以 GDB 重新學習 C 語言程式設計

Jim Huang (黃敬群 /jserv)
from 0xlab
元智大學 / May 5, 2010

課程主軸

觀察程式的執行 (Process)

透過 GDB 進行以下動作：
- 動態攔截程式的執行
- 動態觀察與分析
- 完整顯示執行時期的資訊
- 適度調整程式的邏輯

...

交叉理解 GNU/Linux 系統的運作原理

state-of-the-art
（目前工藝水平）

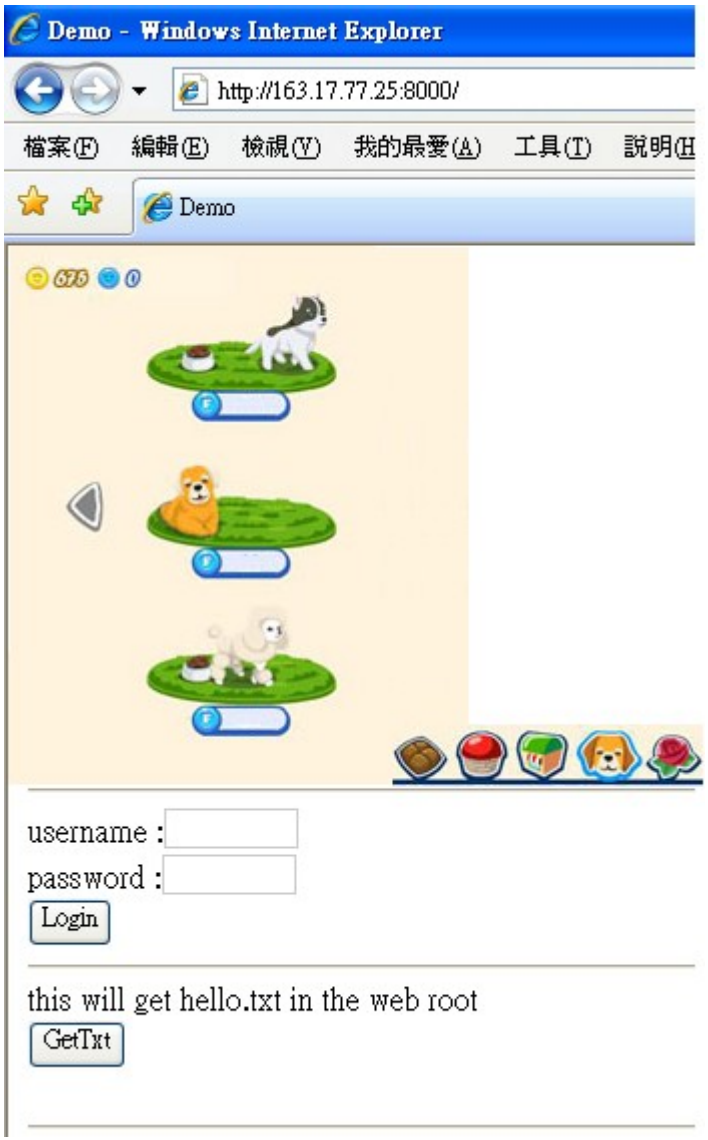
從除錯與分析的結果，思考：
- 引入新增的功能
- 分析其風險與衝擊
- 重新整理，符合預期需求
- 「技巧」

杜威博士：「作中學」

案例探討

背景故事：在修平技術學院資訊網路技術系兼課「Unix 程式設計」

案例探討： Ajax for Embedded



- 「山寨版」開心農場
- 從一個具體而微的 Embedded AJAX 系統出發，透過 GDB 去追蹤分析，進而作擴充
- 掌握 UNIX 系統程式開發的技巧

Rights to copy

http://0xlab.org/

contact@0xlab.org

Corrections, suggestions, contributions and translations are welcome!

Latest update:Mar 27, 2010

Show All Networks | View All Friends

Posted Items

2 posted items. See All

Twitter
7:50pm May 28

I'm a rapper dancer; are you?
2:29am May 17

Notes

3 of 198 notes. See All

The Future As Today, But More So
5:00pm Sep 20

Exploiting The Potential Of Blogs and Social Networks
3:00pm Sep 20

TokBox - A Useful Video-Conferencing Tool Or Something Sinister?
8:00am Sep 19

Groups

30 of 36 groups. See All

Creating Apps for Facebook (New Stanford Course) • The Campaign to register "Socialism" as a Political View on Facebook • The Demon Barbers Rock! • Teaching & Learning with Facebook • CETL Student Network • RLO-CETL - Reusable Learning Objects • Campaign to get a Bath network added on Facebook. • Everton FC- The people's club • Technorati Users • The (semi-)official Sidmouth FolkWeek group • Newbridge Primary School, Bath • Sidmouth Folk Festival 2007 Photos and Stuff • DBpedia • The Semantic Web - Benefits, Education & Outreach • JISC Digitisation Programme • UCISA • Wikimedia Commons • Facebook Applications • Lager vs Beer • The

WordPress

UK Web Focus
Reflections On The Web

243 posts
926 comments

Recent posts:

The Future As Today, But More So
Thu 20 Sep 2007 — 3 comments

My Background When I was young we didn't have a TV and it wasn't until I was 7 or so that my family caught up, and I discovered why my school friends were so excited about Doctor Who. And at that time we didn't have a telephone, so when my parents wanted to ring their friends, it i [...]

Exploiting The Potential Of Blogs and Social Networks
Thu 20 Sep 2007 — 2 comments

In November 2006 UKOLN ran a day's workshop on Exploiting The Potential Of Wikis which was held at Austin Court, Birmingham. The feedback for the event was very positive, with positive comments made not just about the content of the workshop but also the venue. This year, on 26th November 2007 [...]

TokBox - A Useful Video-Conferencing Tool Or Something Sinister?
Wed 19 Sep 2007 — 6 comments

The TokBox Video Chat Tool The TokBox instant video chat tool was reviewed by TechCrunch in August 2007. As with several of the Web 2.0 services I've mentioned on this blog, Tokbox is very easy to set up and use: simply register for a (free) account and, assuming you have a Webcam and micropho [...]

Recent comments:

AJ Cann on The Future As Today, But More So
Mon 24 Sep 2007

Nope, not mass market Phil. IM technology is on the cusp of mass market. Email used to be mass mar [...]

一系列「服務」的整合
前端與後端


Embedded Ajax 的目標：
用最少的資源，實做基本
的 Web framework 功能
facebook
Profile edit Friends Networks Inbox
home account privacy logout
Search
Applications edit
Photos
Groups
Events
Marketplace
Where I've Been
Video
Posted Items
To upload a picture so that your friends can find you, click here.
Edit My Profile
You are online now.
Notes
Tell everyone what's on your mind with Facebook notes. To write a new note, click here.
Brian Kelly
is using his bath.ac.uk as his main Facebook ID!
Updated just a moment ago edit
Profile
Networks: None
Birthday: August 25, 1956
Mini-Feed
Displaying 4 stories.
See All
July 7
Brian updated their Where I've Been map. 6:52pm
Brian added the Where I've Been application. 6:45pm
Brian added the Video application. 6:44pm
Welcome to Facebook! We're glad you're here. 6:42pm
Information
Contact Info [edit]
Email: b.kelly@ukoln.ac.uk
To fill out the rest of your profile, click here.
The Wall
No wall posts.
Wall-to-Wall | See All
Write something on your own Wall...
Post
Add Attachment
facebook
login sign up tour
Email:
Password:
Login
Forgot Password?
Already a Member? Login
facebook
Facebook is a social utility that connects you with the people around you.
Everyone can use Facebook — Sign Up
upload photos or publish notes • get the latest news from your friends • post videos on your profile • tag your friends • use privacy settings to control who sees your info • join a network to see people who live, study, or work around you
Find your friends
Facebook © 2007
about developers jobs advertisers polls terms privacy help

AJAX

- 依據維基百科的解釋

AJAX 全稱爲”Asynchronous JavaScript and XML”（非同步 JavaScript 和 XML），是一種設計互動式網頁應用的網頁開發技術

使用 XHTML+CSS 來表示訊息；

使用 JavaScript 操作 DOM (Document Object Model) 進行動態顯示及交互；

使用 XML 和 XSLT 進行數據交換及相關操作；

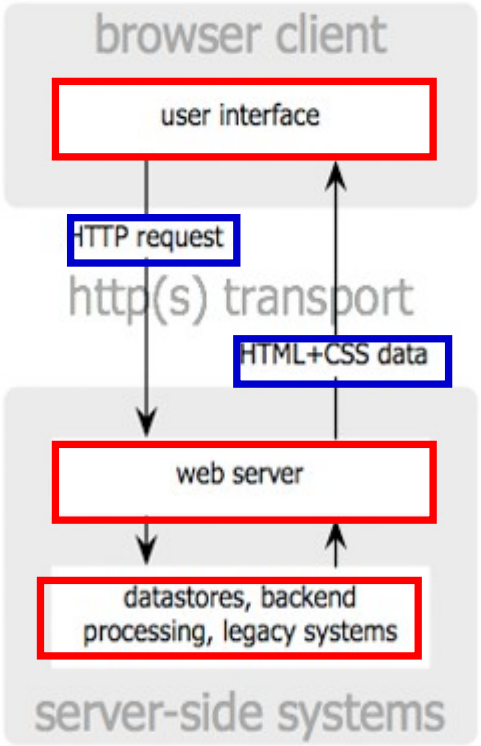
使用 XMLHttpRequest 對象與 Web 伺服器進行非同步數據交換；

使用 JavaScript 將所有的東西綁在一起。

使用 SOAP 以 XML 的格式來傳送方法名和方法參數。

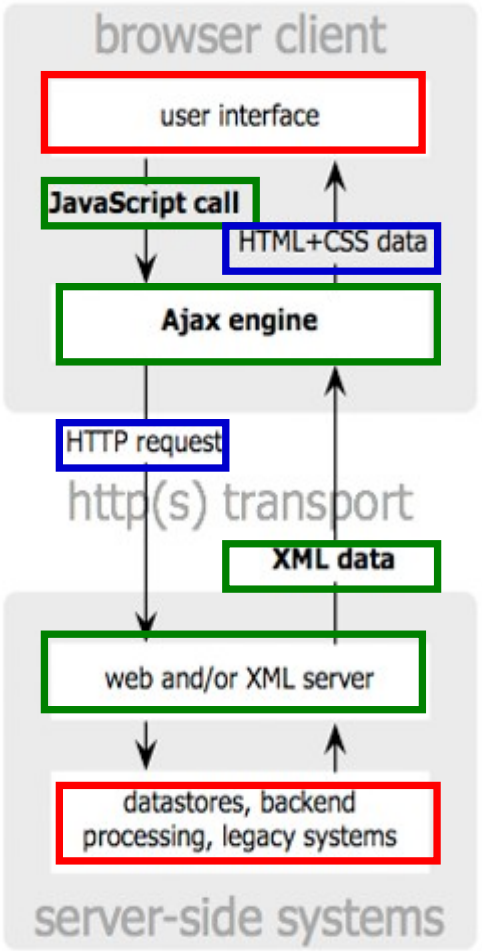
AJAX = Asynchronous JavaScript and XML

Synchronous



classic
web application model

Jesse James Garrett / adaptivepath.com



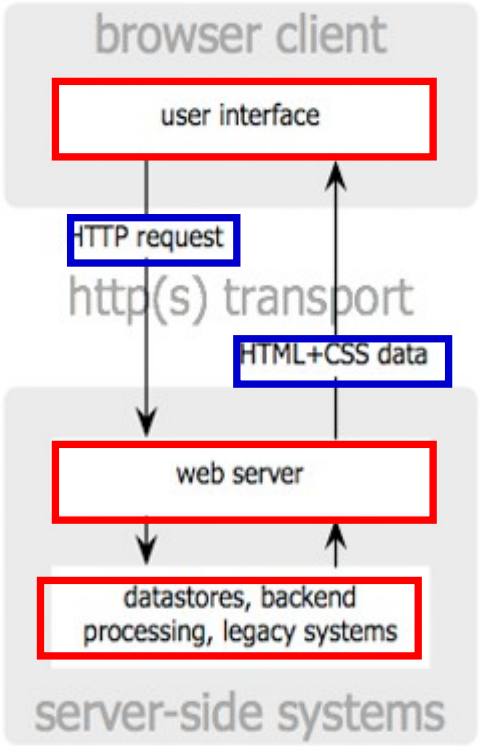
Ajax
web application model

Asynchronous

AJAX = Asynchronous JavaScript and XML

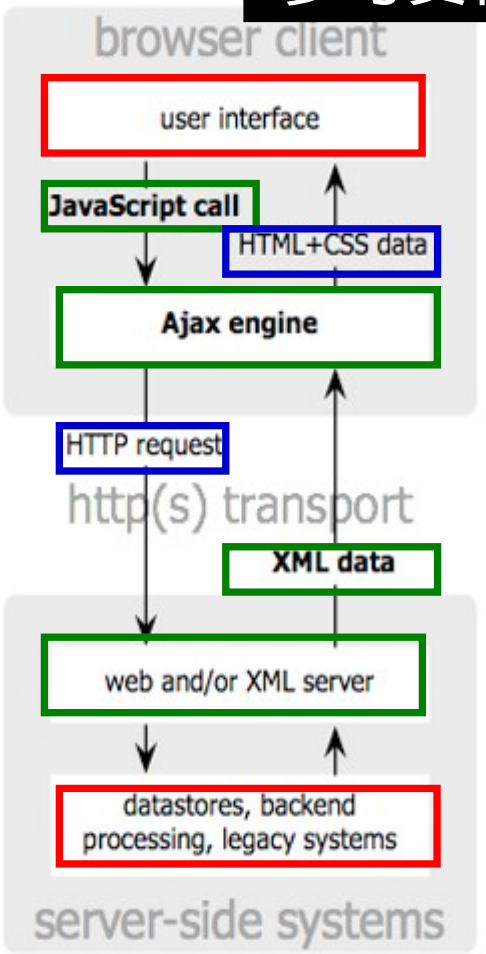
eServ 的前端 (web browser)
參考實做透過流行的 jQuery

