

程式人

月刊
雜誌

Programmer



讀書做善事、寫書做公益 – 歡迎程式人認養專欄或捐出您的網誌
參考價：NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體
羅慧夫顱顏基金會 彰化銀行 (009) 帳號：5234-01-41778-800



愛心條碼

程式人雜誌

開放公益出版品

2013 年 3 月號

本期內容

- 授權聲明
- 程式人短訊
 - 法律短訊-創作共用授權間的相容性問題與疑惑
 - 書籍短訊：電腦類的免費電子書網站
 - 網路短訊-MOOC 大量線上教育
- 程式人介紹
 - 開放原始碼之父-Richard Stallman
 - LLVM編譯器基礎架構的靈魂人物-Chris Lattner
- 程式人頻道
 - 看影片學 C# 視窗程式設計
- 程式人文集
 - Arduino入門教學(3) – LED 控制實驗 (作者：Copper Maa)
 - JavaScript (3) – Closure 與匿名函數 (作者：陳鍾誠)
 - R 統計軟體：(1) 簡介與基本操作 – (作者：陳鍾誠)
 - 從 C# 看作業系統：(1) Thread 與 Deadlock 實作 – (作者：陳鍾誠)
- 雜誌訊息
 - 讀者訂閱

- [投稿須知](#)
- [參與編輯](#)
- [公益資訊](#)

授權聲明

本雜誌採用 創作共用：[姓名標示、相同方式分享](#) 授權，若您想要修改本書產生衍生著作時，至少應該遵守下列授權條件：

1. 標示原作者姓名
2. 採用 創作共用：[姓名標示、相同方式分享](#) 的方式公開衍生著作。

另外、當本雜誌中有文章或素材並非採用 [姓名標示、相同方式分享](#) 時，將會在該文章或素材後面標示其授權，此時該文章將以該標示的方式授權釋出，請修改者注意這些授權標示，以避免產生侵權糾紛。

例如有些文章可能不希望被作為「商業性使用」，此時就可能會採用創作共用：[姓名標示、非商業性、相同方式分享](#) 的授權，此時您就不應當將該文章用於商業用途上。

最後、懇請勿移除公益捐贈的相關描述，以便讓愛心得以持續散播！

程式人短訊

- 法律短訊-創作共用授權間的相容性問題與疑惑
 - 相容性問題
 - 前幾期授權錯誤的更正方式
 - 筆者的疑惑
- 書籍短訊：電腦類的免費電子書網站
- 網路短訊-MOOC 大量線上教育

法律短訊-創作共用授權間的相容性問題與疑惑

相容性問題

在前幾期的程式人雜誌中，我們都有引用並修改了維基百科的文章，然後採用 [姓名標示、非商業性、相同方式分享](#) 的 CC 授權方式分享整本雜誌，後來我收到 jserv 兄的 email 之後，才知道原來維基百科是採用 [姓名標示、相同方式分享](#) 的授權，而且更慘的是，這兩個授權互不相容，不能修改後以另一個授權釋出。

在 [Remixing OER: A Guide to License Compatibility](#) 這篇文章中，Creative Commons 組織說明了 CC 授權的相

容性，並且畫出了一個相容矩陣，如下圖所示。

Compatibility chart		Terms that can be used for a derivative work or adaptation					
		BY	BY-NC	BY-NC-ND	BY-NC-SA	BY-ND	BY-SA
Status of original work	BY						
	BY-NC						
	BY-NC-ND						
	BY-NC-SA						
	BY-ND						
	BY-SA						

圖、CC 授權的相容性矩陣

其中 BY-NC-SA 就是 姓名標示、非商業性、相同方式分享 授權，而 BY-SA 就是 姓名標示、相同方式分享 授權，而文章中對 BY-SA 授權的說明如下。

This license lets others copy, share, modify and build upon your work even for commercial purposes, as long as

they credit you and license new creations derived from your work under the same conditions.

由這段文字看來，SA (Share-Alike) 屬性的授權就必須用同樣授權進行分享，因此就不能將 BY-SA 改為 BY-NC-SA 授權分享了，雖然 BY-NC-SA 的分享條件事實上比 BY-SA 更嚴格。

前幾期授權錯誤的更正方式

由於 BY-SA 與 BY-NC-SA 兩者完全不能相容，因此本雜誌前兩期的做法，有些文章乃是修改自維基百科，然後整本雜誌標示 [姓名標示](#)、[非商業性](#)、[相同方式分享](#) 的方式就有錯誤了，因此筆者會將前兩期有修正自維基百科的文章改標示為 [姓名標示](#)、[相同方式分享](#) 授權，以更正此一錯誤。

這種方式我們目前認為應該可以符合 BY-SA 的授權規定，因為 BY-SA 的授權當中有下列描述：

「彙編」指文學或藝術創作的合集，例如百科全書及詩文選集，表演、錄音物或廣播，或除列舉於後述第1條第(h)項所列之著作以外的其他著作或保護標的。由於其內容之選擇與編排具有智慧創作性，且本著作於其中是以未經修改的完整形式，與一個或更多的作品，彼此間成為分離且獨立之著作，而共同彙集成的完整合集。為本授權條款之目的，構成彙編的著作，不會被視為改用作品（定義如上）。

而本雜誌基本上是一堆文章的彙編，因此我們將藉由此一方式修正前兩期的授權錯誤！

本期開始，「程式人雜誌」將採用與維基百科相容的 [姓名標示](#)、[相同方式分享](#) 授權，然後請投稿者盡可

能用此一授權進行投稿，如果投稿者仍然覺得「非商業性」是必要條件的話，那我們將在該篇文章中單獨標示授權，以便盡可能的符合維基百科的授權規定。

筆者的疑惑

不過，筆者百思不解的一個問題是，為何 Lawrence Lessig 要用 Share-Alike 這個條件，將相容性鎖死，而不改用像 Share-Publicly (公開分享, SP) 這樣的條件呢？因為如果用 SP 取代 SA 的話，整個授權矩陣不就可以變成如下形式，授權間的相容性就會提高不少了阿！

原作\改作	BY	BY-NC	BY-NC-ND	BY-NC-SP	BY-ND	BY-SP
BY	○	○	○	○	○	○
BY-NC		○	○	○		
BY-NC-ND						
BY-NC-SP			○	○		
BY-ND						
BY-SP			○	○	○	○

由於筆者並非法律專家，也很難對「著作權法」目前的「合理使用」之範圍有明確的判斷力，但我們非常努力的想遵守「著作權法」與「創作共用」的規定，如果本雜誌在授權上仍有侵權問題的話，請來信告知，我們會盡可能符合法律規定。

另外、筆者希望知道這個問題答案的人，可以來信或留言告知上述問題的答案，以解筆者之惑阿！【本文由陳鍾誠撰寫】

書籍短訊：電腦類的免費電子書網站

FreeComputerBooks 是筆者很常用來下載免費電子書的網站，這個網站其實是一個資訊類電子書的入口網站，經營者把他所找得到的資訊類電子書連結都蒐集在此，以下是該網站的網址。

- <http://freecomputerbooks.com/>

以下是該網站的入口畫面，網站中包含了很詳細的分類，如果您需要合法的免費電腦書籍，可以考慮從

FreeComputerBooks.com

Free Computer, Mathematics, Technical Books and Lecture Notes, etc.

[Home](#)[All](#)[Mobile](#)[Math](#)[C](#)[C++](#)[C#](#)[Java](#)[JS](#)[Python](#)[Web](#)[Windows](#)[Linux](#)[DB](#)[Network](#)

Book Categories

FOLLOW ME ON [twitter](#)

- [All Categories](#)
- [Recent Books](#)
- [IT Certificates Studies](#)
- [IT Guides and Trainings](#)
- [IT Technical Documents](#)
- [Books by O'Reilly®](#) COOL
- [Other Free Book Sites](#)
- + [Computer Languages](#)
- + [Computer Science](#)
- + [Databases, SQL, Data Analysis and Mining](#)

Your request is forbidden

Sorry, but the URL you'r requesting is prohibited!

Search Results

What's **Big Data**? What's the current trends of it? Check [here](#).

