

Remote Debugging

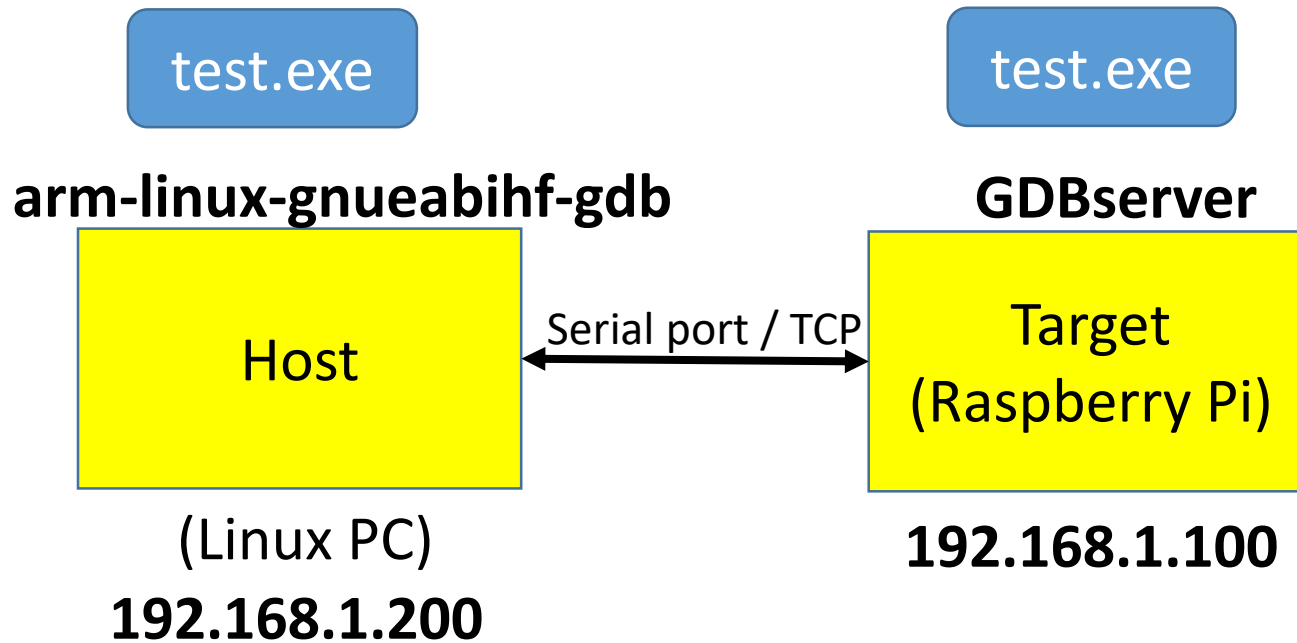
開發教師：陳鵬升

國立中正大學 資訊工程學系

What can You Learn?