Synchronous



classic
web application model

Jesse James Garrett / adaptivepath.com



Ajax
web application model

Asynchronous

以 **server-side**
為切入點
→ **eServ**

取得 eServ

- 實作環境： Ubuntu GNU/Linux 9.10
- 下載網頁

  http://code.google.com/p/eserv/downloads/list
- 檔名： eserv-preview.tar.bz2
- 準備動作：

```
# cd /tmp
# wget http://eserv.googlecode.com/files/eserv-preview.tar.bz2
# tar jxcvf eserv-preview.tar.bz2
# cd eserv
# make
```

為了課程打造一個中小型的
Embedded Ajax 系統 – eServ
用「解析」的角度去看待

GDB 基礎指令

command	說明
`run [args]`	開始執行
`start [args]`	開始執行（並自動在 main() break）
`Ctrl-C`（組合按鍵）	中斷程式

`list [LINENUM]`
`list [FUNCTION]`
`list [FILE:LINENUM]`
	列出程式碼（重複下 list 指令可接著列出下面十行）

`print [EXP]`	顯示 expression 的值
`break [LINENUM]` `break [FUNCTION]`	設一個 break point
`next [TIMES]`	執行到下一個 statement（不會進入 function）
`step [TIMES]`	執行到下一個 statement（會進入 function）
`until [LINENUM]`	執行到某行
`continue [TIMES]`	執行到被中斷為止
`finish`	執行到結束此 function（還可用 return 直接回傳並結束）
`info [SUBCOMMAND]`	顯示一些資訊（如 breakpoints： info break）
`help [SUBCOMMAND]`	說明（如 help list）

觀察

main.c

```
#include "libeserv/http.h"


int main()
{
    char buf[16];
    ex_init();
    while (scanf("%16s", buf) > 0) {
        if (strncmp("quit" , buf, 4) == 0)
        break;
        ex_sleep(200);
    }
    ex_uninit();
    return 0;
}
```

原則：善用 GDB 一類的工具，以協助追蹤系統的運作，而不要一味迷失在程式碼的茫茫大海中

唯一不需要額外追蹤的程式碼片段，稍候作實驗分析

觀察

gdb 操作

```
jserv@venux:/tmp/eserv$ gdb ./eserv
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 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 "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/eserv/eserv...done.
(gdb)
```

以 IA32/x86 架構 Linux 為本議程探討對象

本簡報圖例：

凡紅色框搭配粗體字者，表示使用者輸入

凡綠色框住部份，表示系統 / 程式輸出，需留意

觀察

```
(gdb) list
1   #include "libeserv/http.h"
2
3   int main()
4   {
5       char buf[16];
6       ex_init();
7       while (scanf("%16s", buf) > 0) {
8           if (strncmp("quit" , buf, 4) == 0)
9               break;
10          ex_sleep(200);
(gdb) break 8
Breakpoint 1 at 0x804b3b0: file main.c, line 8.
```

```
(gdb) run
Starting program: /tmp/eserv/eserv
[Thread debugging using libthread_db enabled]
[New Thread 0xb7fe6b70 (LWP 14341)]
...
[Thread 0xb77e5b70 (LWP 14400) exited]
```

設定停止執行的條件

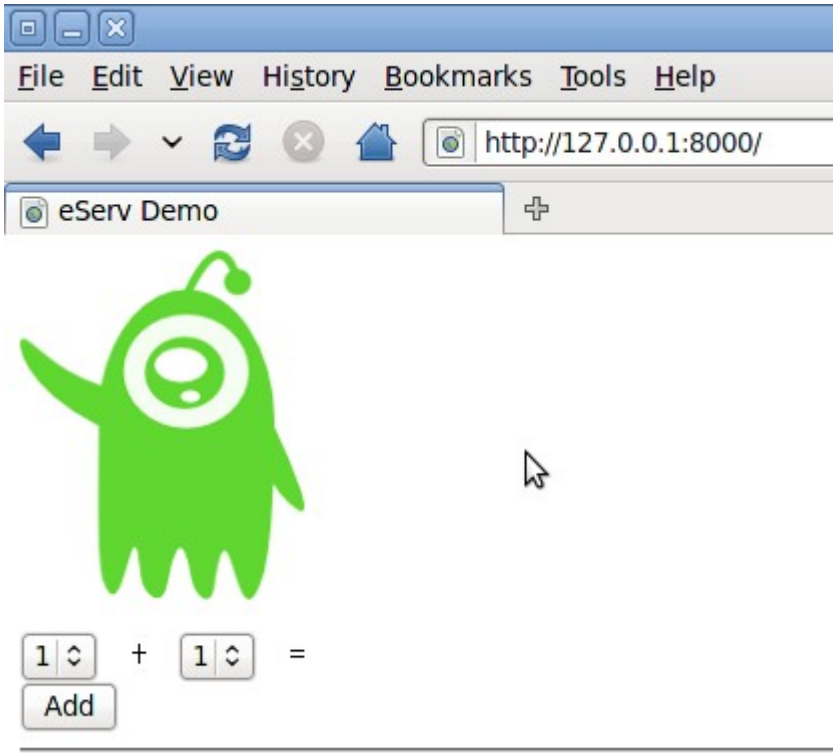
只要 stdin 沒有適當的輸入 (quit)，
main() 就持續等待

觀察

```
(gdb) run
Starting program: /tmp/eserv/eserv
[Thread debugging using libthread_db enabled]
[New Thread 0xb7fe6b70 (LWP 14544)]
 eServ is running...
[New Thread 0xb77e5b70 (LWP 14609)]
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 1507
Cache-Control: max-age=86400
ETag: 5e3.4baccc9b
Server: eServ/1.1
```

開啟網頁瀏覽器
網址： http://127.0.0.1:8000/

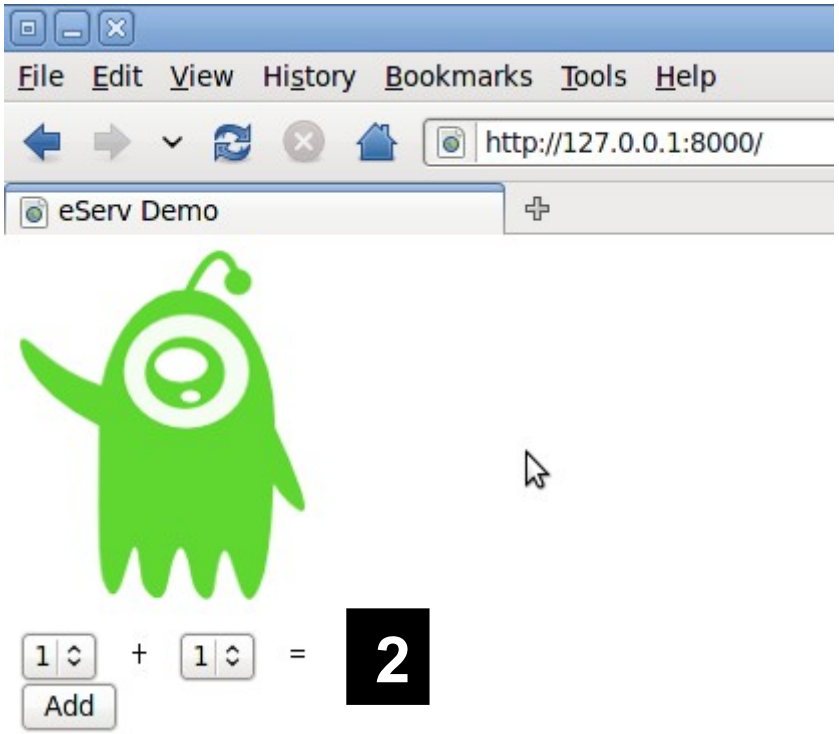
CFLAGS += -D_DEBUG



觀察

```
[New Thread 0xb7fe6b70 (LWP 15153)]
[New Thread 0xb77e5b70 (LWP 15159)]


--add.cgi--
rAdd:1
lAdd:1
[Thread 0xb77ddb70 (LWP 22930) exited]

^C
Program received signal SIGINT, Interrupt.
0x00afc422 in __kernel_vsyscall ()
(gdb) info threads
  2 Thread 0xb7fdeb70 (LWP 22917)  0x00afc422 in __kernel_vsyscall ()
* 1 Thread 0xb7fdf6c0 (LWP 22914)  0x00afc422 in __kernel_vsyscall ()
(gdb)
```

按下 Ctrl-C
把控制權搶回



按下網頁左側的 [Add]，
GDB 執行畫面隨之更新

觀察

目前在 Thread 1

```
(gdb) info threads
  2 Thread 0xb7fdeb70 (LWP 22917)  0x00afc422 in __kernel_vsyscall ()
* 1 Thread 0xb7fdf6c0 (LWP 22914)  0x00afc422 in __kernel_vsyscall ()
(gdb) thread 2
[Switching to thread 2 (Thread 0xb7fdeb70 (LWP 22917))]#0  0x00afc422 in
__kernel_vsyscall ()
(gdb) where
#0  0x00afc422 in __kernel_vsyscall ()
#1  0x00943e88 in accept () at ../sysdeps/unix/sysv/linux/i386/socket.S:97
#2  0x08049d1d in ex_http_start () at libeserv/http.c:83
#3  0x0093c80e in start_thread (arg=0xb7fdeb70) at pthread_create.c:300
#4  0x008787ce in clone () at ../sysdeps/unix/sysv/linux/i386/clone.S:130
(gdb)
```

切換到 Thread 2

GDB 自 eserv 搶回控制權的瞬間，
有兩個 Thread 正運作，
對應不同的 thread context

指令格式：
thread apply all 指令

```
(gdb) thread apply all bt

Thread 2 (Thread 0xb7fdeb70 (LWP 24537)):
#0  0x00545422 in __kernel_vsyscall ()
#1  0x0047de88 in accept () at ../sysdeps/unix/sysv/linux/i386/socket.S:97
#2  0x08049d1d in ex_http_start () at libeserv/http.c:83
#3  0x0047680e in start_thread (arg=0xb7fdeb70) at pthread_create.c:300
#4  0x001dc7ce in clone () at ../sysdeps/unix/sysv/linux/i386/clone.S:130

Thread 1 (Thread 0xb7fdf6c0 (LWP 24534)):
#0  0x00545422 in __kernel_vsyscall ()
...
#6  0x0015d4f8 in __scanf (format=0x804d3bd "%s") at scanf.c:35
#7  0x0804b3dd in main () at main.c:7
(gdb)
```

實驗：停止程式執行 (1/2)


```
jserv@venux:/tmp/eserv$ gdb ./eserv
GNU gdb (GDB) 7.1-ubuntu
...
(gdb) list
1    #include "libeserv/http.h"
2
3    int main()
4    {
5          char buf[16];
6          ex_init();
7          while (scanf("%16s", buf) > 0) {
8                if (strncmp("quit", buf, 4) == 0)
9                     break;
10               ex_sleep(200);
(gdb) b 7
Breakpoint 1 at 0x804afde: file main.c, line 7.
(gdb) r
```

原本程式會呼叫 **scanf()**，持續等待 **stdin**，而實驗目標則是忽略 **scanf()**，並進行符合預期的停止程式執行動作

將中斷點設定於迴圈發生點
如果 **buf** 內容為 **quit**，則可 (模擬) 正常結束執行

實驗：停止程式執行 (2/2)

```
(gdb) r
Starting program: /home/jserv/experimental/eserv/
[Thread debugging using libthread_db enabled]
[New Thread 0xb7fe6b70 (LWP 2427)]


Breakpoint 1, main () at main.c:7
7         while (scanf("%16s", buf) > 0) {
(gdb) p sprintf(buf, "quit")
$1 = 4
(gdb) jump 8
Continuing at 0x804afe0.
eServ terminated.
[Thread 0xb7fe6b70 (LWP 2427) exited]

Program exited normally.
(gdb)
```

