

 Native breakpoint works via lldb console but does not work if set in code editor.

+14

Hotlists (7)


Mark as Duplicate





Comments (16)DependenciesDuplicates (0)Blocking (0)Resources (7)

FixedBugP1




[AOSP] assigned

[AOSP] Released


[AOSP] win10

[AOSP] Version-Studio-2.2

Migrated-Tools-C++ Debugger

 STATUS UPDATE No update yet.

Edit

 DESCRIPTION jo...@gmail.com created issue [#1](#)

Studio Build: "Android Studio 2.2.3"
Version of Gradle Plugin:"com.android.tools.build:gradle:2.2.3"
Version of Gradle: "https://services.gradle.org/distributions/gradle-2.14.1-all.zip"
Version of Java: Using JDK embedded with Android Studio 2.2.3
OS: Windows 10

Issue: I am developing an application which uses Java and C++. Using the new ExternalNativeBuild functionality available in the Android Plugin 2.2 DSL, my project uses ndk-build files to compile and also links in some pre-built C++ binaries (previously compiled w/ ndk-build).

Using this approach, the project builds and runs as expected however I am having difficulty hitting breakpoints in *pre-built* libs when I set them via the editor. However, when I set the equivalent break when expected. Oddly enough, I am always able to hit a breakpoint in the app's JNI code regardless of how the breakpoint was set.

Steps to Reproduce:

Sadly the source being debugged can not be distributed as our company's license does not support that. So, I will instead describe the issue with steps & pictures and hopefully that will suffice.

1) Set a breakpoint in JNI_OnLoad. This can be done through the Editor or through the LLDB Console.
2) Start debugging the application, and see the breakpoint in JNI_OnLoad getting hit. (Screenshot1.png)
3) While the breakpoint in JNI_OnLoad is still active, set another breakpoint via the editor in some pre-built library function that will be executed after the program execution continues. (Screenshot2.png)
4) While the breakpoint in JNI_OnLoad is still active, open the LLDB console in the Debug Pane, and manually set another breakpoint in the same pre-built library function one line after the previous one.
5) Continue execution and you should see that the breakpoint set via the editor is skipped while the breakpoint set via the LLDB console is hit. (Screenshot4.png) This shows that the pre-built library is able to load them properly but Android Studio does not seem to be supplying the file / line information to LLDB correctly.


Notes: When trying to figure out the cause of this issue I did notice one peculiarity. After steps 2 & 3 shown earlier, I ran the command 'breakpoint list' via the LLDB console and compared what it output:


1: file = 'C:\Users\Alex\Perforce\alex_SATAN_2274\ts3d\dev\Visualize\MASTER\hps\internals\tests\fire\source\android\fire\app\src\main\jni\fire_jni.cpp', line = 11, exact_match = 0, locations 1.1: where = libfire.so::JNI_OnLoad(JavaVM *, void *) + 72 at fire_jni.cpp:11, address = 0xc8b6bdac, resolved, hit count = 1


2: file = 'C:\Users\Alex\Perforce\alex_SATAN_2274\ts3d\dev\Visualize\MASTER\hoops_3df\Dev_Tools\hoops_3dgs\source\std_draw_tree.cpp', line = 5689, exact_match = 0, locations = 0 (pending)


3: file = 'std_draw_tree.cpp', line = 5691, exact_match = 0, locations = 1, resolved = 1, hit count = 0
3.1: where = libhps_core.so::HD_Standard_Draw_Tree(HOOPS::Rendition_Pointer<HOOPS::Internal_Net_Rendition> const&, HOOPS::Segment const*, HOOPS::Include_Path const&) + 156 at std_draw_tree.cpp:5691, hit count = 0


As you can see, breakpoint #2 was set by the editor and references the full / absolute path of the file where the BP was set where the breakpoint set in the LLDB console lists only the file name (I should android studio only be supplying the filename to LLDB or is it correct to pass the whole path?


 **Screenshot1.PNG**
148 KB [View](#) [Download](#)


 **Screenshot2.PNG**
155 KB [View](#) [Download](#)

 **Screenshot3.PNG**
158 KB [View](#) [Download](#)


 **Screenshot4.PNG**
169 KB [View](#) [Download](#)

 Mentioned issues (1)

 Links (5)

 **Mentioned issues (1)**

P3 Android is not open Bluetooth 4.0 for developers "OK, nevermind. I've just seen that you filed [bug 36949180](#) . We can continue this there and close this,"

 **Links (5)**

"<https://services.gradle.org/distributions/gradle-2.14.1-all.zip>..."

"For reference, I was thinking of <https://code.google.com/p/android/issues/detail?id=225565> , which had a workaround of using absolute paths for building."

