

Android GDB Debug

William.L

wiliwe@gmail.com

2010-11-16

Environment -

1. Ubuntu 9.10
2. Android 2.2 (Froyo)
3. nVidia T250 Ventana board

Introduction

- Depends on the Android tool : **Android Debug Bridge (ADB)**
- Android 2.2(Froyo)源碼目錄中，prebuild 目錄下有現成的 GDB 程式(目前使用的 ARM EABI 版本為 4.4.0)
- **AndroidFroyoSrc /prebuilt/linux-x86/toolchain/arm-eabi-4.4.0/bin/arm-eabi-gdb**
- ARM EABI GDB 用法：**arm-eabi-gdb ExecutableWithDebugSymbol**
- ExecutableWithDebugSymbol 位在目錄 **out/target/product/ventana/symbols/system/bin/**

I) GDB Server 端(Android 裝置)使用步驟

- 使用 **adb shell** 登入遠端 Android 裝置
- 裝置後執行指令
 - # **gdbserver :5039 /data/Executable &** 或
 - # **gdbserver :5039 --attach ExecutablePID &**
- Port number **5039**, 可自行替換其它埠號, 但不可跟其它網路服務使用的埠號衝突!

II) GDB Client 端(PC/NB)使用步驟

- **arm-eabi-gdb ExecutableWithDebugSymbol**, 進入 GDB
- GDB 中設定偵錯用函式庫搜路徑 [**AndroidSrc 使用絕對路徑**]
 - # **set solib-absolute-prefix AndroidFroyoSrc/out/target/product/BoardModel/symbols/**
 - # **set solib-search-path**

AndroidFroyoSrc

/out/target/product/ventana/symbols/system/lib:AndroidSrc/out/target/product/ventana/symbols/system/bin

- target remote : 5039 → 此 port number 需跟 GDB Server 使用的相同

III) 常用 GDB 指令

- * **b** - 設中斷點(breakpoint)
- * **display** - 檢視(watch)變數 / 暫存器內容
- * **c** - continue
- * **n(ext)** - 單步執行(step over, 不進入函式)
- * **s(tep)** - 單步執行(step into, 進入函式)
- * **bt** - backtrace, 查看 function 的 callstack, stack 左方有數字表示各 function 在 stack 中的層數。

Standalone Program Debug

測試用程式，係將一自 framebuffer 取出的圖寫至 framebuffer。執行的程式名稱爲 **testFB**。

Android 2.2 之後，已內建 GDB server。在 ADB shell 下執行 `gdbserver`，即可啓動 GDB server。自己的工作機器(PC/NB)，稱作 **Host**。

1. 將 Host 的 5039 埠映對到 **模擬器** 或 **設備** 的 5039 埠
(`adb forward Host-Port Dev/Emulator-Port`)

