



嵌入式軟體聯盟  
Embedded Software Consortium



# Building a Cross Debugger for ARM Linux

開發學生：林盈志、高毓廷、黃資閔

高效能計算實驗室

國立中正大學 資訊工程學系

教育部顧問室嵌入式軟體聯盟

<http://esw.cs.nthu.edu.tw>

# What can You Learn?

- ❖ During embedded-system development
  - Why do we need cross debugger
  - What is the role of cross debugger
- ❖ How to build a cross debugger
- ❖ How to do source-level cross debug with ARM emulator

# Equipment Requirement

## ❖ PC with

- Linux OS installed Ubuntu 16.04
- Native gcc compiler installed
- Internet access

# Software Requirement

❖ GNU debugger source code (**version 8.1 or newer**)

➤ <ftp://www.gnu.org/gnu/gdb/gdb-8.1.tar.gz>

# Outline

- ❖ Introduction to GDB
- ❖ Build a cross debugger
- ❖ Testing and Practice: command-line mode

# Outline

- ❖ Introduction to GDB
- ❖ Build a cross debugger
- ❖ Testing and Practice: command-line mode

# GNU Debugger (GDB)

- ❖ Allows you to see what is going on `inside' another program while it executes
  - Start your program, specifying anything that might affect its behavior
  - Make your program stop on specified conditions
  - Examine what has happened, when your program has stopped
  - Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another

# GDB: The GNU Project Debugger

## GDB: The GNU Project Debugger

[\[bugs\]](#) [\[GDB Maintainers\]](#) [\[contributing\]](#) [\[current git\]](#) [\[documentation\]](#) [\[download\]](#) [\[home\]](#) [\[irc\]](#) [\[links\]](#) [\[mailing lists\]](#) [\[news\]](#) [\[schedule\]](#) [\[song\]](#) [\[wiki\]](#)

## GDB: The GNU Project Debugger

### What is GDB?

GDB, the GNU Project debugger, allows you to see what is going on 'inside' another program while it executes -- or what another program was doing.

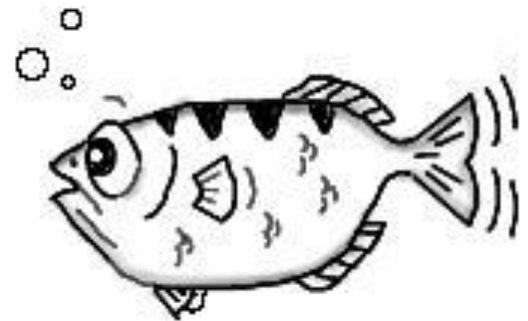
GDB can do four main kinds of things (plus other things in support of these) to help you catch bugs in the act:

- Start your program, specifying anything that might affect its behavior.
- Make your program stop on specified conditions.
- Examine what has happened, when your program has stopped.
- Change things in your program, so you can experiment with correcting the effects of one bug

Those programs might be executing on the same machine as GDB (native), on another machine (remote), or on a remote target, as well as on Mac OS X.

### What Languages does GDB Support?

GDB supports the following languages (in alphabetical order):