約有 1,440 項結果 (搜尋時間：0.22 秒)



O'Reilly® Programming Pig - Free Computer Books

This book is an ideal learning tool and reference for Apache Pig, the programming language that helps you describe and run large data projects on Hadoop.

freecomputerbooks.com/Programming-Pig.html

FreeComputerBooks 網站的畫面

另外、Freetechbooks 也是一個同性質的網站，網址如下：

- <http://www.freetechbooks.com/>

有了這兩個網站，相信程式人會很容易的找到您要的英文免費電子書。(不過可惜的是、中文的世界筆者還沒看到類似的入口網站，如果有讀者知道請記得告訴我!)【本文由陳鍾誠撰寫】

網路短訊-MOOC 大量線上教育

最近 MOOC (Massive Open Online Course) 突然變成了一個熱門名詞，翻譯成中文就是 (大規模開放線上課程)。數位時代雜誌與商業週刊都不約而同用這個主題當作封面故事，可見這個名詞的火紅程度真的很高。

2002 年 MIT 提出 Open Course Ware (OCW) 開放課程之後，很多學校都跟進採用 OCW 的方式釋出教材。在台灣、交通大學大概是最早採用 OCW 的學校，後來台大等許多學校也陸續跟進了，以下是台灣 OCW 的相關網址。

學校	OCW 網址
交通大學	http://ocw.nctu.edu.tw/
台灣大學	http://ocw.aca.ntu.edu.tw/
清華大學	http://ocw.nthu.edu.tw

中山大學	http://ocw.nsysu.edu.tw/
臺灣師範大學	http://ocw.lib.ntnu.edu.tw/
政治大學	http://ocw.nccu.edu.tw/
輔仁大學	http://ocw.fju.edu.tw/
南台科技大學	http://ocw.stust.edu.tw/

OCW 的特色是把學校老師的教材上傳，這些教材通常包含「講義、投影片、授課錄影、指定作業」等等，這種由學校主動要求老師將教材有系統性上傳做法在當年也算是一大進步，讓很多人受惠良多 (包含我在內)。

但是如果您看過這些課程，可能會發現自己很難從頭到尾看完，因為要把一門課從頭到尾看完，可能得每次看好幾個小時，然後看個十幾二十次，這對於網路世代講求速效的人們而言，真的很不容易。

MOOC 可以說是在 OCW 的基礎上更進了一步，主要特色是必須在網路上登錄註冊才可以開始上課，課程內容、編排與呈現方法都與 OCW 有所差別，教師不再是將上課從頭到尾進行錄影下來，而是專門針的一個主題，進行 10 分鐘左右的講解，然後會隨即附上一兩個作業或測驗，讓學員能夠透過該作業檢驗該主題是否已經學會了。由於需要登入才能上課，因此這些作業都可以直接在網路上作答。

← → ↺ https://www.edx.org/courses/MITx/6.002x/2012_Fall/courseware/Week_1/Administrivia_ar ☆

Diigolet 翻譯精靈 五行健康操 - YouTu... 翻譯精靈 http://hilite.me/ 翻譯精靈 中華電信客戶服務首... >> 其他書籤

Courseware Course Info Textbook Discussion Wiki Progress

Overview

Week 1

Administrivia and Circuit Elements
Lecture Sequence

Circuit Analysis Toolchest
Lecture Sequence


Week 1 Tutorials

Homework 1
Homework due September 16, 2012

Lab 1
Lab due September 16, 2012

S1V1: MOTIVATION FOR 6.002X

6.002x is Exciting!



What's behind this?

You will learn, what, all of fun things in this course. So, for example, you will learn what's behind this What's behind the iPhone

What are some of the foundational technology and

cool ideas that drive this and many other things in our lives

圖、筆者的在 edX 上 MIT 電子電路課程的網頁畫面

如果你寫不出作業，想找同學討論，沒關係，課程上都附有討論區，讓同一門課的學員可以互相討論，充分理解之後再進行作答也可以。

那麼、這些作業與測驗由誰來批改呢？如果是選擇題的話，當然就可以由電腦直接批改沒有問題，但是如果是問答、數學證明或程式設計題的話，這些線上課程動不動就有五萬人同時登入上課，那麼老師一個人要改五萬份考卷不是改到手斷掉，於是有些 MOOC 平台會要求學員每個人要負責改隨機選出的 3-5 人作業，這樣老師的手就不會斷掉了 (但是學生的手會很痠！)。

當然、在這樣的機制上，作弊也是有可能的，但是這些課程登入時都會要求你按下誠實條款。畢竟、不誠則無物 (這不知道是哪個偉人所說的，我也忘了！)

Exercise Set 3
graded exercises due
February 27 23:30

available options.

: 5.0 POINTS

Each part is worth 1 point.

a) income

☐ categorical

☐ qualitative ordinal

☐ quantitative

☐ quantitative discrete

b) score on a test where each answer gets 5 points if it is right, and 0

圖、筆者在 edX 上 MIT 統計學測驗的網頁畫面

MOOC 就是將這種「教學短片 => 作業測驗 => 討論作答」的方式，整合成一個網站，所有的學習都在這個網站上完成，這可以說是一種新型的教育模式，或許不久之後台灣也會有學校開始嘗試這種做法也說不定。

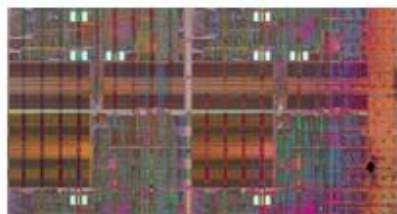
目前筆者所知道的 MOOC 平台網站有三個，分別是 Coursera, Udacity, edX，網址如下所示。

MOOC 平台 網址	
Coursera	https://www.coursera.org/
Udacity	https://www.udacity.com/
edX	https://www.edx.org/

這些網站的背後，其實也都有相對應的學校在支援，而且最近開始也吸引了不少資金的投入，或許真的會形成一股教育新勢力也說不定。

更棒的是，當您修完課之後，只要付一些證書的費用 (通常一兩百美金)，就可以取得證明書。當然、您得先通過這些測驗，至少要及格才行。

以下就是筆者慘痛的經驗，上去註冊之後每天都曠課，測驗完全沒有做，因此筆者在 MIT 的電子學課程上，竟然以零分的成績，被死當了，有圖為證：



MITx

Course Completed - Dec 25, 2012

6.002x Circuits and Electronics

Your final grade: **0%**. Grade required for a certificate: **60%**.

[View Archived Course](#)

[Unregister](#)

圖、筆者在 edX 上 MIT 電子電路課程被死當的畫面

雖然說這些證明書目前在台灣還沒有甚麼法律效用，不能拿來當作大學畢業證書，但是能拿到一張「MIT、史丹佛」等名校的某某課程證書，真的是酷炫到不行。這樣、不用出國也能到名校留學了！【本文由陳鍾誠撰寫】

程式人介紹

- 開放原始碼之父-Richard Stallman
- LLVM編譯器基礎架構的靈魂人物-Chris Lattner
 - 參考文獻

開放原始碼之父 - Richard Stallman

Richard Stallman 是開放原始碼領域最知名的人物之一，也是創造整個開源文化的始祖，他於 1983 年創建了 GNU 計畫，想建立一個類似 UNIX 的作業系統，雖然這個作業系統卻遲遲未能完成，但是 GNU 卻創造出了完成這個作業系統的大部分工具，像是 gcc, gdb, glibc, Emacs 等等，這讓後來的 (Linus Torvalds) 能夠在此基礎上創造出 Linux。



圖、Richard Stallman 2010 年的照片

另外、Stallman 也創造出 GPL (GNU General Public License) 這個重要的開放原始碼授權，並且找了律師一起制定相關的法律條款，位開放原始碼的發展奠定了法律基礎。2001 年時，Lawrence Lessig 也是受到這類授權的啟發而創造出了 Creative Commons 創作共用授權，讓開放原始碼文化擴散到一般領域。

根據 Stallman 的傳記，他是因為想要改良一台 Xerox 9700 的雷射印表機的功能，槓上了全錄公司，由於該印表機沒有附上驅動程式的原始碼，並且拒絕 Stallman 等人的索取程式碼，所以才引發了他推展開放原始碼運動的想法。

這個故事告訴我們，有時候毫不起眼的小事，會改變你的一生，然後讓你變成偉人！

【本文由陳鍾誠取材並修改自維基百科】

LLVM編譯器基礎架構的靈魂人物-Chris Lattner

前一篇文章中，我們介紹了開放原始碼之父 Richard Stallman，他所創造的 GNU 組織，設計出了 gcc 的整套編譯器與工具鏈，對開放原始碼世界造成非常深遠的影響。現在、我們將介紹 Apple 所採用新一代編譯器 LLVM 的重要靈魂人物：Chris Lattner。

