本文主要講解開發 Android 平台下的 LKM(Linux Kernel Module)的步驟，以及如何使用 Android Emulator 除錯 LKM。

一、編譯 android 核心

1.首先執行模擬器(emulator 指令所在目錄為 androidsdk/tools/，可將其添加至系統環境變數 PATH 中)

```
emulator -avd android4
```

注意：本人使用的是 android4 的版本，android2.x 的版本也可使用

2. goldfish 核心下載

```
git clone http://android.googlesource.com/kernel/goldfish.git
```

3.從模擬器中將/proc/config.gz 檔案複製到 goldfish(即 kernel)目錄

```
cd goldfish/
adb pull /proc/config.gz .
```

4.將 config.zg 解壓

```
gunzip config.gz
mv config .config
```

5.進行編譯

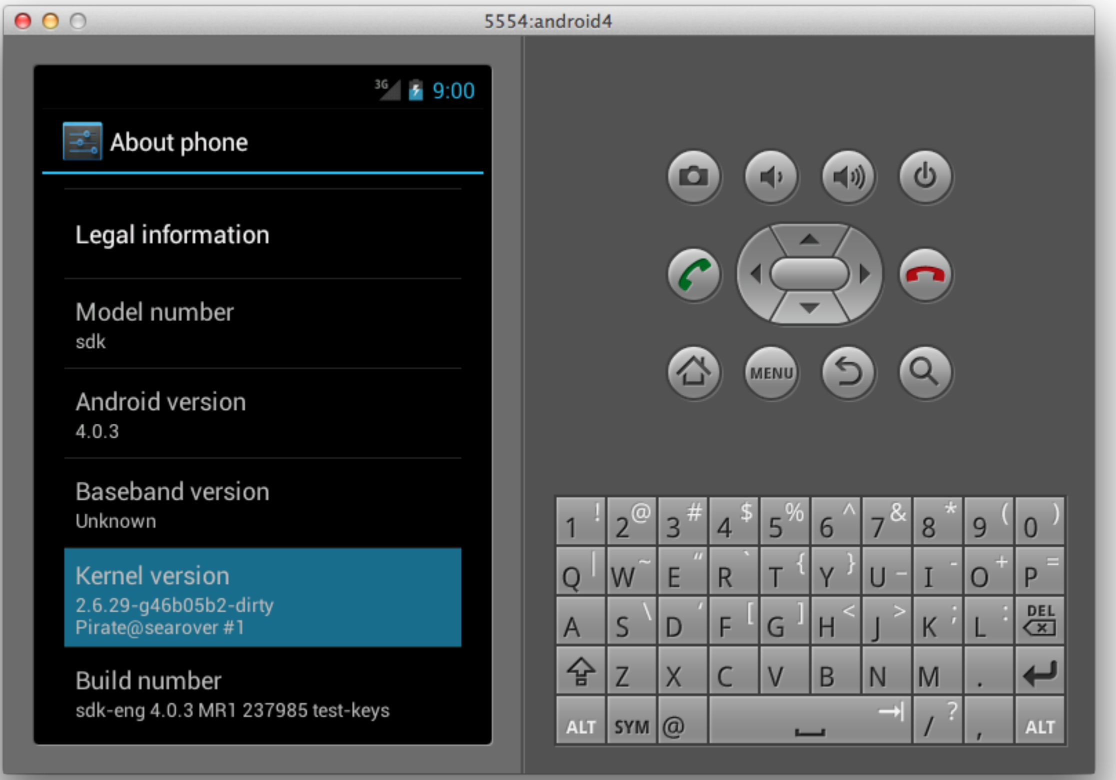
```
make ARCH=arm CROSS_COMPILE=arm-eabi- -j4
```

注意：本人所用 arm toolchain 為 NDKr5c 版本，NDKr7 所用的 gcc 4.4.3 版本編譯后的模組在加載時會出問題，切莫使用。也可自己編譯 arm toolchain

6.通過 emulator 執行剛剛編譯好的 kernel

```
emulator -kernel /Volumes/Software/Android/kernel/goldfish/arch/arm/boot/zImage -avd android4
```

7.通過"About phone"，可以檢視目前核心資訊



Legal information

Model number
sdk

Android version
4.0.3

Baseband version
Unknown

Kernel version
2.6.29-g46b05b2-dirty
Pirate@searover #1

Build number
sdk-eng 4.0.3 MR1 237985 test-keys

輔助說明：也可使用如下指令檢視核心版本

```
cat /proc/version
```

二、編譯"Hello World!"模組，源碼詳見 helloworld.zip

1.解壓并進入到 helloworld 目錄

```
unzip helloworld.zip
cd helloworld
```

2.編譯 hello 模組，編譯成功之會將得到 hello.ko

```
make
```

注意：如果編譯時出現下述錯誤

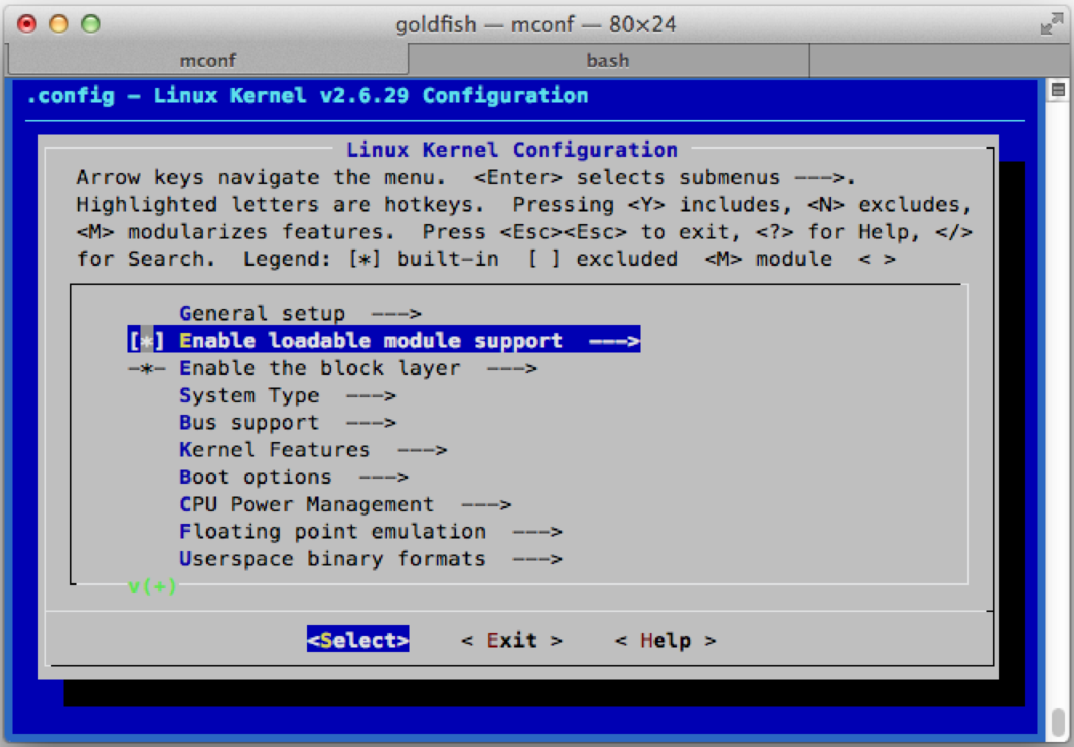
error: variable '__this_module' has initializer but incomplete type