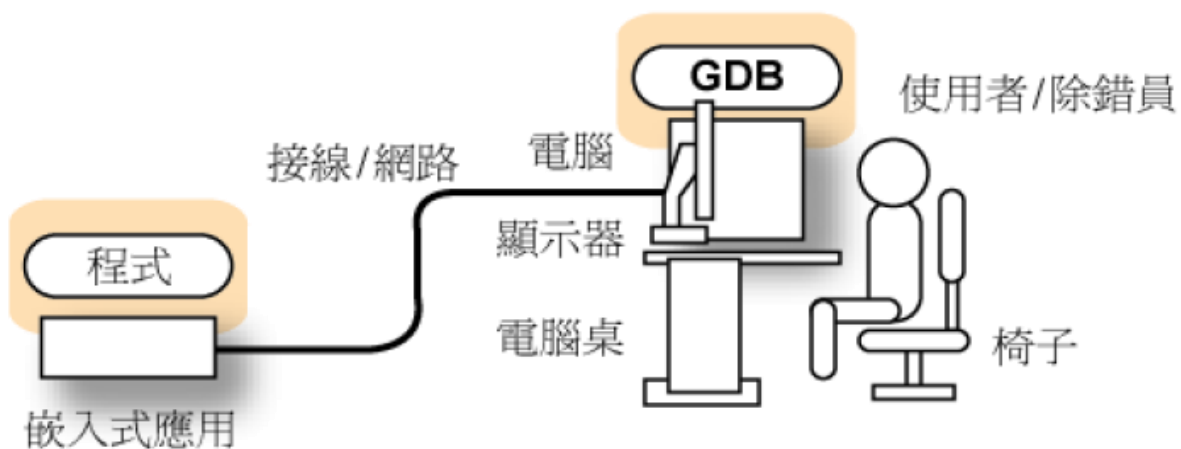




## 近端/本機型除錯



## 遠端/遙控型除錯



# Graphic User Interface to GDB

❖ It makes GDB easier to use

- DDD
- Insight
- Eclipse CDT
- gdbgui
- ...

**Graphic User Interface**

**GDB**

# DDD (Data Display Debugger)

The screenshot shows the DDD (Data Display Debugger) interface. The title bar indicates the file path: `DDD: /public/source/programming/ddd-3.2/ddd/cxxtest.C`. The menu bar includes File, Edit, View, Program, Commands, Status, Source, Data, and Help. The toolbar contains icons for Lookup, Find, Break, Watch, Print, Disasm, Plot, Hide, Rotate, Set, and Undisp.

The main display area shows a linked list structure. The first node is labeled `1: list (List *) 0x804df80`. It points to a node with `value = 85`, `self = 0x804df80`, and `next = 0x804df90`. This node points to another node with `value = 86`, `self = 0x804df90`, and `next = 0x804df90`. The list is terminated by a null pointer.

The command window shows the following commands and their results:

```
list->next           = new List(a_global + start++);
list->next->next       = new List(a_global + start++);
list->next->next->next  = list;
```

The status bar shows the current command: `(void) list; // Display this`. The command window also displays the memory address `(List *) 0x804df80`.

A "DDD Tip of the Day #5" dialog box is open, displaying a message: "If you made a mistake, try **Edit→Undo**. This will undo the most recent debugger command and redisplay the previous program state." The dialog has buttons for Close, Prev Tip, and Next Tip.

The bottom status bar shows the command: `(gdb) graph display *(list->next->next->self) dependent on 4` and the current state: `(gdb) list = (List *) 0x804df80`.



**DDD**  
v3.3  
DataDisplayDebugger

# Insight

The screenshot displays the Insight debugger interface with the following components:

- Source Window:** Shows the source code of `pi1.c` at line 79. The code is a C program with a `while` loop and nested `for` loops. Line 79 is highlighted: `shift(&mf[i - 1], &mf[i], temp - 2, temp * kf);`. The status bar indicates the program stopped at line 79 at address `0x804882b`.
- Stack:** Shows the call stack with frames `libc_start_main`, `main`, and `shift`.
- Local Variables:** Displays the current state of local variables:
  - `argc = (int) 2`
  - `argv = (char **) 0xbffffab4`
  - `i = (int) 20`
  - `endp = (char *) 0xbffffa68 "\210Ã*Ã,Ã,ÃK\001B\002"`
  - `args = (struct captured_main_args) {...}`
- Registers:** Shows the state of CPU registers:

| Register | Value      | Symbol |
|----------|------------|--------|
| eax      | 0x0        | st0    |
| ecx      | 0xffffffff | st1    |
| edx      | 0xa0       | st2    |
| ebx      | 0x50       | st3    |
| esp      | 0xbffffa28 | st4    |
| ebp      | 0xbffffa68 | st5    |
- Console Window:** Shows the GDB command line with the following text:

```
(gdb) b shift
Breakpoint 3 at 0x8048b17: file pi3.c, line 7.
(gdb) c
Continuing.
```