2000年，Chris Lattner 自奧勒岡州波特蘭大學（University of Portland）計算機科學系畢業後，進入伊利諾伊大學厄巴納-香檳分校 (University of Illinois at Urbana-Champaign) 就讀。在此期間，他與維克拉姆·艾夫（Vikram Adve）發起 LLVM 專案，並於2003年發表。

LLVM 的全名是 Low Level Virtual Machine，因為 LLVM 的設計的初始目標是為所有靜態及動態語言創造出動態的編譯技術，但後來 LLVM 不斷的擴大之後，專案範圍已經不再侷限於虛擬機，成為以編譯器為核心的一個基礎架構，包含編譯、連結、除錯等等相關工具，其原始碼採用 BSD 的授權協定。

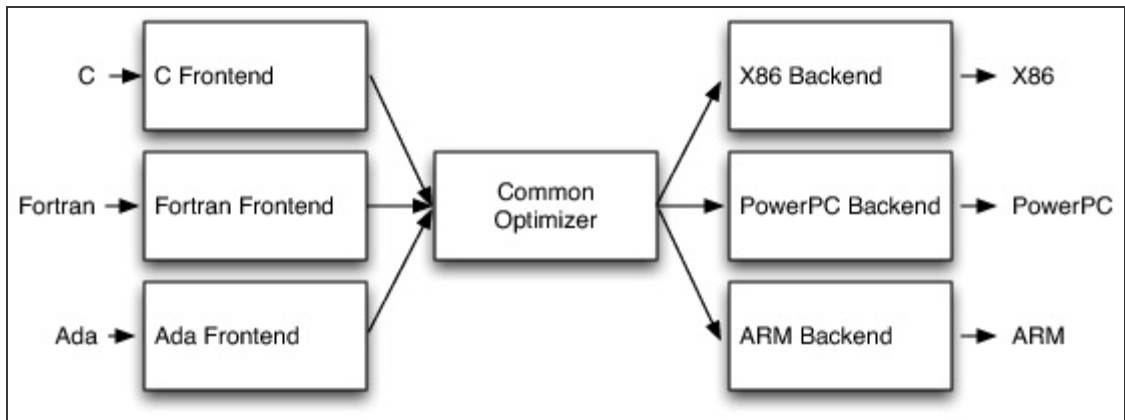
現在 LLVM 已經單純成為一個縮寫標記，適用於 LLVM 底下的所有專案，包含 LLVM 中介碼(LLVM IR)、LLDB 除錯工具、LLVM C++ 標準函式庫等等。

2005年，蘋果公司僱用了 Chris Lattner 及他的團隊，目的是將 LLVM 帶入蘋果公司，創立編譯器團隊並改進蘋果系統的速度與品質，並且同時啟動了 Clang 這個子計畫，負責改進蘋果系統的主要開發語言 Objective-C (同時支援 C/C++)、將 LLVM 納入到蘋果開發工具 Xcode 中，並且開始孕育並培養 LLVM 的開放原始碼社群。

2010 年，ACM 組織的 SIGPLAN 團體頒贈了「程式語言軟體獎」(Programming Languages Software Award) 給 Chris Lattner，以表彰他在 LLVM 上的貢獻，

LLVM 目前支援 Ada、C、C++、D、Fortran 及 Objective-C 等語言，也有人寫了 Haskell 的 LLVM 編譯器。現在、LLVM 已經成為 GNU 編譯器工具鏈 gcc 的一個競爭方案。而且、由於 gcc 已經是個歷史悠久的專案了，相對的很多程式碼也會比較陳舊而難以維護，因此 LLVM 或許能取代 gcc 而成為未來開放原始碼領域的主要編譯工具也說不定，因為 LLVM 的架構感覺上比 gcc 更清楚，更容易維護。

LLVM 編譯器基本上屬於彈性的三層式編譯器架構，基本上可分為 Frontend (前端)、IR 與最佳化、Backend (後端) 等幾部分，這種方式可以有效的將「高階語言」與「目標語言」兩者分離開來，形成一個多對多的架構，會比較容易就可以加入新的高階語言，或者產生新的目標語言，此種架構如下圖所示。



圖、典型的三層式編譯器架構

在編譯器領域，已經有不少人開始高度注意 LLVM 這個專案，並且實際在各個領域，您可以在 <http://llvm.org/docs/> 當中找到 LLVM 的相關技術資訊。

特別值得一提的是，筆者正在進行的「開放電腦計畫」這個企圖自行建立「CPU、Assembler、Compiler、OS、虛擬機」的計畫中，原本進度稍嫌緩慢，但是在我大哥決定為此計畫的處理器 CPU0 寫一個 LLVM Backend 程式，並寫出了 [Tutorial: Creating an LLVM Backend for the Cpu0 Architecture](#) 這本書籍之後，就往前邁進了很多。有了 LLVM Backend 的技術，就讓我們有機會可以用標準的 C 語言將現成的小型作業系統，像是 UNIXv6、MINIX 或 RTThread 等作業系統，移植到 CPU0 上，以便完成整個計畫了。

參考文獻

- Chris Lattner's Homepage – <http://www.nondot.org/sabre/>
- Chris Lattner – <http://www.techcn.com.cn/index.php?doc-view-148143.html>
- LLVM 2.0 – <http://www.youtube.com/watch?v=029YXzHtRy0>
- Mac OS X 背后的故事（八）三好学生Chris Lattner的LLVM编译工具链 – <http://www.programmer.com.cn/9436/>
- <http://zh.wikipedia.org/wiki/LLVM>
- http://en.wikipedia.org/wiki/Chris_Lattner
- LLVM, Chris Lattner – <http://www.aosabook.org/en/llvm.html>

【本文由陳鍾誠取材並修改自維基百科】

程式人頻道

- 看影片學 C# 視窗程式設計
- C# 基礎程式設計
- C# 視窗程式設計 – 使用 Windows Form 技術
- C# 網路程式設計 – 使用 Socket 函式庫

看影片學 C# 視窗程式設計

C# 是微軟平台的主力語言，用來寫 Windows 系統的程式非常好用，也是很多學生學習視窗程式設計的捷徑。

但是在學習視窗程式設計的時候，只看書往往會覺得有點學習障礙，這是因為 C# 的開發工具 Visual Studio 實在設計得太好的關係。

Visual Studio 我們可以用視覺化拖拉的方式，輕易的進行視窗界面設計，這些拖拉動作所產生的界面程式碼數量很多，我們通常很難直接用打字的方式設計，於是 C# 的視窗程式就和 Visual Studio 這個開發工具緊密的結合在一起了。

傳統的紙本書籍很難傳達這種視覺化界面操作的訊息，於是很多人買了一堆 C# 視窗程式設計書籍後，還

是沒有辦法學會視窗程式。

針對這個問題，筆者的建議是利用書籍搭配教學影片一起學習，這樣既能傳達與程式相關的文字訊息，又能傳達與視覺化操作的影音訊息，兩者適當的搭配就能比較有效率的學會視窗程式設計了。

筆者自己在金門大學中教授視窗程式設計也有數年了，為了讓學生學會這個領域，我用自己熟悉的 `wikidot` 創建了一個針對 C# 而設計的網站，網址是 <http://cs0.wikidot.com>，如果您想學習 C# 視窗程式，建議您可以從這裡開始出發。

在網站中，我們從基本的 C# 語法開始介紹、然後進入物件導向、接著才進入函式庫、檔案與 Windows Form 視窗程式，對於這些主題，筆者幾乎都會將自己上課的內容進行全程錄影，初學者如果能從頭開始閱讀這些文章並觀賞影片，應該可以很容易的進入 C# 視窗程式的領域才對。

C# 基礎程式設計

主題	教學影片
C# 命令列編譯器 <code>csc</code>	http://youtu.be/biz51oMWRKk
C# Visual Studio Express 安裝與第一個程式	http://youtu.be/d7Hi4BOFmj8