GDB 的 print(簡寫 p) 指令有著 C 語言直譯器的效果

更改 **buf** 的內容，使其為 **quit**，注意要透過 **sprintf()** 一類的函式呼叫，考慮到 **buf** 的型態

忽略第 **7** 行的執行，強迫跳躍到第 **8** 行

達成！

搭配 break commands

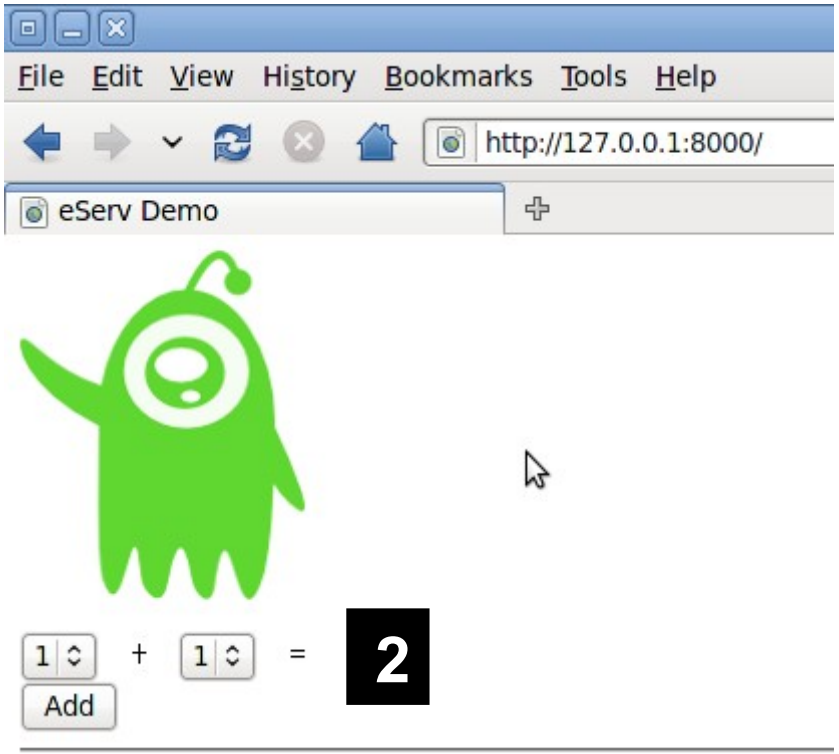
```
# gdb ./eserv
GNU gdb (GDB) 7.1-ubuntu
...
(gdb) b 7
Breakpoint 1 at 0x804afde: file main.c, line 7.
(gdb) commands
Type commands for when breakpoint 1 is hit, one per line.
End with a line saying just "end".
> p sprintf(buf, "quit")
> jump 8
> end
(gdb) r
Starting program: /tmp/eserv/eserv
[Thread debugging using libthread_db enabled]
[New Thread 0xb7fe6b70 (LWP 6257)]
```

當 break 觸發時，執行預先設定的命令

```
Breakpoint 1, main () at main.c:7
7           while (scanf("%16s", buf) > 0) {
$1 = 4
eServ terminated.
[Thread 0xb7fe6b70 (LWP 6257) exited]


Program exited normally.
```

追蹤網頁 CGI 的執行

```
[New Thread 0xb7fe6b70 (LWP 15153)]
[New Thread 0xb77e5b70 (LWP 15159)]


--add.cgi--
rAdd:1
lAdd:1
[Thread 0xb77ddb70 (LWP 22930) exited]
```




```
# grep "rAdd" *
```

用程式輸出，試著找出可能對應的原始程式碼

```
cgi_custom.c:   const char *lAdd, *rAdd;
cgi_custom.c:   rAdd = get_param_info(pHttp, "rAdd");
cgi_custom.c:   sum = atoi(lAdd) + atoi(rAdd);
Binary file cgi_custom.o matches
Binary file eserv matches
```

cgi_custom.c

```
#include "libeserv/cgi.h"


int cgi_page_sum(ExHttp *pHttp)
{
    const char *lAdd, *rAdd;
    int sum;
    char buf[32];
    printf("\n--add.cgi--\n");


    print_param(pHttp);
    lAdd = get_param_info(pHttp, "lAdd");
    rAdd = get_param_info(pHttp, "rAdd");
    sum = atoi(lAdd) + atoi(rAdd);
    ...
```

不要急著看原始程式碼，
讓 GDB 協助分析

(1) 對可疑處設定中斷點

```
jserv@venux:/tmp/eserv$ gdb ./eserv
Reading symbols from /tmp/eserv/eserv...done.
(gdb) b cgi_page_sum
Breakpoint 1 at 0x804b479: file cgi_custom.c, line 4.
(gdb) r
Starting program: /tmp/eserv/eserv
[Thread debug
[New Thread 0xb7fdeb70 (LWP 5528)]
[New Thread 0xb77ddb70 (LWP 5529)]
[Thread 0xb77ddb70 (LWP 5529) exited]
[New Thread 0xb77ddb70 (LWP 5532)]
[Switching to Thread 0xb77ddb70 (LWP 5532)]

Breakpoint 1, cgi_page_sum (pHttp=0xb77dc8cc) at cgi_custom.c:4
27   {
(gdb)
```

按下 [Add] 後，會觸發中斷點

繼續對 eserv 分析

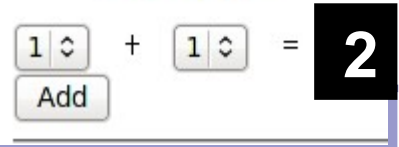
停在 CGI 的處理函式 (Callback function)

(2) GDB 協助指出程式碼

```
((gdb) list
1    #include "libeserv/cgi.h"
2
3    int cgi_page_sum(ExHttp *pHttp)
4    {
5        const char *lAdd, *rAdd;
6        int sum;
7        char buf[32];
8        printf("\n--add.cgi--\n");
9
10       print_param(pHttp);
```

```
(gdb) list
11       lAdd = get_param_info(pHttp, "lAdd");
12       rAdd = get_param_info(pHttp, "rAdd");
13       sum = atoi(lAdd) + atoi(rAdd);
14
15       sprintf(buf, "%d", sum);
16       ex_send_msg(pHttp, NULL, buf, strlen(buf));
17       return 0;
18   }
19
20   int cgi_page_txt(ExHttp *pHttp)
```

程式將 1+1=2 的算術結果，回傳給 AJAX 前端的處理



(3) 嘗試找出回傳值

```
(gdb) b 15
Breakpoint 2 at 0x804b4e5: file cgi_custom.c, line 15.
(gdb) c
Continuing.
```

```
13      sum = atoi(lAdd)+atoi(rAdd);
14
15      sprintf(buf ,"%d" ,sum);
16      return 0;
```

```
--add.cgi--
rAdd:1
lAdd:1
```

```
Breakpoint 2, cgi_page_sum (pHttp=0xb77dc8cc) at cgi_customm.c:15
38      sprintf(buf ,"%d" ,sum);
(gdb) p sum
$1 = 2
```

(4) 動態修改 CGI 回傳值

```
(gdb) n
16      ex_send_msg(pHttp, NULL, buf, strlen(buf));
(gdb) p buf
$2 =
"2\000\004\b\236\311}\267$\310}\267\004\000\000\000\324\220\004\b7\324\004\b\21
1\311}\267\001\000\000"
(gdb) p sprintf(buf, "<img src='alien.png'>")
$3 = 21
(gdb) c
```

由 buf 的內容知道，即回傳給 AJAX 前端的字串

插入一段 HTML 字串後，立即更新於瀏覽器中：由原本「單一圖片」，變成「兩個圖片」

小技巧：讓 GDB 更好用

GDB 會讀取 .gdbinit 的設定

```
# cat ~/.gdbinit
set history save on
set history size 10000
set history filename ~/.gdb_history
set print pretty on
set print static-members off
set charset ASCII
```

讓 GDB 啓動指令的歷史紀錄功能

設定 GDB 顯示的模式

```
# cat ~/.gdb_history
b cgi_page_sum
r
list
b 15
c
p sum
...
```

在 GDB 的命令提示畫面，按下 Up/Down 即可顯示指令的歷史紀錄

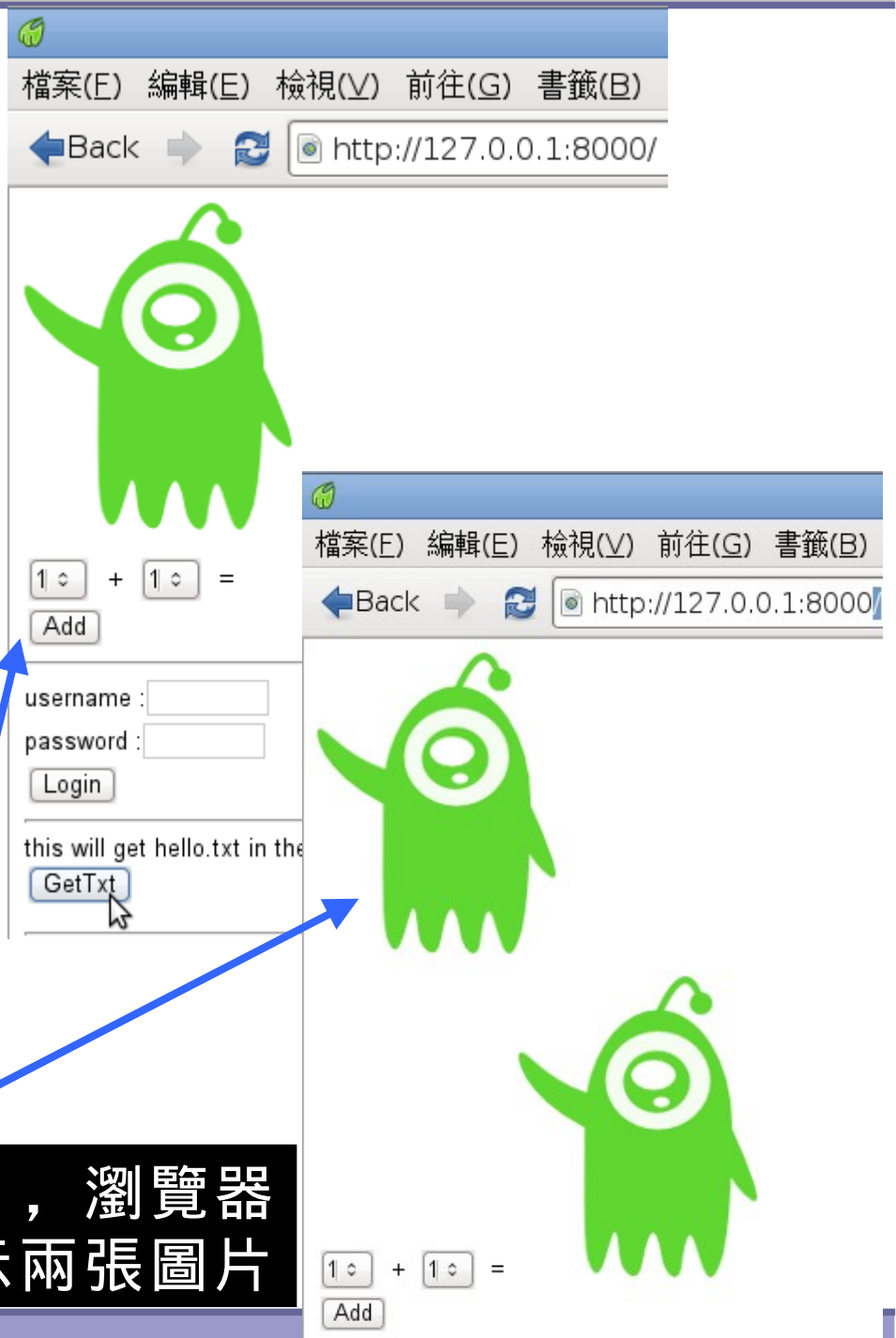
簡單的自動化

```
# cat gdb-macro
b cgi_page_sum
r
b 15
c
p sum
n
p buf
p sprintf(buf, "<img src='alien.png'>")
c
# gdb -x gdb-macro eserv
```

稍早範例的 GDB 指令取自產生的 ~/.gdb_history 檔案

特意拿掉 list 指令，避免還得切回到 GDB 提示畫面

批次執行之前的指令



一旦按下 [Add]，瀏覽器立刻顯示兩張圖片

查詢變數與符號的型態

```
(gdb) list
11      lAdd = get_param_info(pHttp, "lAdd");
12      rAdd = get_param_info(pHttp, "rAdd");
13      sum = atoi(lAdd) + atoi(rAdd);
14
15      sprintf(buf, "%d", sum);
16      ex_send_msg(pHttp, NULL, buf, strlen(buf));
17      return 0;
18  }
19
20  int cgi_page_txt(ExHttp *pHttp)
(gdb) p buf
$4 = "<img src='alien.png'>\000\004\b\211I~\267\001\000\000"
(gdb) whatis buf
type = char [32]
```

cgi_custom.c

```
1   #include "libeserv/cgi.h"
2
3   int cgi_page_sum(ExHttp *pHttp)
4   {
5           const char *lAdd, *rAdd;
6           int sum;
7           char buf[32];
8           printf("\n--add.cgi--\n");
9
10          print_param(pHttp);
```