adb forward tcp:5039 tcp:5039

2. 使用 ADB 連至 **模擬器** 或 **設備**

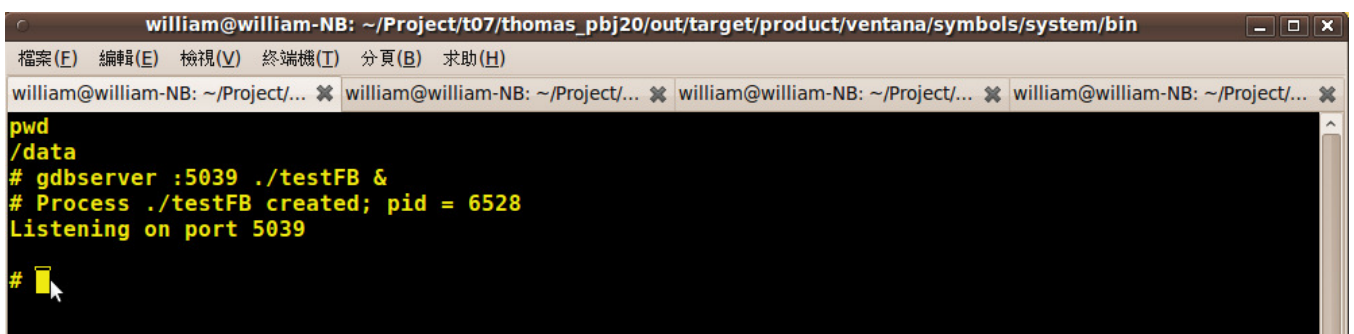
adb shell

3. ADB 連線成功後，執行下列指令：

#> gdbserver :5039 ./testFB &

or

gdbserver :5039 --attach PID &



```
william@william-NB: ~/Project/t07/thomas_pbj20/out/target/product/ventana/symbols/system/bin
檔案(F) 編輯(E) 檢視(V) 終端機(T) 分頁(B) 求助(H)
william@william-NB: ~/Project/... ✖ william@william-NB: ~/Project/... ✖ william@william-NB: ~/Project/... ✖ william@william-NB: ~/Project/... ✖
pwd
/data
# gdbserver :5039 ./testFB &
# Process ./testFB created; pid = 6528
Listening on port 5039
#
```

, where PID is the ID of the process executing **testFB** program.



4. 於 Host 上，執行 codebase 中的 arm-eabi-gdb client 程式。參數為含有 debugging symbol 的 testFB 程式。

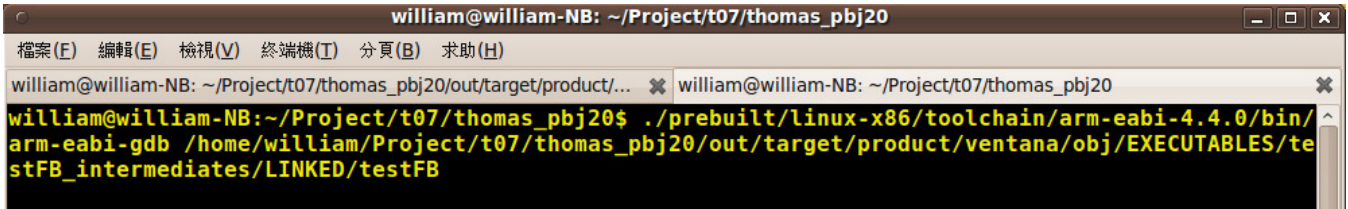
AndroidFroyoSrc/prebuilt/linux-x86/toolchain/arm-eabi-4.4.0/bin/arm-eabi-gdb

AnroidFroyoSrc/out/target/product/ventana/obj/EXECUTABLES/testFB_intermediates/LINKED/testFB

or

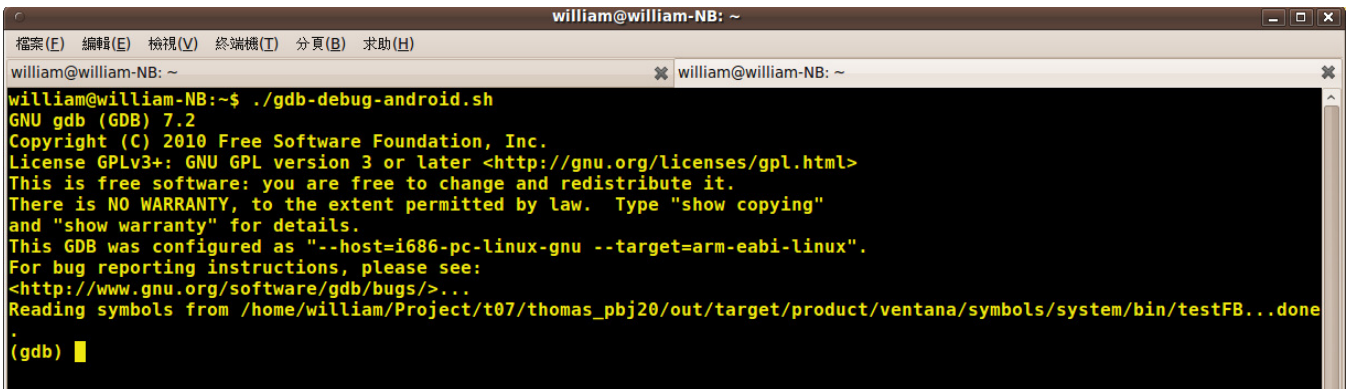
AndroidFroyoSrc /prebuilt/linux-x86/toolchain/arm-eabi-4.4.0/bin/arm-eabi-gdb

