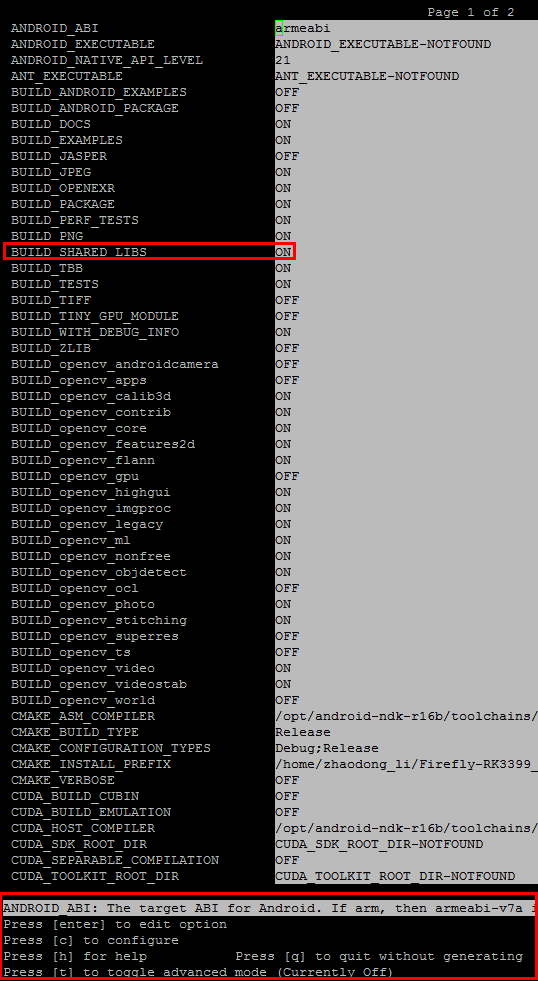
1. 精细化配置opencv, 这里以armeabi平台为例

(1). cd build\_armeabi

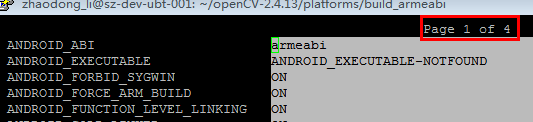
(2). Ccmake 空格 . .代表当前目录

(3). 执行完步骤(2),可见如下界面

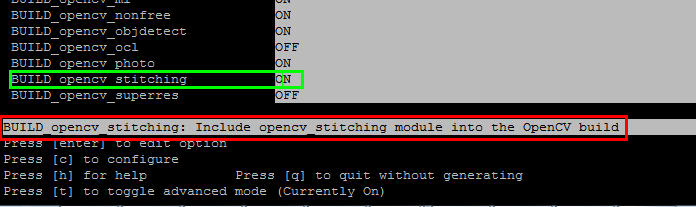
该选项BUILD\_SHARED\_LIBS 默认是OFF，这里配置成了ON, 即编译出来的是动态库