GDB 清楚展現
C 語言的類型

也可用 ptype

詳細取得記憶體資訊

```
(gdb) p buf
$4 = "<img src='alien.png'>\000\004\b\211I~\267\001\000\000"
(gdb) whatis buf
type = char [32]
(gdb) p *buf
$5 = 60 '<'
(gdb) p &buf
$6 = (char (*)[32]) 0xb77e47ec
(gdb) x/32c 0xb77e47ec
0xb77e47ec:  60 '<'   105 'i'  109 'm'  103 'g'  32 ' '   115 's'  114 'r'  99 'c'
0xb77e47f4:  61 '='   39 '\''  97 'a'   108 'l'  105 'i'  101 'e'  110 'n'  46 '.'
0xb77e47fc:  112 'p'  110 'n'  103 'g'  39 '\''  62 '>'   0 '\000' 4 '\004'8 '\b'
0xb77e4804:  -119 '\211'  73 'I'   126 '~'  -73 '\267'   1 '\001' 0 '\000' 0 '\000' 0 '\000'
(gdb)
```

x = examine
- (gdb) **x/nfu** 位址
- 格式:
  - 印出 n 個資料項
  - f - 輸出格式
  - u - 資料項的單位

離線分析

```
(gdb) p &buf
$6 = (char (*)[32]) 0xb77e47ec
(gdb) x/32c 0xb77e47ec
0xb77e47ec:  60 '<'   105 'i'  109 'm'  103 'g'  32 ' '   115 's'   114 'r'   99 'c'
0xb77e47f4:  61 '='   39 '\''  97 'a'   108 'l'  105 'i'  101 'e'   110 'n'   46 '.'
0xb77e47fc:  112 'p'  110 'n'  103 'g'  39 '\''  62 '>'   0 '\000'  4 '\004'8 '\b'
0xb77e4804:  -119 '\211'  73 'I'   126 '~'  -73 '\267'   1 '\001' 0 '\000' 0 '\000' 0 '\000'
(gdb)dump memory here-is-my-string.bin 0xb77e47ec 0xb77e4802
```

x = examine
- (gdb) **x/nfu** 位址
- 格式:
  - 印出 n 個資料項
  - f - 輸出格式
  - u - 資料項的單位

驗證寫入的記憶體資料

```
# cat here-is-my-string.bin
<img src='alien.png'>
```

- (gdb) **dump memory** 輸出檔名 起始位置 終止位置
- (gdb) **restore** 輸入檔名 **binary** 起始位置

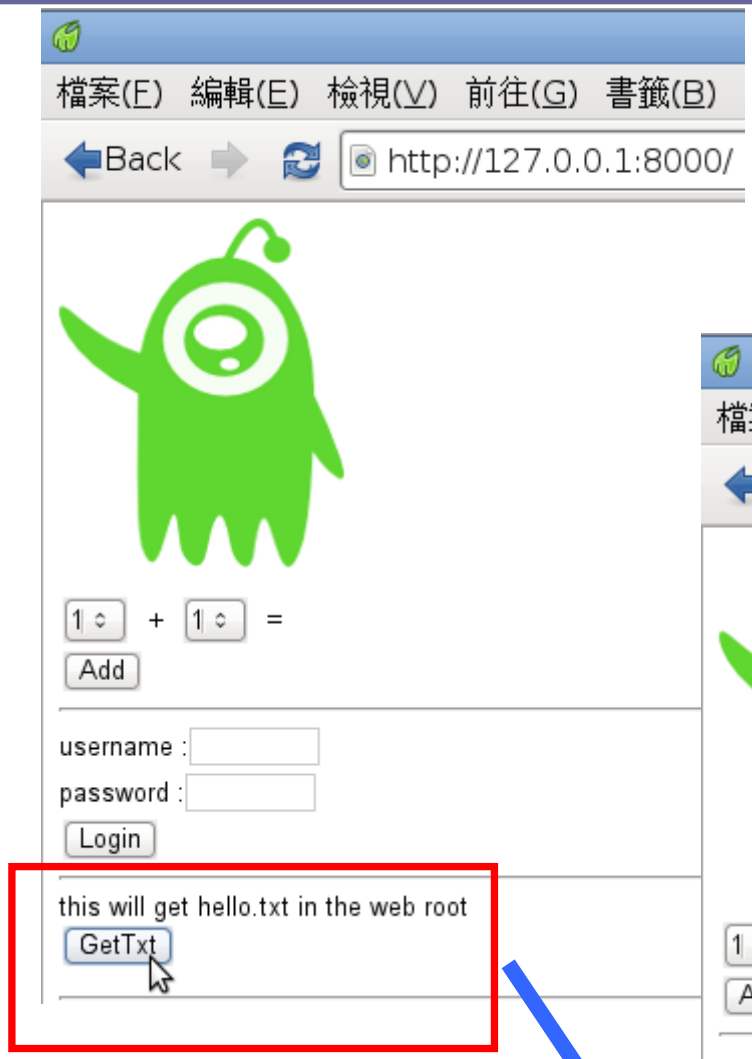
可搭配外部工具使用

牛刀小試

原本按下會印出 'hello.txt' 的內容

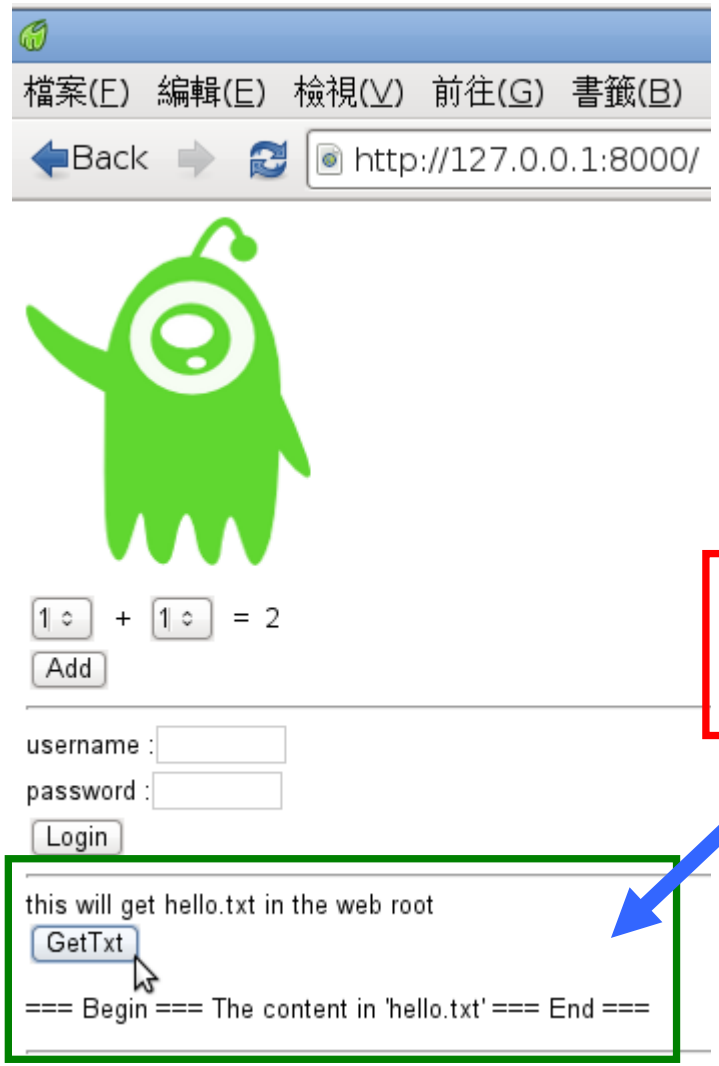
牛刀小試

注意：
不修改原始程式碼
不修改 'hello.txt' 內容
全程透過 gdb 去操作



現在直接輸出 HIT 字樣

牛刀小試（提示）



提示 (1)

會用到的 gdb 指令有

break, run, continue, print

提示 (2)

參考 cgi_custom.c 的 cgi_page_sum 函式：

```
int cgi_page_sum(ExHttp *pHttp)
{
    ...
    char buf[32];
    ...

    sprintf(buf ,"%d" ,sum);
    ex_send_msg(pHttp ,NULL ,buf ,strlen(buf));
```

原本按下會印出 'hello.txt' 的內容

牛刀小試（提示）

提示 (3)

參考 cgi_custom.c 的 cgi_page_sum 函式：

```
int cgi_page_sum(ExHttp *pHttp)
{
    ...
    char buf[32];
    ...

    sprintf(buf ,"%d" ,sum);
     ex_send_msg(pHttp ,NULL ,buf ,strlen(buf));
```

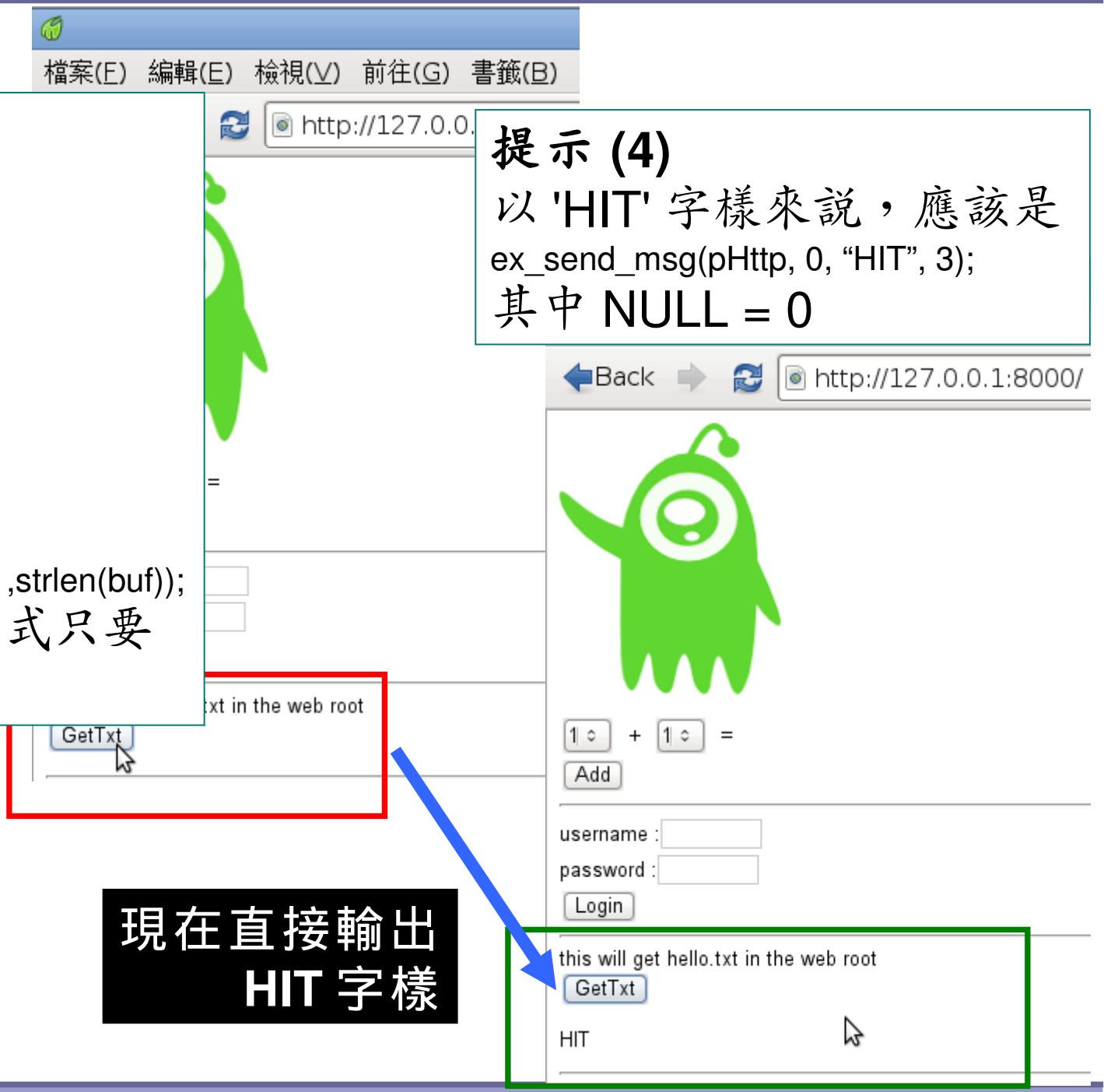
表示 ex_send_msg() 函式只要帶入合適的參數即可

提示 (4)

以 'HIT' 字樣來說，應該是

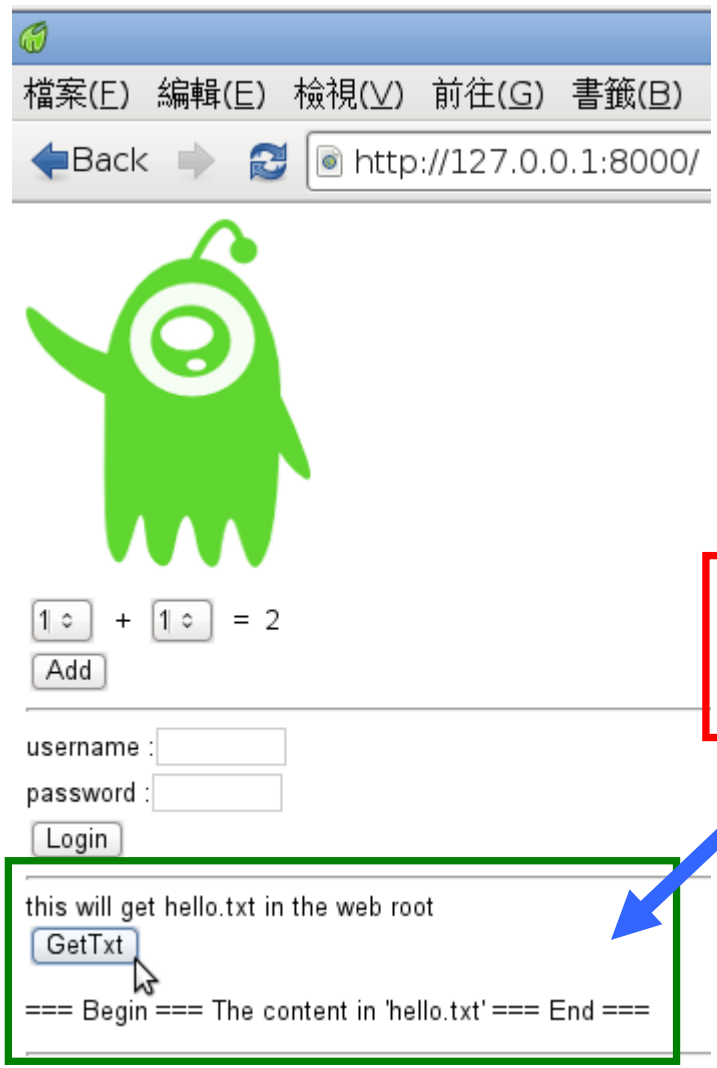
ex_send_msg(pHttp, 0, “HIT”, 3);