C# 基本型態宣告與觀察	http://youtu.be/T81pMd32tMY
C# 的基本運算操作	http://youtu.be/yMOJjz7VmaE
C# 運算式家庭作業	http://youtu.be/JTUTdZjLBxI
C# 的 if 語法	http://youtu.be/qM1DpkVqEkQ
C# 陣列與迴圈	http://youtu.be/dNGpHr2Ycxk
C# 迴圈 for, while 詳細講解	http://youtu.be/Wk-RZvjgG2o
C# 的函數呼叫	http://youtu.be/dnkKId0ZO1c
C# 的物件導向/封裝	http://youtu.be/HLNAbXHSQHs
C# 的物件導向/建構函數	http://youtu.be/rS7osPMWJKE
C# 的物件導向/繼承 1	http://youtu.be/pNkvMXp28AE
C# 的物件導向/繼承 2	http://youtu.be/GKxxE3UdGms
C# 的物件導向/多型	http://youtu.be/G1QQyXEpSCk

C# 物件導向練習題講解	http://youtu.be/Om50P0T_aVU
C# 的錯誤處理	http://youtu.be/7bCefu8BB00

C# 視窗程式設計 - 使用 Windows Form 技術

主題	教學影片
C# 文字型計算機	http://youtu.be/A1KVnrfVF7k
C# 實作字典查詢程式	http://youtu.be/yS3G-H_hrFU
C# Array 與 List 物件的使用	http://youtu.be/EHSGtKpRprI
C# HashTable 與 Dictionary 物件的使用	http://youtu.be/hYH-npRmmKM
C# 將小字典擴充為小翻譯系統	http://youtu.be/_sFsTo41PXs
C# Timer 與碼錶	http://youtu.be/UA0rizekLow

C# Timer 與小時鐘	http://youtu.be/NJ6B5-vO_88
C# Timer 與移動球	http://youtu.be/6Gs4MPzt6q4
C# 檔案處理	http://youtu.be/3EyPcAddd70
C# 文字編輯器 1	http://youtu.be/xymT54EI53E
C# 文字編輯器 2	http://youtu.be/xz5sKvZjLZI
C# 畫圖功能示範	http://youtu.be/8aAql8R4WSg
C# 小畫板 (2)	http://youtu.be/HkOkWRQ_Ad4
C# 小畫板 (3)	http://youtu.be/VXaQu_yYu08
C# 瀏覽器控制 1/3	http://youtu.be/CIwYabPN7qA
C# 瀏覽器控制 2/3	http://youtu.be/sJ6cfuL3-ZA
C# 瀏覽器控制 3/3	http://youtu.be/YThlDxk-E7U
C# DataGridView 元件的使用	http://youtu.be/XzsZRqLAi30

C# 製作賣紅茶的 POS 系統	http://youtu.be/XX0wHhvkkiE
------------------	---

對於已經熟悉微軟程式設計的人而言，可能會發現我們使用的是較舊的 Windows Form 視窗架構，而非較新的 WPF (Windows Presentation Foundation) 架構，因為筆者並沒有採用 WPF 進行課堂教學的關係，不過還好，「曹祖聖」先生已經為微軟錄製了一系列的 WPF 的影音開發教材，如果您有興趣，可以到以下網頁中觀賞學習。(由於該網頁的影音採用微軟的 Silverlight，因此您必須先安裝 Silverlight 虛擬機後才能觀賞這些影片)

主題	影片網址
MSDN 教學短片 - WPF	http://msdn.microsoft.com/zh-tw/netframework/cc963622

C# 網路程式設計 - 使用 Socket 函式庫

主題	教學影片
C# 網路 UDP 程式設計	http://youtu.be/DxdfeURkNzU
C# 設計簡易的 WebServer	http://youtu.be/Fol-ZLQ2GKA

C# 多人聊天室	http://youtu.be/pMUw-NIm9yE
C# 網路爬蟲 (Crawler)	http://youtu.be/eYwu0ElaOIY

程式人文集

- Arduino入門教學(3) – LED 控制實驗 (作者：Copper Maa)

- Lab1 Blinking a LED

- 實驗目的
 - 材料
 - 接線
 - 電路圖
 - 程式碼
 - 說明：
 - 範例照片／影片

- 實驗二：使用按鍵控制 LED 燈號

- 實驗目的
 - 材料
 - 接線
 - 電路圖
 - 程式碼
 - 說明：
 - 範例照片／影片
 - 動動腦

■ 說明：

- JavaScript (3) – Closure 與匿名函數 (作者：陳鍾誠)
 - 參考文獻
- R 統計軟體：(1) 簡介與基本操作 – (作者：陳鍾誠)
 - 簡介
 - 安裝
 - 基本操作
- 從 C# 看作業系統：(1) Thread 與 Deadlock 實作 – (作者：陳鍾誠)
 - Process 與 Thread
 - C# 中的 Thread 概念
 - 以 C# 體驗 Deadlock 死結