需要配置核心選項，首先執行

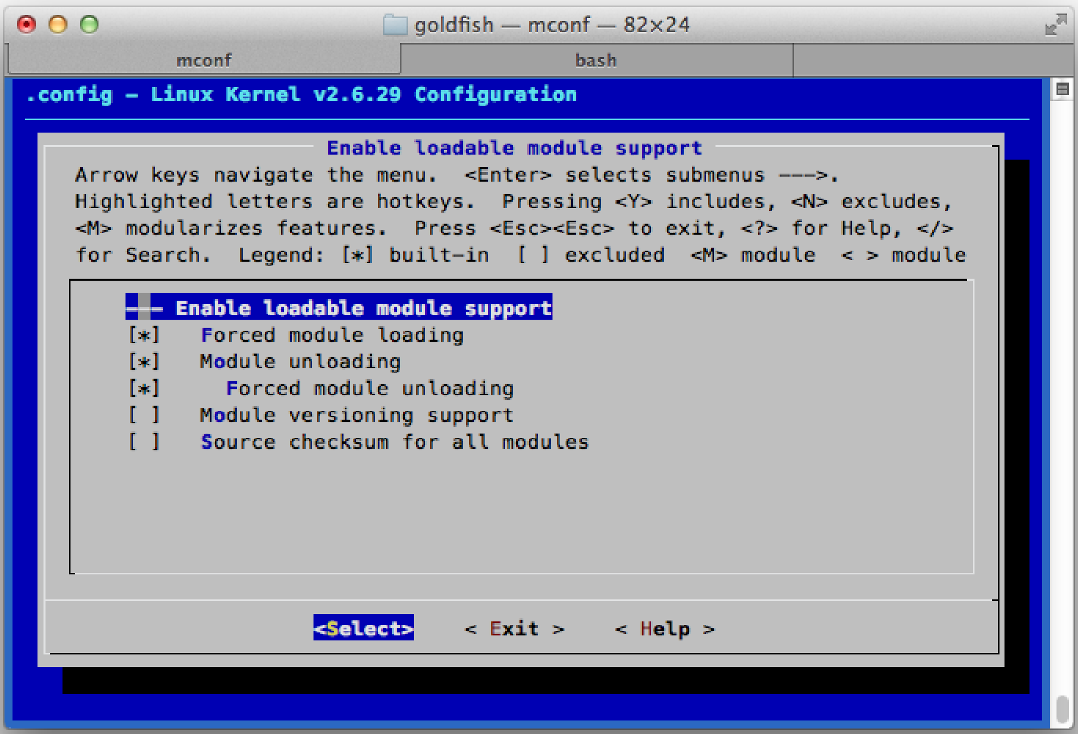
```
make ARCH=arm CROSS_COMPILE=arm-eabi- menuconfig
```

進入核心配置介面，勾選下列選項

```
[*] Enable loadable module support ---> (選中這一項，按白字元即可)
    [*] Forced module loading (選中上述一項，按轉鍵即可看到此項)
    [*] Module unloading
        [*] Forced module unloading
```



```
goldfish — mconf — 80×24
mconf                                   bash

.config - Linux Kernel v2.6.29 Configuration
 ─────────────────────────────────────────────────────────────────────────────
  ┌──────────────────── Linux Kernel Configuration ─────────────────────┐
  │  Arrow keys navigate the menu.  <Enter> selects submenus --->.      │
  │  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  │  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  │  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >   │
  │ ┌────────────────────────────────────────────────────────────────┐ │
  │ │        General setup  --->                                     │ │
  │ │    [*] Enable loadable module support  --->                    │ │
  │ │    -*- Enable the block layer  --->                            │ │
  │ │        System Type  --->                                       │ │
  │ │        Bus support  --->                                       │ │
  │ │        Kernel Features  --->                                   │ │
  │ │        Boot options  --->                                      │ │
  │ │        CPU Power Management  --->                              │ │
  │ │        Floating point emulation  --->                          │ │
  │ │        Userspace binary formats  --->                          │ │
  │ └────v(+)────────────────────────────────────────────────────────┘ │
  ├─────────────────────────────────────────────────────────────────────┤
  │                 <Select>    < Exit >    < Help >                    │
  └─────────────────────────────────────────────────────────────────────┘
```



```
.config - Linux Kernel v2.6.29 Configuration
                       Enable loadable module support
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module

      --- Enable loadable module support
      [*]   Forced module loading
      [*]   Module unloading
      [*]     Forced module unloading
      [ ]   Module versioning support
      [ ]   Source checksum for all modules

                  <Select>    < Exit >    < Help >
```