其中 NULL = 0



現在直接輸出 HIT 字樣

牛刀小試解答 (1/4)




```
$ gdb ./eserv
GNU gdb (GDB) 7.1-ubuntu
(gdb) break cgi_page_txt
Breakpoint 1 at 0x804b54b: file
cgi_custom.c, line 22.
(gdb) run
Starting program:
```

設定必要的中斷點

原本按下會印出 'hello.txt' 的內容

牛刀小試解答 (2/4)

```
$ gdb ./eserv
GNU gdb (GDB) 7.1-ubuntu
(gdb) break cgi_page_txt
Breakpoint 1 at 0x804b54b: file cgi_custom.c, line 22.
(gdb) run
Starting program: /tmp/eserv/eserv
[Thread debugging using libthread_db enabled]
[New Thread 0xb7fdfb70 (LWP 7015)]
[New Thread 0xb77deb70 (LWP 7018)]
```

當瀏覽器按下 [GetTxt] 按鈕時，就會觸發

```
[Switching to Thread 0xb77deb70 (LWP 7020)]

Breakpoint 1, cgi_page_txt (pHttp=0xb77dd8cc) at cgi_custom.c:22
22      printf("\n--txt.cgi--\n");
```

牛刀小試解答 (3/4)

```
Breakpoint 1, cgi_page_txt (pHttp=0xb77dd8cc) at cgi_custom.c:22
22      printf("\n--txt.cgi--\n");
(gdb) list
17      return 0;
18  }
19
20  int cgi_page_txt(ExHttp *pHttp)
21  {
22      printf("\n--txt.cgi--\n");
23      print_param(pHttp);
24      ex_send_file(pHttp, "hello.txt");
25
26      return 0;
(gdb) break 24
Breakpoint 2 at 0x804b562: file cgi_custom.c, line 24.
```

觀察相關的程式碼

原本的程式邏輯：將 'hello.txt' 檔案內容印出

牛刀小試解答 (4/4)

```
(gdb) break 24
Breakpoint 2 at 0x804b562: file cgi_custom.c, line 24.
(gdb) continue
Continuing.

--txt.cgi--

Breakpoint 2, cgi_page_txt (pHttp=0xb77dd8cc) at cgi_custom.c:24
2       ex_send_file(pHttp ,"hello.txt");
(gdb) print ex_send_msg(pHttp, 0, "HIT", 3)
$1 = 0
(gdb) continue
Continuing.
[Thread 0xb77deb70 (LWP 7020) exited]
```

原本的程式邏輯：將 'hello.txt' 檔案內容印出

換成 ex_send_msg 函式的呼叫

牛刀小試自動化

```
# cat gdb-macro-2
b cgi_custom.c:24
r
print ex_send_msg(pHttp, 0, "HIT", 3)
c
$ gdb -x gdb-macro-2 ./eserv
GNU gdb (GDB) 7.1-ubuntu
...
Breakpoint 1 at 0x804b13e: file cgi_custom.c, line 24.
...
--txt.cgi--
...
Breakpoint 1, cgi_page_txt (pHttp=0xb77e48d0) at cgi_custom.c:24
24      ex_send_file(pHttp, "hello.txt");
$1 = 3
```

換成 ex_send_msg 函式的呼叫

回頭觀察 Thread (1/3)

```
(gdb) thread apply all bt

Thread 2 (Thread 0xb7fdeb70 (LWP 24537)):
#0  0x00545422 in __kernel_vsyscall ()
#1  0x0047de88 in accept () at ../sysdeps/unix/sysv/linux/i386/socket.S:97
#2  0x08049d1d in ex_http_start () at libeserv/http.c:83
#3  0x0047680e in start_thread (arg=0xb7fdeb70) at pthread_create.c:300
#4  0x001dc7ce in clone () at ../sysdeps/unix/sysv/linux/i386/clone.S:130

Thread 1 (Thread 0xb7fdf6c0 (LWP 24534)):
#0  0x00545422 in __kernel_vsyscall ()
...
#6  0x0015d4f8 in __scanf (format=0x804d3bd "%s") at scanf.c:35
#7  0x0804b3dd in main () at main.c:7
(gdb)
```

指令格式：
thread apply all 指令

共有兩個 Thread
在 Linux 上， Thread 等同於 LWP (Light-Weight Process)

回頭觀察 Thread (2/3)

如果一開始就設定中斷點於 cgi_page_sum 函式

```
$ gdb ./eserv
(gdb) b cgi_page_sum
Breakpoint 1 at 0x804b479: file cgi_custom.c, line 4.
(gdb) run
Starting program: /tmp/eserv/eserv
[Thread debugging using libthread_db enabled]
[New Thread 0xb7fdfb70 (LWP 4411)]
[New Thread 0xb77deb70 (LWP 4413)]
[Switching to Thread 0xb77deb70 (LWP 4413)]
```

當瀏覽器按下 [Add] 按鈕時，就會觸發

```
Breakpoint 1, cgi_page_sum (pHttp=0xb77dd8cc) at cgi_custom.c:4
27  {
(gdb) info threads
```

這時候發現有三個 Thread

```
* 3 Thread 0xb77deb70 (LWP 4413)  cgi_page_sum (pHttp=0xb77dd8cc) at cgi_custom.c:4
  2 Thread 0xb7fdfb70 (LWP 4411)  0x0066c422 in __kernel_vsyscall ()
  1 Thread 0xb7fe06c0 (LWP 4408)  0x0066c422 in __kernel_vsyscall ()
```

回頭觀察 Thread (3/3)

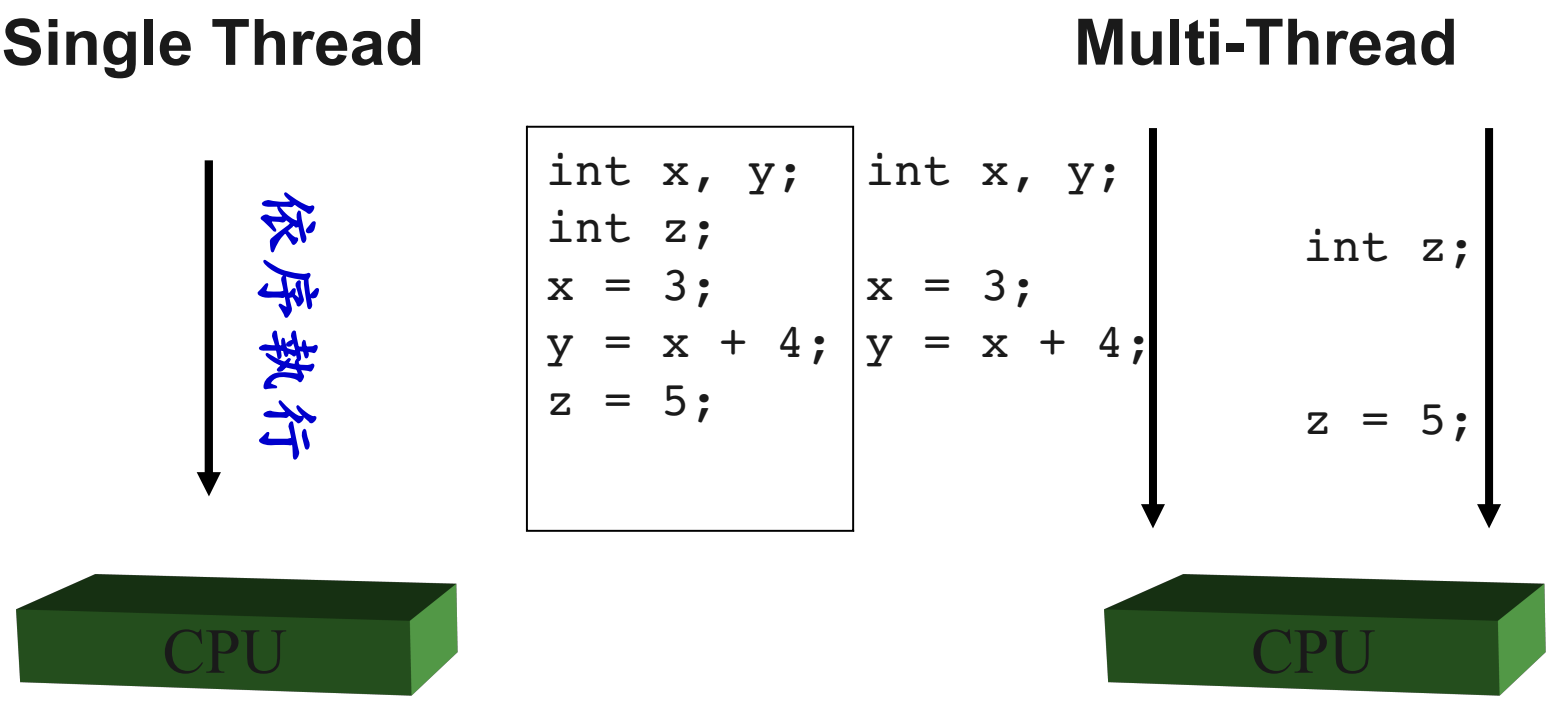
```
(gdb) thread apply all bt

Thread 3 (Thread 0xb77deb70 (LWP 4413)):
#0  cgi_page_sum (pHttp=0xb77dd8cc) at cgi_custom.c:4
#1  0x0804b6a6 in cgi_handler (pHttp=0xb77dd8cc, handle=0x804b46c) at cgi.c:23
...
#6  0x00d6993e in clone () at ../sysdeps/unix/sysv/linux/i386/clone.S:130

Thread 2 (Thread 0xb7fdfb70 (LWP 4411)):
#0  0x0066c422 in __kernel_vsyscall ()
...
#4  0x00d6993e in clone () at ../sysdeps/unix/sysv/linux/i386/clone.S:130

Thread 1 (Thread 0xb7fe06c0 (LWP 4408)):
#0  0x0066c422 in __kernel_vsyscall ()
...
#7  0x0804b3dd in main () at main.c:7
```

真實運作的網頁伺服器
同時有三個 **Thread** 運作，
負責不同的操作

Thread (執行緒)

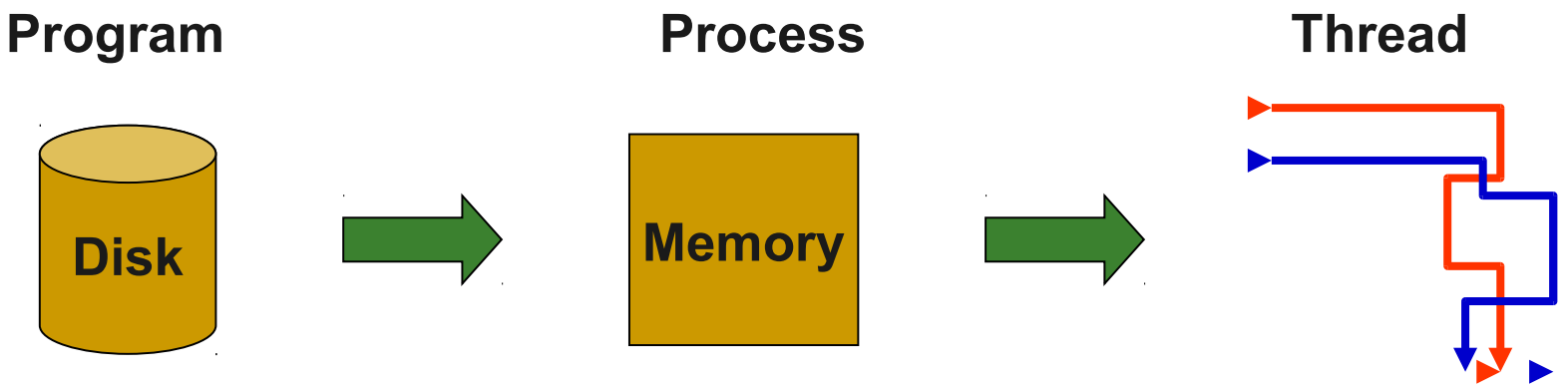
- 定義：

程式的執行軌跡



Program, Process, Thread

- 程式 (Program)
  - 儲存於硬碟中的可執行檔
- 行程 (Process)
  - 載入記憶體中的可執行檔
- 執行緒 (Thread)
  - Process 中的一段程式碼執行軌跡稱爲 Thread ，是電腦中最小的執行單位



eServ 網路伺服器

Thread 1

```
Thread 1 (Thread 0xb7fe06c0 (LWP 4408)):
#0  0x0066c422 in
#1  0x00d5a01b in
#2  0x00d03e6b in
#3  0x00d056fb in
#4  0x00d06b28 in
#5  0x00ce9238 in
#6  0x00cea508 in
#7  0x0804b3dd in main () at main.c:7
```

Thread 2

```
Thread 2 (Thread 0xb7fdfb70 (LWP 4411)):
#0  0x0066c422 in __kernel_vsyscall
#1  0x005cce88 in accept
#2  0x08049d1d in ex_http_start
#3  0x005c580e in start_thread
#4  0x00d6993e in clone
```

Thread 3

```
int cgi_page_sum(ExHttp *pHttp)
{
    const char *lAdd ,*rAdd;
    int sum;
    char buf[32];
    printf("\n--add.cgi--\n");
    ...
```

add.cgi

AJAX = Asynchronous JavaScript and XML

瀏覽器

HTTP Request

HTTP 資料回傳

持續將 XML 資料更新

打造電子相簿

顯示 No.1

注意：
瀏覽器的「上一頁」按鍵沒有顯示，表示資料都在同一頁面顯示。此為應用 AJAX 的示範，不需刷新全部頁面，藉此提昇反應速度

開始只有一段提示

理解程式運作的方式之一，去驗證細節與延展擴充既有系統

打造電子相簿

顯示 No.1

Demo - 綠網頁瀏覽器

檔案(F) 編輯(E) 檢視(V) 前往(G) 書籤(B) 工具(T) 視窗(W)

Back http://127.0.0.1:8000/

this will play gallery step-by-step.