# gdbgui (browser-based)

gdbgui - gdb in a browser

127.0.0.1:5000

Load Binary examples/cpp/sin\_cpp.a

Enter source file path to view, or load binary then populate and select from dropdown

jump to line fetch disassembly reload file/hide disassembly /home/csmith/git/gdbgui/examples/cpp/sin.cpp:13

```
1 #include <math.h> /* sin */
2
3 #define PI 3.14159265
4
5 int main ()
6 {
7     double angle = 0, result = 0;
8
9     static const double RAD_TO_DEG = 3.14159265 / 180;
10    while (angle <= 360){
11        result = sin(angle * RAD_TO_DEG);
12
13        angle += 20;
14    }
15    return 0;
16 }
```

0x4005e6 push %rbp  
0x4005e7 mov %rsp,%rbp  
0x4005ea sub \$0x10,%rsp  
0x4005ee pxor %xmm0,%xmm0  
0x4005f2 movsd %xmm0,-0x10(%rbp)  
0x4005f7 pxor %xmm0,%xmm0  
0x4005fb movsd %xmm0,-0x8(%rbp)  
0x400640 jmp 0x400600 <main()+26>  
0x40060f movsd 0xc1(%rip),%xmm0 # 0x4006d8 <\_ZZ4mainE10RAD\_TO\_DEG>  
0x400617 mulsd -0x10(%rbp),%xmm0  
0x40061c callq 0x4004d0 <sin@plt>  
0x400621 movq %xmm0,%rax  
0x400626 mov %rax,-0x8(%rbp)  
0x40062a movsd -0x10(%rbp),%xmm1  
0x40062f movsd 0xb1(%rip),%xmm0 # 0x4006e8  
0x400637 addsd %xmm1,%xmm0  
0x40063b movsd %xmm0,-0x10(%rbp)  
0x400642 mov \$0x0,%eax  
0x400647 leaveq  
0x400648 retq

main sin.cpp:13 0x400642

local variables

RAD\_TO\_DEG: 0.017453292500000002 const double  
angle: 380 double  
result: -7.1795860596832236e-09 double

expressions

result

result: -7.1795860596832236e-09 double

Tree

memory

0x400642 0x400661 8

| address  | hex                     | char     |
|----------|-------------------------|----------|
| 0x400642 | b8 00 00 00 00 c9 c3 0f | .....    |
| 0x40064a | 1f 80 00 00 00 00 41 57 | .....AW  |
| 0x400652 | 41 56 41 89 ff 41 55 41 | AVA..AUA |

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) enter gdb command. To interrupt inferior, send SIGINT.

# Outline

- ❖ Introduction to GDB
- ❖ Build a cross debugger
- ❖ Testing and Practice: command-line mode

# Prepare to Build Cross Debugger

❖ Cross compiler, binutils are ready

❖ Download gdb source package

➤ `ftp://www.gnu.org/gnu/gdb/gdb-8.1.tar.gz`

❖ Uncompress source package

```
$ tar -zxvf gdb-8.1.tar.gz
```

# GDB Source Package (1)-1

- ❖ gdb-8.1/configure (and supporting files)
  - script for configuring GDB and all its supporting libraries
- ❖ gdb-8.1/gdb
  - the source specific to GDB itself
- ❖ gdb-8.1/bfd
  - source for the [Binary File Descriptor](#) library
- ❖ gdb-8.1/include
  - GNU include files



# GDB Source Package (1)-2

## ❖ gdb-8.1/libiberty

- Source for the “**liberty**” free software library
- It is a collection of subroutines used by various GNU programs.
- Current members include:
  - **getopt** -- get options from command line
  - **obstack** -- stacks of arbitrarily-sized objects
  - **strerror** -- error message strings corresponding to errno
  - **strtol** -- string-to-long conversion
  - **strtoul** -- string-to-unsigned-long conversion

# GDB Source Package (2)

## ❖ gdb-8.1/opcodes

- source for the library of opcode tables and disassemblers

## ❖ gdb-8.1/readline

- source for the GNU command-line interface

## ❖ gdb-8.1/sim

- Source for various simulators
  - ARM
  - MIPS
  - PowerPC
  - ...