注意：修改核心選項后要重新編譯核心

3.加載編譯好的 hello.ko 模組
1)將 hello_m.ko 複製到模擬器中
adb push hello.ko /sdcard/

3)加載 hello.ko 模組
adb shell
insmod /sdcard/hello.ko

4)通過下述指令列印核心資訊，看到"Hello, World!"，表示模組加載成功。
dmesg

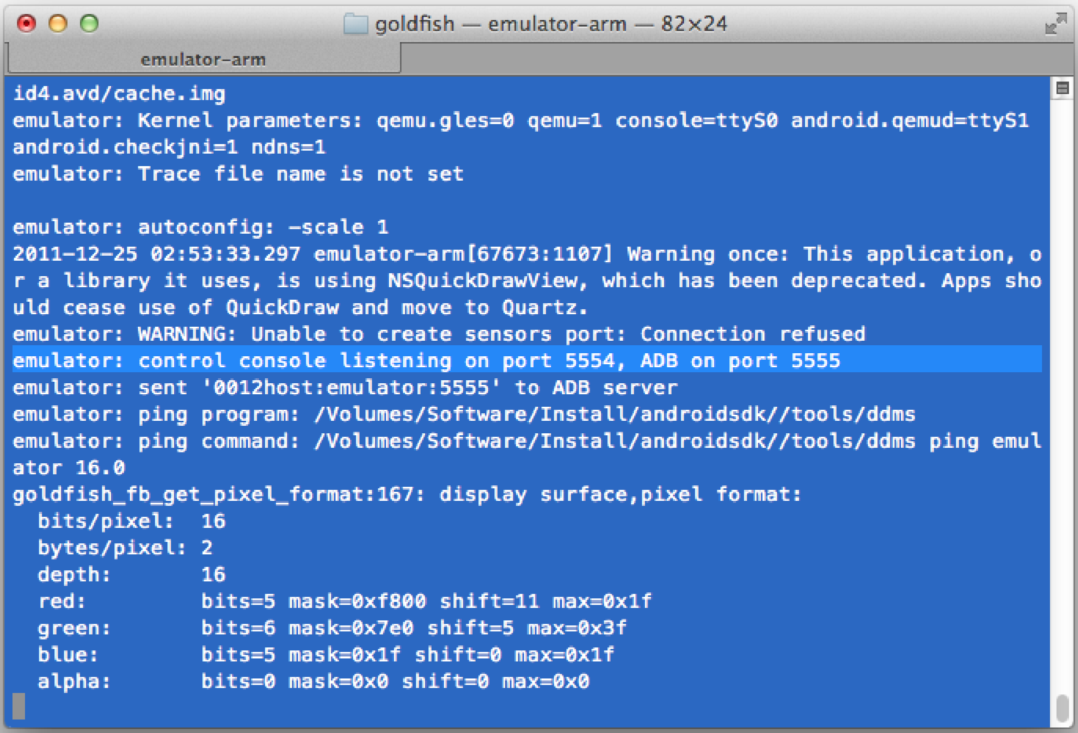
5)檢視已加載模組
lsmod

6)卸載 hello.ko 模組
rmmod hello
卸載模組后可再次執行 dmesg，將會看到"Goodbye!"，表示模組卸載成功。
注意：此時執行的模擬器一定要通過-kernel 參數指定編譯好的 zImage

三、使用 GDB 除錯程式和核心

1.除錯程式

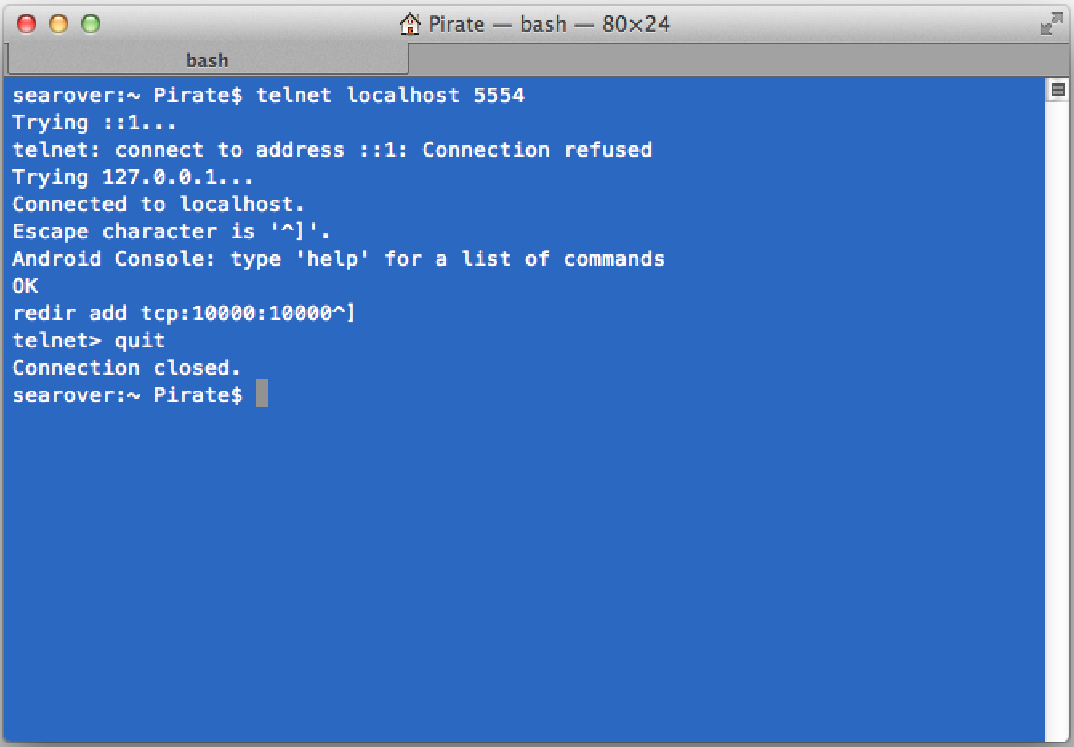
1)首先啟動模擬器(此時啟動需要加入-netfast 參數，每個參數的含義自行查詢)
emulator -avd android4 -verbose -netfast



```
id4.avd/cache.img
emulator: Kernel parameters: qemu.gles=0 qemu=1 console=ttyS0 android.qemud=ttyS1
android.checkjni=1 ndns=1
emulator: Trace file name is not set

emulator: autoconfig: -scale 1
2011-12-25 02:53:33.297 emulator-arm[67673:1107] Warning once: This application, o
r a library it uses, is using NSQuickDrawView, which has been deprecated. Apps sho
uld cease use of QuickDraw and move to Quartz.
emulator: WARNING: Unable to create sensors port: Connection refused
emulator: control console listening on port 5554, ADB on port 5555
emulator: sent '0012host:emulator:5555' to ADB server
emulator: ping program: /Volumes/Software/Install/androidsdk//tools/ddms
emulator: ping command: /Volumes/Software/Install/androidsdk//tools/ddms ping emul
ator 16.0
goldfish_fb_get_pixel_format:167: display surface,pixel format:
  bits/pixel:  16
  bytes/pixel: 2
  depth:       16
  red:         bits=5 mask=0xf800 shift=11 max=0x1f
  green:       bits=6 mask=0x7e0 shift=5 max=0x3f
  blue:        bits=5 mask=0x1f shift=0 max=0x1f
  alpha:       bits=0 mask=0x0 shift=0 max=0x0
```