AndroidFroyoSrc/out/target/product/ventana/symbols/system/bin/testFB



```
william@william-NB: ~/Project/t07/thomas_pbj20
檔案(F) 編輯(E) 檢視(V) 終端機(T) 分頁(B) 求助(H)
william@william-NB: ~/Project/t07/thomas_pbj20/out/target/product/... x william@william-NB: ~/Project/t07/thomas_pbj20 x
william@william-NB:~/Project/t07/thomas_pbj20$ ./prebuilt/linux-x86/toolchain/arm-eabi-4.4.0/bin/arm-eabi-gdb /home/william/Project/t07/thomas_pbj20/out/target/product/ventana/obj/EXECUTABLES/testFB_intermediates/LINKED/testFB
```

5. 於 Host 上，執行 gdb-debug-android.sh script 以進入 GDB client 的畫面如下：



```
william@william-NB: ~
檔案(F) 編輯(E) 檢視(V) 終端機(T) 分頁(B) 求助(H)
william@william-NB: ~ x william@william-NB: ~ x
william@william-NB:~$ ./gdb-debug-android.sh
GNU gdb (GDB) 7.2
Copyright (C) 2010 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=i686-pc-linux-gnu --target=arm-eabi-linux".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/william/Project/t07/thomas_pbj20/out/target/product/ventana/symbols/system/bin/testFB...done
(gdb) █
```

gdb-debug-android.sh 內容如下：

```
#!/bin/bash
export ANDROID_SRC_ROOT=/home/william/Project/Android_SrcRoot

#export APP_PROCESS_TO_DEBUG=/out/target/product/ventana/obj/EXECUTABLES/testFB_intermediates/LINKED/testFB
#export APP_PROCESS_TO_DEBUG=/out/target/product/ventana/symbols/system/bin/HelloWorld
export APP_PROCESS_TO_DEBUG=/out/target/product/ventana/obj/EXECUTABLES/HelloWorld_intermediates/HelloWorld

adb push snapshot.bmp /data/snapshot.bmp
adb forward tcp:5039 tcp:5039

$ANDROID_SRC_ROOT/prebuilt/linux-x86/toolchain/arm-eabi-4.4.0/bin/arm-eabi-gdb
$ANDROID_SRC_ROOT$APP_PROCESS_TO_DEBUG
```

此 shell script 中，將事先用 DDMS 抓出的 snapshot 圖，"snapshot.bmp"，adb push 到 Android 裝置中"/data"目錄；此圖由 **testFB** 程式所使用。

6. 於 Host 的 GDB client 中，設定 debug 用的函式庫之絕對路徑

(gdb) **set solib-absolute-prefix AnroidFroyoSrc /out/target/product/ventana/symbols/**

