Browsing and debugging AOSP code: revisited

Chris Simmonds, 2net

Overview

- This is an update to a talk I did in May 2022
  Video - https://youtu.be/SEgBapOKI2g
  Slides - https://2net.co.uk/slides/aosp-aaos-meetup/2022-may-debug.pdf
- What's new:
  - attaching to a running device without using DDMS(*)
  - quick debug sessions using JDB

(*) DDMS = Dalvik Debug Monitor Service, an old debug tool that I have been using for ever but is now broken in A 13. Turns out there is a better way (and has been for many years)

Recap

Create an IntelliJ project file for AOSP (assumed to be in ~/aosp)

```
$ cd aosp
$ m idegen
$ development/tools/idegen/idegen.sh
find: 'out/target/product/marvin/recovery/root/d': Permission denied
find: 'out/target/product/marvin/root/d': Permission denied
[...]
```

Creates: `android.ipr`

For more information, see `development/tools/idegen/README`

Configuring Android Studio

Indexing the whole of AOSP takes a lot of resources

Increase heap to 5 GB

In Help->Edit Custom VM Options, add:

```
-Xmx5g
```

For reference, the VM Options are stored in
`$HOME/.config/Google/AndroidStudio2022.2/studio64.vmoptions`

Importing AOSP code

In Android Studio, open `$AOSP/android.ipr`

If you see a dialog box offering to convert the project format, press Cancel

Wait for it to complete "Updating indexes" (background task, bottom right); takes hours

Do NOT migrate this project to Gradle. Ever.

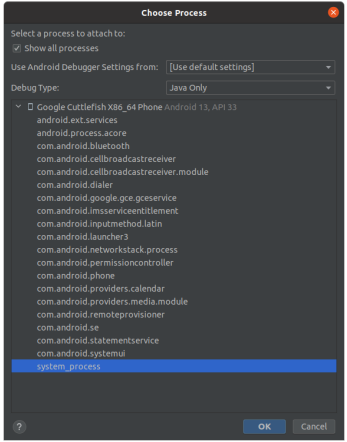
Now you can use Studio to browse the code, search for classes, etc.

Some people consider this a better alternative to using vim

Debugging using Android Studio 1/3

Suppose you want to debug `system_server`

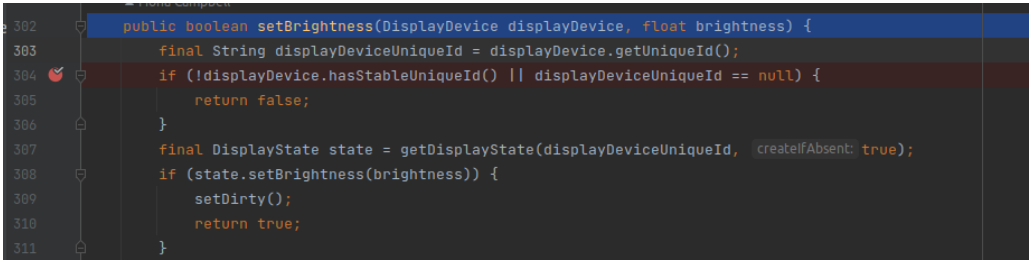
- Boot target device
- In Android Studio, Run -> Attach Debugger to Android Process (*)
- Tick Show all processes
- Scroll down list and select `system_process` and click OK



(*) This is greyed out until Studio has finished indexing all files

Debugging using Android Studio 2/3

Set a breakpoint, e.g. in
`com.android.server.display.BrightnessSetting.setBrightness`

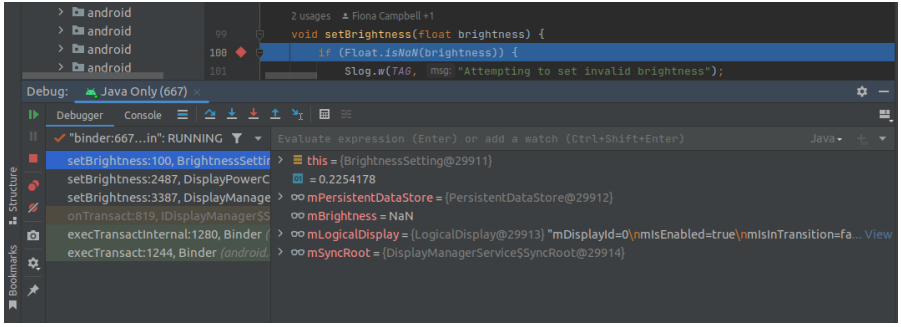


Debugging using Android Studio 3/3

Resume Program

On device, open Settings -> Display -> Brightness level

Move the slider, and it should hit the breakpoint



JDB

- JDB is a command-line debugger, part of Open JDK
- Pros: easy to set up, nothing new to install, no need for `IDEGen`, fast
- Cons: a lot of typing; it's JDB

Debugging with JDB 1/2

Find the PID of `system_server`

```
$ adb shell ps -A | grep system_server
system         664   427 2533808 318092 0                   0 S system_server
```

Connect port 8700 to the JDWP thread for PID 664:

```
$ adb forward tcp:8700 jdwp:664
8700
```

Start JDB, giving the path to the component we are debugging:

```
$ jdb -attach localhost:8700 -sourcepath frameworks/base/services/core/java
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
Initializing jdb ...
>
```

Debugging with JDB 2/2

Set a method breakpoint:

```
> stop in com.android.server.display.BrightnessSetting.setBrightness
Set breakpoint com.android.server.display.BrightnessSetting.setBrightness
```

Change brightness level in Settings app, and the breakpoint is hit:

```
Breakpoint hit: "thread=main", com.android.server.display.BrightnessSetting.setBrightness(), l
100             if (Float.isNaN(brightness)) {
```

Print out the brightness and resume the thread:

```
main[1] print brightness
 brightness = 0.26377952
main[1] resume
All threads resumed.
>
```

That’s all

Any questions?