(4). 按键盘上面的字母t,进入高级模式，高级模式会多出很多配置选项



每个选项的含义都在这里描述：

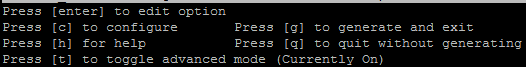


例如：当前光标在绿色框处，绿色框选项的说明见红色框. 每个选项的含义都需要自己去看

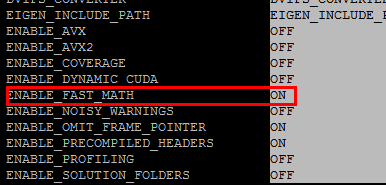
(5). 配置完成后按字母c, 会弹出如下界面, 再按下字母e



(6).再按下字母g, 生成并退出



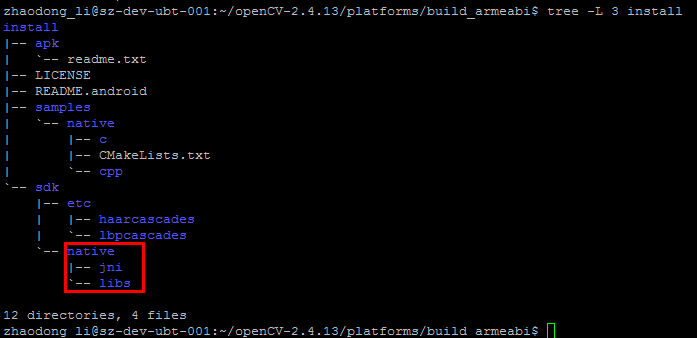
说明：intel架构平台请使能该选项, 可提高opencv的计算速度



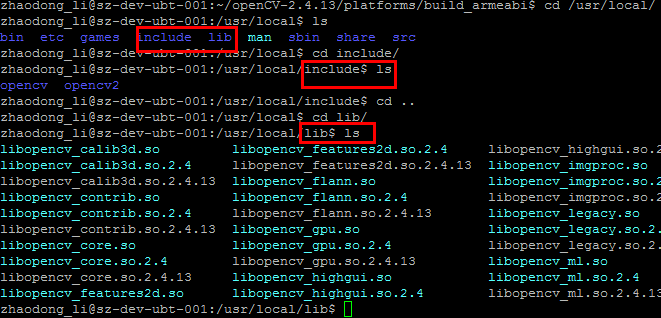
## 5. 不同平台下的OpenCV的编译安装

(1). 编译: make -jN, 10.71服务器N=16, 因为服务器是8核的，每个核启动2个线程，所以是16

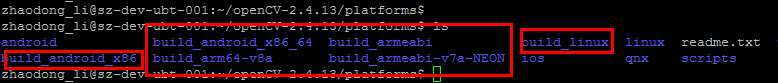
(2). 安装: make install , 生成的include和lib文件，见下图红框处；特别说明的是, Ubuntu发行版在安装的时候需执行该命令：sudo make install ,否则没权限创建目录