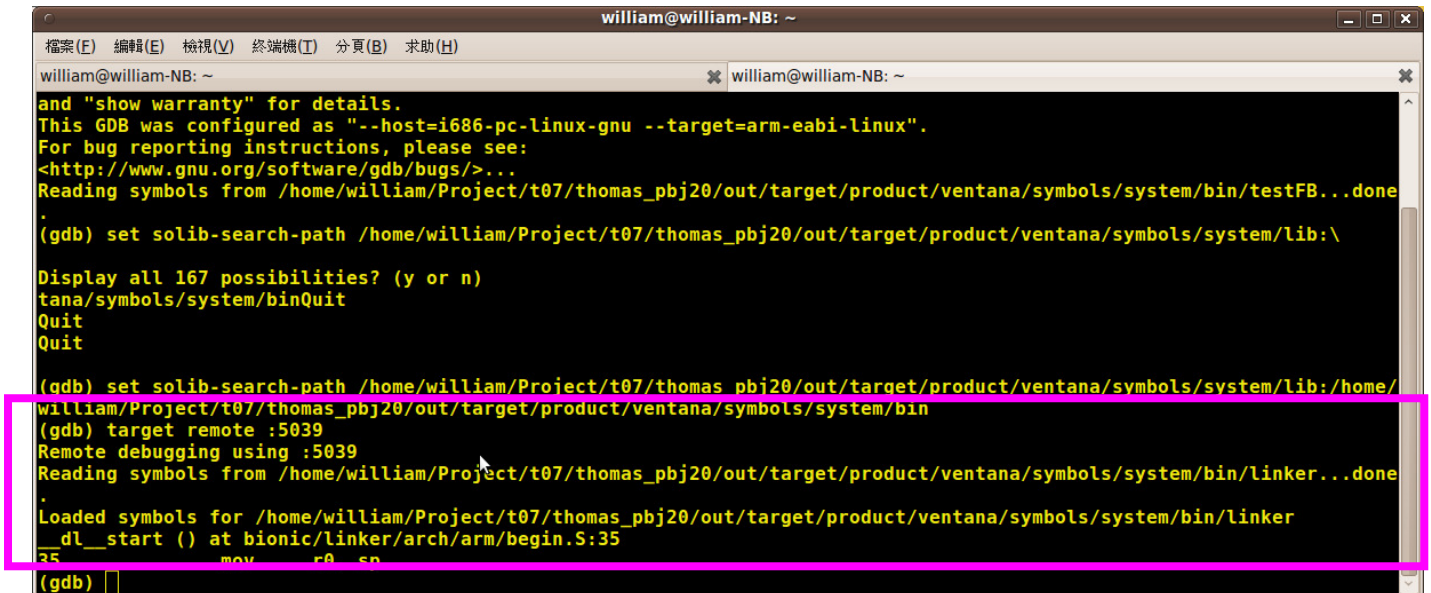
(gdb) set solib-search-path

AnroidFroyoSrc/out/target/product/ventana/symbols/system/lib:AnroidSrc/out/target/product/ventana/symbols/system/bin:AnroidSrc/out/target/product/ventana/system/lib

7. 現在開始偵錯動作...

於 Host 的 GDB client 中執行指令:

(gdb) **target remote :5039**



```
william@william-NB: ~
檔案(F) 編輯(E) 檢視(V) 終端機(T) 分頁(B) 求助(H)
william@william-NB: ~
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-linux-gnu --target=arm-eabi-linux".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/william/Project/t07/thomas_pbj20/out/target/product/ventana/symbols/system/bin/testFB...done
(gdb) set solib-search-path /home/william/Project/t07/thomas_pbj20/out/target/product/ventana/symbols/system/lib:\
Display all 167 possibilities? (y or n)
tana/symbols/system/binQuit
Quit
Quit
(gdb) set solib-search-path /home/william/Project/t07/thomas_pbj20/out/target/product/ventana/symbols/system/lib:/home/
william/Project/t07/thomas_pbj20/out/target/product/ventana/symbols/system/bin
(gdb) target remote :5039
Remote debugging using :5039
Reading symbols from /home/william/Project/t07/thomas_pbj20/out/target/product/ventana/symbols/system/bin/linker...done
Loaded symbols for /home/william/Project/t07/thomas_pbj20/out/target/product/ventana/symbols/system/bin/linker
_dl_start () at bionic/linker/arch/arm/begin.S:35
35      mov     r0, sp
(gdb) |
```

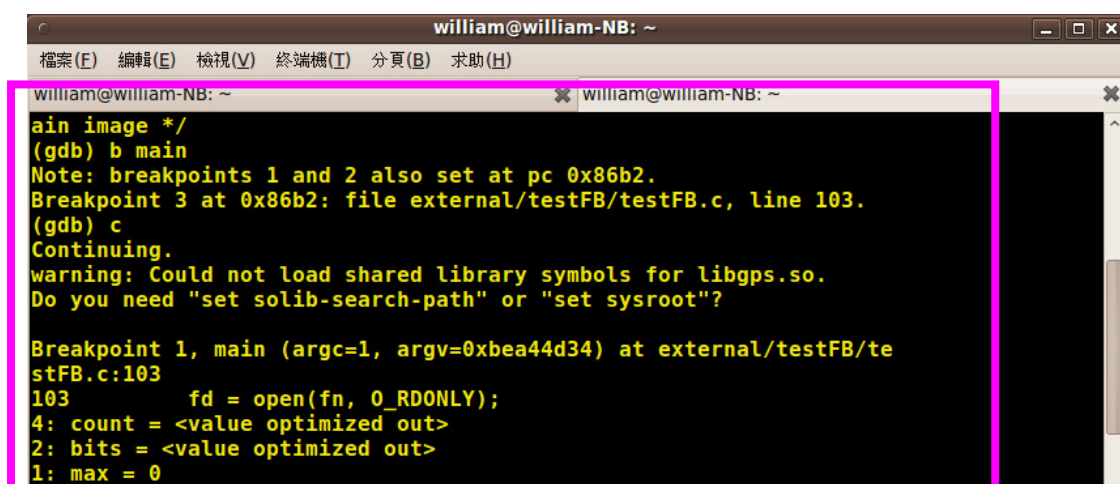
[In Debugging]