- 如何進行remote debugging
- 編譯GDB server

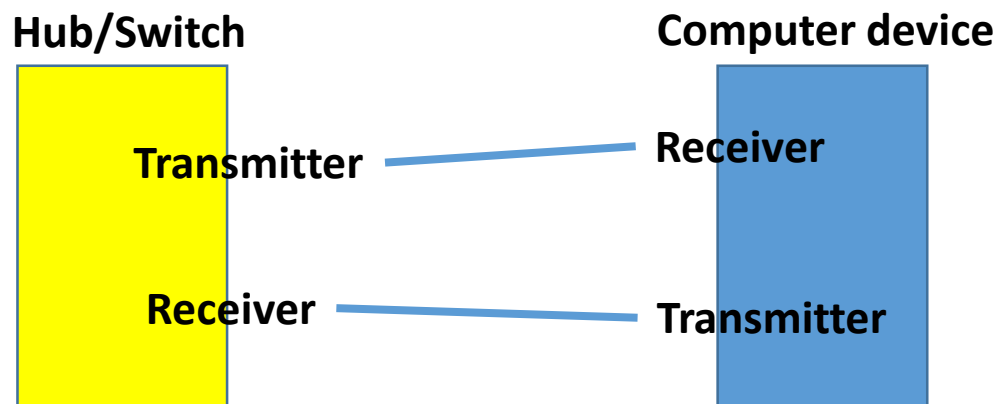
Remote Debugging



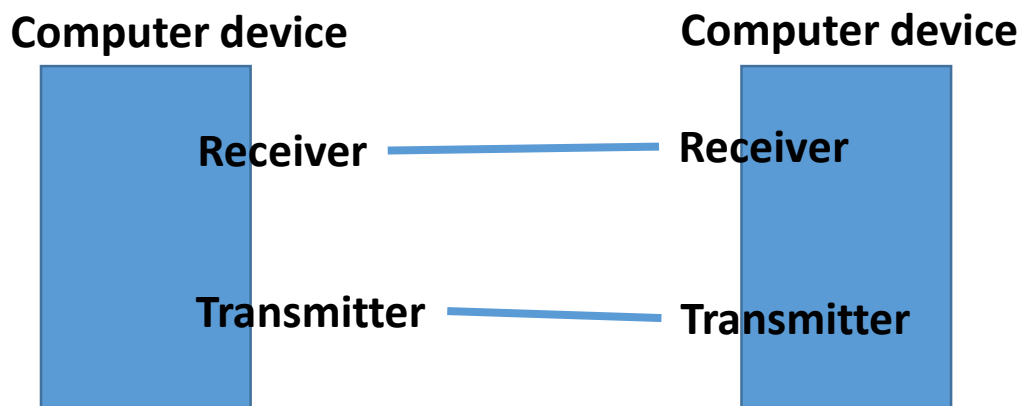
Network Configuration (1)

- 使用跳線的網路線連結target machine與host machine
- Setup an IP on target machine
- Setup an IP on virtual machine
- 關閉無線網路

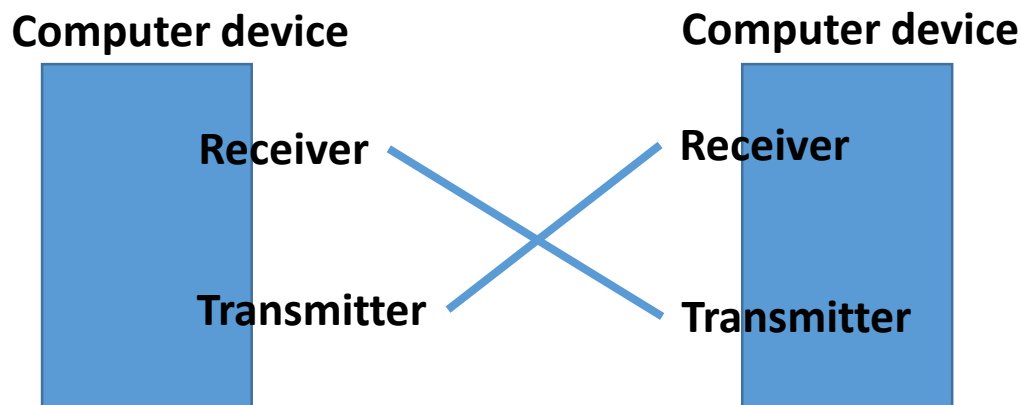
為什麼需要跳線的網路線？



為什麼需要跳線的網路線？



為什麼需要跳線的網路線？



Network Configuration (2)

- 設定 Linux PC 與 Raspberry Pi 的IP

- 方法1:

```
$ ifconfig eth0 192.168.1.100 broadcast 192.168.1.255  
netmask 255.255.255.0
```

```
$ ifconfig eth0 (顯示網路設定狀況)
```


Network Configuration (3)

- 方法2:

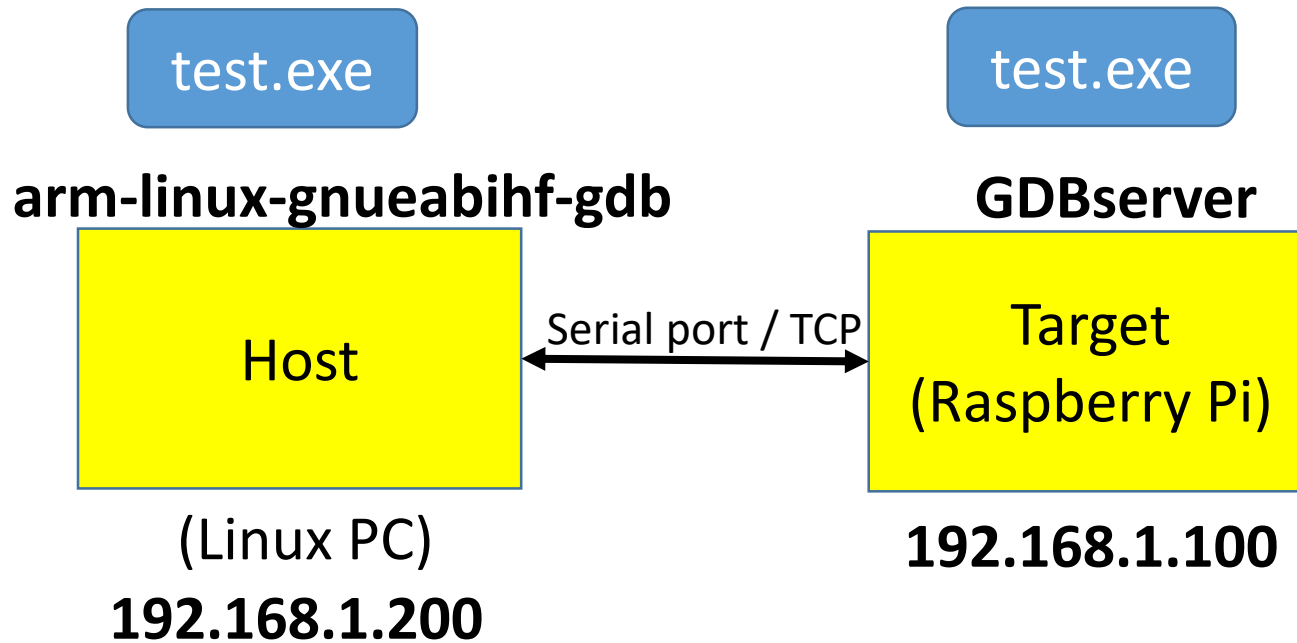
Edit the file: /etc/network/interfaces

```
auto eth0
iface eth0 inet static
address 192.168.1.100
netmask 255.255.255.0
Broadcast 192.168.1.255
gateway 192.168.1.254
```

重新開機

```
$ reboot
```

Remote Debugging



GDBserver (1)

編譯GDBserver

- 新版的GDBserver需要C++編譯器
- 如果你的cross toolchain沒有支援C++，請：
 - 重新編譯cross toolchain，設定
`--enable-languages=c, c++`
 - 下載別人已經編譯好的cross toolchain

GDBserver (2)

- Change to the directory <GDB-source>/gdb/gdbserver/

如果你的cross compiler執行檔為arm-linux-gnueabi-hf-gcc

```
$ export CC=arm-linux-gnueabi-hf-gcc
```

```
$ export CXX=arm-linux-gnueabi-hf-g++
```

```
$ export CFLAGS=-static
```

```
$ export CXXFLAGS=-static
```

```
$ ./configure --host=arm-linux-gnueabi-hf  
--target=arm-linux-gnueabi-hf
```

```
$ make
```

若一切順利，可以再同目錄下發現gdbserver的執行檔

Tested Program

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int i = 0;
6      int j = 0;
7
8      for (i=0; i<100; i++) {
9          j = j + 1;
10     }
11
12     printf("j = %d\n", j);
13     return 0;
14 }
```

Generate Debugged Program

- 產生 `test_static.exe`、`test.exe`、`test_static_g.exe`、`test_g.exe` 四個執行檔 (程式內容自訂)

