---

gdbclient

The current version of envsetup.sh has a gdbclient command that handles much of the setup. You have make sure that: 1) you do this from the same window used to build the software on the device you are debugging and 2) verify that the symbols in the object files in the build tree match up with what is installed on the device or emulator.

1. **Cmd: gdbclient <executable name> <port number> <task name>**

<executable name>: file name in system/bin dir

<port number>: default is :5039 (need the colon before the number)

<task name>: obtained by running "ps" on the target. GDB uses it to identify the PID internally.

E.g.

>>> gdbclient mediaserver :5039 mediaserver

>>> gdbclient app_process :5039 com.android.systemui

```
mandy_liu@mandy:~/vb_share/53782_EvitareUL_Generic_WWE_JB_CRC_Stable$ gdbclient app_process :5039 com.android.sy
stemui
[1] 11574
Attached; pid = 8557
Listening on port 5039
GNU gdb (GDB) 7.3.1-gg2
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=x86_64-linux-gnu --target=arm-linux-android".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/mandy_liu/vb_share/53782_EvitareUL_Generic_WWE_JB_CRC_Stable/out/target/product/evita
reul/symbols/system/bin/app_process...done.
Remote debugging from host 127.0.0.1
warning: .dynamic section for "/home/mandy_liu/vb_share/53782_EvitareUL_Generic_WWE_JB_CRC_Stable/out/target/pro
duct/evitareul/symbols/system/bin/linker" is not at the expected address (wrong library or version mismatch?)
warning: Could not load shared library symbols for 12 libraries, e.g. libkikin.so.
Use the "info sharedlibrary" command to see the complete listing.
Do you need "set solib-search-path" or "set sysroot"?
epoll_wait () at bionic/libc/arch-arm/syscalls/epoll_wait.S:10
10          ldmfd   sp!, {r4, r7}
(gdb)
```

Then you can run commands like “info threads”, “break”, “step” etc.

2. **Cmd: bt**

Show trace of where you are currently. Which functions you are in. Prints stack backtrace.

```
(gdb) bt
#0  epoll_wait () at bionic/libc/arch-arm/syscalls/epoll_wait.S:10
#1  0x40146414 in android::Looper::pollInner (this=0x50dad018, timeoutMillis=989)
    at customize/native/libs/utils/Looper.cpp:218
#2  0x401466f4 in android::Looper::pollOnce (this=0x50dad018, timeoutMillis=989, outFd=0x0, outEvents=0x0,
    outData=0x0) at customize/native/libs/utils/Looper.cpp:189
#3  0x401fb0aa in pollOnce (timeoutMillis=<optimized out>, this=<optimized out>)
    at frameworks/native/include/utils/Looper.h:176
#4  android::NativeMessageQueue::pollOnce (this=0x5262dd48, env=0x4154d778, timeoutMillis=<optimized out>)
    at customize/base/core/jni/android_os_MessageQueue.cpp:97
#5  0x4081e234 in dvmPlatformInvoke () at dalvik/vm/arch/arm/CallEABI.S:258
#6  0x40850cc6 in dvmCallJNIMethod (args=0x4106bdd8, pResult=0x40013020, method=0x4c065918, self=0x40013010)
    at dalvik/vm/Jni.cpp:1155
#7  0x40827664 in dalvik_mterp () at dalvik/vm/mterp/out/InterpAsm-armv7-a-neon.S:16239
#8  0x4082d3c4 in dvmInterpret (self=0x40013010, method=0x4c05c7b8, pResult=0xbeb6a810)
    at dalvik/vm/interp/Interp.cpp:1964
#9  0x4086708a in dvmInvokeMethod (obj=<optimized out>, method=<optimized out>, argList=<optimized out>,
    params=0x418d5aa0, returnType=0x4154e2a8, noAccessCheck=false) at dalvik/vm/interp/Stack.cpp:737
#10 0x4086fd3a in Dalvik_java_lang_reflect_Method_invokeNative (args=<optimized out>, pResult=0x40013020)
    at dalvik/vm/native/java_lang_reflect_Method.cpp:101
#11 0x40827664 in dalvik_mterp () at dalvik/vm/mterp/out/InterpAsm-armv7-a-neon.S:16239
#12 0x4082d3c4 in dvmInterpret (self=0x40013010, method=0x4c0c8568, pResult=0xbeb6a9a8)
    at dalvik/vm/interp/Interp.cpp:1964
---Type <return> to continue, or q <return> to quit---
```

