调试ndk代码可以在eclipse或者android studio中，android studio的ndk代码动态调试已经非常强大了，之前调试开源项目ppsspp的c++代码，已经体会到非常便捷了。

下面是使用gdb远程调试ndk代码，入口函数是main（include $(BUILD\_EXECUTABLE)

），功能代码编码测试结束后，添加几个jni的接口，编译成so（include $(BUILD\_SHARED\_LIBRARY)），就可以提供给安卓应用程序使用了。

同时也是使用gdb调试ndk executable程序的方法，目前还没有找到Android Studio或eclipse远程调试Android ELF可执行文件的方法（可以调试so）。

可以在windows下做开发（熟悉vs的使用），在linux环境下动态调试；或者直接在linux环境下开发和调试。

下面的代码编写和调试在Ubuntu下完成，使用ndk r22。

1、代码

**mian.cpp**

#include <stdio.h>

long add(long a, long b){

printf("fun add\n");

return a + b;

}

int main(int argc, char\*\* argv){

long ret = 0;

ret = add(1,2);

printf("fun main: %ld\n", ret);

return ret;

}

**Android.mk**

LOCAL\_PATH := $(call my-dir)

include $(CLEAR\_VARS)

LOCAL\_MODULE := test-debug-ndk

LOCAL\_SRC\_FILES := main.cpp \

LOCAL\_ARM\_MODE := arm

LOCAL\_CFLAGS += -fPIE

LOCAL\_LDFLAGS += -fPIE -pie

include $(BUILD\_EXECUTABLE)

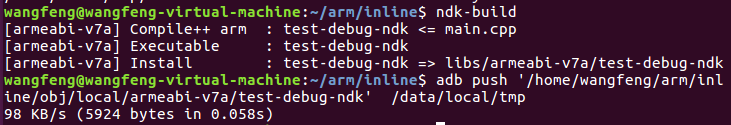
**Application.mk**

APP\_PLATFORM := android-16

APP\_ABI := armeabi-v7a

#APP\_STL := c++\_static

编译代码：



注意这里push的是obj\local\armeabi-v7a目录下的test-debug-ndk，这个是包含符号的

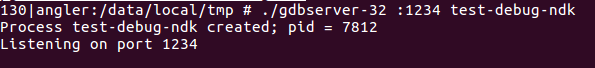
2、动态调试

terminal 1

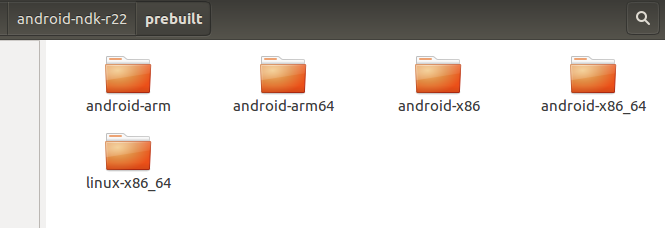
adb forward tcp:1234 tcp:1234

terminal 2

./gdbserver-32 :1234 test-debug-ndk（以启动程序的方式，./ gdbserver-32 :1234 –attach pid是以附加进程的方式）