從上圖像記行可以看到主控台監聽的通訊埠號為 5554，ADB 的通訊埠號為 5555。

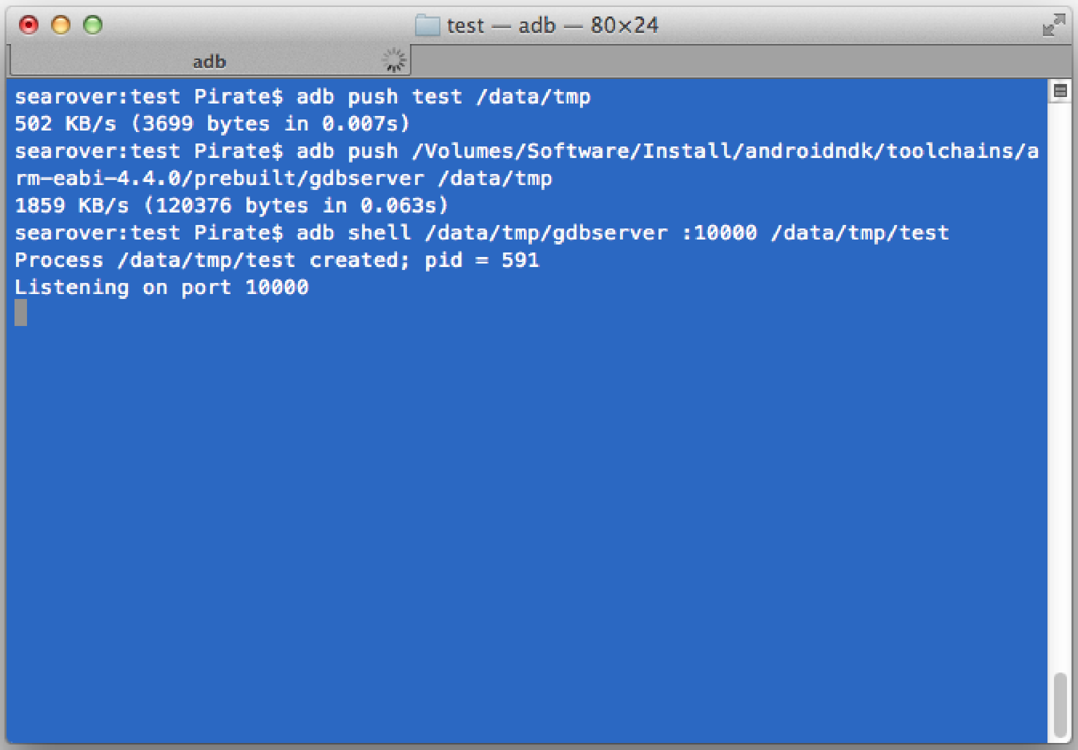
2)接下來進行網路重導，將存取主機 10000 通訊埠的要求重導至模擬器 10000 通訊埠
telnet localhost 5554
在 telnet 中匯入：redir add tcp:10000:10000
接下來按下 CTRL 和](右中括弧)鍵，回到 telnet 輔助說明符，匯入 quit 結束 telnet。



```
searover:~ Pirate$ telnet localhost 5554
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Android Console: type 'help' for a list of commands
OK
redir add tcp:10000:10000^]
telnet> quit
Connection closed.
searover:~ Pirate$
```

至此重導完成。
注意：通訊埠重導也可使用 adb forward tcp:10000: tcp:10000 指令完成

3)接下來在模擬器中開啟 gdbserver，用以監聽主機發出的除錯要求。我們除錯一個 test 程式，源碼詳見 test.zip
把要除錯的 test 程式和 NDK 中的 gdbserver 拷貝到模擬器的/data/tmp 目錄中，然后開啟對程式除錯的監聽
adb push test /data/tmp/
adb push androidndk/toolchains/arm-eabi-4.4.0/prebuilt/gdbserver /data/tmp
adb shell /data/tmp/gdbserver :10000 /data/tmp/test

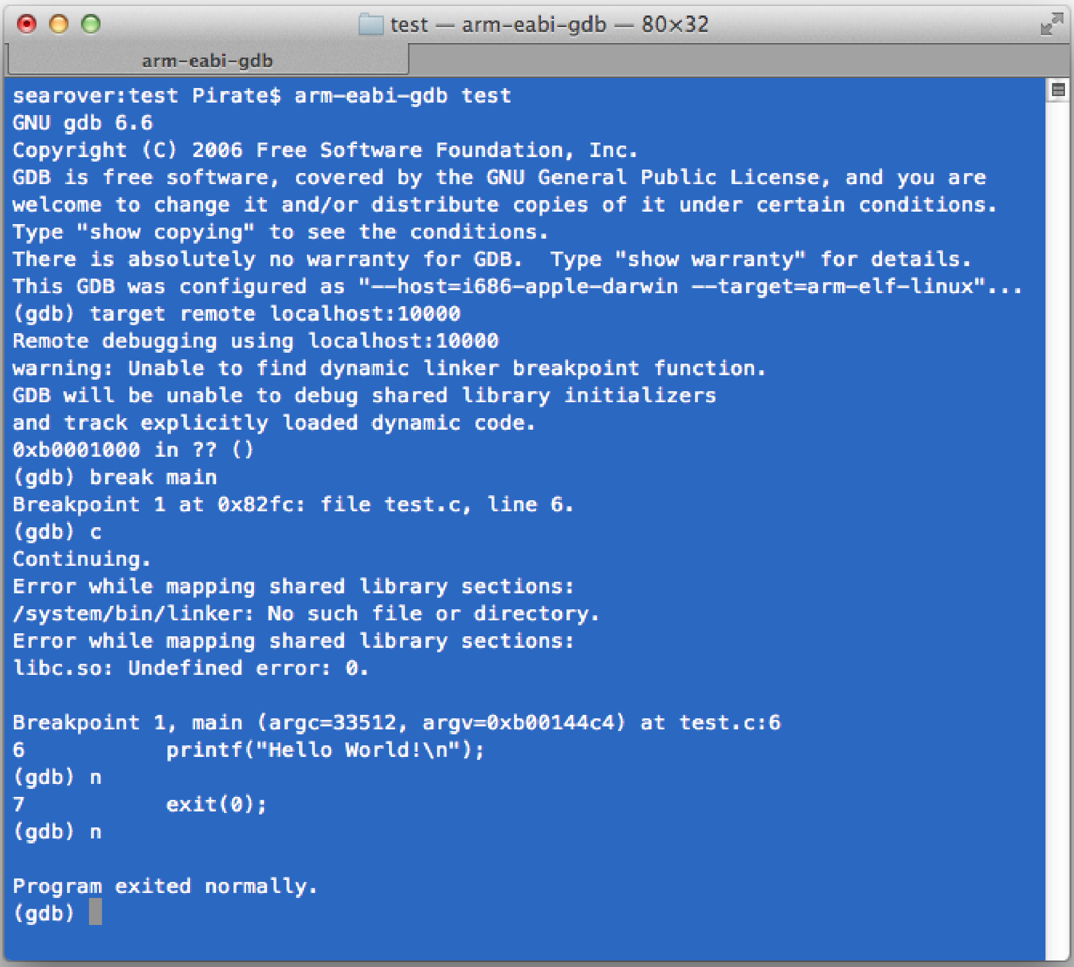


```
searover:test Pirate$ adb push test /data/tmp
502 KB/s (3699 bytes in 0.007s)
searover:test Pirate$ adb push /Volumes/Software/Install/androidndk/toolchains/a
rm-eabi-4.4.0/prebuilt/gdbserver /data/tmp
1859 KB/s (120376 bytes in 0.063s)
searover:test Pirate$ adb shell /data/tmp/gdbserver :10000 /data/tmp/test
Process /data/tmp/test created; pid = 591
Listening on port 10000
```

也可以在程式執行后，使用-attach 指定 PID 來進行除錯
adb shell gdbserver :10000 --attach <PID>
注意：此時要重新開啟一個新的終端機，然后執行下面的作業