# Build Cross Debugger (1)

- ❖ Build cross binutils (`--prefix=/MY_DIR`)
- ❖ Add `/MY_DIR/bin` to `PATH`
- ❖ Build cross compiler (`--prefix=/MY_DIR`)

前一個實驗

```
$ cd myWORK
```

```
$ mkdir build_gdb
```

## ❖ Configure GDB

```
$ ../gdb-8.1/configure --prefix=/MY_DIR \  
    --target=arm-linux-gnueabihf \  
    --enable-tui=yes
```

❖ make

❖ make install

**/home/pschen/WORK/crossgcc2**

# Build Cross Debugger (2)

- ❖ `--enable-tui`
  - enable full-screen terminal user interface (TUI)
- ❖ `--enable-gdbtk`
  - enable gdbtk graphical user interface (GUI)
- ❖ `--enable-profiling`
  - enable profiling of GDB
- ❖ `--enable-sim`
  - Link gdb with simulator
- ❖ Check GDB document for others

# Outline

- ❖ Introduction to GDB
- ❖ Build a cross debugger
- ❖ Testing and Practice: command-line mode

# test.c

```
#include <math.h>
#include <stdio.h>

int main(void)
{
    int a;
    double b;

    a = 10;
    b = 20 + cos((double) a);
    printf("%f\n", b);

    return 0;
}
```

# Practice on Source-Level Debug (Command-line Mode)

## ❖ Use cross compiler to compile test.c

- "-g" option: add **debug information** which makes possible source-level debug

```
$ /home/pschen/WORK/crossgcc2/bin/arm-linux-gnueabihf-  
gcc -static -g test.c -o test.exe
```

# GDB Command-Line Mode (1)

**\$ arm-linux-gnueabihf-gdb**

```
pschen@SmallTurtleLinux1: ~/WORK/crossgcc2/bin
pschen@SmallTurtleLinux1:~/WORK/crossgcc2/bin$ ./arm-linux-gnueabihf-gdb
GNU gdb (GDB) 8.1
Copyright (C) 2018 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-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) █
```



# GDB Command-Line Mode (2)

\$ arm-linux-gnueabihf-gdb -tui

```
pschen@SmallTurtleLinux1: ~/RaspberryPi3/test
test.c
5      {
6          int a;
7          double b;
8
9          a = 10;
10         b = 20 + cos((double) a);
11
12         return 0;
13     }
14
15
16
17
18
19
exec No process In:
---Type <return> to continue, or q <return> to quit---
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"...
Reading symbols from ./a.out...done.
(gdb)
```

# GDB Command-Line Mode (3)

❖ (gdb) file test.exe

Reading symbols from /root/test.exe...done.

❖ (gdb) break 10

Insert a breakpoint at line 10

Breakpoint 1 at 0x836e: file test.c, line 10.

❖ (gdb) r

run

❖ (gdb) n

next

❖ (gdb) s

step (step in)

❖ (gdb) info register

List all register contents

# GDB Command-Line Mode (4)

❖ (gdb) quit => exit GDB

❖ (gdb) x/10 0x0

- dump memory content from 0x0
- repeat count = 10

|       |            |            |            |            |
|-------|------------|------------|------------|------------|
| 0x0:  | 0xe59ff838 | 0xea00089b | 0xe59ff838 | 0xe59ff838 |
| 0x10: | 0xe59ff838 | 0xe59ff838 | 0xe59ff838 | 0xe59ff838 |
| 0x20: | 0x00000000 | 0x00000000 |            |            |