注意这里的gdbserver-32是从ndk r22的android-arm从push到/data/local/tmp的



terminal 3

gdb-multiarch

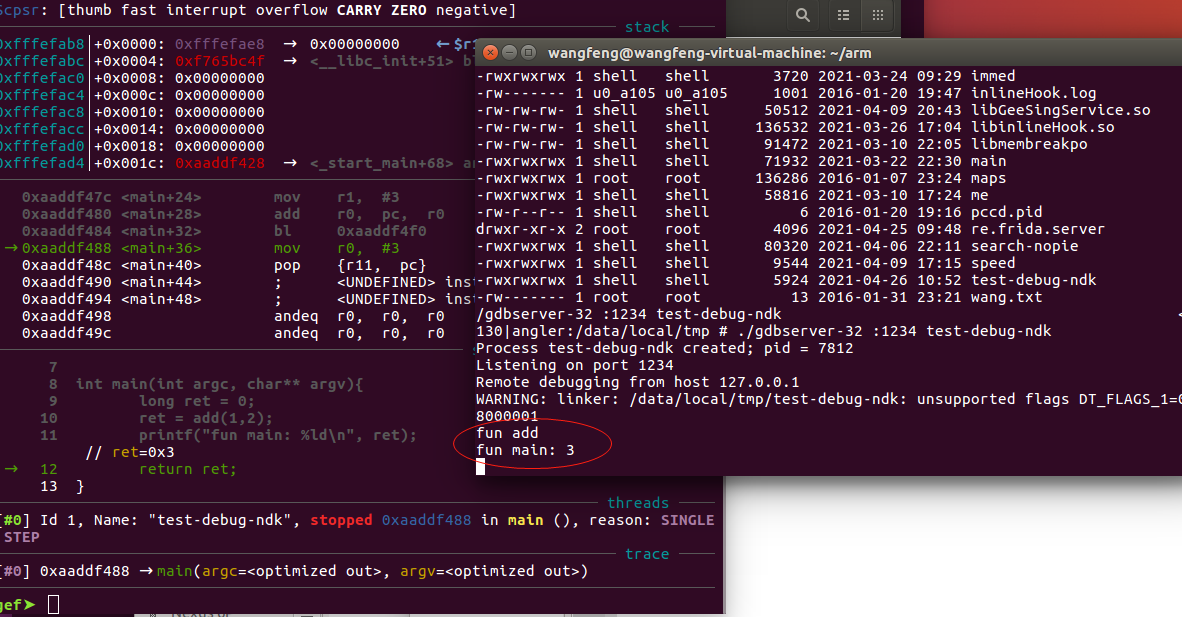
target remote :1234

b main

c

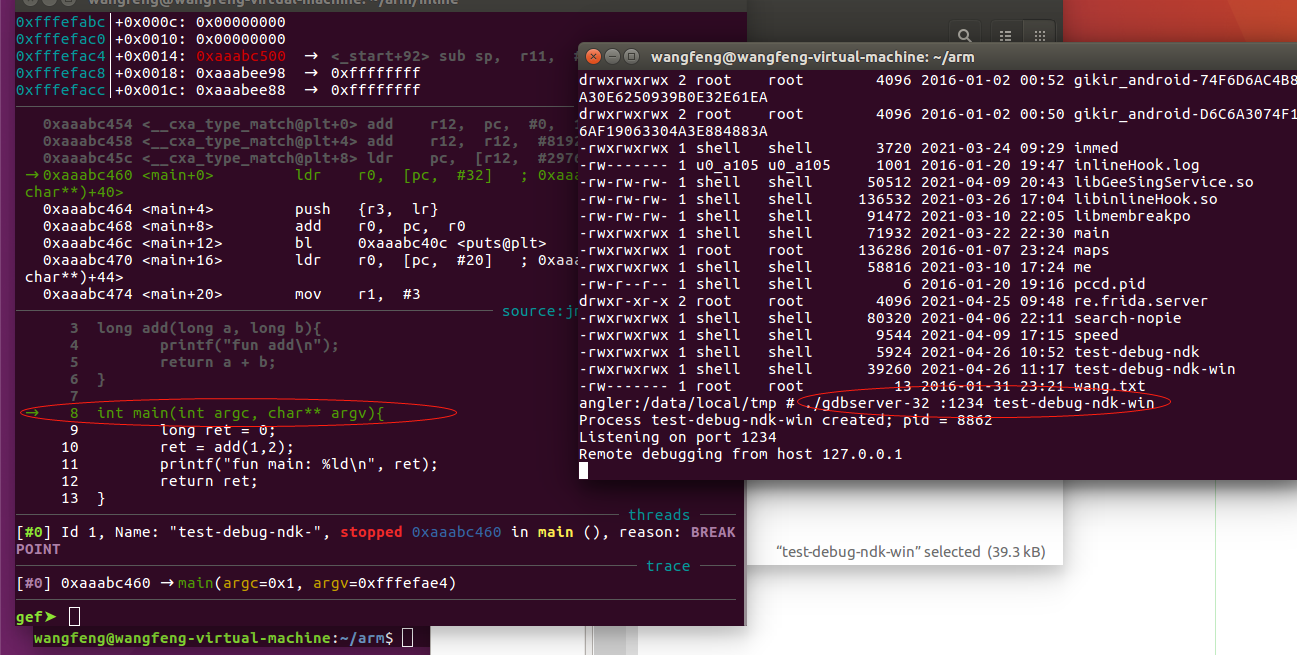


n

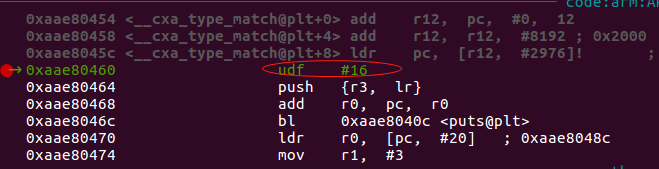


quit

尝试了下在windows下ndk r10e编译的程序test-debug-ndk-win，使用linux环境下ndk r22的gdbserver，也是可以动态调试的，这里注意使用obj\local\armeabi-v7a目录下的程序



问题：udf #16



遇到udf #16就无法单步调试了