8. 於 main()設定中斷點(break point)，讓 GDB 執行並於第一個中斷點停止執行程式。

於 Host 的 GDB client 中執行指令:

(gdb) **b main**

(gdb) **c**



```
william@william-NB: ~
檔案(F) 編輯(E) 檢視(V) 終端機(T) 分頁(B) 求助(H)
william@william-NB: ~
ain image */
(gdb) b main
Note: breakpoints 1 and 2 also set at pc 0x86b2.
Breakpoint 3 at 0x86b2: file external/testFB/testFB.c, line 103.
(gdb) c
Continuing.
warning: Could not load shared library symbols for libgps.so.
Do you need "set solib-search-path" or "set sysroot"?

Breakpoint 1, main (argc=1, argv=0xbea44d34) at external/testFB/te
stFB.c:103
103      fd = open(fn, O_RDONLY);
4: count = <value optimized out>
2: bits = <value optimized out>
1: max = 0
```

顯示程式碼：

(gdb) **list**

```
(gdb) list
98         char *fn="/data/snapshot.bmp";
99
100        if (vt_set_mode(1))
101            return -1;
102
103        fd = open(fn, O_RDONLY);
104        if (fd < 0) {
105            LOGE("cannot open '%s'\n", fn);
106            goto fail_restore_text;
107        }
```

繼續執行下一行 statement

(gdb) next

使用 backtrace 指令，觀察 call stack

(gdb)bt

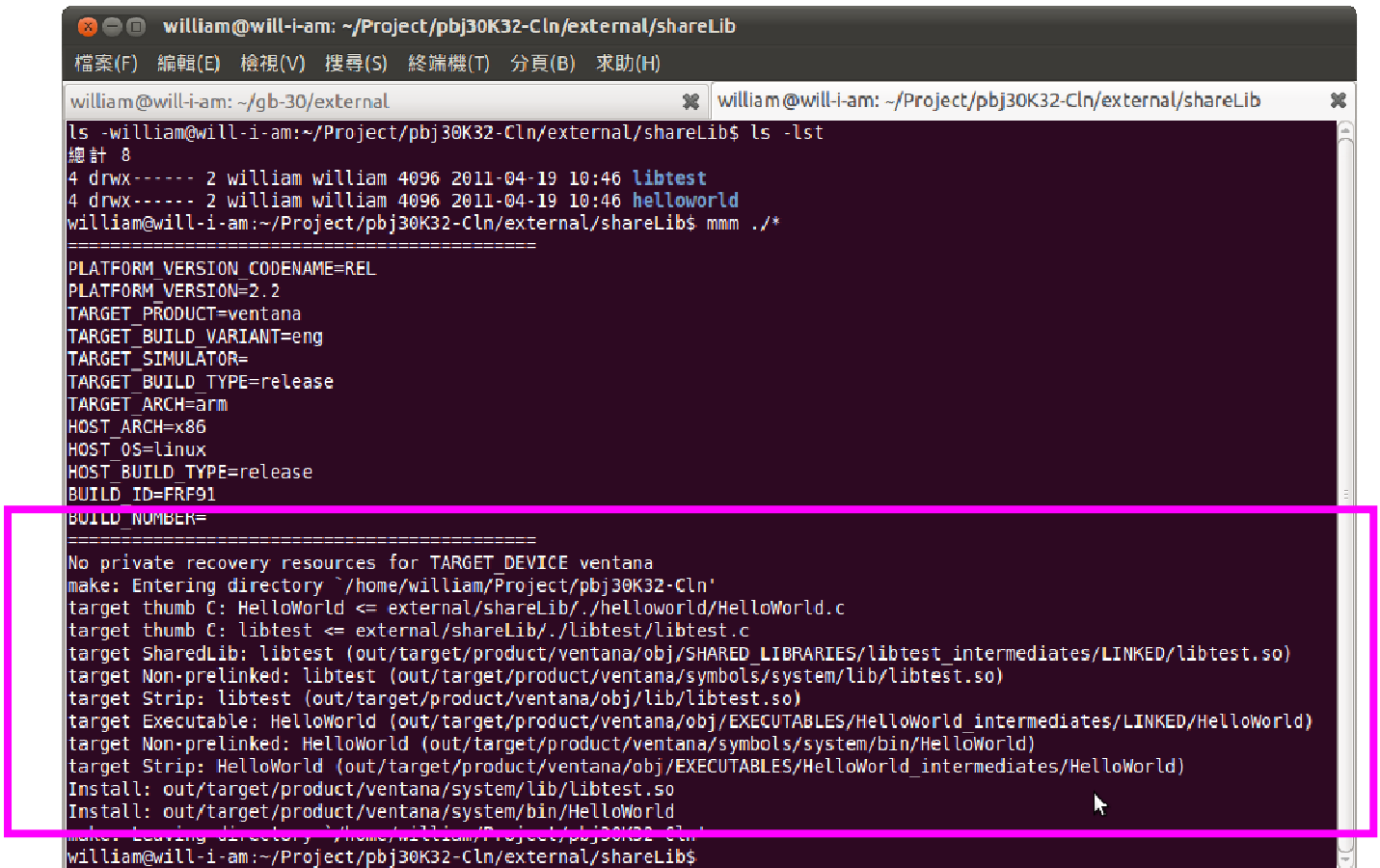
```
46         if (ioctl(fb->fd, FBIOGET_FSCREENINFO, &fb->fi) < 0)
47             goto fail;
48         if (ioctl(fb->fd, FBIOGET_VSCREENINFO, &fb->vi) < 0)
49             goto fail;
50
(gdb) n
104         if (fd < 0) {
(gdb) n
110         if (fstat(fd, &s) < 0) {
(gdb) n
114         data = mmap(0, s.st_size, PROT_READ, MAP_SHARED, fd, 0);
(gdb) n
115         if (data == MAP_FAILED)
(gdb) list
110         if (fstat(fd, &s) < 0) {
111             goto fail_close_file;
112         }
113
114         data = mmap(0, s.st_size, PROT_READ, MAP_SHARED, fd, 0);
115         if (data == MAP_FAILED)
116             goto fail_close_file;
117
118         if (fb_open(&fb))
119             goto fail_unmap_data;
(gdb) n
118         if (fb_open(&fb))
(gdb) s
fb_open (fb=<value optimized out>) at external/testFB/testFB.c:42
42         fb->fd = open("/dev/graphics/fb0", O_RDWR);
(gdb) bt
#0  fb_open (fb=<value optimized out>) at external/testFB/testFB.c:42
#1  0x00008750 in main (argc=<value optimized out>, argv=<value optimized out>) at external/testFB/testFB.c:118
(gdb)
```