# Try It !

## ❖ Command

- n 單步執行
- s 單步執行 (step in)
- info register 觀看暫存器的值
- help 說明
- quit 結束

# Remote Debugging Using QEMU

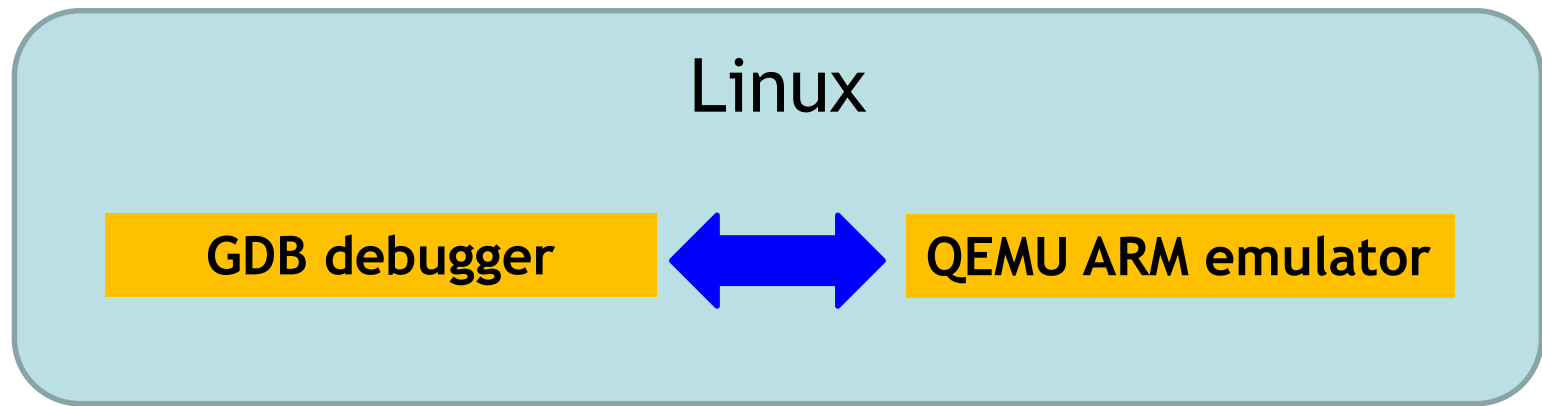


GDB debugger



Target program

# Remote Debugging Using QEMU



# 安裝 QEMU

## ❖ 安裝 QEMU user mode emulator

```
$ apt-get install qemu-user
```

## ❖ 安裝成功，在 /usr/bin 目錄下有執行檔 qemu-arm

## ❖ 檢視相關使用參數

```
$ qemu-arm -help
```

# qemu-arm (1)

pschen@LAPTOP-H2KVDJV3: ~

```
pschen@LAPTOP-H2KVDJV3:~$ qemu-arm -help
usage: qemu-arm [options] program [arguments...]
Linux CPU emulator (compiled for arm emulation)

Options and associated environment variables:
```

| Argument             | Env-variable      | Description   |
|----------------------|-------------------|---|
| -h                   |                   | print this help   |
| -help                |                   |   |
| -g port              | QEMU_GDB          | wait gdb connection to 'port'   |
| -L path              | QEMU_LD_PREFIX    | set the elf interpreter prefix to 'path'                              |
| -s size              | QEMU_STACK_SIZE   | set the stack size to 'size' bytes                                    |
| -cpu model           | QEMU_CPU          | select CPU (-cpu help for list)                                       |
| -E var=value         | QEMU_SET_ENV      | sets targets environment variable (see below)                         |
| -U var               | QEMU_UNSET_ENV    | unsets targets environment variable (see below)                       |
| -0 argv0             | QEMU_ARGV0        | forces target process argv[0] to be 'argv0'                           |
| -r uname             | QEMU_UNAME        | set qemu uname release string to 'uname'                              |
| -B address           | QEMU_GUEST_BASE   | set guest_base address to 'address'                                   |
| -R size              | QEMU_RESERVED_VA  | reserve 'size' bytes for guest virtual address space                  |
| -d item[,...]        | QEMU_LOG          | enable logging of specified items (use '-d help' for a list of items) |
| -dfilter range[,...] | QEMU_DFILT        | filter logging based on address range                                 |
| -D logfile           | QEMU_LOG_FILENAME | write logs to 'logfile' (default stderr)                              |
| -p pagesize          | QEMU_PAGESIZE     | set the host page size to 'pagesize'                                  |
| -singlestep          | QEMU_SINGLESTEP   | run in singlestep mode  |
| -strace              | QEMU_STRACE       | log system calls  |
| -seed                | QEMU_RAND_SEED    | Seed for pseudo-random number generator                               |
| -trace               | QEMU_TRACE        | [[enable=]<pattern>][,events=<file>][,file=<file>]                    |