Next

No. 1

顯示 No.2

Demo - 綠網頁瀏覽器

檔案(F) 編輯(E) 檢視(V) 前往(G) 書籤(B) 工具(T) 視窗(W) 求助(H)

Back http://127.0.0.1:8000/

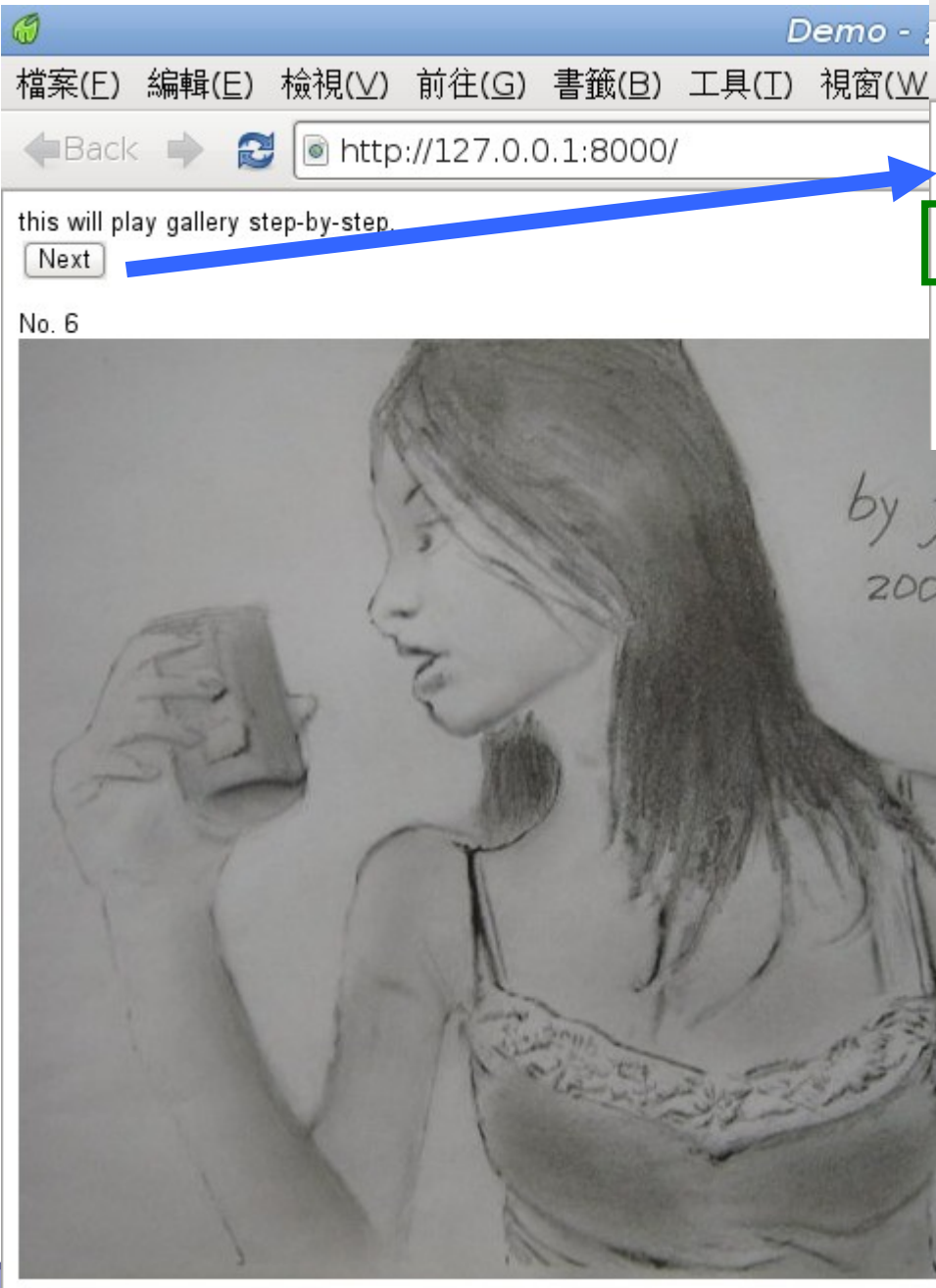
this will play gallery step-by-step.

Next

No. 2

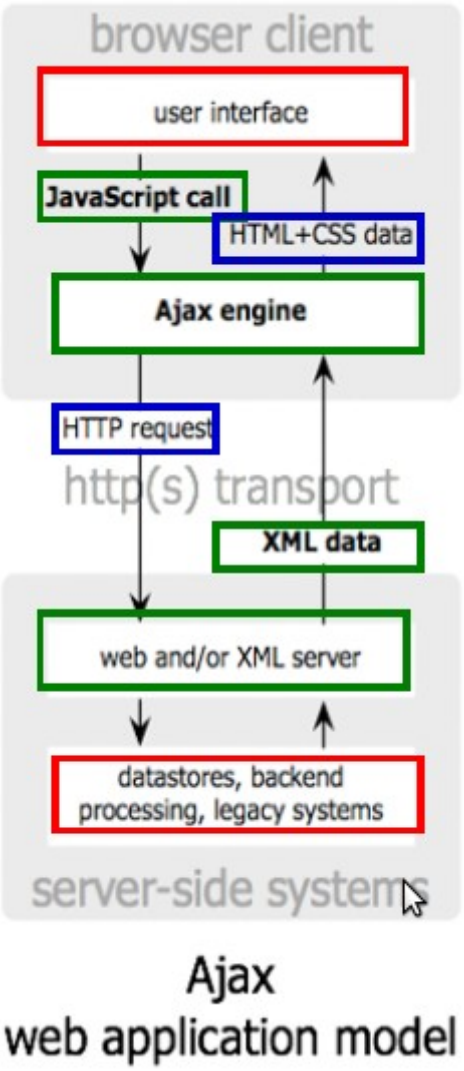
打造電子相簿

顯示 No.6 (最後一張)



當沒有足以顯示的圖片，告知使用者即將重新開始

```
<body>
    <span>this will play gallery step-by-step.</span><br />
    <button type="button" id="btnGetTxt">Next</button>
    <p id="txt" />
    <hr />
</body>
```



Ajax
web application model

www/gallery.html


```
<script>
$("#btnGetTxt").click(
    function() {
        $.get("gallery.cgi",
        function(data) {
            $("#txt")[0].innerHTML = data;
        })
    }
)
</script>
```

自 **txt.cgi** 取得資料 **(GET** 方法 **)**，並將識別為 **txt** 的 **HTML** 元素內容更新為 **web server** 的回傳值

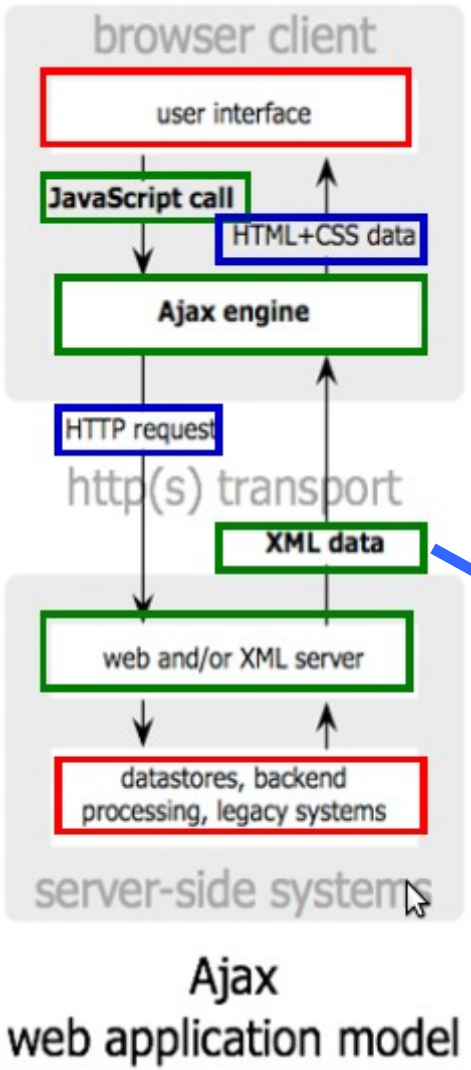
```
<body>
    <span>this will play gallery step-by-step.</span><br />
    <button type="button" id="btnGetTxt">Next</button>
    <p id="txt" />
    <hr />
</body>
```

當識別為 **btnGetTxt** 的 **HTML** 元素
(即 **Next** 按鈕)
被按下 (即 **click()** 動作) 時，
會觸發 **JavaScript** 動作

www/gallery.html


```
<script>
$("#btnGetTxt").click(
    function() {
        $.get("gallery.cgi",
        function(data) {
            $("#txt")[0].innerHTML = data;
        })
    }
)
</script>
```

自 **txt.cgi** 取得資料 **(GET** 方法 **)**，並將識別為 **txt** 的 **HTML** 元素內容更新為 **web server** 的回傳值



Ajax
web application model

```
<body>
    <span>this will play gallery step-by-step.</span><br />
    <button type="button" id="btnGetTxt">Next</button>
    <p id="txt" />
    <hr />
</body>
```

```
<script>
$("#btnGetTxt").click(
    function() {
        $.get("gallery.cgi",
        function(data) {
            $("#txt")[0].innerHTML = data;
        })
    }
)
</script>
```

自 **txt.cgi** 取得資料 **(GET** 方法 **)**，並將識別為 **txt** 的 **HTML** 元素內容更新為 **web server** 的回傳值

cgi_custom.c

```
int cgi_page_gallery(ExHttp *pHttp)
{
    char buf[32];
    printf("\n--gallery.cgi--\n");

    print_param(pHttp);
     ...
```

cgi_custom.c

```
int cgi_page_gallery(ExHttp *pHttp)
{
    static int count = 0;
    char buf[40];
    printf("\n--gallery.cgi--\n");

    print_param(pHttp);

    count++;
    if (count > 3) {
        sprintf(buf, "All of pictures are shown. Reset");
        count = 0;
    }
    else {
        sprintf(buf ,"No. %d<br /><img src='%d.jpg'>", count, count);
    }
    ex_send_msg(pHttp, NULL, buf ,strlen(buf));
    return 0;
}
```