4)當模擬器開啟 gdbserver 后，即可開始進行應用的除錯
arm-eabi-gdb test
注意：此處的 test，為主機上的可執行檔案

5)最后，在 gdb 中執行下述指令，就可以在(gdb)中除錯斷點，進行程式的除錯了。
target remote localhost:10000



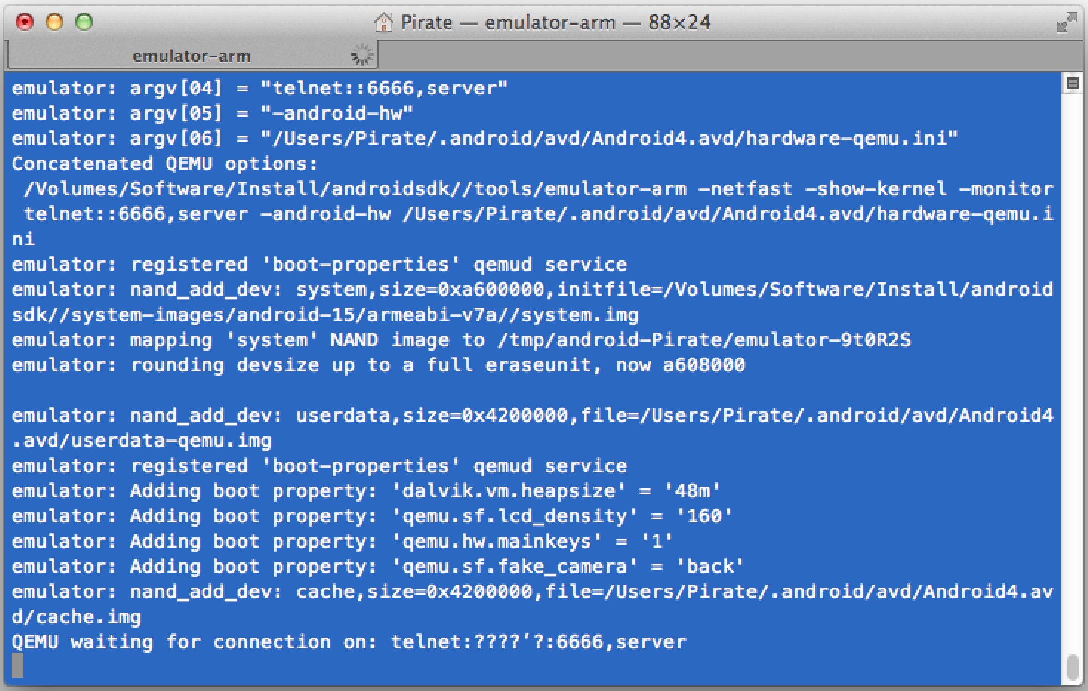
```
searover:test Pirate$ arm-eabi-gdb test
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=i686-apple-darwin --target=arm-elf-linux"...
(gdb) target remote localhost:10000
Remote debugging using localhost:10000
warning: Unable to find dynamic linker breakpoint function.
GDB will be unable to debug shared library initializers
and track explicitly loaded dynamic code.
0xb0001000 in ?? ()
(gdb) break main
Breakpoint 1 at 0x82fc: file test.c, line 6.
(gdb) c
Continuing.
Error while mapping shared library sections:
/system/bin/linker: No such file or directory.
Error while mapping shared library sections:
libc.so: Undefined error: 0.

Breakpoint 1, main (argc=33512, argv=0xb00144c4) at test.c:6
6            printf("Hello World!\n");
(gdb) n
7            exit(0);
(gdb) n

Program exited normally.
(gdb) 
```

(gdb) break main \\在 main 函數中除錯斷點
(gdb) c \\設定斷點后，繼續執行
(gdb) n \\執行下一行
GDB 指令的使用可自行尋找相關資料

附，可在 gdb 中可以設定動態鏈結程式庫的搜索路徑
set solib-search-path
out/target/product/generic/symbols/system/lib:out/target/product/generic/symbols/sys-tem/bin

2.除錯核心

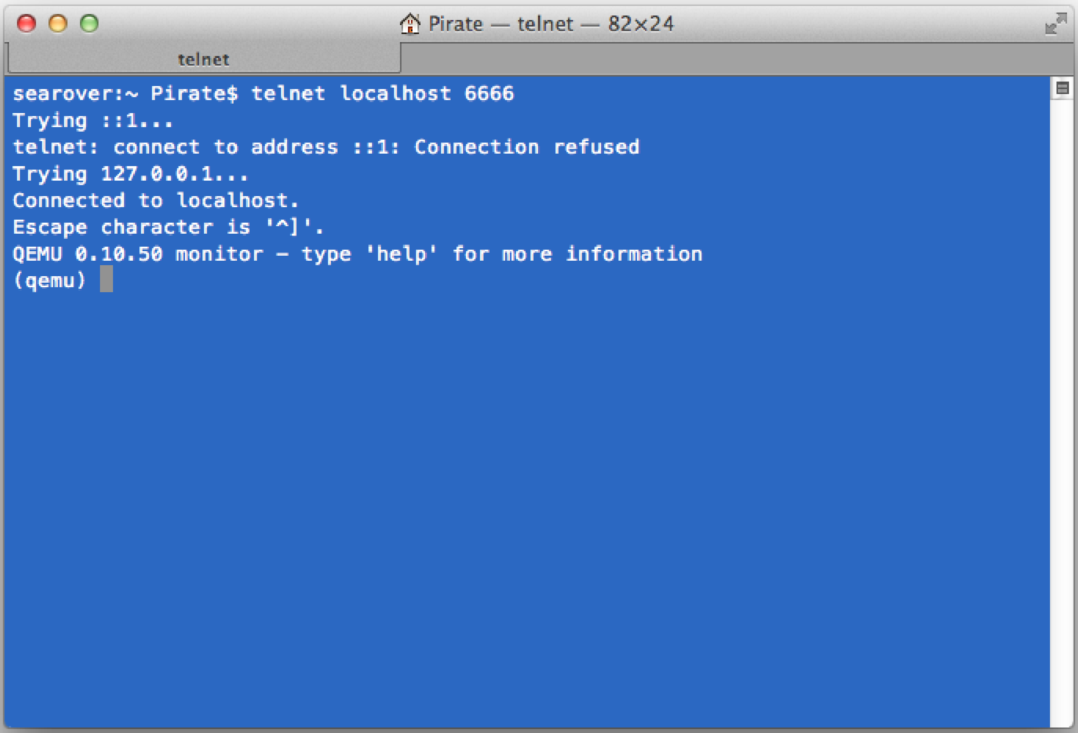
1)啟動模擬器，在除錯核心時一定要指定-kernel 參數，以除錯自己編譯的核心
emulator -verbose -show-kernel -netfast -kernel
/Volumes/Software/Android/kernel/goldfish/arch/arm/boot/zImage -avd android4 -qemu
-monitor telnet::6666,server



```
emulator: argv[04] = "telnet::6666,server"
emulator: argv[05] = "-android-hw"
emulator: argv[06] = "/Users/Pirate/.android/avd/Android4.avd/hardware-qemu.ini"
Concatenated QEMU options:
 /Volumes/Software/Install/androidsdk//tools/emulator-arm -netfast -show-kernel -monitor
 telnet::6666,server -android-hw /Users/Pirate/.android/avd/Android4.avd/hardware-qemu.i
ni
emulator: registered 'boot-properties' qemud service
emulator: nand_add_dev: system,size=0xa600000,initfile=/Volumes/Software/Install/android
sdk//system-images/android-15/armeabi-v7a//system.img
emulator: mapping 'system' NAND image to /tmp/android-Pirate/emulator-9t0R2S
emulator: rounding devsize up to a full eraseunit, now a608000

emulator: nand_add_dev: userdata,size=0x4200000,file=/Users/Pirate/.android/avd/Android4
.avd/userdata-qemu.img
emulator: registered 'boot-properties' qemud service
emulator: Adding boot property: 'dalvik.vm.heapsize' = '48m'
emulator: Adding boot property: 'qemu.sf.lcd_density' = '160'
emulator: Adding boot property: 'qemu.hw.mainkeys' = '1'
emulator: Adding boot property: 'qemu.sf.fake_camera' = 'back'
emulator: nand_add_dev: cache,size=0x4200000,file=/Users/Pirate/.android/avd/Android4.av
d/cache.img
QEMU waiting for connection on: telnet:????'?:6666,server
```

