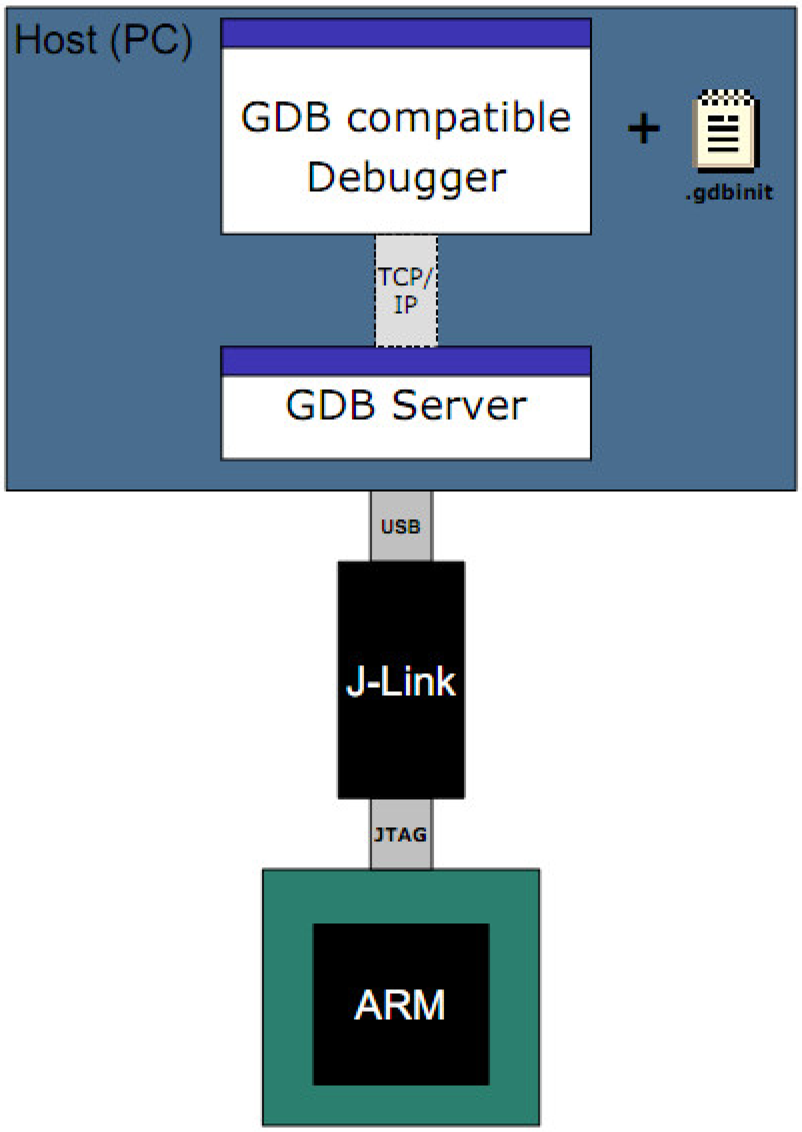
安裝JLlink驅動軟件後，通過附帶的J-Llink GDB Server軟件可以實現利用gdb調試開發板。經過數天的摸索，終於試驗成功，並可以通過這種方式調試U-boot。

一、 原理

借用一下JlinkGdbServer用戶手冊中的一副圖片，說明一下gdb、gdbserver、jlink和開發板之間的聯繫。gdb在開始運行時會首先在用戶目錄下尋找.gdbinit文件，並執行文件裡面記錄的初始化命令，執行完畢之後再轉入正常的流程，接受用戶輸入的命令。

由於gdb和gdb server都有Linux和Windows的版本，這裡面我為了方便，gdb選用的是虛擬機Linux環境下3.14版本的arm-linux-gdb，通過TCP/IP的方式連接到主機上的J-Llink GDB Server。



二、.gdbinit文件

利用gdb調試開發板的時候需要先執行一些初始化命令，否則有可能不能正常調試。初始化命令其實也就是 禁止看門狗，初始化時鐘等等，類似於uboot的start.s文件，參考GDB Server用戶手冊中給出的格式和一些網上資源，自己寫了一個.gdbinit文件。