## qemu-arm (2)

- -g port: 啟動qemu內部類似gdbserver的程式，等待遠端GDB的連線
- -L path: 設定cross toolchain執行檔的路徑 (set the elf interpreter prefix to 'path')

切換到test.exe所在的目錄，執行：

```
$ qemu-arm -g 12345 ./test.exe
```

# qemu-arm (3)

❖ 開啟另一個終端機視窗

```
$ arm-linux-gnueabihf-gdb ./test.exe
```

File Edit View Search Terminal Help

```
pschen@SmallTurtleLinux1:~/WORK$ /home/pschen/WORK/crossgcc2/bin/arm-linux-gnueabi-gdb ./test.exe
```

```
GNU gdb (GDB) 8.1
```

```
Copyright (C) 2018 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-pc-linux-gnu --target=arm-linux-gnueabi".
```

```
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"...
```

```
Reading symbols from ./test.exe...done.
```

```
(gdb) target remote localhost:12345
```

```
Remote debugging using localhost:12345
```

```
_start () at ../ports/sysdeps/arm/start.S:79
```

```
79      mov fp, #0
```

```
(gdb)
```

```
(gdb)
```

# qemu-arm (4)

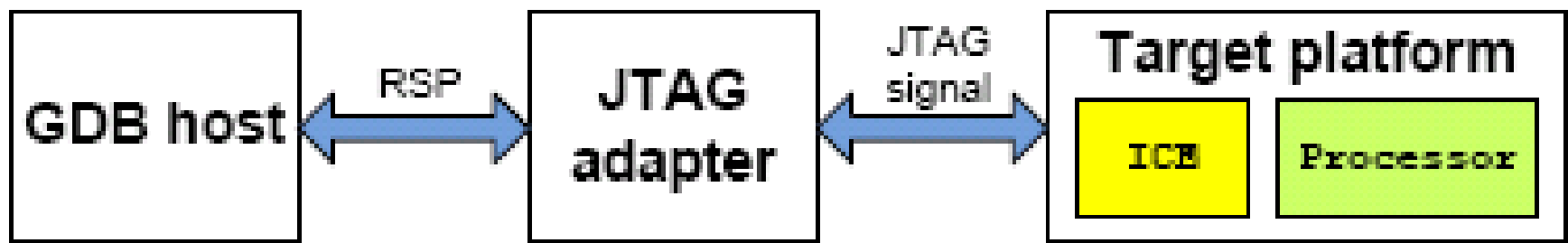
## ❖ 設定remote debugging

```
(gdb) target remote localhost:12345
```

```
(gdb) break main           // 設定中斷點
```

```
(gdb) c                    // 繼續執行程式
```