```
$ arm-linux-gnueabihf-gcc -static test.c -o  
test_static.exe
```

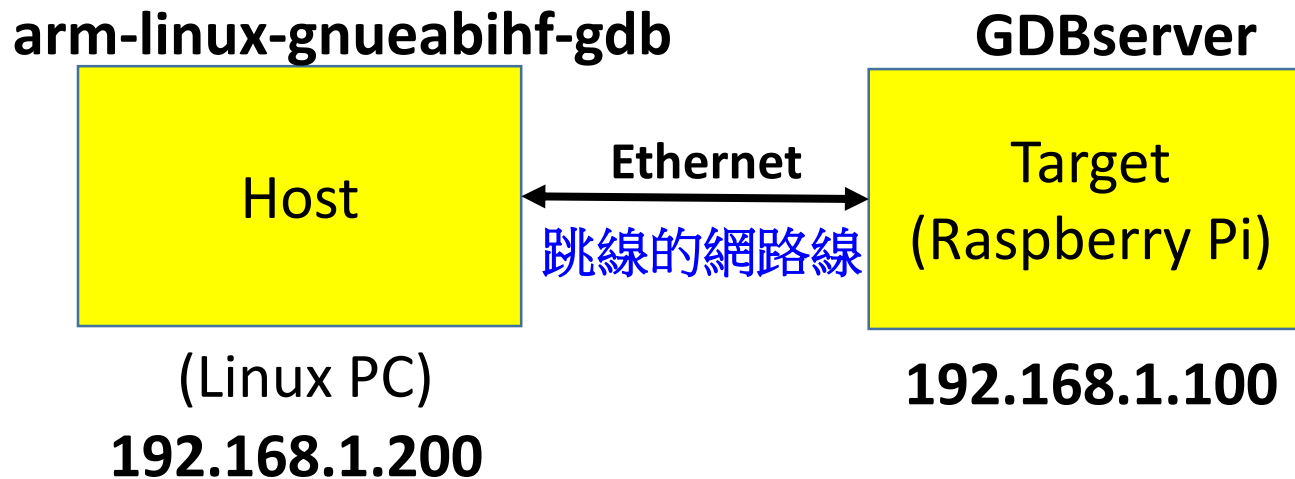
```
$ arm-linux-gnueabihf-gcc test.c -o test.exe
```