3. Cmd: info threads

List threads in use.

```
[New Thread 8566]
[New Thread 8567]
[New Thread 8568]
[New Thread 8569]
[New Thread 8578]
[New Thread 8579]
[New Thread 8582]
[New Thread 8710]
[New Thread 8711]
  Id   Target Id         Frame
  14   Thread 8711       __ioctl () at bionic/libc/arch-arm/syscalls/__ioctl.S:9
  13   Thread 8710       __ioctl () at bionic/libc/arch-arm/syscalls/__ioctl.S:9
  12   Thread 8582       __futex_syscall3 () at bionic/libc/arch-arm/bionic/futex_arm.S:58
  11   Thread 8579       epoll_wait () at bionic/libc/arch-arm/syscalls/epoll_wait.S:10
  10   Thread 8578       epoll_wait () at bionic/libc/arch-arm/syscalls/epoll_wait.S:10
  9    Thread 8569       __ioctl () at bionic/libc/arch-arm/syscalls/__ioctl.S:9
  8    Thread 8568       __ioctl () at bionic/libc/arch-arm/syscalls/__ioctl.S:9
  7    Thread 8567       __futex_syscall3 () at bionic/libc/arch-arm/bionic/futex_arm.S:58
  6    Thread 8566       __futex_syscall3 () at bionic/libc/arch-arm/bionic/futex_arm.S:58
  5    Thread 8565       __futex_syscall3 () at bionic/libc/arch-arm/bionic/futex_arm.S:58
  4    Thread 8564       recvmsg () at bionic/libc/arch-arm/syscalls/recvmsg.S:9
  3    Thread 8563       __rt_sigtimedwait () at bionic/libc/arch-arm/syscalls/__rt_sigtimedwait.S:10
  2    Thread 8561       __futex_syscall3 () at bionic/libc/arch-arm/bionic/futex_arm.S:58
* 1    Thread 8557       epoll_wait () at bionic/libc/arch-arm/syscalls/epoll_wait.S:10
(gdb)
```

Cmd: thread <thread number> + **bt**

Switch debug thread to a specific thread and show the backtrace.

>>> thread 3

```
(gdb) thread 3
[Switching to thread 3 (Thread 12920)]
#0  __rt_sigtimedwait () at bionic/libc/arch-arm/syscalls/__rt_sigtimedwait.S:10
10          ldmfd   sp!, {r4, r7}
(gdb) bt
#0  __rt_sigtimedwait () at bionic/libc/arch-arm/syscalls/__rt_sigtimedwait.S:10
#1  0x400d8452 in sigwait (set=<optimized out>, sig=0x4ff74e58) at bionic/libc/unistd/sigwait.c:63
#2  0x40856012 in signalCatcherThreadStart (arg=<optimized out>) at dalvik/vm/SignalCatcher.cpp:285
#3  0x4085924c in internalThreadStart (arg=0x41523d50) at dalvik/vm/Thread.cpp:1793
#4  0x400d0e4c in __thread_entry (func=0x40859201 <internalThreadStart(void*)>, arg=0x41523d50,
    tls=<optimized out>) at bionic/libc/bionic/pthread.c:240
#5  0x400d0580 in pthread_create (thread_out=0x4dbd0fa0, attr=0xbeb6a780,
    start_routine=0x40859201 <internalThreadStart(void*)>, arg=0x41523d50) at bionic/libc/bionic/pthread.c:384
#6  0x00000000 in ?? ()
(gdb)
```

4. Cmd:

c: Continue executing until next break point/watchpoint.

n: Execute next line of code. Will not enter functions.

s: Step to next line of code. Will step into a function.

l: List next 5 lines of source code.

5. **Cmd: detach**

Detach from the debugging process.

---

gdbserver

---

command to device shell (via adb)

% command to host pc shell

1. **# Cmd: gdbserver :<port> <executable>** or **gdbserver :<port> --attach <pid>**

On the device, launch a new command or attach to an existing process

>>> # gdbserver :5039 /data/bin/memtest

>>> # gdbserver :5039 --attach 3124

2. **% Cmd: adb forward tcp:<port> tcp:<port>**

Forward the tcp port to device with adb.

>>> % adb forward tcp:5039 tcp:5039

3. Start a special version of gdb that lives in the "prebuilt" area of the source tree

% Cmd: <source_path>/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin/arm-eabi-gdb

>>> % ~/vb_share/53782_EvitareUL_Generic_WWE_JB_CRC_Stable/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin/arm-eabi-gdb ~/vb_share/53782_EvitareUL_Generic_WWE_JB_CRC_Stable/out/target/product/evitareul/obj/EXECUTABLES/app_process_intermediates/LINKED/app_process

The executable is the program to debug (usually app_process for an application). Make sure to use the copy of the executable in the symbols directory, not the primary android directory, because the one in the primary directory has been stripped of its debugging information.

P.S. You can load symbols here or use the *file* cmd when enter gdb mode

4. Tell gdb where to find the shared libraries that will get loaded.

% Cmd: set solib-absolute-prefix <source_path>/out/target/product/<product name>/symbols

% Cmd: set solib-search-path <source_path>/out/target/product/<product name>/symbols/system/lib

>>> % (gdb) set solib-absolute-prefix /home/mandy_liu/vb_share/53782_EvitareUL_Generic_WWE_JB_CRC_Stable/out/target/product/evitareul/symbols

>>> % (gdb) set solib-search-path /home/mandy_liu/vb_share/53782_EvitareUL_Generic_WWE_JB_CRC_Stable/out/target/product/evitareul/symbols/system/lib

Make sure you specify the correct directories—gdb may not tell you if you make a mistake.

5. Connect to the device by issuing the gdb command

% Cmd: target remote :<port>

>>> % (gdb) target remote :5039

The :<port> tells gdb to connect to the localhost port <port>, which is bridged to the device by adb.

6. You may need to inspire gdb to load some symbols by typing:

% Cmd: shared

You should be connected and able to debug as you normally would. You can ignore the error about not finding the location for the thread creation breakpoint. It will be found when the linker loads libc into your process before hitting main().

7. **% Cmd: set scheduler-locking on**

Don't let other threads get scheduled while we're debugging. You should "set scheduler-locking off" before issuing a "continue", or else your thread may get stuck on a futex or other spinlock because no other thread can release it.