"...and gcc emit the filenames as given on the command line. Looks like cmake defaults to absolute paths based on <http://stackoverflow.com/questions/9607155/make-gcc-put-relative-filenames-in-object-files>"

"<http://goo.gl/pKn3Ah>"

"I think it is issuetracker.google.com/issues/37137270 ."

uc...@google.com <uc...@google.com> [#2](#)

Thank you for this feedback. The team may reach out for more information on triaging or reproducing this issue.

an...@google.com <an...@google.com> [#3](#)

Assigned to an...@google.com.

It needs to pass the whole path in general, otherwise having two files with the same base name will result in more breakpoints than intended.

Thanks for your initial investigation, helps us out a lot :)

Is the full path in breakpoint 2 correct? Looks like it is in Screenshot2.PNG.

There is a general issue with source mapping for externally built libraries. Basically it may have been built with the source mapping only having relative paths, which would make the absolute match. I consider this the most likely cause. I don't know much about ndk-build, but you may be able to fix this by setting LOCAL_PATH to the appropriate absolute path when building.

You can confirm this is the problem by looking at the debug_line section of the library, search for the base filename and look at the directory table for it. objdump can do this on linux, not sur

In case I'm wrong:

How is the prebuilt library being built? Does it have optimizations enabled, which could mean that no breakpoints are possible on line 5689? Can you try using the console to set a breakpoint example of a line where a breakpoint set from the editor could not be hit?

jo...@gmail.com <jo...@gmail.com> [#4](#)

[Comment deleted]

jo...@gmail.com <jo...@gmail.com> [#5](#)

- Yes, the full path presented in breakpoint 2 is in fact correct, it just doesn't seem to be reflected in the shared object.

- The prebuilt libraries are built via ndk-build as well (usually on the same machine) without any optimizations enabled. I can verify that I can hit a breakpoint on line 5689 if set through the LL

- By running objdump to reveal the filenames present in the resulting binaries, I can verify that the debug_line contains JUST the filename and not the whole path. However, I do not see any fe build. I've tried altering how our ndk-build makefiles (for our prebuilt libraries) reference the source files many ways but in no case has ndk-build associated debug symbols with an ABSOLUT that have RELATIVE paths to filenames but that did not fix any of the Editor based debugging issues.

So I guess I have two main questions after some investigation:

1) Since it seems Android Studio needs to attach a breakpoint with an absolute file path does that mean Android Studio will never support debugging native debug binaries that were built on would have absolute paths only valid on the other system)?

2) If a absolute path does need to be in the debug info in order for editor breakpoints to work generically, do you know if this problem could be resolved by switching our Android C++ build sy supports it)? i.e. Is there a way to force CMake to encode the full path into the debug info?

Thanks for any insights,
Alex

jo...@gmail.com <jo...@gmail.com> [#6](#)

After thinking about this a bit more, I realized I wanted to ask some additional questions before you responded.

3) The "breakpoints list" lldb command seems to show exact & partial match locations for a set breakpoint. Wouldn't it be possible to first set a breakpoint for just "filename.cpp", check to se is, then we know there are no filename collisions), then if there are multiple matches remove that bp and set a new one for the full path. If this is not feasible (multiple commands across lldb to do it in a single command.

4) If no approach listed previously is workable / feasible is there any way to override the behavior of how the breakpoint is set via the editor (i.e. what command it runs?)

Thanks again, and sorry for all the run-on sentences.
Alex

an...@google.com <an...@google.com> [#7](#)

For reference, I was thinking of <https://code.google.com/p/android/issues/detail?id=225565>, which had a workaround of using absolute paths for building.

Answers to your questions:

1) It does, it just takes a bit of setup. In the run configuration, you can add commands to run on lldb startup, you'll need something like 'settings set target.source-map /buildbot/path /my/pa works for relative files. Pavel (cc'ed) probably knows.

2) I bet clang and gcc emit the filenames as given on the command line. Looks like cmake defaults to absolute paths based on <http://stackoverflow.com/questions/9607155/make-gcc-put->

3) That wouldn't work in the case of a module being loaded after a breakpoint was set, it could have a file with the same name, falling back to the absolute path would get us back to the prob adding a file to your project cause breakpoints to stop working in some cases, which would be very confusing.

4) Not currently, but if we can't find a good solution we would definitely consider it. I think there should be an lldb command to make this work though.

jo...@gmail.com <jo...@gmail.com> [#8](#)

Ah, I didn't think about shared objects / modules being loaded after the breakpoint gets set, that makes sense. I think I have all I need to know to get this working and I'll respond back to this

Thank you for the insight, and thank you guys for all the recent work in Android Studio regarding C++ support. As a developer who relies on the C++ / ndk integration it is nice to see some from recognizes its importance.