圖中 ”#0” 及 ”#1” 分別表示目前正在執行的 function 及呼叫此 function 的 function。若還有其它數字，數字愈大，表示是 call stack 中愈下方的 function。

<Shared Library Debug>

1. 準備測試程式 **HelloWorld** 及一共享函式庫 **libtest.so** (詳情請見附加檔案)。

以下是 libtest.so 編譯過程及結果。



```
william@will-i-am: ~/Project/pbj30K32-Cln/external/shareLib
檔案(F) 編輯(E) 檢視(V) 搜尋(S) 終端機(T) 分頁(B) 求助(H)

william@will-i-am: ~/gb-30/external
william@will-i-am: ~/Project/pbj30K32-Cln/external/shareLib

ls -william@will-i-am:~/Project/pbj30K32-Cln/external/shareLib$ ls -lst
總計 8
4 drwx----- 2 william william 4096 2011-04-19 10:46 libtest
4 drwx----- 2 william william 4096 2011-04-19 10:46 helloworld
william@will-i-am:~/Project/pbj30K32-Cln/external/shareLib$ mmm ./*
=====
PLATFORM VERSION CODENAME=REL
PLATFORM_VERSION=2.2
TARGET_PRODUCT=ventana
TARGET_BUILD_VARIANT=eng
TARGET_SIMULATOR=
TARGET_BUILD_TYPE=release
TARGET_ARCH=arm
HOST_ARCH=x86
HOST_OS=linux
HOST_BUILD_TYPE=release
BUILD_ID=FRF91
BUILD_NUMBER=
=====
No private recovery resources for TARGET_DEVICE ventana
make: Entering directory `/home/william/Project/pbj30K32-Cln'
target thumb C: HelloWorld <= external/shareLib/./helloworld/HelloWorld.c
target thumb C: libtest <= external/shareLib/./libtest/libtest.c
target SharedLib: libtest (out/target/product/ventana/obj/SHARED_LIBRARIES/libtest_intermediates/LINKED/libtest.so)
target Non-prelinked: libtest (out/target/product/ventana/symbols/system/lib/libtest.so)
target Strip: libtest (out/target/product/ventana/obj/lib/libtest.so)
target Executable: HelloWorld (out/target/product/ventana/obj/EXECUTABLES/HelloWorld_intermediates/LINKED/HelloWorld)
target Non-prelinked: HelloWorld (out/target/product/ventana/symbols/system/bin/HelloWorld)
target Strip: HelloWorld (out/target/product/ventana/obj/EXECUTABLES/HelloWorld_intermediates/HelloWorld)
Install: out/target/product/ventana/system/lib/libtest.so
Install: out/target/product/ventana/system/bin/HelloWorld
make: Leaving directory `/home/william/Project/pbj30K32-Cln'
william@will-i-am:~/Project/pbj30K32-Cln/external/shareLib$
```