说明：Ubuntu发行版的opencv 生成在该路径下：



## 6. 目前已经编译出了不同OpenCV平台下的库



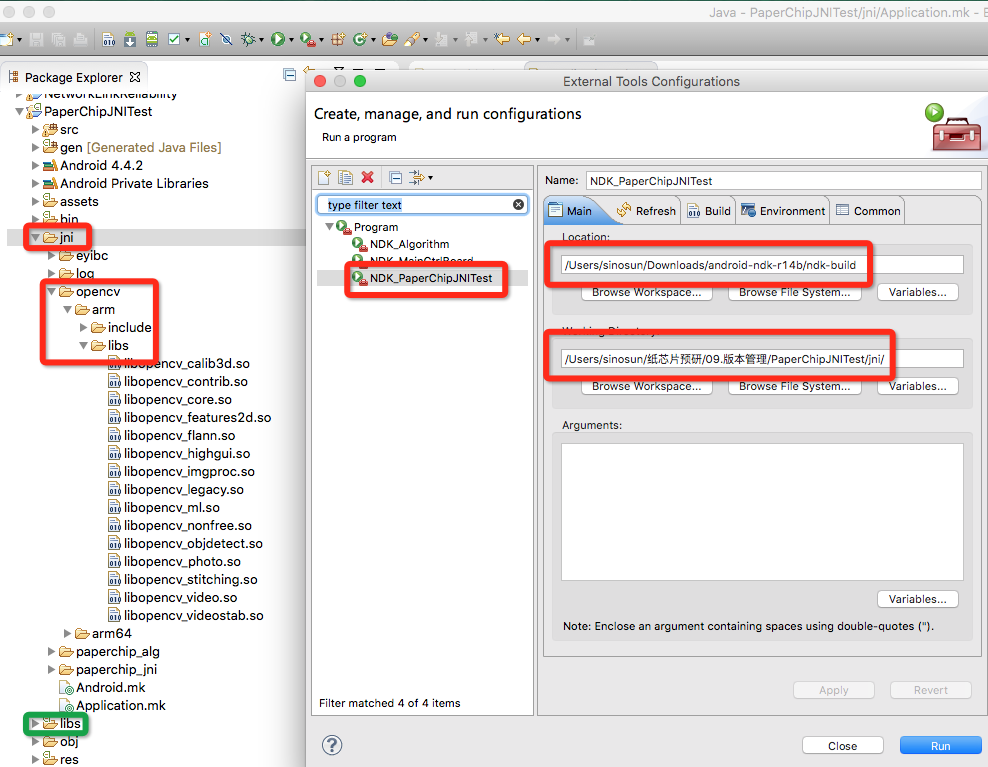
## 7. 如何在Eclipse(Mac)中配置NDK

(1) . 下载window平台的NDK版本

(2). 在Eclipse的JNI目录中引入opencv库, 配置如下：

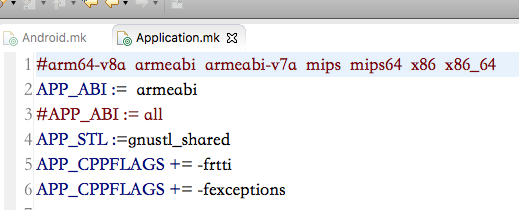
示例工程在该路径：

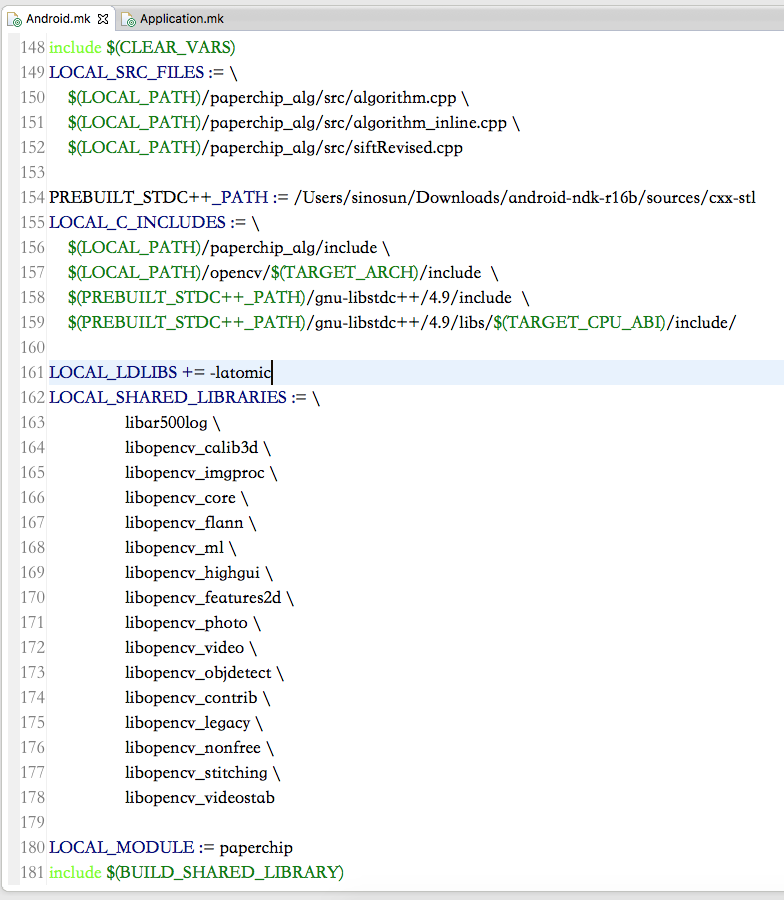




绿色方框是最终生成动态库的目录