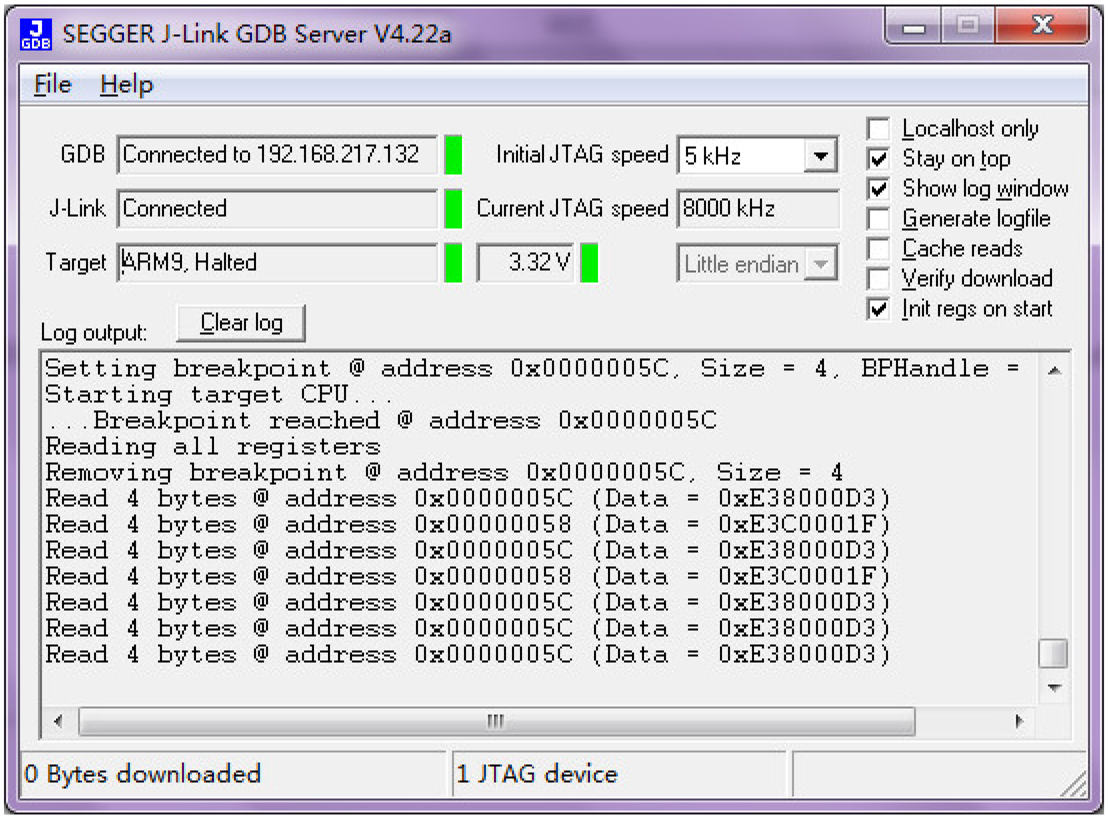
```
#
```

```
# J-LINK GDB SERVER initialization

#

# This connects to a GDB Server listening

# for commands on localhost at tcp port 2331

target remote 192.168.217.1:2331 #首先連接主機的GDB Server，端口都是2331。

#注意主機的GDB Server勾掉"Localhost only"選項，否則連接不上

# Set JTAG speed to 30 kHz

monitor speed 30



# Set GDBServer to little endian

monitor endian little



# Reset the chip to get to a known state.

monitor reset



#

# CPU core initialization

#



# Set the processor mode

monitor reg cpsr = 0xd3
```

```
#disable watchdog

monitor MemU32 0x53000000 = 0x00000000


#disable interrupt

monitor MemU32 0x4A000008 = 0xFFFFFFFF #INTMSK

monitor MemU32 0x4A00000C = 0x00007FFF #INTSUBMSK


#set clock

monitor MemU32 0x4C000000 = 0x00FFFFFF

monitor MemU32 0x4C000014 = 0x00000005

monitor MemU32 0x4C000004 = 0x0005C011


#config sdram

monitor MemU32 0x48000000 = 0x22011110 #conw

monitor MemU32 0x48000004 = 0x00000700 #bank0

monitor MemU32 0x48000008 = 0x00000700 #bank1

monitor MemU32 0x4800000C = 0x00000700 #bank2

monitor MemU32 0x48000010 = 0x00000700 #bank3

monitor MemU32 0x48000014 = 0x00000700 #bank4

monitor MemU32 0x48000018 = 0x00000700 #bank5
monitor MemU32 0x4800001C = 0x00018005 #bank6

monitor MemU32 0x48000020 = 0x00018005 #bank7

monitor MemU32 0x48000024 = 0x008E04F4 #vREFRESH

monitor MemU32 0x48000028 = 0xB1 #vBANKSIZE -- 128M/128M --- should

monitor MemU32 0x4800002c = 0x30 #vMRSRB6

monitor MemU32 0x48000030 = 0x30 #vMRSRB7


# Set auto JTAG speed

monitor speed auto


# Setup GDB FOR FASTER DOWNLOADS

set remote memory-write-packet-size 1024

set remote memory-write-packet-size fixed


# Load the program executable called "image.elf"

# load image.elf

b _start

load

continue
```