看到"QEMU waiting for connection …"資訊，代表核心除錯的監聽成功。
附：也可以通過以下指令直接開啟 gdbserver 的監聽
emulator -verbose -show-kernel -netfast -kernel /Volumes/Software/Android/kernel/goldfish/arch/arm/boot/zImage -avd android4 -qemu -gdb tcp::1234,ipv4

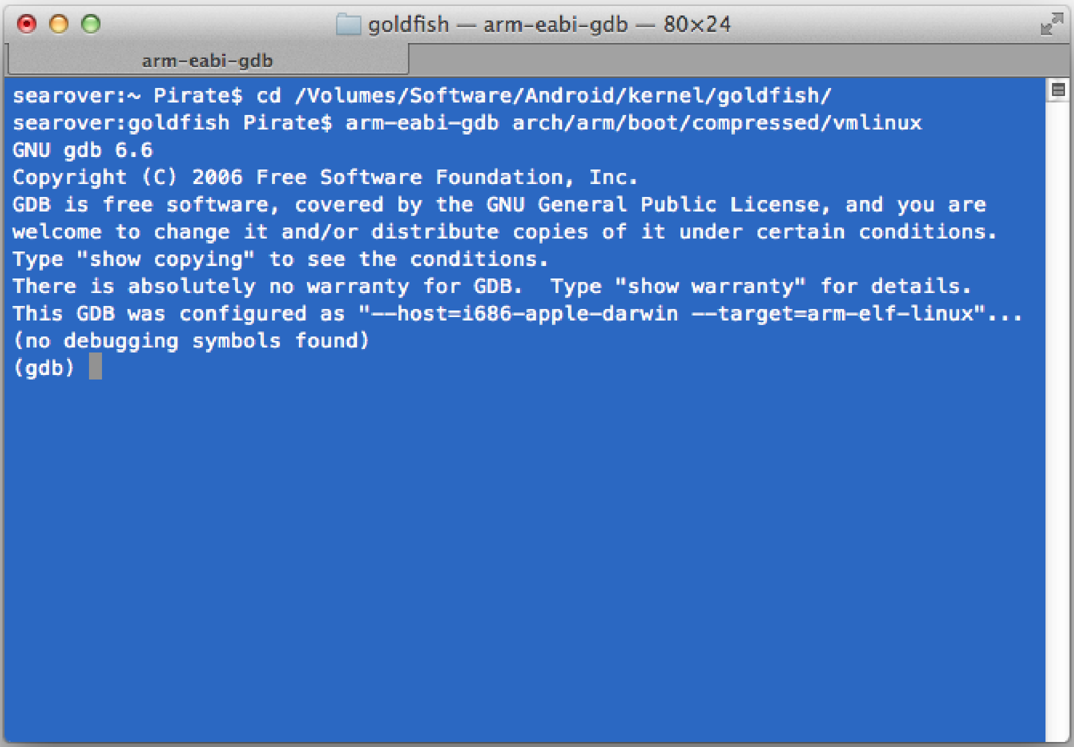
2)使用 telnet 連線模擬器，開始進行核心除錯
telnet localhost 6666



```
searover:~ Pirate$ telnet localhost 6666
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
QEMU 0.10.50 monitor - type 'help' for more information
(qemu) 
```

注意：此時需要啟動 gdbserver 服務，以監聽除錯要求
(qemu)gdbserver

3)接下來開啟一個新的終端機，進入 goldfish 目錄
cd goldfish/
arm-eabi-gdb arch/arm/boot/compressed/vmlinux
goldfish 下的 vmlinux 檔案鏈結到 arch/arm/boot/compressed/vmlinux，所以也可以直接執行 arm-eabi-gdb vmlinux



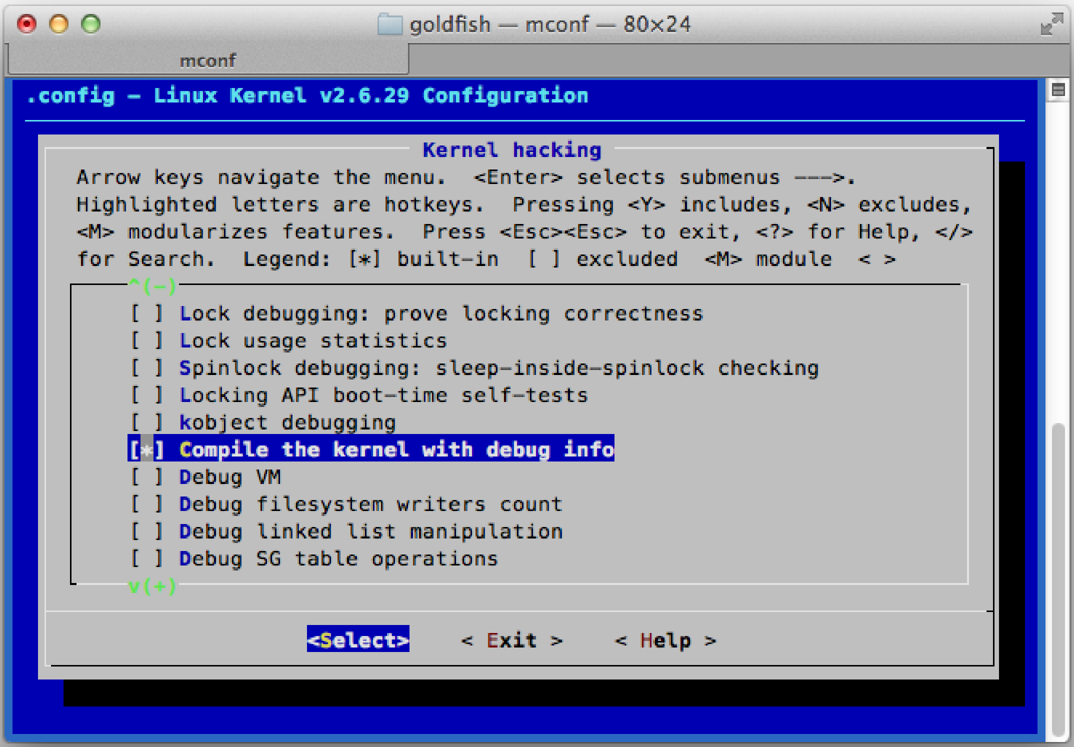
```
searover:~ Pirate$ cd /Volumes/Software/Android/kernel/goldfish/
searover:goldfish Pirate$ arm-eabi-gdb arch/arm/boot/compressed/vmlinux
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=i686-apple-darwin --target=arm-elf-linux"...
(no debugging symbols found)
(gdb) 
```