(2). 为JNI编译Android.mk和Application.mk文件



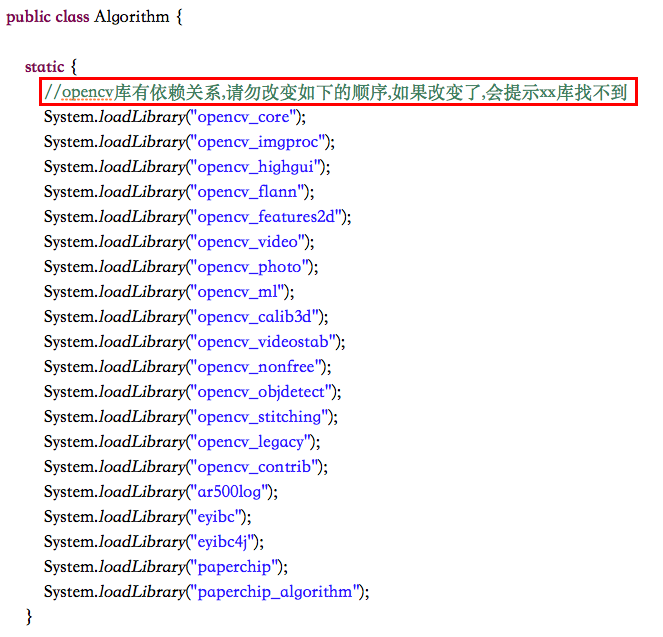


说明：如果遇到该错误 undefined reference to '\_\_atomic\_fetch\_add\_4'，请在Android.mk文件引入该库：

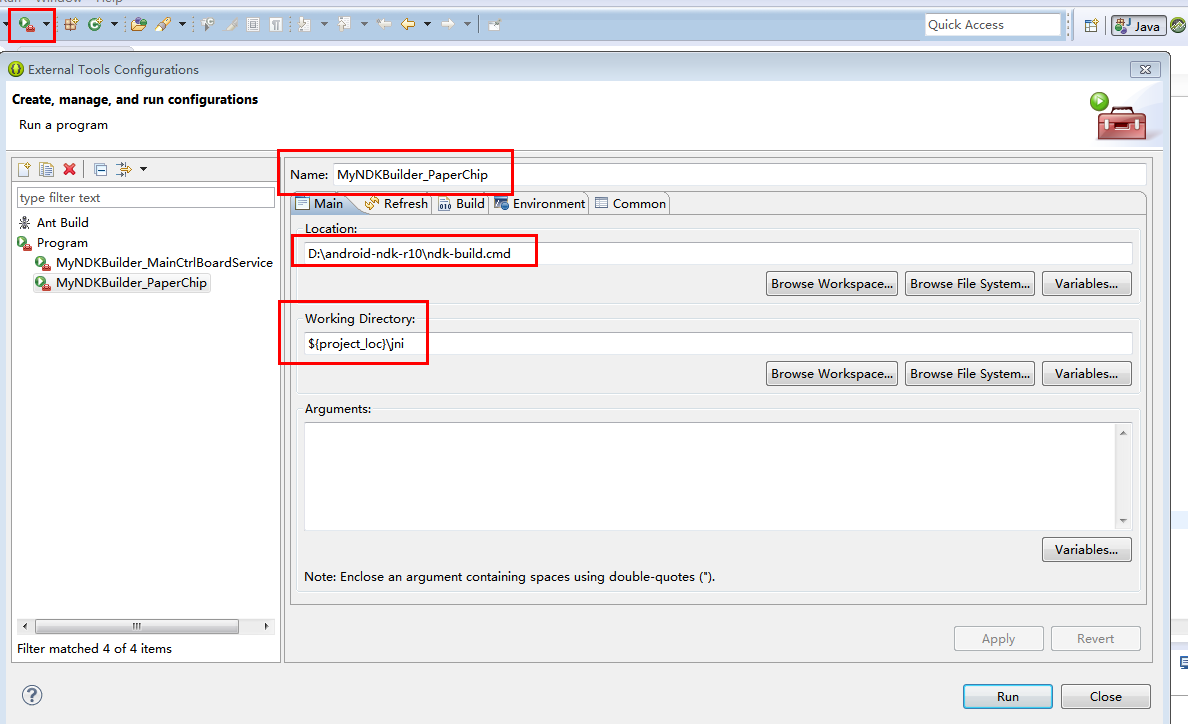
LOCAL\_LDLIBS += -latomic

(3). Java层如何加载opencv的动态库

Java层在加载opencv库的时候，发现有依赖关系，该依赖关系的顺序已经调好了，否则会提示XX库找不到.