Keep up the good work,

Alex

la...@google.com <la...@google.com> [#9](#)

> 1) It does, it just takes a bit of setup. In the run configuration, you can add commands to run on lldb startup, you'll need something like 'settings set target.source-map /buildbot/path /my/ works for relative files. Pavel (cc'ed) probably knows.

LLDB should always have access to a full (non-relative) path. If the path is not specified in the debug_line section directly. LLDB will pick it up from the working directory of the compiler (emb debug_info section).

To see what paths LLDB thinks your source files are at you can run the "image dump line-table foo.cpp" command in the lldb console. After that you should be able to use the source-map cor this is something we'd want to ever fully automate, but I do see some room for improvement here. For instance, if we notice a situation where a file is not found by the full path but the same dialog like, "Hey, it looks like your source files are under a different path than what they were compiled under. Would you like to set up this path mapping?"

That said, if the files have not actually moved and the paths differ only by some ".." components, all of this dancing should not be necessary. LLDB should pick that up automatically. However work correctly on windows in 2.2, so it may be worth trying to upgrade to 2.3 and see if things work there.

ks...@google.com <ks...@google.com> [#10](#)

Thanks for the bug report/attention. Related to this, we're trying to find some people who'd be interested in participating in a Android Studio team lldb user study. If any lldb users would be signup here:

goo.gl/pKn3Ah

jo...@gmail.com <jo...@gmail.com> [#11](#)

[Comment deleted]

jo...@gmail.com <jo...@gmail.com> [#12](#)

Hey All,

I had to sidetrack this issue for a bit while I tackled other things, but I have a had a few hours to look at this again and I have found out some things that may help bring closure to this issue.

1) The source file paths indicated by the lldb command "image dump line-table <filename>" confirm that the source file paths embedded within my shared libraries are absolute and do exist I

2) I can hit editor breakpoints for the paths described in (1) in Android Studio 2.3 (Canary) while I cannot hit them in the official 2.2 release.