注意上面的輔助說明資訊"(no debugging symbols found)"，說明編譯核心時沒有除錯資訊(沒有開啟 CONFIG_DEBUG_INFO 選項)。下面進行核心配置開啟 CONFIG_DEBUG_INFO

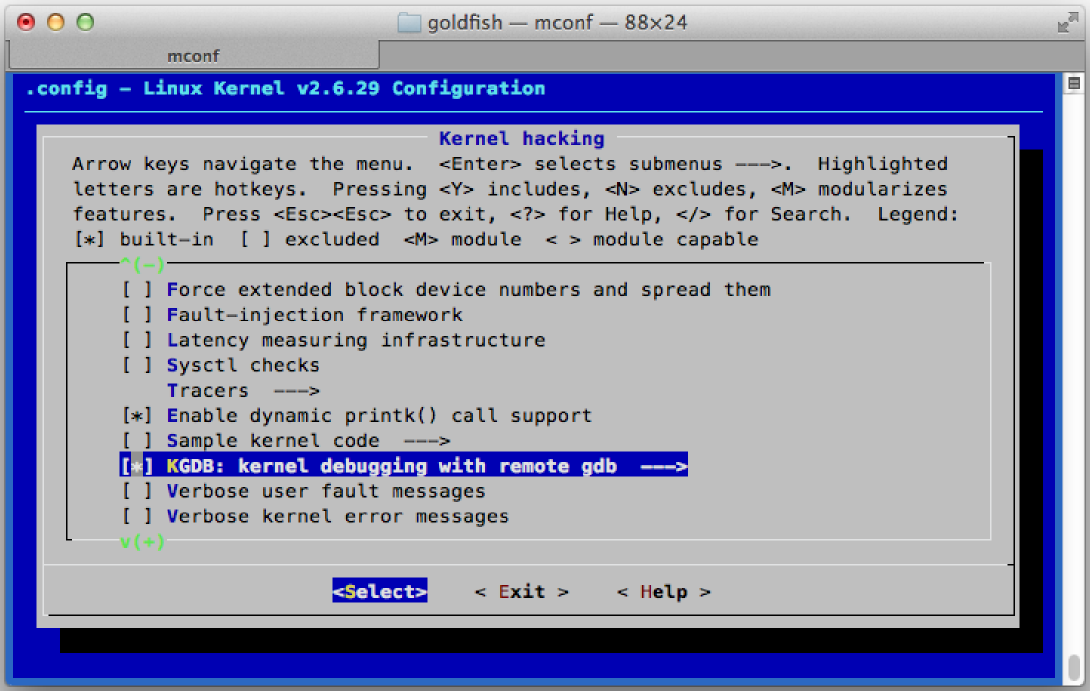
make ARCH=arm CROSS_COMPILE=arm-eabi- menuconfig

進入核心配置介面，勾選下列選項

```
[*] Kernel hacking
    [*] Compile the kernel with debug info
    [*] KGDB: kernel debugging with remote gdb --->
    [*] Enable dynamic printk() call support
```

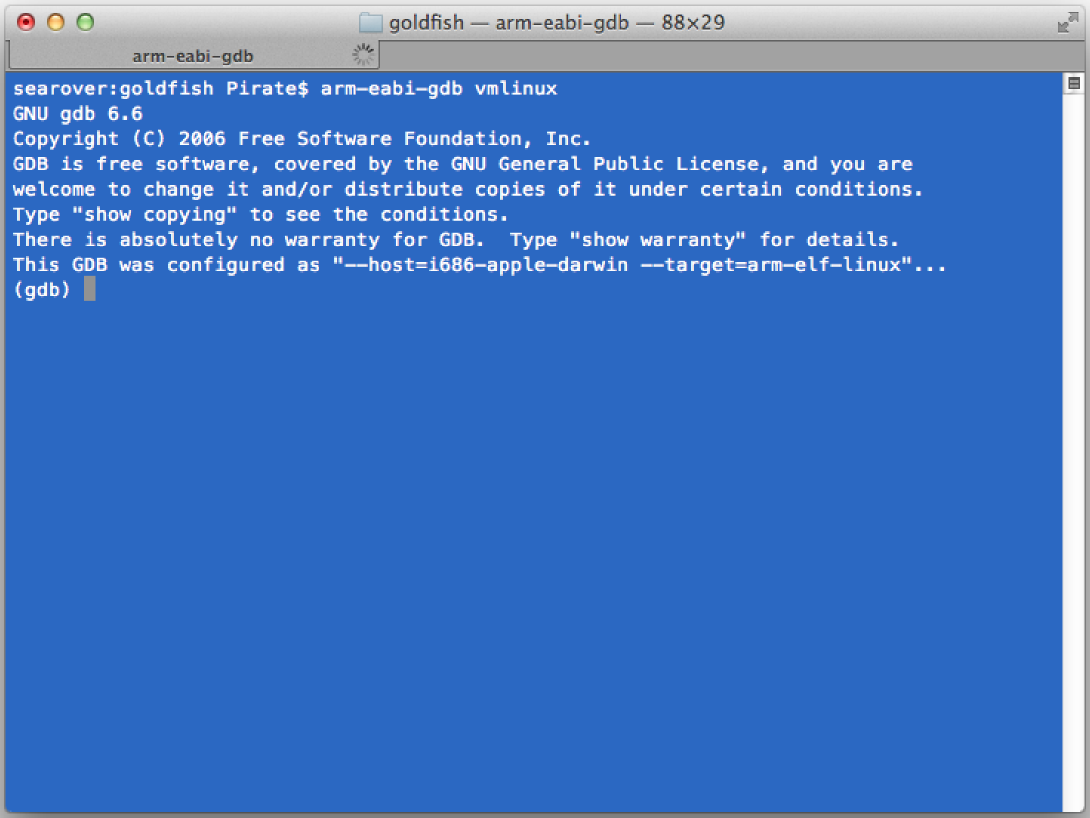


```
goldfish — mconf — 80×24
mconf
.config - Linux Kernel v2.6.29 Configuration

                          Kernel hacking
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >
      ^(-)
      [ ] Lock debugging: prove locking correctness
      [ ] Lock usage statistics
      [ ] Spinlock debugging: sleep-inside-spinlock checking
      [ ] Locking API boot-time self-tests
      [ ] kobject debugging
      [*] Compile the kernel with debug info
      [ ] Debug VM
      [ ] Debug filesystem writers count
      [ ] Debug linked list manipulation
      [ ] Debug SG table operations
      v(+)

                 <Select>    < Exit >    < Help >
```



```
.config - Linux Kernel v2.6.29 Configuration
                              Kernel hacking
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
  letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend:
  [*] built-in  [ ] excluded  <M> module  < > module capable
      ^(-)
      [ ] Force extended block device numbers and spread them
      [ ] Fault-injection framework
      [ ] Latency measuring infrastructure
      [ ] Sysctl checks
          Tracers  --->
      [*] Enable dynamic printk() call support
      [ ] Sample kernel code  --->
      [*] KGDB: kernel debugging with remote gdb  --->
      [ ] Verbose user fault messages
      [ ] Verbose kernel error messages
      v(+)

                   <Select>    < Exit >    < Help >
```