```
$ arm-linux-gnueabihf-gcc -static -g test.c -o  
test_static_g.exe
```

```
$ arm-linux-gnueabihf-gcc -g test.c -o test_g.exe
```

- 將 `gdbserver` 執行檔、四個測試執行檔、source code 放置到 Pi 上 (目錄可以自己決定，ex: 放置在 /root 目錄下)

- 使用跳線的網路線連結Linux PC與Raspberry Pi
- 使用 **ifconfig** 命令確認Pi與Linux PC的IP設定是否正確



Target Machine (Raspberry Pi)

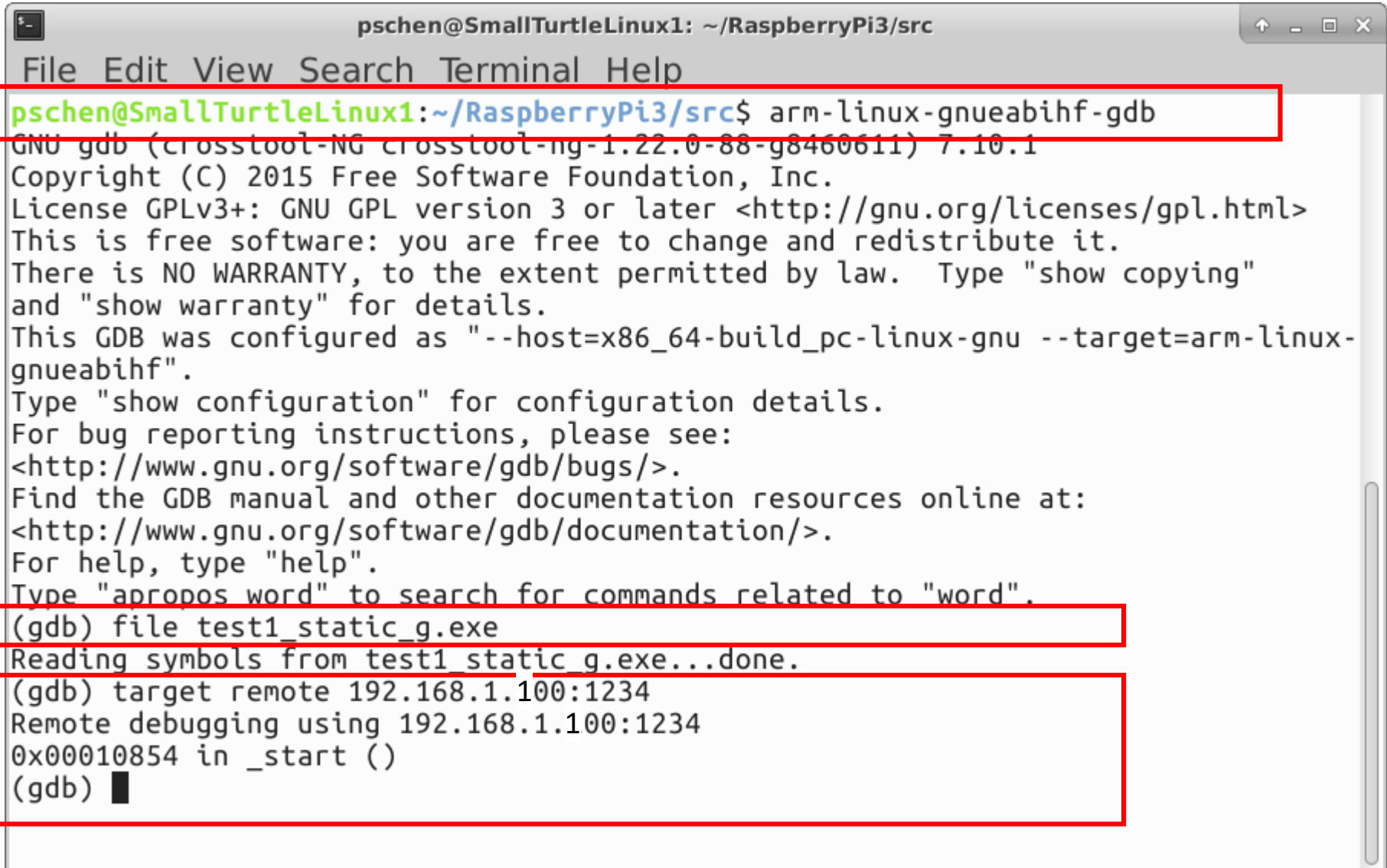
- 登入 Raspberry Pi
- 切換到gdbserver執行檔的目錄

```
$ ./gdbserver 192.168.1.200:1234 ./test_static_g.exe
```

```
Process ./test_static_g.exe created; pid=145
```

```
Listening on port 1234
```


Host Machine (Linux PC)



A terminal window titled "pschen@SmallTurtleLinux1: ~/RaspberryPi3/src" with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of "arm-linux-gnueabihf-gdb", which displays version information and license details for GNU GDB 7.10.1. It then shows the command "(gdb) file test1_static_g.exe" and the response "Reading symbols from test1_static_g.exe...done.". The next command is "(gdb) target remote 192.168.1.100:1234", followed by "Remote debugging using 192.168.1.100:1234" and "0x00010854 in _start ()". The prompt "(gdb) " is shown at the end.