3) Directly after hitting a breakpoint in an external source file (source file used in building of external library) the lldb session is terminated. If I run "adb logcat" while running android studio I time: (I assume this was not an intended / ci result of setting a breakpoint)

```
02-27 15:23:28.891 5008 5097 F libc : Fatal signal 5 (SIGTRAP), code 1 in tid 5097 (Thread-2)
02-27 15:23:28.893 362 362 W : debuggerd: handling request: pid=5008 uid=10268 gid=10268 tid=5097
02-27 15:23:28.986 5102 5102 F DEBUG : *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
02-27 15:23:28.986 5102 5102 F DEBUG : Build fingerprint: 'google/bullhead/bullhead:7.1.1/N4F26O/3582057:user/release-keys'
02-27 15:23:28.986 5102 5102 F DEBUG : Revision: 'rev_1.0'
02-27 15:23:28.986 5102 5102 F DEBUG : ABL: 'arm'
02-27 15:23:28.986 5102 5102 F DEBUG : pid: 5008, tid: 5097, name: Thread-2 >>> com.techsoft3d.hps.sandbox <<<
02-27 15:23:28.987 5102 5102 F DEBUG : signal 5 (SIGTRAP), code 1 (TRAP_BRKPT), fault addr 0xf714dd56
02-27 15:23:28.987 5102 5102 F DEBUG : r0 00000001 r1 d4cfbc38 r2 00000001 r3 00000001
02-27 15:23:28.987 5102 5102 F DEBUG : r4 f719f174 r5 d4eff978 r6 00000000 r7 00000032
02-27 15:23:28.987 5102 5102 F DEBUG : r8 d4cfbc38 r9 d4cfbbb8 sl 000013e9 fp 00000005
02-27 15:23:28.987 5102 5102 F DEBUG : ip 00000032 sp d4cfbb98 lr f714c037 pc f714dd56 cpsr 000f0030
02-27 15:23:29.024 5102 5102 F DEBUG :
02-27 15:23:29.024 5102 5102 F DEBUG : backtrace:
02-27 15:23:29.024 5102 5102 F DEBUG : #00 pc 0000fd56 /system/bin/linker (__dl_Z11page_offsetx+5)
02-27 15:23:29.024 5102 5102 F DEBUG : #01 pc 0000e033 /system/bin/linker (__dl_notify_gdb_of_libraries+10)
02-27 15:23:29.024 5102 5102 F DEBUG : #02 pc 00002ac3 /system/bin/linker (__dl_ZL24debuggerd_signal_handlerIP7siginfoPv+702)
02-27 15:23:29.024 5102 5102 F DEBUG : #03 pc 00047b10 /system/bin/linker
02-27 15:23:29.024 5102 5102 F DEBUG : #04 pc 025da41c /data/app/com.techsoft3d.hps.sandbox-2/lib/arm/libhps_core.so
02-27 15:23:29.024 5102 5102 F DEBUG : (_Z21HD_Standard_Draw_TreeRKN5HOOPS17Rendition_PointerINS_22Internal_Net_RenditionEEEEPKNS_7SegmentERKNS_12Include_PathE+143)
02-27 15:23:29.024 5102 5102 F DEBUG : #05 pc 025d6d3f /data/app/com.techsoft3d.hps.sandbox-2/lib/arm/libhps_core.so
02-27 15:23:29.024 5102 5102 F DEBUG : #06 pc 025d65bb /data/app/com.techsoft3d.hps.sandbox-2/lib/arm/libhps_core.so
02-27 15:23:29.024 5102 5102 F DEBUG : (_Z16HD_Adjust_WLimitRN5HOOPS17Rendition_PointerINS_22Internal_Net_RenditionEEEEPKNS_7SegmentERKNS_12Include_PathE+562)
02-27 15:23:29.024 5102 5102 F DEBUG : #07 pc 027902ef /data/app/com.techsoft3d.hps.sandbox-2/lib/arm/libhps_core.so
02-27 15:23:29.024 5102 5102 F DEBUG : (_Z13HD_New_WindowRN5HOOPS17Rendition_PointerINS_22Internal_Net_RenditionEEEEPKNS_7SegmentERKNS_12Include_PathEPNS_18Deferred_Draw_TreeE+4282)
02-27 15:23:29.024 5102 5102 F DEBUG : #08 pc 027924c3 /data/app/com.techsoft3d.hps.sandbox-2/lib/arm/libhps_core.so (_Z22HD_Draw_Deferred_TreesPN5HOOPS15Display_Con
02-27 15:23:29.024 5102 5102 F DEBUG : #09 pc 0260b9bd /data/app/com.techsoft3d.hps.sandbox-2/lib/arm/libhps_core.so (_Z18HD_Standard_UpdatePN5HOOPS15Display_Context
02-27 15:23:29.024 5102 5102 F DEBUG : #10 pc 025f863d /data/app/com.techsoft3d.hps.sandbox-2/lib/arm/libhps_core.so
02-27 15:23:29.024 5102 5102 F DEBUG : #11 pc 025fcd9 /data/app/com.techsoft3d.hps.sandbox-2/lib/arm/libhps_core.so (_ZN20Standard_Driver_Task7Do_TaskEb+220)
02-27 15:23:29.024 5102 5102 F DEBUG : #12 pc 02623229 /data/app/com.techsoft3d.hps.sandbox-2/lib/arm/libhps_core.so
02-27 15:23:29.024 5102 5102 F DEBUG : #13 pc 02482eeb /data/app/com.techsoft3d.hps.sandbox-2/lib/arm/libhps_core.so (_ZN5HOOPS13System_Thread7bounce_EPv+50)
```

02-27 15:23:29.024 5102 5102 F DEBUG : #14 pc 000470b3 /system/lib/libc.so (__ZL15__pthread_startPv+22)
02-27 15:23:29.024 5102 5102 F DEBUG : #15 pc 00019e3d /system/lib/libc.so (__start_thread+6)
02-27 15:23:30.415 4379 5104 W ActivityManager: Force finishing activity com.techsoft3d.hps.sandbox/.MobileSurfaceActivity

Should I report this bug as a new issue?

Thanks,

Alex



la...@google.com <la...@google.com> [#13](#)

Hi,

sorry about the delay. I think we can keep the same bug open for this, as your general problem of "hitting breakpoints in an external file" is not resolved, although it sounds like you were able

The logcat output is a consequence of the debugger exiting (the app hits a breakpoint we've left there, and the system kills it as there is no debugger present). I the problem in this case is so you check whether there are any exceptions logged in your idea.log file after this happens?

I am going to try to see if I can reproduce this. If not, I am going to need more information about your setup.



la...@google.com <la...@google.com> [#14](#)

Marked as fixed.

OK, nevermind. I've just seen that you filed [bug 36949180](#). We can continue this there and close this,



ma...@smule.com <ma...@smule.com> [#15](#)

> OK, nevermind. I've just seen that you filed [bug 36949180](#). We can continue this there and close this,

The linked bug does not seem related. Where did you continue this discussion?



an...@google.com <an...@google.com> [#16](#)

I think it is issuetracker.google.com/issues/37137270.