三、調試U-Boot(Flash中運行)

U-Boot默認編譯出來有幾種格式的文件，調試的時候只使用其中的兩種格式：elf格式的文件用於gdb調試，bin文件用於下載到開發板去執行。先將u-boot.bin文件經過J-Link FLASH下載到開發板的nor flash之後，linux下執行下面的命令開始調試。成功的話，有下面的結果。



GDB: Connected to 192.168.217.132
J-Link: Connected
Target: ARM9, Halted
3.32 V
Initial JTAG speed: 5 kHz
Current JTAG speed: 8000 kHz
Little endian
[ ] Localhost only
[✓] Stay on top
[✓] Show log window
[ ] Generate logfile
[ ] Cache reads
[ ] Verify download
[✓] Init regs on start

```
Setting breakpoint @ address 0x0000005C, Size = 4, BPHandle =
Starting target CPU...
...Breakpoint reached @ address 0x0000005C
Reading all registers
Removing breakpoint @ address 0x0000005C, Size = 4
Read 4 bytes @ address 0x0000005C (Data = 0xE38000D3)
Read 4 bytes @ address 0x00000058 (Data = 0xE3C0001F)
Read 4 bytes @ address 0x0000005C (Data = 0xE38000D3)
Read 4 bytes @ address 0x00000058 (Data = 0xE3C0001F)
Read 4 bytes @ address 0x0000005C (Data = 0xE38000D3)
Read 4 bytes @ address 0x0000005C (Data = 0xE38000D3)
Read 4 bytes @ address 0x0000005C (Data = 0xE38000D3)
```

0 Bytes downloaded
1 JTAG device

```
root@ desktop:/home/desktop/debug# arm-linux-gdb u-boo.elf

GNU gdb (Sourcery G++ Lite 2008q3-72) 6.8.50.20080821-cvs

Copyright (C) 2008 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-none-linux-gnu
```

```
eabi".

    For bug reporting instructions, please see:

    <https://support.codesourcery.com/GNUToolchain/>...

    0x00000000 in ?? ()

    JTAG speed set to 30 kHz

    Target endianess set to "little endian"

    Resetting target

    Writing register (CPSR = 0x000000D3)

    Writing 0x00000000 @ address 0x53000000

    Writing 0xFFFFFFFF @ address 0x4A000008

    Writing 0x00007FFF @ address 0x4A00000C

    Writing 0x00FFFFFF @ address 0x4C000000

    Writing 0x00000005 @ address 0x4C000014

    Writing 0x0005C011 @ address 0x4C000004

    Writing 0x22011110 @ address 0x48000000

    Writing 0x00000700 @ address 0x48000004

    Writing 0x00000700 @ address 0x48000008

    Writing 0x00000700 @ address 0x4800000C
Writing 0x00000700 @ address 0x48000010

Writing 0x00000700 @ address 0x48000014

Writing 0x00000700 @ address 0x48000018

Writing 0x00018005 @ address 0x4800001C

Writing 0x00018005 @ address 0x48000020

Writing 0x008E04F4 @ address 0x48000024

Writing 0x000000B1 @ address 0x48000028

Writing 0x00000030 @ address 0x4800002C

Writing 0x00000030 @ address 0x48000030

Select auto JTAG speed (8000 kHz)

The target may not be able to correctly handle a memory-write-packet-size

of 1024 bytes. Change the packet size? (y or n) [answered Y; input not from terminal]

/root/.gdbinit:60: Error in sourced command file:

No executable file specified.

Use the "file" or "exec-file" command.

(gdb) n

start_code () at start.S:121

121 mrs r0, cpsr
```

```
Current language: auto; currently asm

(gdb) n

start_code () at start.S:122

122 bic r0, r0, #0x1f

(gdb)

start_code () at start.S:123

123 orr r0, r0, #0xd3

(gdb)
```