Arduino入門教學(3) – LED 控制實驗（作者：Copper Maa）

Lab1 Blinking a LED

實驗目的

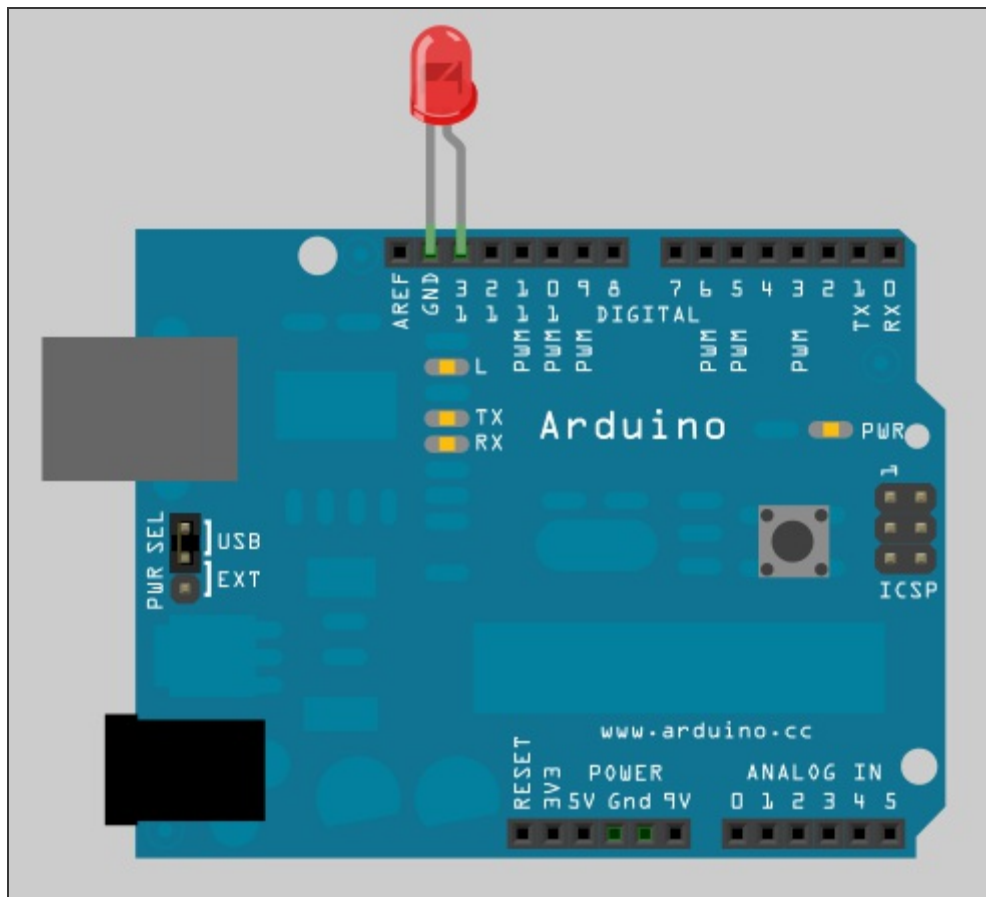
讓一顆燈號閃爍，每隔一秒切換一次燈號。

材料

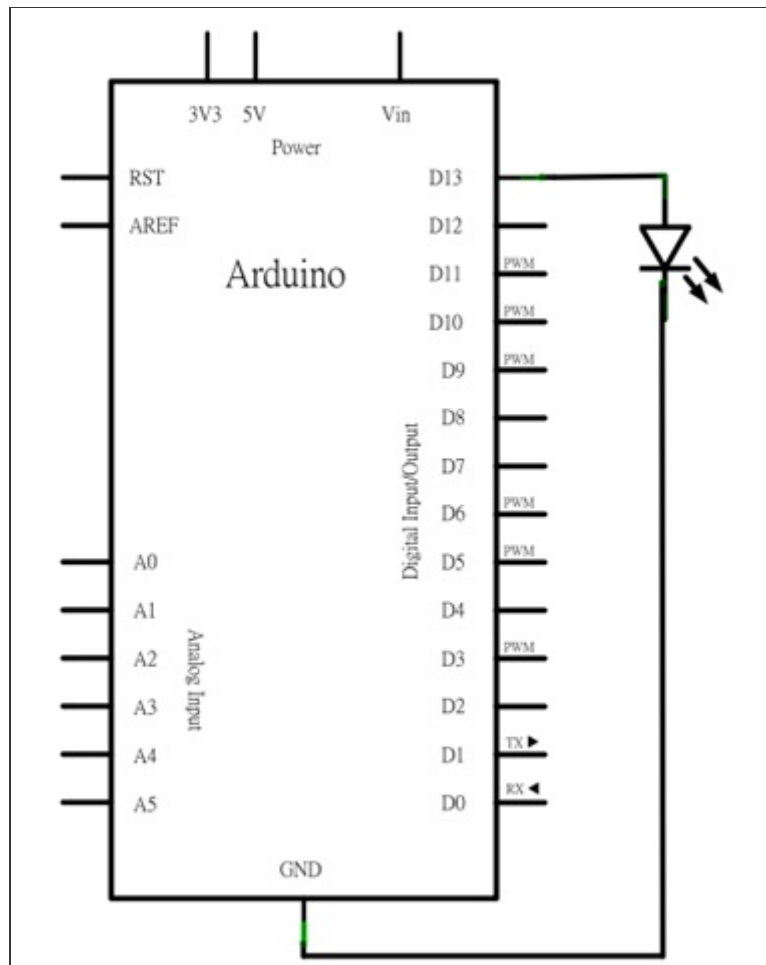
- Arduino 主板 x 1
- LED x 1

接線

- 把 LED 接到 Arduino 板子上，LED 長腳(陽極)接到 pin13，短腳(陰極)接到 GND，如下圖：



電路圖



如果你有電子背景，你可能注意到了一件事，我們的 LED 沒有串接電阻，這是因為 Arduino 的輸出電流很小不會燒壞 LED，所以才敢這麼做。少接一顆電阻是為了簡化，這樣初學者會學得比較輕鬆。一般來說，LED 串接一顆電阻是個好主意。

程式碼

```
/*  
Blink  
Turns on an LED on for one second, then off for one second, repeatedly.  
  
This example code is in the public domain.  
*/  
  
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(13, HIGH); // set the LED on  
  delay(1000);           // wait for a second
```

```
digitalWrite(13, LOW); // set the LED off
delay(1000);           // wait for a second
}
```

說明：

- L11: pinMode(13, OUTPUT) 這行把 pin13 設置成 output pin
- L15: digitalWrite(13, HIGH) 這行供應 5V 電壓到 pin13，藉此打開 LED 燈號
- L16: delay(1000) 讓 CPU 閒置一秒鐘，讓 LED 燈號亮著一秒鐘
- L17: digitalWrite(13, LOW) 關閉燈號
- L18: delay(1000) 讓 CPU 閒置一秒鐘，讓 LED 燈號關閉一秒鐘

註：這支是 Arduino 內建的範例程式，點選 File > Examples > 1.Basics > Blink 就可以找到。

範例照片／影片

Youtube 上正好有段講解 Blinking a LED 的影片，我們來看看他的示範：

影片名稱	網址
How-to Tuesday: Arduino 101 the LED	http://youtu.be/pMV2isNm8JU

實驗二：使用按鍵控制 LED 燈號

實驗目的

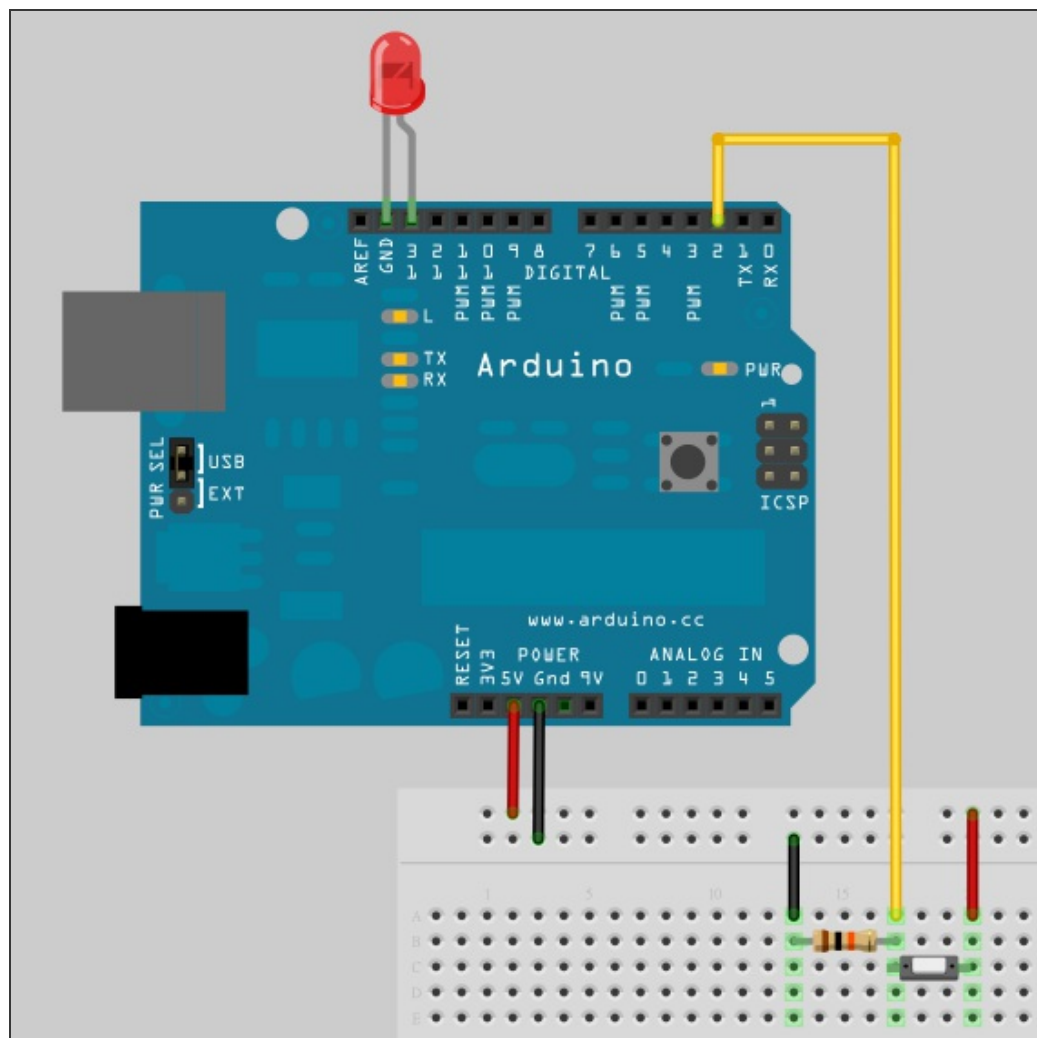
使用按鍵 (PushButton)控制 LED 燈號的開關，當按鍵被按下時打開 LED 燈號，按鍵放開時關閉 LED 燈號。

