

學號：404410030

姓名：鄭光宇

Email：jengku@gmail.com

Lab.2 Cross Toolchain for ARM Linux (GDB)

實驗目的：

藉由實際從原始碼編譯一次 gdb，了解 gdb 的編譯步驟，讓我們之後如果要修改、加入 gdb 程式碼時，可以更有相關經驗、概念。並利用這次實驗練習 remote debugging，了解 remote debugging 的流程與基本原理。

實驗步驟：

實驗分成兩大步驟：編譯 gdb、使用 gdb, gdbserver 實現 remote debugging。

接續 Lab.1，我們要將 gdb 編譯進上次我們在 Lab.1 編譯好的 toolchains，詳細步驟如下：

我們寫一個 bash script，下載原始碼、編譯 gdb 到 toolchains 中，每個步驟都有附上註解。

```
#!/bin/bash

sudo apt-get install texinfo libncurses5-dev libx11-dev # 安裝相關套件

### 下載、解壓 gdb
wget ftp://ftp.gnu.org/gnu/gdb/gdb-8.1.tar.gz
tar zxvf gdb-8.1.tar.gz

### 建立用來編譯 gdb 的臨時目錄
rm -rf build_gdb # 清除舊的臨時目錄
mkdir build_gdb
cd build_gdb

### 在臨時目錄中，編譯、設定 gdb，並指定 target 名稱，與安裝路徑
### 安裝路徑可以利用執行這份 bash script 時，以環境變數 TOOL_CHAIN_PATH 傳入
### 例如：TOOL_CHAIN_PATH=/home/peter/toolchains ./install_gdb.sh
../gdb-8.1/configure --prefix=${TOOL_CHAIN_PATH} \
                    --target="arm-linux-gnueabi" \
                    --enable-tui

### 以五個 jobs 加速 gdb 編譯
make -j5
### 安裝到設定 gdb 時指定的 prefix
make install
```

編譯好 gdb 後，我們先在 raspberry pi 上，安裝 gdbserver，用來完成 remote debugging：

```
sudo apt-get install gdbserver
```

之後，我們將 raspberry pi 的網域設定到與 host 相同網域，之後輸入 `ifconfig` 指令查詢 raspberry pi 的 IP。

```
pi@raspberrypi:~$ ifconfig | grep -e wlan -e inet
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.20.10.13 netmask 255.255.255.240 broadcast 172.20.10.15
    inet6 fe80::229e:a4f3:541b:77a prefixlen 64 scopeid 0x20<link>
```

將 IP 抄下來。

我們在 host 上編譯實驗要用的 test.exe：

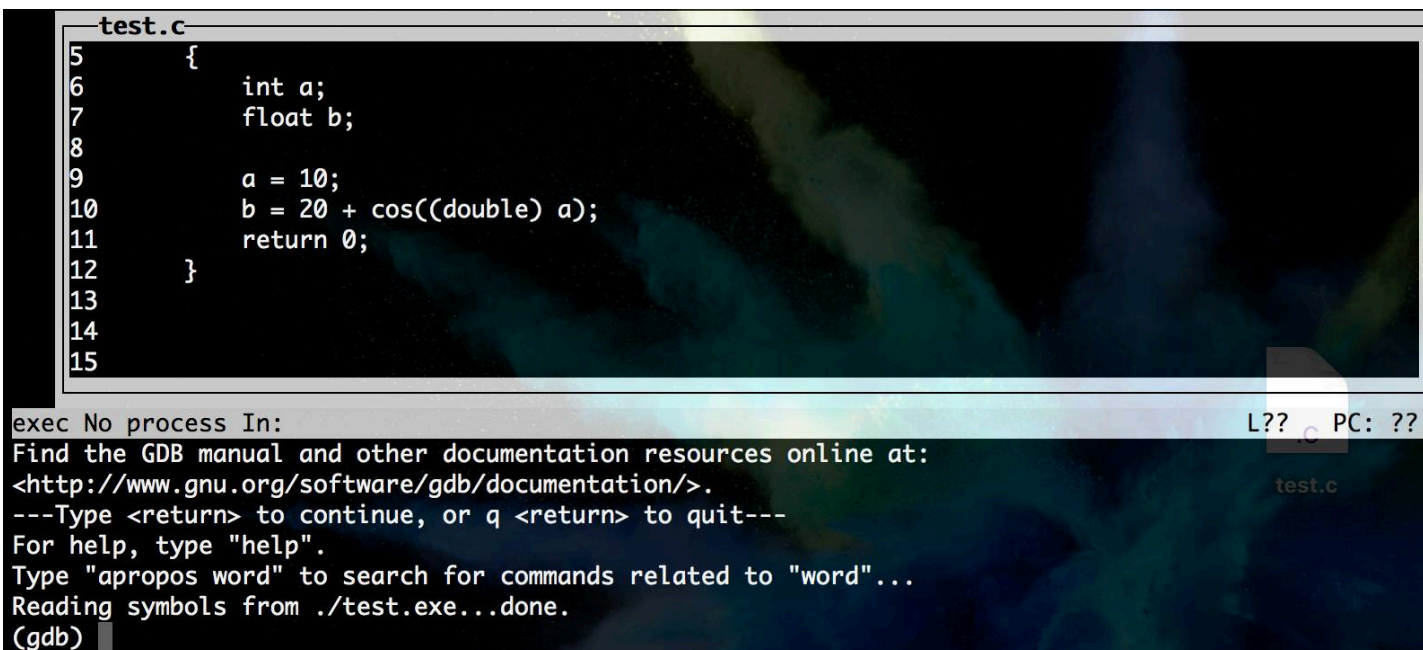
```
ubuntu@ed9cd29fe287:~$ ./rpi_toolchains/crossgcc2/bin/arm-linux-gnueabi-gcc --sysroot="$PWD/rpi_toolchains/sysroot" --static -g test.c -lm -o test.exe
ubuntu@ed9cd29fe287:~$ ls -s test.exe
2888 test.exe
ubuntu@ed9cd29fe287:~$
```