2. 分別將編譯好的 **HelloWorld** 及 **libtest.so**，用 ADB push 至 **/data** 及 **/system/lib**。

#> adb push **AndroidFroyoSrc**

/out/target/product/ventana/obj/EXECUTABLES/HelloWorld_intermediates/LINKED/HelloWorld
/data

#> adb push **AndroidFroyoSrc**

/out/target/product/ventana/obj/SHARED_LIBRARIES/Libtest_intermediates/LINKED/libtest.so
/system/lib

```
# gdbserver :5039 ./HelloWorld &
# Process ./HelloWorld created; pid = 2690
Listening on port 5039

# Remote debugging from host 127.0.0.1

# ls
lost+found
dontpanic
misc
local
data
app-private
app
property
dalvik-cache
gps
docking_log
docking_plug
headphone_plug
system
backup
snapshot.bmp
libtest.so
HelloWorld
```

3. 在 Android 機器中，執行 GDB server 指令

`gdbserver :5039 ./HelloWorld &`

，之後再執行 `ps` 指令查看 HelloWorld process 的 ID。

```
app_23    2001    884    225716 20752 ffffffff afd0ebac S com.android.mms
app_25    2040    884    225968 25000 ffffffff afd0ebac S com.google.android.apps.maps:FriendService
app_26    2056    884    219076 22464 ffffffff afd0ebac S com.google.android.gm
app_37    2066    884    215420 21416 ffffffff afd0ebac S com.android.email
app_39    2081    884    213044 20268 ffffffff afd0ebac S com.android.bluetooth
app_45    2095    884    211924 20228 ffffffff afd0ebac S com.android.deskclock
app_11    2107    884    212280 19528 ffffffff afd0ebac S com.android.music
app_20    2117    884    213900 20856 ffffffff afd0ebac S com.cooliris.media
app_30    2150    884    211212 19280 ffffffff afd0ebac S com.android.defcontainer
app_5     2645    884    212440 20236 ffffffff afd0ebac S com.android.calendar
system    2664    884    212800 20512 ffffffff afd0ebac S com.compal.swbutton:remote
root      2810    901    676    332    c00b6294 afd0e84c S /system/bin/sh
root      2817    2810   820    488    c014e8b8 afd0dd04 S gdbserver
root      2820    2817   596    236    c00c5b88 000083ae T ./HelloWorld
root      2830    2810   824    324    00000000 afd0d93c R ps
```

4. 在 Android 機器中，執行指令

`cat /proc/2820/maps`

即可見到 libtest.so 被分配的記憶體位址：`0x80000000`。

```
# cat /proc/2820/maps
00008000-00009000 r-xp 00000000 b3:1e 24056 /data/HelloWorld
00009000-0000a000 rwpx 00001000 b3:1e 24056 /data/HelloWorld
40000000-40008000 r-xs 00000000 00:04 685 /dev/ashmem/system_properties (deleted)
80000000-80001000 r-xp 00000000 b3:19 878 /system/lib/libtest.so
80001000-80002000 rwpx 00001000 b3:19 878 /system/lib/libtest.so
afb00000-afb16000 r-xp 00000000 b3:19 848 /system/lib/libm.so
afb16000-afb17000 rwpx 00016000 b3:19 848 /system/lib/libm.so
afc00000-afc01000 r-xp 00000000 b3:19 736 /system/lib/libstdc++.so
afc01000-afc02000 rwpx 00001000 b3:19 736 /system/lib/libstdc++.so
afd00000-afd41000 r-xp 00000000 b3:19 116 /system/lib/libc.so
afd41000-afd44000 rwpx 00041000 b3:19 116 /system/lib/libc.so
afd44000-afd4f000 rwpx 00000000 00:00 0
b0001000-b000c000 r-xp 00001000 b3:19 1025 /system/bin/linker
b000c000-b000d000 rwpx 0000c000 b3:19 1025 /system/bin/linker
b000d000-b0021000 rwpx 00000000 00:00 0
bead7000-bead8000 rwpx 00000000 00:00 0 [stack]
#
```