材料

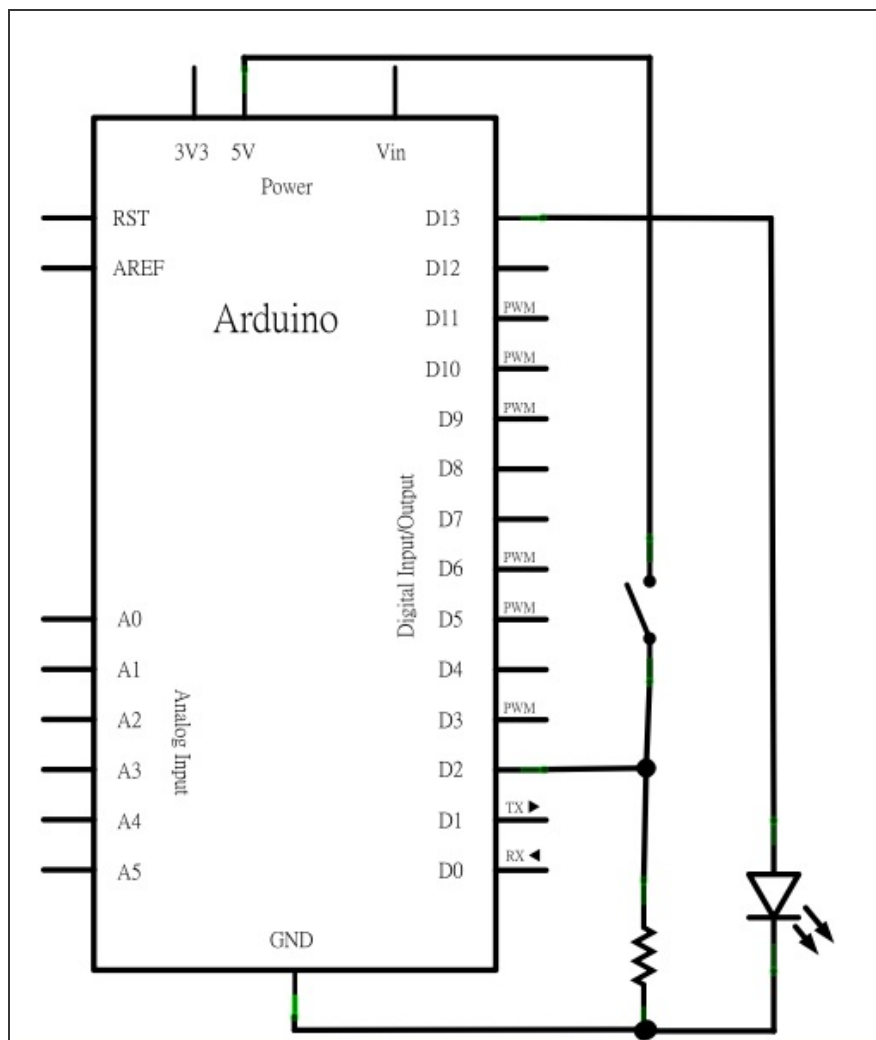
- Arduino 主板 x 1
- LED x 1
- PushButton 或 Switch 開關 x 1
- 10K 電阻 x 1
- 麵包板 x 1
- 單心線 x N

接線

- 把 LED 接到 pin13，長腳(陽極)接到 pin13，短腳(陰極)接到 GND
- PushButton 一支腳接到 +5V
- pin2 接到 Pushbutton 的另一支腳，同一支腳位接一個 10K 電阻連到 GND



電路圖



程式碼

```
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin = 13;      // the number of the LED pin

// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status

void setup() {
    // initialize the LED pin as an output:
    pinMode(ledPin, OUTPUT);
    // initialize the pushbutton pin as an input:
    pinMode(buttonPin, INPUT);
}

void loop(){
    // read the state of the pushbutton value:
    buttonState = digitalRead(buttonPin);

    // check if the pushbutton is pressed.
```

```
// if it is, the buttonState is HIGH:
```

```
if(buttonState == HIGH) {
```

```
// turn LED on:
```

```
  digitalWrite(ledPin, HIGH);
```

```
}
```

```
else {
```

```
// turn LED off:
```

```
  digitalWrite(ledPin, LOW);
```

```
}
```

```
}
```

說明：

- L01~L02: 定義按鍵與 LED 的腳位，按鍵接在 pin2, 而 LED 接在 pin13
- L16: 讀取按鍵的狀態並保存到 `buttonState` 變數裏
- L20~L26: 這支程式的目的是按下按鍵時要打開 LED 燈號，放開按鍵時要關閉 LED 燈號，因此，假如 `buttonState` 為 HIGH，代表按鍵狀態是按下(pressed)的，此時要打開 LED，反之，假如 `buttonState` 為 LOW，代表按鍵狀態是放開的，此時要關閉 LED

註：這支是 Arduino 內建的範例程式，點選 File > Examples > 2.Digital > Button 就可以找到。

範例照片／影片

Youtube 上正好有段講解使用按鍵控制 LED 的影片，我們來看看他的示範：

影片名稱	網址
How-to Tuesday: Arduino 101 the button	http://youtu.be/XUuXq4J4u14

注意！影片所用的範例程式，它是按下按鍵時關閉 LED 燈號，放開按鍵時打開 LED 燈號，邏輯與本文的範例相反。

動 動 腦

在不修改程式碼的條件下，讓 LED 變成在正常情況下是亮的，而當按下按鍵時讓 LED 燈號關掉。提示：條件是不能修改程式碼，要改變這支程式的行為，你只能夠動接線。

說明：

- L01~L02: 定義按鍵與 LED 的腳位，按鍵接在 pin2, 而 LED 接在 pin13
- L16: 讀取按鍵的狀態並保存到 `buttonState` 變數裏
- L20~L26: 這支程式的目的是按下按鍵時要打開 LED 燈號，放開按鍵時要關閉 LED 燈號，因此，假如 `buttonState` 為 HIGH，代表按鍵狀態是按下(pressed)的，此時要打開 LED，反之，假如 `buttonState` 為 LOW，代表按鍵狀態是放開的，此時要關閉 LED

註：這支是 Arduino 內建的範例程式，點選 File > Examples > 2.Digital > Button 就可以找到。

(本文作者為馬萬圳，原為網誌上的兩篇文章，經作者授權給程式人雜誌後由陳鍾誠編輯為此文，原文連結如下 <http://coopermaa2nd.blogspot.tw/2010/12/arduino-lab1-blinking-led.html>,
<http://coopermaa2nd.blogspot.tw/2010/12/arduino-lab2-led.html>)

JavaScript (3) - Closure 與匿名函數 (作者：陳鍾誠)

在以下的程式中，我們同時展示了 JavaScript 當中的全域變數、區域變數與匿名函數、Closure 等機制。在本文中，我們會用這個程式逐步解釋一些 JavaScript 語言中重要但詭異的概念。

很多語言都有全域變數與區域變數之分，在 JavaScript 當中看來也是如此，但事實上 JavaScript 的全域變數其實只是最外層領域的區域變數而已，在瀏覽器當中，這個最外層領域就是 window，所以以下程式中最後一部分的 sum 與 windows.sum 都同樣是 8。

程式：closure.htm

```
<html>
<body>

<script type="text/javascript">
var sum = 0;
function add(x) {
```

```
sum += x;
document.write("typeof:x="+typeof x+" y="+typeof y+" z="+typeof z+" sum="+typeof sum+"<br/>");
var y = 2;
return function() {
    var z = x+y;
    document.write("value :x="+x+" y="+y+" z="+z+" sum="+sum+"<br/>");
    document.write("typeof:x="+typeof x+" y="+typeof y+" z="+typeof z+" sum="+typeof sum+"<br/><br/>");
}
}
```

```
f = add(3);
```

```
f();
```

```
add(5());
```

```
document.write("typeof:x="+typeof x+" y="+typeof y+" z="+typeof z+" sum="+typeof sum+"<br/><br/>");
```

```
</script>
```

```
<script type="text/javascript">
```

```
document.write("sum="+sum+" window.sum="+window.sum);
```

```
</script>
```

```
</body>
```

```
</html>
```

因此、JavaScript 可以說沒有全域變數的概念，全域變數的只不過是最外層物件的區域變數罷了。

在 JavaScript 每一層領域當中，都可以定義區域變數，例如在上述範例中，`sum` 是最外層變數，因此所有的程式區段都可以存取這個變數。

參數也是一種區域變數，像是 `add(x)` 當中的 `x`，也有領域概念，其作用範圍僅適用於 `add` 裏面。而 `add` 當中所定義的區域變數 `y`，則只有在定義之後才會生效，因此其領域範圍是從 `var y=2;` 這行程式開始，一直到 `add` 函數結束為止。

JavaScript 當中的函數，可以沒有名稱，這種函數稱為匿名函數，像是上述範例的函數 `add` 中，就傳回了一個匿名函數，如下所示：

```
function add(x) {  
    sum += x;  
    document.write("typeof:x="+typeof x+" y="+typeof y+" z="+typeof z+" sum="+typeof sum+"<br/>");  
    var y = 2;  
    return function() {
```



```
var z = sum+x+y;  
document.write("value :x="+x+" y="+y+" z="+z+" sum="+sum+"<br/>");  
document.write("typeof:x="+typeof x+" y="+typeof y+" z="+typeof z+" sum="+typeof sum+"<br/><br/>");  
}  
}
```