```
pschen@SmallTurtleLinux1: ~/RaspberryPi3/src
File Edit View Search Terminal Help
pschen@SmallTurtleLinux1:~/RaspberryPi3/src$ arm-linux-gnueabihf-gdb
GNU gdb (crosstool-NG crosstool-ng-1.22.0-88-g8460611) 7.10.1
Copyright (C) 2015 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-build_pc-linux-gnu --target=arm-linux-
gnueabihf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) file test1_static_g.exe
Reading symbols from test1_static_g.exe...done.
(gdb) target remote 192.168.1.100:1234
Remote debugging using 192.168.1.100:1234
0x00010854 in _start ()
(gdb) █
```

嘗試設中斷點、除錯、觀看變數值 ...

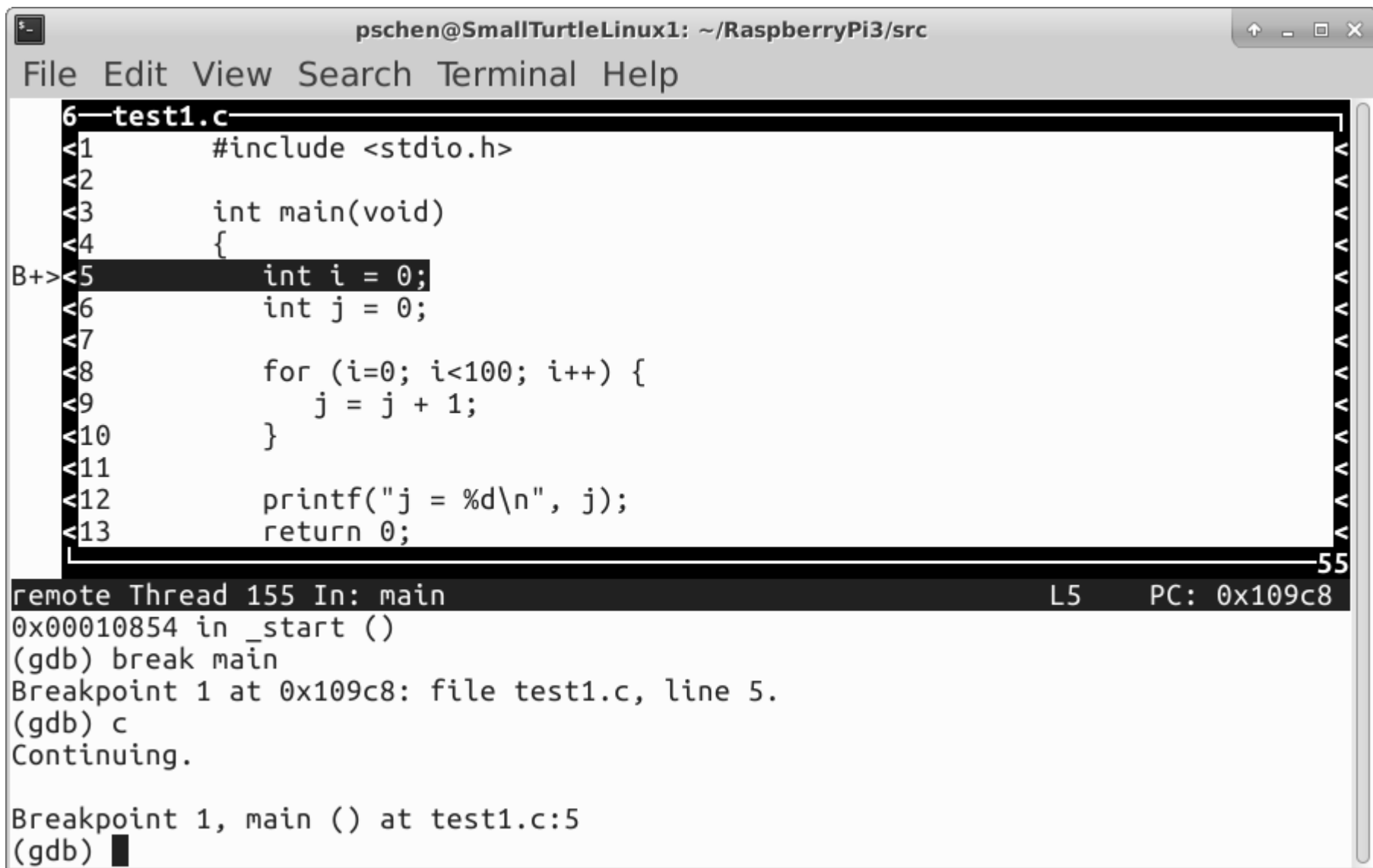
```
pschen@SmallTurtleLinux1: ~/RaspberryPi3/src
File Edit View Search Terminal Help
gnueabihf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) file test1_static_g.exe
Reading symbols from test1_static_g.exe...done.
(gdb) target remote 192.168.1.100:1234
Remote debugging using 192.168.1.100:1234
0x00010854 in start ()
(gdb) break main
Breakpoint 1 at 0x109c8: file test1.c, line 5.
(gdb) c
Continuing.