在 raspberry pi (target) 上開啟 gdbserver，IP 為之前 ifconfig 上看到的那一組，port 隨意指定，指定要執行 `test.exe`，並記住這個 port 號。被除錯的 `test.exe` 會停在程式剛執行的地方：

```
pi@raspberrypi:~$ gdbserver 172.20.10.13:3456 ./test.exe
Process ./test.exe created; pid = 929
Listening on port 3456
```

在 host 端，開啟 gdb (用 `-tui` 模式開啟)：

```
ubuntu@ed9cd29fe287:~$ ./rpi_toolchains/crossgcc2/bin/arm-linux-gnueabi-gdb -tui ./test.exe
```



The screenshot shows the GDB TUI interface. At the top, the source code of `test.c` is displayed in a window titled "test.c". The code is as follows:

```
5  {
6      int a;
7      float b;
8
9      a = 10;
10     b = 20 + cos((double) a);
11     return 0;
12 }
13
14
15
```

Below the code window, the GDB prompt is visible. It shows the message "exec No process in:" and "L?? PC: ??". Below this, it says "Find the GDB manual and other documentation resources online at: <http://www.gnu.org/software/gdb/documentation/>." and "---Type <return> to continue, or q <return> to quit---". It also shows "For help, type 'help'." and "Type 'apropos word' to search for commands related to 'word'...". Finally, it says "Reading symbols from ./test.exe...done." and "(gdb)".

輸入 `b 10`，在第十行設定斷點 (breakpoint)：

```
test.c
5      {
6          int a;
7          float b;
8
9          a = 10;
10         b = 20 + cos((double) a);
11         return 0;
12     }
13
14
15
b+
exec No process In: L?? PC: ??
---Type <return> to continue, or q <return> to quit---
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./test.exe...done.
(gdb) b 10
Breakpoint 1 at 0x109f4: file test.c, line 10.
(gdb)
```

輸入 `target remote 你的IP:你的port`，開始 remote debug：

```
test.c
[ No Source Available ]
remote Thread 929.929 In: _start L79 PC: 0x10874
Reading symbols from ./test.exe...done.
(gdb) b 10
Breakpoint 1 at 0x109f4: file test.c, line 10.
(gdb) target remote 172.20.10.13:3456
Remote debugging using 172.20.10.13:3456
_start () at ../ports/sysdeps/arm/start.S:79
```

輸入 `continue`，遠端的 `test.exe` 會從 gdbserver 幫我們自動設的斷點繼續執行到我們設定的斷點 (第十行)：

```
test.c
6      int a;
7      float b;
8
9      a = 10;
B+> 10     b = 20 + cos((double) a);
11     return 0;
12 }
13
14
15
16

remote Thread 929.929 In: main L10 PC: 0x109f4
Remote debugging using 172.20.10.13:3456
_start () at ../ports/sysdeps/arm/start.S:79
(gdb) continue
Continuing.

Breakpoint 1, main () at test.c:10
(gdb) █
```

接下來就可以練習 remote debugging 了。

試著使用 step :

```
Breakpoint 1, main () at test.c:10
(gdb) s
__cos (x=10) at ../sysdeps/ieee754/dbl-64/s_sin.c:519
(gdb) █

(gdb) continue
Continuing.
[Inferior 1 (process 929) exited normally]
(gdb) █
```

其他，用 gdb 印出記憶體、register 等資訊：

remote Thread 933.933 In: main

(gdb) info register

r0	0x1	1	
r1	0x7efff354		2130703188
r2	0x7efff35c		2130703196
r3	0xa	10	
r4	0x7efff218		2130702872
r5	0x1728c	94860	
r6	0x0	0	
r7	0x10134	65844	
r8	0x0	0	
r9	0x0	0	
r10	0x0	0	
r11	0x7efff1fc		2130702844
r12	0x0	0	
sp	0x7efff1f0		0x7efff1f0
lr	0x16db0	93616	
pc	0x109f4	0x109f4	<main+20>
cpsr	0x60000010		1610612752
fpscr	0x0	0	

No symbol 0x0 in current context.

(gdb) p &a

\$1 = (int *) 0x7efff1f4

(gdb) x/10 0x7efff1f4

0x7efff1f4:	10	0	93616	0
0x7efff204:	1	2130703188	68064	0
0x7efff214:	0	65844		

(gdb) p &b

\$2 = (float *) 0x7efff1f0

(gdb) x/10 0x7efff1f0

0x7efff1f0:	65844	10	0	93616
0x7efff200:	0	1	2130703188	68064
0x7efff210:	0	0		

(gdb) █

問題與討論：

What is "GDB server"?

一個用來做遠端除錯的程式，我們會在 target 端開啟 gdbserver，去控制 target 上要被除錯的程式，透過 TCP 或者 Serial Port 與 host 溝通。使用 gdbserver 時，target 端不需要實際安裝 gdb，而且 gdbserver 遠比 gdb 小，因此可以快速地 porting，很快達成遠端 debug。使用 gdbserver 時，我們需要一份已經編譯好的可執行檔在 target 端上，而 host 端需要同一份可執行檔，與它的 source code。

What is "GDB stub"?

GDB stub 是一個用來監測、控制要被 debug 的程式的程式，會和要被 debug 的程式 link 在一起，藉由接收 Remote Serial Protocol 與 host 通訊，將 host 的指令，利用 gdb stub 去控制要被除錯的程式。