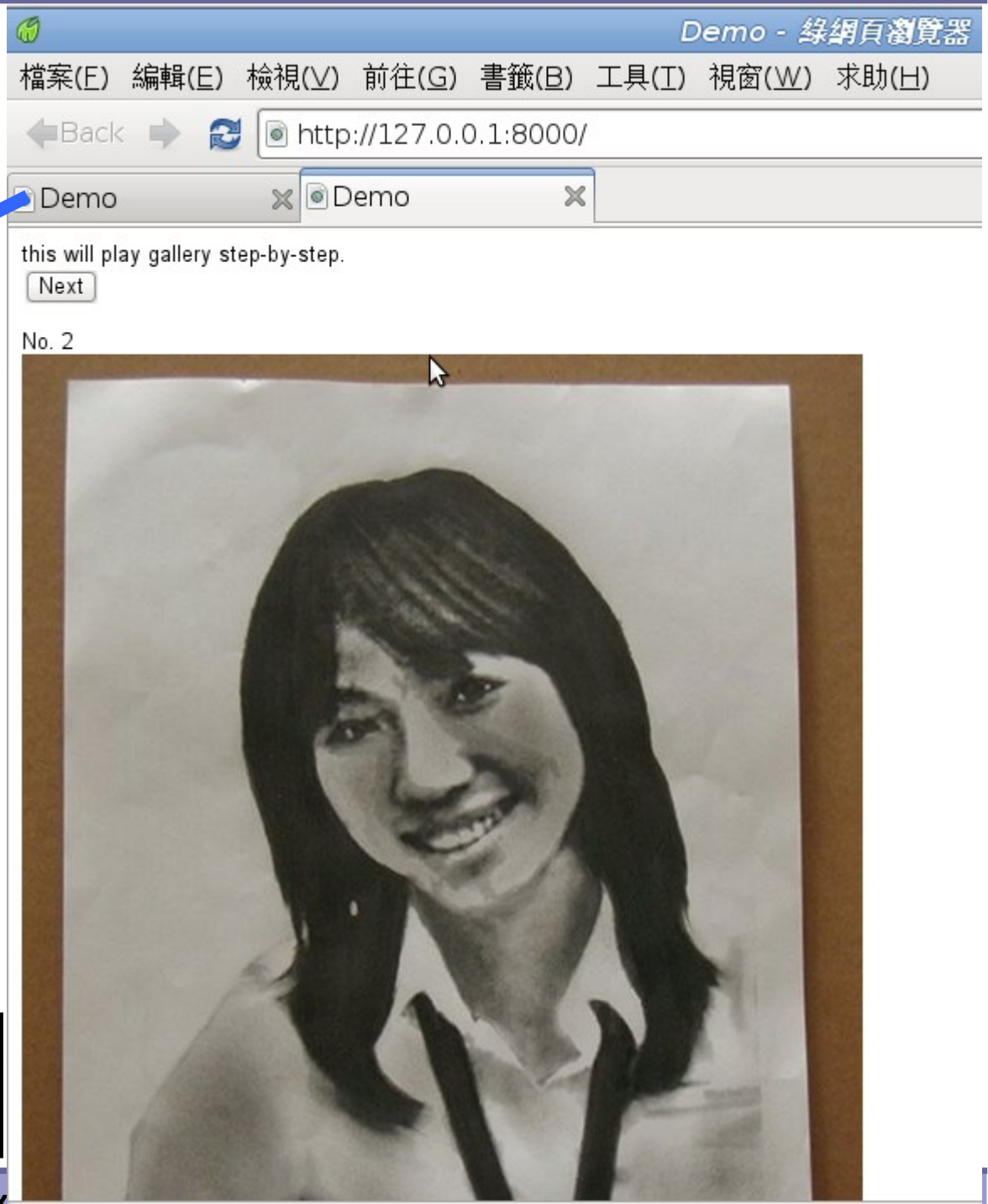
cgi_custom.c

```
int cgi_page_gallery(ExHttp *pHttp)
{
    static int count = 0;
    char buf[40];
    printf("\n--gallery.cgi--\n");

    print_param(pHttp);

    count++;
    if (count > 3) {
        sprintf(buf, "All of pictures are shown. Reset");
        count = 0;
    }
    else {
        sprintf(buf ,'No. %d<br /><img src='%d.jpg'>", count, count);
    }
    ex_send_msg(pHttp, NULL, buf ,strlen(buf));
    return 0;
}
```

```
<script>
$("#btnGetTxt").click(
    function() {
        $.get("txt.cgi",
        function(data) {
            $("#txt")[0].innerHTML = data;
        })
    }
)
</script>
```

當打開兩個頁面時

左邊的頁面原本是 **No.1**



當我們建立新的網路連線時，
編號已經跳到 **No.2**

當打開兩個頁面時

0xlab

左邊的頁面原本是 No.1

進階題材：克服多執行緒環境的**資料隔離**議題，需額外建立針對 HTTP session 的調整

cgi_custom.c

```
int cgi_page_gallery(ExHttp *pHttp)
{
    static int count = 0;
     ...
    count++;
    if (count > 3) {
        sprintf(buf, "All of pictures are shown. Reset");
        count = 0;
    }
```

從左右兩個 HTTP 連線狀態來看，變數 count 是共用的

追蹤程式執行流程 (1/9)

如果一開始就設定中斷點於 cgi_page_sum 函式

```
$ gdb ./eserv
(gdb) b cgi_page_sum
Breakpoint 1 at 0x804b479: file cgi_custom.c, line 4.
(gdb) run
Starting program: /tmp/eserv/eserv
[Thread debugging using libthread_db enabled]
[New Thread 0xb7fdfb70 (LWP 4411)]
[New Thread 0xb77deb70 (LWP 4413)]
[Switching to Thread 0xb77deb70 (LWP 4413)]
```

當瀏覽器按下 [Add] 按鈕時，就會觸發

```
Breakpoint 1, cgi_page_sum (pHttp=0xb77dd8cc) at cgi_custom.c:4
27   {
(gdb)
```

追蹤程式執行流程 (2/9)

試著透過 gdb 去追蹤程式流程

```
(gdb) bt
#0  cgi_page_sum (pHttp=0xb77e48d0) at cgi_custom.c:4
#1  0x0804ae6f in cgi_handler (pHttp=0xb77e48d0, handle=0x804b048)
    at libeserv/cgi.c:23
#2  0x0804a1f3 in cgiProcess (pHttp=0xb77e48d0) at libeserv/request.c:160
#3  0x0804a2cc in replyHandler (pHttp=0xb77e48d0) at libeserv/request.c:192
#4  0x0804a4b8 in requestHandler (s=0x6) at libeserv/request.c:246
#5  0x0013396e in start_thread () from /lib/tls/i686/cmov/libpthread.so.0
#6  0x002149de in clone () from /lib/tls/i686/cmov/libc.so.6
(gdb) up
#1  0x0804b6a6 in cgi_handler (pHttp=0xb77dd8cc, handle=0x804b46c) at cgi.c:93
93      return pf(pHttp);
```

切換到上一層

看起來沒有呼叫 cgi_page_sum 函式？！

追蹤程式執行流程 (3/9)

```
(gdb) list
18
19  int cgi_handler(ExHttp *pHttp, void *handle)
20  {
21      int (*pf)(ExHttp *) = handle;
22
23      return pf(pHttp);
24  }
25
26  int errorLog(ExHttp *pHttp, const char *mess)
27  {
```

怎麼看都不像有呼叫？
(cgi_page_sum)

探索隱藏在原始程式碼背後的奧秘：
Function pointer & dispatcher

追蹤程式執行流程 (4/9)

```
(gdb) bt
#0  cgi_page_sum (pHttp=0xb77e48d0) at cgi_custom.c:4
#1  0x0804ae6f in cgi_handler (pHttp=0xb77e48d0, handle=0x804b048)
    at libeserv/cgi.c:23
#2  0x0804a1f3 in cgiProcess (pHttp=0xb77e48d0) at libeserv/request.c:160
#3  0x0804a2cc in replyHandler (pHttp=0xb77e48d0) at libeserv/request.c:192
#4  0x0804a4b8 in requestHandler (s=0x6) at libeserv/request.c:246
#5  0x0013396e in start_thread () from /lib/tls/i686/cmov/libpthread.so.0
#6  0x002149de in clone () from /lib/tls/i686/cmov/libc.so.6
(gdb) up
#1  0x0804b6a6 in cgi_handler (pHttp=0xb77dd8cc, handle=0x804b048) at
libeserv/cgi.c:23
93      return pf(pHttp);
(gdb) p cgi_page_sum
$1 = {int (ExHttp *)} 0x804b048 <cgi_page_sum>
```

cgi_handler 的參數 handle 就是函式 cgi_page_sum 的位址

追蹤程式執行流程 (5/9)

```
(gdb) up
#2  0x0804a1f3 in cgiProcess (pHttp=0xb77e48d0) at libeserv/request.c:160
160         if (cgi_handler(pHttp, handle) < 0) {
(gdb) list
155         if (decodeParam(pHttp) < 0) {
156             errorLog(pHttp , "param decode error");
157             ret = -3;
158             break;
159         }
160         if (cgi_handler(pHttp, handle) < 0) {
161             errorLog(pHttp, "handler error");
162             ret = -4;
163         }
164     } while (0);
```

如果能理解參數 handle 的傳遞與指派狀況，應能知曉系統的設計

試圖追蹤此段運作原理

追蹤程式執行流程 (6/9)

顯示中斷點設置的狀態

```
(gdb) info breakpoints
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x0804b479 in cgi_page_sum at cgi_custom.c:4
breakpoint already hit 1 time
(gdb) del 1
(gdb) break cgiProcess
Breakpoint 2 at 0x804a517: file libeserv/request.c, line 142.
(gdb) c
Continuing.
--add.cgi--
[Thread 0xb77deb70 (LWP 11071) exited]
[New Thread 0xb77deb70 (LWP 11100)]
[Switching to Thread 0xb77deb70 (LWP 11100)]
```

del = delete：移除指定的中斷點

設定 cgiProcess 函式為新的中斷點

恢復之前的 CGI 執行

```
Breakpoint 2, cgiProcess (pHttp=0xb77dd8cc) at libeserv/request.c:142
142     int ret = 0;
```

追蹤程式執行流程 (7/9)

0xlab

```
(gdb) watch handle
Hardware watchpoint 3: handle
(gdb) c
Continuing.
Hardware watchpoint 3: handle
Old value = (void *) 0xb6fe398c
New value = (void *) 0x0
cgiProcess (pHttp=0xb77dd8cc) at libeserv/request.c:145
145             if((handle=cgi_page_find(pHttp->url)) == NULL){
```

不同於 **breakpoint** 綁定在某行，
一個 **watchpoint** 可能會在任意行 **break**

command	說明
watch [exp]	當 exp 值變動時 break
rwatch [exp]	當 exp 被讀取時 break
awatch [exp]	當 exp 被讀取或被更動時則 break
info watch	察看目前的 watch point

變數 handle 的
內容值從「未定義」
(0xb6fe398c)
指定為 NULL

libeserv/request.c

```
static int cgiProcess(ExHttp *pHttp)
{
  int ret = 0;

  void *handle = NULL;
  do {
    if((handle=cgi_page_find(pHttp->url)) == NULL){
    ...
```

追蹤程式執行流程 (8/9)

```
(gdb) c
Continuing.
Hardware watchpoint 3: handle

Old value = (void *) 0x0
New value = (void *) 0x804b048
0x0804a15a in cgiProcess (pHttp=0xb6fe38d0) at libeserv/request.c:145
145         if ((handle = cgi_page_find(pHttp->url)) == NULL) {
```

handle 的值一旦變動，就會觸發 watchpoint

從 URL 找出特定 CGI handler 的進入點

```
#1  0x0804b6a6 in cgi_handler (pHttp=0xb77dd8cc, handle=0x804b048)
(gdb) p cgi_page_sum
$1 = {int (ExHttp *)} 0x804b048 <cgi_page_sum>
```

追蹤程式執行流程 (9/9)

```
(gdb) c
Continuing.


--txt.cgi--
```

最後 watchpoint 隨著函式結束 (變數 handle 的可見區域) 而消滅

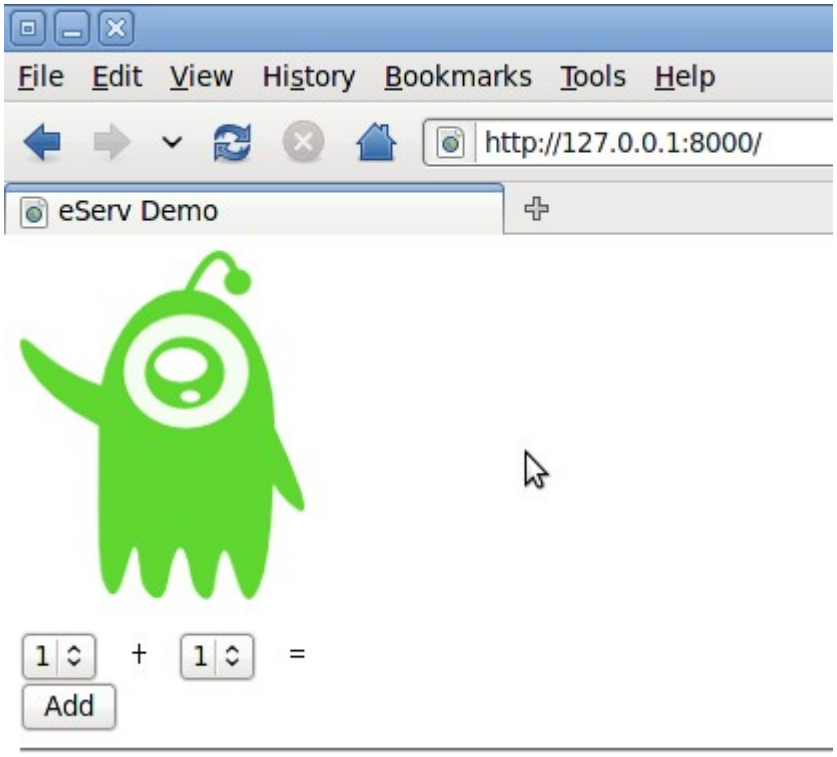
```
Watchpoint 3 deleted because the program has left the block in
which its expression is valid.
0x0804a2cc in replyHandler (pHttp=0xb6fe38d0) at libeserv/request.c:192
192         ret = cgiProcess(pHttp);
```

Callback function 運用的技巧

CGI Callback 是如何被註冊？

```
(gdb) run
Starting program: /tmp/eserv/eserv
[Thread debugging using libthread_db enabled]
[New Thread 0xb7fe6b70 (LWP 14544)]
eServ is running...
```