Breakpoint 1, main () at test1.c:5
5          int i = 0;
(gdb) n
6          int j = 0;
(gdb) watch i
Hardware watchpoint 2: i
```

嘗試設中斷點、除錯、觀看變數值 ...

```
pschen@SmallTurtleLinux1: ~/RaspberryPi3/src
File Edit View Search Terminal Help
5      int i = 0;
(gdb) n
6      int j = 0;
(gdb) watch i
Hardware watchpoint 2: i
(gdb) info r
r0          0x1          1
r1          0x7eb8de44    2126044740
r2          0x7eb8de4c    2126044748
r3          0x0          0
r4          0x7eb8dd08    2126044424
r5          0x1115c      69980
r6          0x0          0
r7          0x10114      65812
r8          0x0          0
r9          0x0          0
r10         0x0          0
r11         0x7eb8dcec    2126044396
r12         0x0          0
sp          0x7eb8dce0    0x7eb8dce0
lr          0x10c98      68760
pc          0x109d0      0x109d0 <main+20>
cpsr       0x60000010    1610612752
(gdb) █
```

嘗試使用 “**arm-linux-gnueabihf-gdb -tui**” 文字視窗模式



The screenshot shows a terminal window titled "pschen@SmallTurtleLinux1: ~/RaspberryPi3/src". The window contains a C program named "test1.c" and the GDB debugger interface. The program code is as follows:

```
6—test1.c—
<1      #include <stdio.h>
<2
<3      int main(void)
<4      {
B+><5          int i = 0;
<6          int j = 0;
<7
<8          for (i=0; i<100; i++) {
<9              j = j + 1;
<10         }
<11
<12         printf("j = %d\n", j);
<13         return 0;
55
```

The GDB interface shows the following commands and output:

```
remote Thread 155 In: main                                L5      PC: 0x109c8
0x00010854 in _start ()
(gdb) break main
Breakpoint 1 at 0x109c8: file test1.c, line 5.
(gdb) c
Continuing.

Breakpoint 1, main () at test1.c:5
(gdb) █
```

- **Q1: 請問 test.exe 、 test_g.exe 、 test_static.exe 、 test_static_g.exe 在Pi上執行的結果如何? 它們之間的差別是什麼?**

<請於實驗報告裡回覆>

Backup: Build a Cross Compiler (包含g++)

- `$../gcc-4.9.3/configure \`
`--prefix=/home/pschen/WORK/crossgcc2 \`
`--target=arm-linux-gnueabihf \`
`--enable-languages=c,c++ \`
`--with-sysroot=/home/pschen/WORK/sysroot`
`--with-arch=armv6 --with-fpu=vfp --with-float=hard \`
`--disable-sjlj-exceptions --enable-__cxa_atexit \`
`--disable-libmudflap --enable-libgomp \`
`--disable-libssp --enable-libquadmath \`
`--enable-libquadmath-support \`
`--disable-libsanitizer --enable-lto \`
`--enable-threads=posix --enable-target-optspace \`
`--with-linker-hash-style=gnu --disable-nls \`
`--disable-multilib --enable-long-long \`
`--with-host-libstdcxx='-static-libgcc -Wl,-Bstatic,-lstdc++,-Bdynamic -lm'`
- `%make`
- `%make install`