變數 `z` 是該匿名函數的區域變數，由於設定為 `sum+x+y`，因此存取了外層的 `y`、`x` 與更外層的 `sum`，領域的存取規則是內層可以存取外層的變數，但是外層卻不能存取內層的變數。

筆者認為，JavaScript 中的一個最特別且強大的特性，莫過於函數既可以塞給一般變數，也可以當參數傳遞。例如以下程式區段就是用 `f` 去接收 `add(x)` 函數所傳回來的匿名函數，然後再用 `f()` 函數去執行這個匿名函數而已。

```
f = add(3);  
f();  
  
add(5)();
```

上列程式碼的最後一行 `add(5)()`，只不過是將這兩個動作合起來一起作，也就是呼叫 `add(5)` 後，傳回值是個匿名函數，然後再用「匿名函數()」這樣的方法呼叫函數，去執行該匿名函數而已。(這個功能有點像 C 當中的函數指標)

在上述的 `closure.htm` 程式中，還有個很特別的地方，就是這些匿名函數被傳回來後，就已經回到了最外層的領域範圍了 (`window`)，那麼當該匿名函數執行時，照理說由於 `x, y` 屬於內層 `add` 函數的區域變數，應該會存取不到才對，但為何程式執行結果卻都還是正常，沒有發生錯誤呢？

這牽涉到 JavaScript 當中一個非常特別的機制，稱為 Closure (閉包)。

閉包與 JavaScript 的領域 (Scope) 特性有關，因為 JavaScript 採用 Lexical Scope，也就是變數作用範圍依照程式定義時的上下文 (Context) 所決定，而不是根據執行時期的上下文所決定的。

因此，上述匿名函數所服從的領域規則，仍然是定義時的領域，也就是 `add` 函數的子領域，而不是執行時期 `window` 的子領域，因此仍然可以正確存取 `x,y,z` 等變數，不會發生錯誤。

(JavaScript 的語法到此已經大致說明完畢，在下一期當中，筆者會將焦點轉移到 JavaScript 在瀏覽器中的運用上，看看 JavaScript 的實際用途，我們下期見！)

參考文獻

- Node.js 中文電子書 » JavaScript 與 NodeJS – http://book.nodejs.tw/zh-tw/node_javascript.html#scope-closure

R 統計軟體：(1) 簡介與基本操作 – (作者：陳鍾誠)

簡介

R 軟體是專門為了機率統計而設計的一種開放原始碼軟體，是免費的自由軟體。

市面上有許多與 R 類似的商用軟體，像是 SPSS, SAS, MINITAB, S-PLUS 等，但是這些軟體是要花錢買的。

R 軟體所使用的程式語言，被稱為 R 語言。

R 語言 與 S-PLUS 所使用的語言很類似，兩者都衍生自貝爾實驗室 Rick Becker, Allan Wilks, John Chambers 所創造的 S 語言，R 語言基本上是 GNU 所實作的 S 語言版本。

筆者篆寫此文時，R 所採用的 S 語言演化到了第四版，因此稱為 S4。

安裝

R 軟體的官方網站為 <http://www.r-project.org/>，其中有個相當重要的子網站稱為 CRAN (Comprehensive R Archive Network)，其網址為 <http://cran.r-project.org/>，您可以從這個網站中下載 R 軟體。

舉例而言，筆者使用的是 Windows 作業系統，因此可以從以下網址下載到最新版的 R 軟體。

- <http://cran.r-project.org/bin/windows/base/>

The screenshot shows a web browser window with the address bar displaying `cran.r-project.org/bin/windows/base/`. The page title is "Download R-2.15.2 for Windows (32/64 bit)". The main content area has a light gray background and contains the following elements:

- A link: [Download R 2.15.2 for Windows](#) (47 megabytes, 32/64 bit)
- A link: [Installation and other instructions](#)
- A link: [New features in this version](#)

Below these links, there is a paragraph of text:

If you want to double-check that the package you have downloaded exactly matches the package distributed by R, you can compare the [md5sum](#) of the .exe to the [true fingerprint](#). You will need a version of md5sum for windows: both [graphical](#) and [command line versions](#) are available.

Below the paragraph is a section titled "Frequently asked questions" with a bulleted list of links:

- [How do I install R when using Windows Vista?](#)
- [How do I update packages in my previous version of R?](#)
- [Should I run 32-bit or 64-bit R?](#)

At the bottom of the page, there is a paragraph of text:

Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for

舉例而言，筆者點選時為 Download R 2.15.2 for Windows 這個連結，這會下載位於下列網址的檔案：

- <http://cran.r-project.org/bin/windows/base/R-2.15.2-win.exe>

下載完畢後，請啟動該安裝檔，然後不斷按「下一步」就可以完成安裝了，過程非常簡單。

以下網址中的 Youtube 影片介紹了 R 軟體的下載、安裝、套件、網站、電子書等等，有興趣的朋友可以看看。

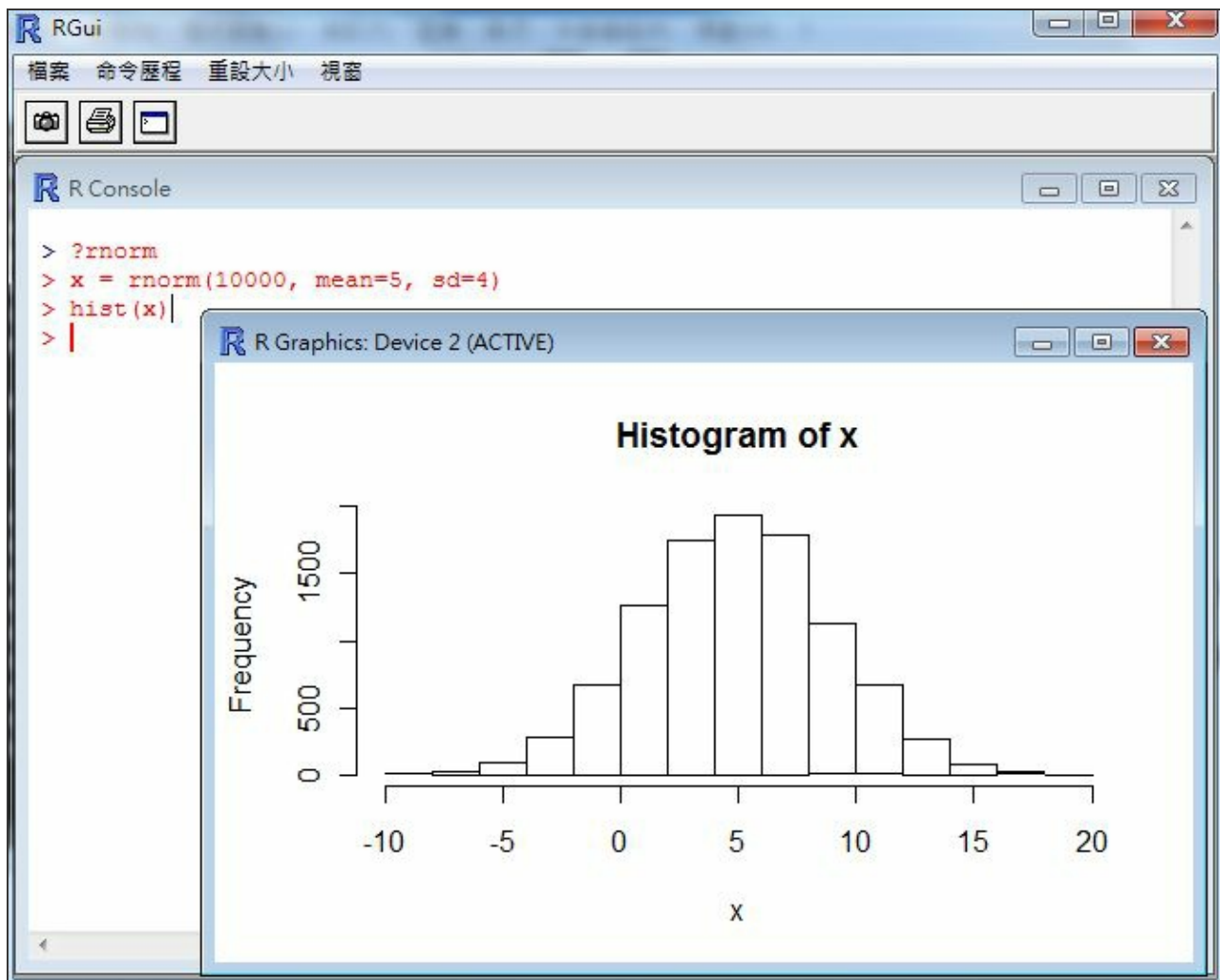
- <http://www.youtube.com/watch?v=AipnE4s8sKk>