## 8. 如何在Eclipse(Window)中配置NDK



说明：在window系统中，Android工程不能包含中文路径，否则编译jni目录里面的文件时，找不到目标路径.

例如该路径：D:\纸芯片预研\04.终端软件\02.源代码\01.trunk\PaperChipJNITest\jni, 是编译不出.so文件的.

## 9. NDK编译STL

1.在jni目录下新建Application.mk; 加入APP\_STL :=  stlport\_static  右边的值还可以换成下面几个：

**system** - 使用默认最小的C++运行库，这样生成的应用体积小，内存占用小，但部分功能将无法支持

**stlport\_static** - 使用STLport作为静态库

**stlport\_shared** - STLport 作为动态库，这个可能产生兼容性和部分低版本的Android固件，目前不推荐使用。

**gnustl\_static**  - 使用 GNU libstdc++ 作为静态库

默认情况下STLPORT是不支持C++异常处理和RTTI，所以不要出现-fexceptions 或-frtti ，如果真的需要，可以使用gnustl\_static来支持标准C++的特性，但生成的文件体积会偏大，运行效率会低一些。

支持C++异常处理，在Application.mk中加入LOCAL\_CPPFLAGS += -fexceptions这句，同理支持RTTI，则加入LOCAL\_CPPFLAGS += -frtti。

 强制重新编译STLPort ，在Application.mk中加入STLPORT\_FORCE\_REBUILD := true 可以强制重新编译STLPort源码，由于一些原因可能自己需要修改下STLPort库，一般普通的开发者无需使用此项

2.在要使用STL的cpp文件中包含相关的头文件，并且使用using namespace std;

补充:

1. APP\_STL的取值：
2. system(default)系统默认的C++运行库
3. stlport\_static以静态链接方式使用的sttport版本的STL
4. stlport\_shared以动态链接方式使用的sttport版本的STL
5. gnustl\_static以静态链接方式使用的gnustl版本的STL
6. gnustl\_shared以动态链接方式使用的gnustl版本的STL
7. gabi++\_static以静态链接方式使用的gabi++
8. gabi++\_shared以动态链接方式使用的gabi++
9. c++\_static以静态链接方式使用的LLVM libc++
10. c++\_shared以动态链接方式使用的LLVM libc++
11. 就目前的情况，LLVM的支持要优于GNU，所以推荐使用“c++\_static”
12. 由于Android系统的碎片化，各个版本的libgnustl\_share.so不一致，就导致崩溃或者运行异常问题**鉴于gnustl\_share存在诸多不稳定问题，所以尽量采用gnustl\_static替代gnustl\_share。**
13. **使用gnustl\_static，尽量避免使用gnustl\_share。此外gnustl将逐步废弃！**