# GDB for ICE/Simulator



# Remote Serial Protocol

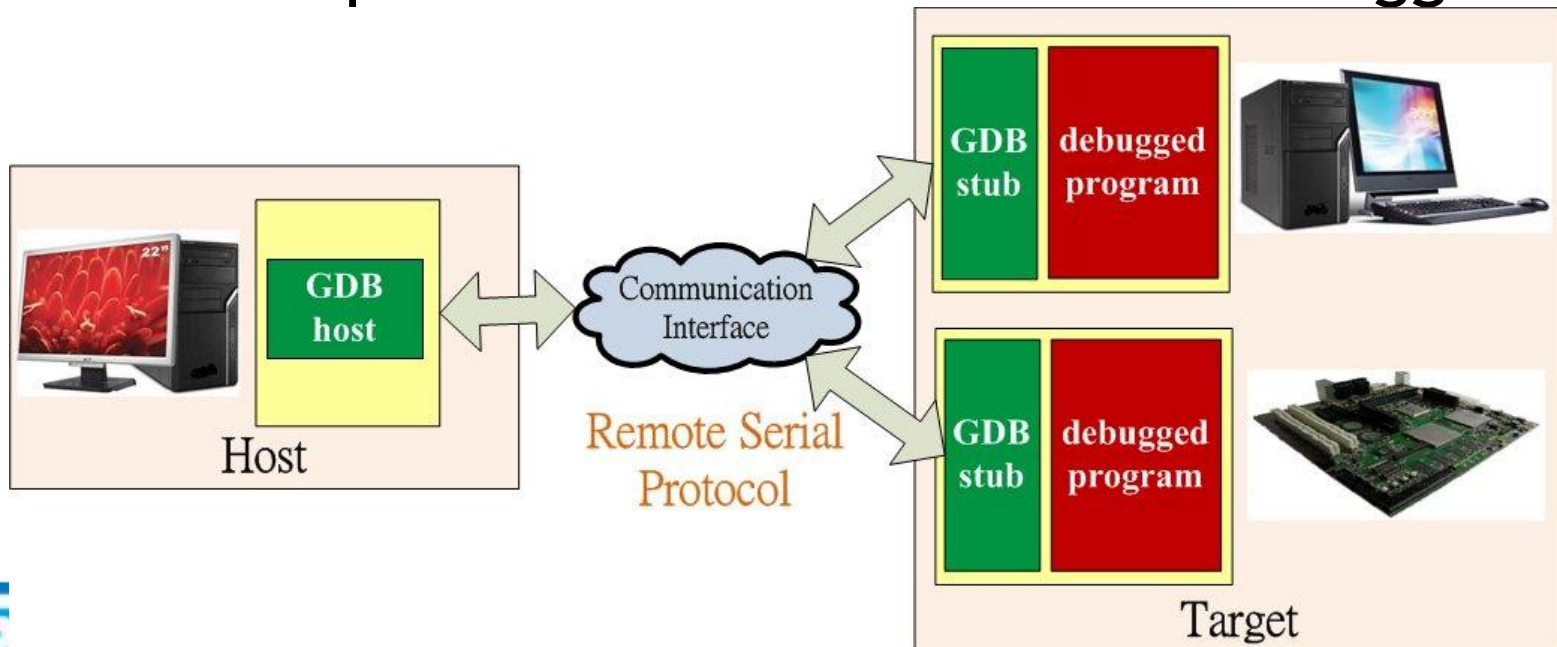
- ❖ All GDB commands and responses are sent by using the **remote serial protocol**
- ❖ ASCII message-based protocol
  - Packet format: **\$packet-data#checksum**
  - Example :

**Set register content**

|         |                          |
|---------|--------------------------|
| GDB:    | <b>\$P10=0040149c#b3</b> |
| Target: | <b>+ \$OK#9a</b>         |

# What is GDB stub

- ❖ An **agent** of the GDB host
  - Monitor and control the debugged program
  - Communicate with GDB host
  - Be compiled and linked with the debugged



# Implementation of the GDB stub

```
int main(void)
{
    set_debug_traps();
    breakpoint();
    int a, b;
    a = 10;
    b = 20 + a;
    return 0;
}
```

debugged\_program.c

## ❖ Set\_debug\_trap()

- Set **exceptionHandler** for each traps

## ❖ Breakpoint()

- Directly halt the execution of the program
- Wait the commands from GDB host



# Reference

## ❖ GNU GDB

- <http://sources.redhat.com/gdb>

## ❖ Insight

- <http://sources.redhat.com/insight>

## ❖ DDD

- <http://www.gnu.org/software/ddd/>

# Questions

- ❖ What is “GDB server”?
- ❖ What is “GDB stub”?

請在實驗報告裡回覆相關問題

# Backup