記錄下此位址。

5. 接下來查看 libtest.so 的 .text 區段中的偏移量(offset)

在 Host 中執行指令：

AndroidFroyoSrc/prebuilt/linux-x86/toolchain/arm-eabi-4.4.0/bin/arm-eabi-objdump -h

AndroidFroyoSrc/out/target/product/ventana/obj/SHARED_LIBRARIES/libtest_intermediates/LINKED/libtest.so |grep .text

```
william@will-i-am:~/Project/pbj30K32-Cln$ ./prebuilt/linux-x86/toolchain/arm-eabi-4.4.0/bin/arm-eabi-objdump -h ./out/target/product/ventana/obj/SHARED_LIBRARIES/libtest_intermediates/LINKED/libtest.so |grep .text
5 .text 00000034 00002e0 000002e0 000002e0 2**2
```

記下位址：0x00000034。

6. Host 中的 GDB client 設定如同 <Standalone Program Debug> 中步驟 4，5，6，7，除了將 "testFB" 改成 "HelloWorld"。

7. 於 Host 中 GDB clientgdb 中載入動態函式庫。

先將上述步驟 4，5 取得的位址作加總： $0x80000000 + 0x00000034 = 0x80000034$ 。

在 GDB client 中執行命令：

#> add-symbol-file

AndroidFroyoSrc/out/target/product/ventana/obj/SHARED_LIBRARIES/libtest_intermediates/LINKED/libtest.so 0x80000034

8. 接下來，如同<Standalone Program Debug>中步驟 8，於 Host GDB client 中，執行各項 GDB 指令。

```
(gdb) b 8
Breakpoint 3 at 0x83bc: file external/shareLib/helloworld/HelloWorld.c, line 8.
(gdb) c
Continuing.

Breakpoint 3, main () at external/shareLib/helloworld/HelloWorld.c:9
9      while(a--) {
(gdb) n
10         printf("while 1\n");
(gdb) n
11         func();
(gdb) s
func () at external/shareLib/./libtest/libtest.c:5
5      printf("\nlibtest 1\n");
(gdb) list
1      #include "stdio.h"
2      #include "libtest.h"
3
4      void func() {
```

上方為 Host 上的 GDB client 中，進入 libtest.so 的原程式中執行 GDB 單步執行指令。

```
bead7000-bead8000 rwxp 00000000 00:00 0
# main 1
while 1

libtest 1
Hello, This is a function in libtest!
libtest 2

while 2
while 1

libtest 1
Hello, This is a function in libtest!
libtest 2

while 2
while 1

libtest 1
Hello, This is a function in libtest!
libtest 2

while 2
while 1

libtest 1
Hello, This is a function in libtest!
libtest 2
```

上方爲程式執行結果。

上述測試用檔案載點 (**GitHub**) :

<https://github.com/wiliwe/android-gdb-example.git>

使用 **Git** 工具(<http://git-scm.com/downloads>)下載

git clone https://github.com/wiliwe/android-gdb-example.git

Reference

1. nVidia GDB document

http://developer.download.nvidia.com/tegra/docs/android_gdb_debugging.pdf

2. GDB usage

<http://www.cmlab.csie.ntu.edu.tw/~daniel/linux/gdb.html>

<http://tetralet.luna.com.tw/index.php?op=ViewArticle&articleId=187&blogId=1>

http://opencsl.openfoundry.org/Lab05_debugger.rst.html

3. Android GDB

<http://blog.wjmjimmie.cn/2010/08/02/android%E7%B3%BB%E7%BB%9F%E4%B8%AD%E8%B0%83%E8%AF%95%E5%8A%A8%E6%80%81%E9%93%BE%E6%8E%A5%E5%BA%93so%E6%96%87%E4%BB%B6%E7%9A%84%E6%AD%A5%E9%AA%A4/>

<http://www.cprogramming.com/debugging/segfaults.html>