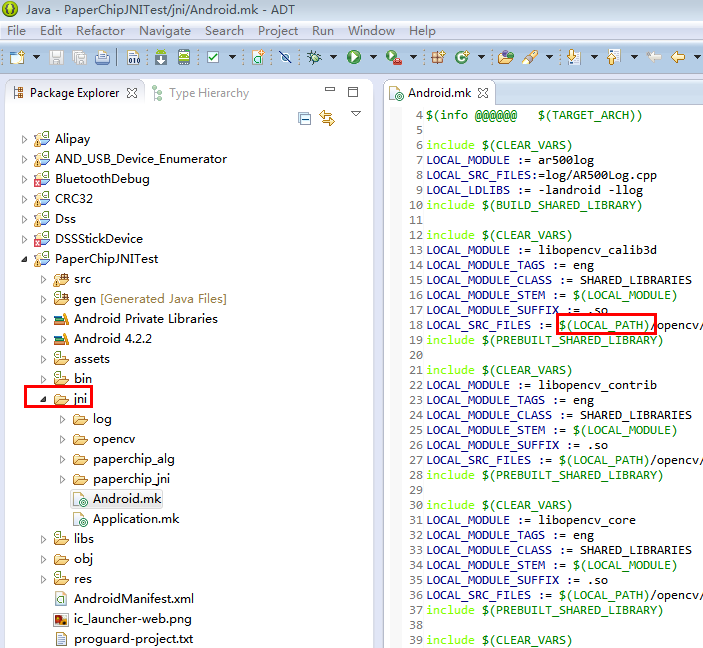
## 10. NDK编译中遇到的问题

(1). 如果使用VS2013 + visualGDB开发JNI, android.mk如果如下这样写会报错, 因为在该环境下,

LOCAL\_SRC\_FILES := jni/opencv/arm/libs/libopencv\_calib3d.so会变成jni/jni/opencv/arm/libs/libopencv\_calib3d.so,

为了在该环境下和Eclipse中都可以编译，故将该参数的值LOCAL\_SRC\_FILES写成如下形式:

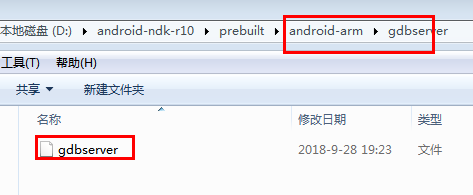
LOCAL\_SRC\_FILES := opencv/arm/libs/libopencv\_calib3d.so



(2). 如果使用VS2013 + visualGDB开发JNI时遇到如下问题，说明NDK中没有对应平台的gdbserver



例如ndkr10版本就没有如下红框中的路径，该路径系我自己添加：



补充：

OpenCV-3.4.2 Ubuntu系统下编译方法：

以下内容以编译armeabai目标平台为例。

在/home/zhaodong\_li/opencv-3.4.2/platforms/scripts目录下创建编译脚本cmake\_android\_armeabi.sh，脚本内容如下：

#!/bin/sh

cd `dirname $0`/..

mkdir -p build\_armeabi

cd build\_armeabi

cmake -DANDROID\_ABI="armeabi" -DANDROID\_TOOLCHAIN\_NAME="arm-linux-androideabi-4.9" -DBUILD\_JAVA=FALSE -DBUILD\_EXAMPLES=FALSE -DBUILD\_ANDROID\_EXAMPLES=FALSE -DINSTALL\_PYTHON\_EXAMPLES=FALSE -DINSTALL\_C\_EXAMPLES=FALSE -DANDROID\_FORCE\_ARM\_BUILD=ON -DANDROID\_STL=gnustl\_static -DANDROID\_STL\_FORCE\_FEATURES=ON -DCMAKE\_TOOLCHAIN\_FILE=../android/android.toolchain.cmake -DANDROID\_NATIVE\_API\_LEVEL=android-21 -DCMAKE\_SYSTEM\_VERSION=21 -DCMAKE\_BUILD\_TYPE=Release $@ ../..

其中“DBUILD\_JAVA=FALSE -DBUILD\_EXAMPLES=FALSE -DBUILD\_ANDROID\_EXAMPLES=FALSE -DINSTALL\_PYTHON\_EXAMPLES=FALSE -DINSTALL\_C\_EXAMPLES=FALSE”是在李沼东的基础上添加的，解决3.0以上版本出错的问题。

更改路径/home/zhaodong\_li/opencv-3.4.2/platforms/android 下的oid.toolchain.cmake内容，增加set( ANDROID\_SUPPORTED\_NDK\_VERSIONS ${ANDROID\_EXTRA\_NDK\_VERSIONS} -r16b -r10d -r10c -r10b -r10 -r9d -r9c -r9b -r9 -r8e -r8d -r8c -r8b -r8 -r7c -r7b -r7 -r6b -r6 -r5c -r5b -r5 "" )

# API level defaults

set( ANDROID\_DEFAULT\_NDK\_API\_LEVEL 21 )

然后执行脚本cmake\_android\_armeabi.sh，在/home/zhaodong\_li/opencv-3.4.2/platforms/build\_armeabi下生成文件OpenCV.mk文件。

运行ccmake .配置cmake编译参数。