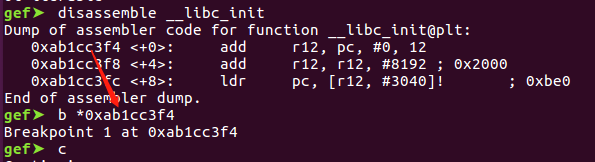
原因暂时还没有找到

有一种可以调试的情况：

在windows下面使用android-ndk-r10e编译生成的可执行程序

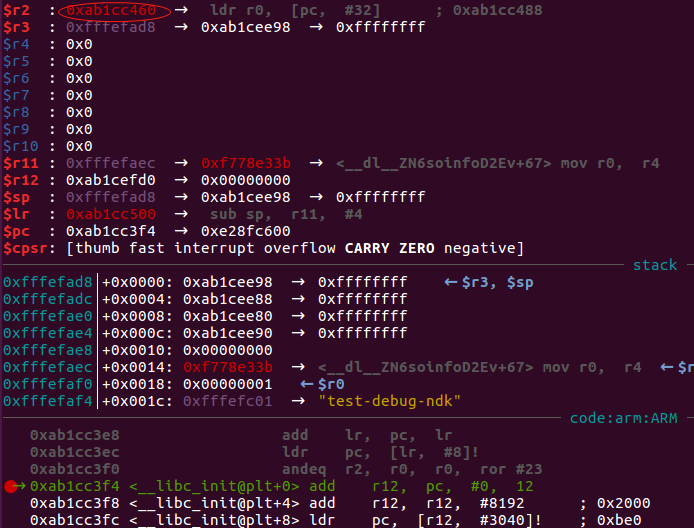
libs/armeabi-v7a下生成的elf是strip了的

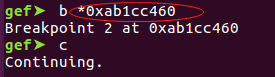


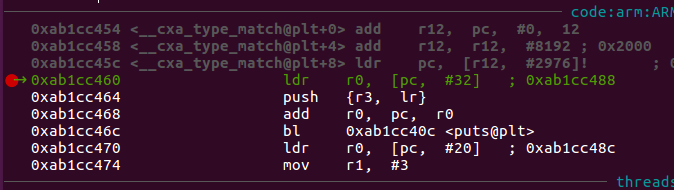


disassemble \_\_libc\_init，如果使用的linux下的android-ndk-r22编译生成的可执行程序，提示的是：

No symbol table is loaded. Use the "file" command.







这样可以单步调试

如果直接b \*0xaaf07460，出现udf #16的问题