四、調試U-Boot(RAM中運行)

U-Boot最開始是在Flash中運行，初始化之後，會將自身代碼搬移到RAM中運行，由於鏈接的時候指定了運行地址是從0x0開始的，所以U-Boot轉移到RAM中運行之後，其真正的運行地址就和編譯出的elf文件中的地址就不對應了，通過反編譯elf文件就可以看出來，相差一個偏移。gdb應該是根據elf文件中記錄的符號表（函數名，變量名）和其對應的運行地址來進行調試的，現在實際的運行地址變了，gdb也就找不到地址所對應的符號了，那怎麼辦呢？

U-Boot在鏈接的時候運行地址是由CONFIG_SYS_TEXT_BASE這個宏指定的，這個宏最初是被定義為0的，也就是運行地址是從0地址開始，現在把它修改為實際的運行地址，重新編譯一個elf文件，現在這個elf文件中的地址就是和實際的運行地址一樣了，再使用這個elf文件來調試的話，就可以繼續調試RAM中運行的代碼了。

```
root@linsen-desktop:/home/linsen/u-boot/relocate# arm-linux-gdb u-boot

GNU gdb (Sourcery G++ Lite 2008q3-72) 6.8.50.20080821-cvs

Copyright (C) 2008 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-none-linux-gnueabi".

For bug reporting instructions, please see:

<https://support.codesourcery.com/GNUToolchain/>...

0x00000000 in ?? ()

JTAG speed set to 30 kHz

Target endianess set to "little endian"

Resetting target

Writing register (CPSR = 0x000000D3)

Writing 0x00000000 @ address 0x53000000

Writing 0xFFFFFFFF @ address 0x4A000008
Writing 0x00007FFF @ address 0x4A00000C

Writing 0x00FFFFFF @ address 0x4C000000

Writing 0x00000005 @ address 0x4C000014

Writing 0x0005C011 @ address 0x4C000004

Writing 0x22011110 @ address 0x48000000

Writing 0x00000700 @ address 0x48000004

Writing 0x00000700 @ address 0x48000008

Writing 0x00000700 @ address 0x4800000C

Writing 0x00000700 @ address 0x48000010

Writing 0x00000700 @ address 0x48000014

Writing 0x00000700 @ address 0x48000018

Writing 0x00018005 @ address 0x4800001C

Writing 0x00018005 @ address 0x48000020

Writing 0x008E04F4 @ address 0x48000024

Writing 0x000000B1 @ address 0x48000028

Writing 0x00000030 @ address 0x4800002C

Writing 0x00000030 @ address 0x48000030

Select auto JTAG speed (8000 kHz)
```

```
The target may not be able to correctly handle a memory-write-packet-size

of 1024 bytes. Change the packet size? (y or n) [answered Y; input not from term

/root/.gdbinit:60: Error in sourced command file:

No executable file specified.

Use the "file" or "exec-file" command.

(gdb) b board_init_r <-------uboot運行到這個函數的時候已經是在RAM中運行了

Breakpoint 1 at 0x33f44060: file board.c, line 455.

(gdb) c

Continuing.



Breakpoint 1, board_init_r (id=0x33b21f70, dest_addr=871636992) at board.c:455

455 gd = id;

(gdb) n

457 gd->flags |= GD_FLG_RELOC; /* tell others: relocation done */

(gdb) n

459 monitor_flash_len = _end_ofs;

(gdb) b do_ping <-------同樣可以調試命令處理函數

Breakpoint 2 at 0x33f4c6d0: file cmd_net.c, line 269.
```

```
        (gdb) c

        Continuing.



        Breakpoint 2, do_ping (cmdtp=0x33fac72c, flag=0, argc=2, argv=0x33b23430) at
cmd_net.c:269

        269 if (argc < 2)

        (gdb) p argv[1]

        $3 = 0x33b23440 "192.168.0.1"
```