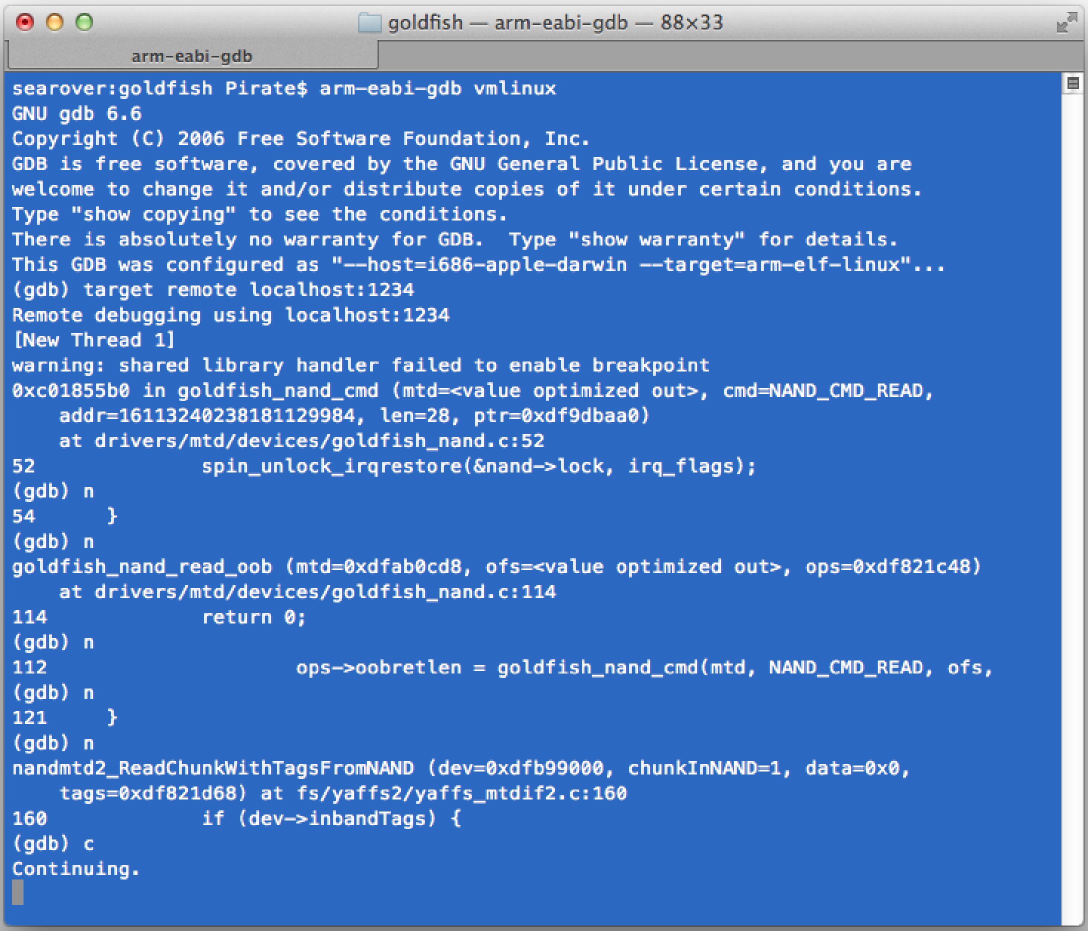
輔助說明：在核心配置介面可以按"/"鍵，尋找"CONFIG_DEBUG_INFO"所在位置
配置完成后，重新編譯核心，再次執行上述步驟



```
searover:goldfish Pirate$ arm-eabi-gdb vmlinux
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=i686-apple-darwin --target=arm-elf-linux"...
(gdb) 
```

看到上述資訊后，即可使用 GDB 進行核心的除錯了。

4)執行 target remote 開始除錯核心，預設通訊埠號為 1234
target remote localhost:1234



```
searover:goldfish Pirate$ arm-eabi-gdb vmlinux
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=i686-apple-darwin --target=arm-elf-linux"...
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
[New Thread 1]
warning: shared library handler failed to enable breakpoint
0xc01855b0 in goldfish_nand_cmd (mtd=<value optimized out>, cmd=NAND_CMD_READ,
    addr=16113240238181129984, len=28, ptr=0xdf9dbaa0)
    at drivers/mtd/devices/goldfish_nand.c:52
52              spin_unlock_irqrestore(&nand->lock, irq_flags);
(gdb) n
54      }
(gdb) n
goldfish_nand_read_oob (mtd=0xdfab0cd8, ofs=<value optimized out>, ops=0xdf821c48)
    at drivers/mtd/devices/goldfish_nand.c:114
114             return 0;
(gdb) n
112                     ops->oobretlen = goldfish_nand_cmd(mtd, NAND_CMD_READ, ofs,
(gdb) n
121     }
(gdb) n
nandmtd2_ReadChunkWithTagsFromNAND (dev=0xdfb99000, chunkInNAND=1, data=0x0,
    tags=0xdf821d68) at fs/yaffs2/yaffs_mtdif2.c:160
160             if (dev->inbandTags) {
(gdb) c
Continuing.
```

各位讀者可嘗試一下除錯 "Hello World!"模組，可以通過如下方式與本人交流：
Nickname : Alan
Emai : pirate310@gmail.com
QQ : 570222203

Linux 核心相關書籍建議：
Linux Kernel Development
Understanding the Linux Kernel
Linux Device Drivers
Essential Linux Device Drivers
Building Embedded Linux Systems

Android Kernel Development 第一部分結束，敬請期待第二部分… Tomorrow will be better!