開啟網頁瀏覽器
網址： http://127.0.0.1:8000/



cgi_page_sum

從 Callback 結果回推註冊過程 (1)

```
160         if (cgi_handler(pHttp, handle) < 0) {
161             errorLog(pHttp, "handler error");
162             ret = -4;
163         }
```

```
$ gdb ./eserv
GNU gdb (GDB) 7.1-ubuntu
...
(gdb) b libeserv/request.c:160
Breakpoint 1 at 0x804a1e1: file libeserv/request.c, line 160.
(gdb) commands
Type commands for when breakpoint 1 is hit, one per line.
End with a line saying just "end".
> quiet
> set $addr=handle
> continue
> end
(gdb) b cgi_page_sum if $addr == &cgi_page_sum
Breakpoint 2 at 0x804b055: file cgi_custom.c, line 4.
```

從之前 watchpoint 的結果得知，handle 是區域變數，一旦離開 cgiProcess 函式範疇，就無法存取

透過 **break commands**，將變數 **handle** 的值放入 **GDB** 變數 **$addr**，隨後仍可得知位址。（重要技巧）

GDB 指令 **quiet** 則要求不再顯示此 **break**

現在 $addr 內含 cgiProcess 函式所找到的 CGI callback function 位址

從 Callback 結果回推註冊過程 (2)

中斷點條件

b cgi_page_sum if $addr == &cgi_page_sum

觸發，表示 **cgi_page_sum** 函式已「被觸發」

```
(gdb) r
...
Breakpoint 1, cgiProcess (pHttp=0xb77e48d0) at libeserv/request.c:160
160           if (cgi_handler(pHttp, handle) < 0) {


Breakpoint 2, cgi_page_sum (pHttp=0xb77e48d0) at cgi_custom.c:4
4     {
gdb) where
#0  cgi_page_sum (pHttp=0xb77e48d0) at cgi_custom.c:4
#1  0x0804ae6f in cgi_handler (pHttp=0xb77e48d0, handle=0x804b048)
    at libeserv/cgi.c:23
#2  0x0804a1f3 in cgiProcess (pHttp=0xb77e48d0) at libeserv/request.c:160
#3  0x0804a2cc in replyHandler (pHttp=0xb77e48d0) at libeserv/request.c:192
#4  0x0804a4b8 in requestHandler (s=0x6) at libeserv/request.c:246
#5  0x0013396e in start_thread () from /lib
#6  0x002149de in clone () from /lib/tls/i6
```

Web/HTML 的 CGI 是如何對應到 eServ 內部註冊的 callback 呢？

從 Callback 結果回推註冊過程 (3)

```
(jserv@venux:/tmp/eserv$ gdb ./eserv
GNU gdb (GDB) 7.1-ubuntu
...
(gdb) b cgi_page_find
Breakpoint 1 at 0x8049aa7: file libeserv/http.c, line 117.
(gdb) r
...
Breakpoint 1, cgi_page_find (pageName=0xb[illegible]e4989 "sum.cgi")
    at libeserv/http.c:117
117     return ex_hash_find(&ExContext.pageMap, pageName);
(gdb) del 1
(gdb) b main
Breakpoint 2 at 0x804af
```

看來需要對付一個 Hash table

libeserv/request.c

```
static int cgiProcess(ExHttp *pHttp)
{
  int ret = 0;

  void *handle = NULL;
  do {
    if((handle=cgi_page_find(pHttp->url)) == NULL){
      ...
```

URL 與 CGI 的對應

從 Callback 結果回推註冊過程 (4)

```
(gdb) r
The program being debugged has...
Start it from the beginning? (y or n) y
Starting program: /tmp/eserv/eserv
...
Breakpoint 2, main () at main.c:4
4    {
(gdb) watch ExContext.pageMap
Watchpoint 3: ExContext.pageMap
(gdb) c
Continuing.
Watchpoint 3: ExContext.pageMap
```

追蹤

```
Old value = {
  ...
  mpool = 0x0,
  ...
}
New value = {
  buckets = 0x0,
  size = 0,
  mpool = 0x804f1c0,
  hashfun = 0,
  hashcmp = 0
}
ex_hash_init (hm=0x804fa14, mp=0x804f1c0,
              _size=97) at libeserv/hash.c:17
17          (ex_hashlist **) ex_mpool_malloc(mp,
```

```
(gdb) whatis ExContext.pageMap
type = ex_hashmap
```

藉由 watch，可窺見 ExContext 結構體內部的修改變化

對照之前的中斷點，可知 ex_hash_find 函式對 ExContext 作處理

```
Breakpoint 1, cgi_page_find (pageName=0xb77e4989 "sum.cgi")
117     return ex_hash_find(&ExContext.pageMap, pageName);
```

從 Callback 結果回推註冊過程 (5)

```
(gdb) c
Continuing.
Watchpoint 2: ExContext.pageMap
Old value = {
  ...
  hashfun = 0x8049097 <ex_hashfun_str>,
  hashcmp = 0x8048e44 <ex_hashcmp_str>
}
New value = {
  ...
  hashfun = 0x8049097 <ex_hashfun_str>,   hashcmp = 0x80497a4 <ex_hashcasecmp_str>


ex_init () at libeserv/http.c:130
130       ExContext.mimeMap.hashcmp = (void *) ex_hashcasecmp_str;
```

ExContext.pageMap 結構體的兩個 callback 都被適當的設定，指向處理 Hash table 的函式

從 callback 結果回推註冊過程 (6)

```
(jserv@venux:/tmp/eserv$ gdb ./eserv
GNU gdb (GDB) 7.1-ubuntu
...
(gdb) b ex_hash_find
Breakpoint 1 at 0x8049022: file libeserv/hash.c, line 64.
(gdb) r
Starting program: /tmp/eserv/e
[Thread debugging using libthr
[New Thread 0xb7fe6b70 (LWP 11
[New Thread 0xb77e5b70 (LWP 11
[Switching to Thread 0xb77e5b7
```

實地走訪 Hash table 的處理方式

```
void* ex_hash_find(const ex_hashmap *hm, const void *key)
{
    int pos = hm->hashfun(key) % hm->size;
    ex_hashlist *nlh = hm->buckets[pos];
    void *ret = NULL;
    while (nlh != NULL) {
        if (hm->hashcmp(nlh->key, key)) {
            ret = nlh->value;
            break;
        }
        nlh = nlh->next;
    }
    return ret;
}
```

```
Breakpoint 1, ex_hash_find (hm=0xb77e48f0, key=0x804c71a) at libeserv/hash.c:64
64      int pos = hm->hashfun(key) % hm->size;
```

從 Callback 結果回推註冊過程 (7)

```
Breakpoint 1, ex_hash_find (hm=0xb77e48f0, key=0x804c71a) at libeserv/hash.c:64
64          int pos = hm->hashfun(key) % hm->size;
(gdb) p hm->hashfun
$1 = (int (*)(const void *)) 0x804a530 <ex_hashfun_uchar>
(gdb) c
Continuing.

Breakpoint 1, ex_hash_find (hm=0x804fa00, key=0x804c6e4) at libeserv/hash.c:64
64          int pos = hm->hashfun(key) % hm->size;
(gdb) p hm->hashfun
$2 = (int (*)(const void *)) 0x8049097 <ex_hashfun_str>
(gdb) p (char *) key
$3 = 0x804c6e4 "html"
```

從參數 **hm** 的內含值可知，程式執行流程中，前後使用兩份 **Hash table**

發現到中斷點設定於 **cgi_page_find** 時，同樣也指定的 **callback – ex_hashfun_str**

驗證 **Hash table** 的功能，將參數 **key** 轉型為 **char *** 得到字串內容為 **html**

從 callback 結果回推註冊過程 (8)

```
(jserv@venux:/tmp/eserv$ gdb ./eserv
GNU gdb (GDB) 7.1-ubuntu
...
(gdb) b ex_hash_find if strcmp((char *) key, "sum.cgi") == 0
Breakpoint 1 at 0x8049022: file libeserv/hash.c, line 64.
(gdb) r
...
Breakpoint 1, ex_hash_find (hm=0x804fa14, key=0xb77e4989) at libeserv/hash.c:64
64        int pos = hm->hashfun(key) % hm->size;
(gdb) watch ret
Hardware watchpoint 2: ret
(gdb) c
Continuing.
Hardware watchpoint 2: ret
Old value = (void *) 0xffffffff
New value = (void *) 0x0
```

當瀏覽器按下 [Add] 按鈕時，就會觸發

從 callback 結果回推註冊過程 (9)

眞相大白：
- 建立 Hash table
- 註冊 Hash function
- 將 CGI 的 pageName 對應到指定的 CGI callback

```
Hardware watchpoint 2: ret



Old value = (void *) 0xffffffff

New value = (void *) 0x0

ex_hash_find (hm=0x804fa14, key=0xb77e4989) at libeserv/hash.c:67

67          while (nlh != NULL) {

(gdb) c

Continuing.

Hardware watchpoint 2: ret



Old value = (void *) 0x0

New value = (void *) 0x804b048

ex_hash_find (hm=0x804fa14, key=0xb77e4989) at libeserv/hash.c:70

(gdb) p cgi_page_sum

$2 = {int (ExHttp *)} 0x804b048 <cgi_page_sum>
```

當我們透過 **watch** 指令去追蹤 **ex_hash_find** 函式時，可發現在 **key="sum.cgi"** 所註冊的 **callback** 就是之前多次分析的 **cgi_page_sum** 函式

```
extern int cgi_page_sum(ExHttp *pHttp);
...
/* customized page handler declare here */
cgi_page cgi_pages[] = {
        {
                .name = "sum.cgi",
                .callback = cgi_page_sum,
        },
        ...
```

cgi_custom.h

```
typedef struct {
    char *name;
    int (*callback)(ExHttp *pHttp);
} cgi_page;
```

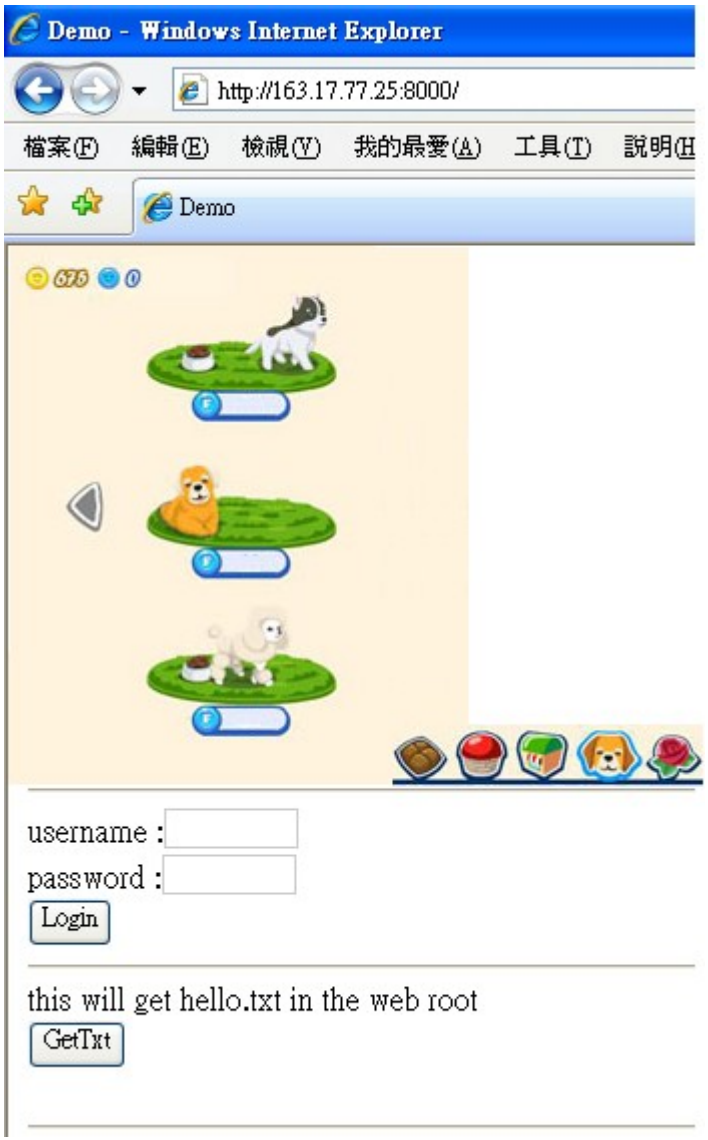
libeserv/cgi.h

```
/* FIXME : flexible inclusion of customized CGI */
#include "../cgi_custom.h"

extern cgi_page cgi_pages[];

void cgi_init()
{
        size_t i;
        for (i = 0; i < sizeof(cgi_pages) / sizeof(cgi_page); i++)
                cgi_page_add(cgi_pages[i].name, cgi_pages[i].callback);
}
```

libeserv/cgi.c

此爲 C99 規格中”Designated Initializers”的實例，允許宣告結構體（struct 或 union）裡頭特定成員的初始值

課程回顧



- 做好開發「山寨版」開心農場的基礎建設，得知如何透過 GDB 去追蹤分析，進而作擴充
- 回顧 C 語言核心概念：指標、記憶體操作、函式呼叫、函式指標
- 回顧重要觀念：callback function、多執行緒、拼湊表 (Hash table)
- 下一課：
  - 解決伺服器的多工同步議題
  - 適度作前端整合
  - 功能實做 (引入資料結構與演算法)