基本操作

為了說明 R 軟體的用法，並用以學習機率統計的概念，本系列文章將運用 R 來說明機率統計的理論，讓程式人可以透過實作學會機率統計，並且學會 R 軟體中的 S 語言。

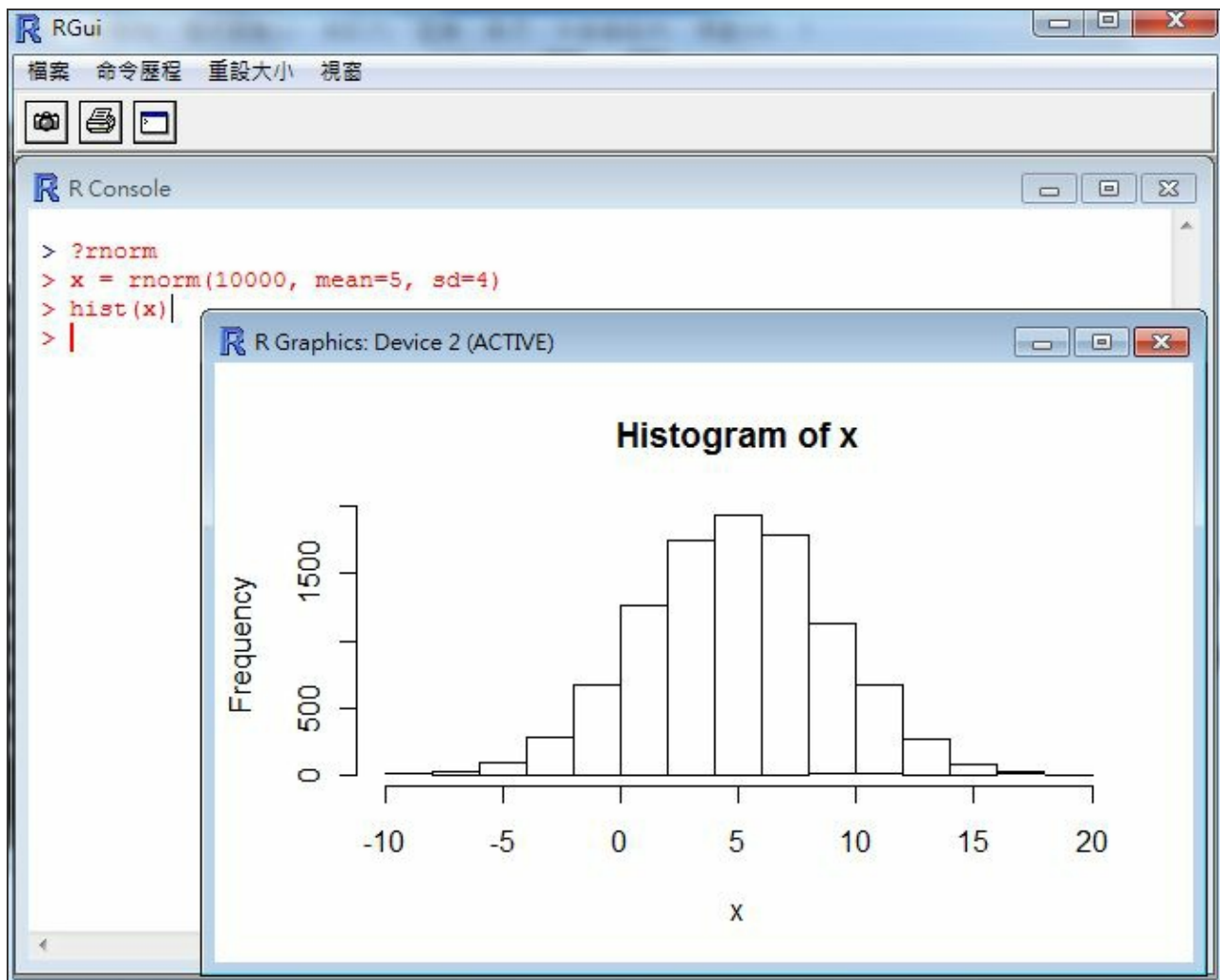
為了避免太過枯燥，我們將不會先介紹 R 的基本語法，而是先用一系列的操作，讓讀者體會 R 的能力，然後再慢慢回到語言的教學上面。

以下是筆者用 R 軟體取樣後會出樣本統計圖的畫面，簡單的幾個指令就可以得到統計結果，是不是很棒呢？



圖、R 軟體執行畫面

第一個指令 `?rnorm` 是要求 R 軟體查詢 `rnorm` 這個指令，R 軟體會顯示以下的說明網頁，您可以看到 `rnorm` 指令是與常態分部 (The Normal Distribution) 有關的。



圖、R 軟體的說明網頁

在 R 軟體中，對於任何一個機率分布 xxxx，都會實作出以 d, p, q, r 為字首的四種函數，例如對於常態分布 Normal Distribution (簡寫為 norm) 而言，就有 dnorm, pnorm, qnorm, rnorm 等四個函數，功能分別如下所示：

函數	說明	語法
dnorm	常態分布的機密度函數	dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm	常態分布的機分布函數	pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm	常態分布的分位數函數	qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm	常態分布隨機樣本函數	rnorm(n, mean = 0, sd = 1)

上表中的 mean 代表平均數，sd 代表 Standard Deviation (標準差)，n 是隨機產生的樣本個數，x 是隨機變數值，q 是累積值，p 是機率值，n 則是產生的樣本數。

您可以發現函數中，有些參數後面有 = 的指定 (像是 mean=0, sd=1, log=FALSE,)，有些卻沒有 (像是 x, q, p, n) 等，這些指定代表預設值，也就是如果您不指定這些參數的值，那麼將會自動代入預設值。

所以 `rnorm(100)` 代表 `rnorm(100, mean = 0, sd = 1)` 的意思，也就是該函數會產生平均數為 `mean=0`，標準差為 `sd=1` 的隨機樣本共 100 個。

關於這些函數的更詳細的說明如下表所示。

字首	函數意義	範例	說明
d	機率密度函數	<code>dnorm(1.96)</code>	$P(X=x)$
p	累積機率函數 (CDF)	<code>pnorm(1.96)=0.975</code>	$P(X\leq x)$
q	計算百分位數	<code>qnorm(0.975)=1.96</code>	q 系列為 p 系列的反函數; 所以 <code>qnorm(pnorm(1.96)) = 1.96</code>
r	抽樣函數	<code>rnorm(100)</code>	傳回 100 個標準常態分布的樣本向量

看懂這些函數之後，讓我們再度列出上圖的操作指令，仔細觀察看看每一個指令的意義。

```
?rnorm
```

```
x = rnorm(10000, mean=5, sd=4)
```

```
hist(x)
```

指令 `x = rnorm(10000, mean=5, sd=4)` 代表我們要用平均值為 5, 標準差為 4 的常態分布，隨機產生 10000 個樣本，然後將這些樣本存到 `x` 陣列當中。

指令 `hist(x)` 代表要用這些樣本畫出統計的直方圖 (Histogram)，於是就畫出了圖中的那個長條狀圖形。

現在、請讀者試著看看下列操作，看看您是否能夠讀懂這些操作的意義。

```
rnorm(10, 3, 2)
```

```
> x
```

```
[1] 2.5810213 0.5399127 5.0005020 5.3402693 2.7900723 3.9638088 5.2119685
```

```
[8] 2.2209882 2.9935943 7.0308419
```

```
> a=dnorm(1.96)
```

```
> a
```

```
[1] 0.05844094
```

```
> b=pnorm(1.96)
```

```
> b
```

```
[1] 0.9750021
```

```
> c=qnorm(b)
> c
[1] 1.96
> d=rnorm(10)
> d
[1] -0.32913677 0.77788306 -1.80862496 0.16694598 -0.65656254 -1.76305925
[7] 1.18237502 0.19651748 -0.07898685 0.73970933
>
```

在下一期當中，我們將會用 R 統計軟體，示範如何進行敘述統計的操作，並說明這些操作的意義，我們下期見！

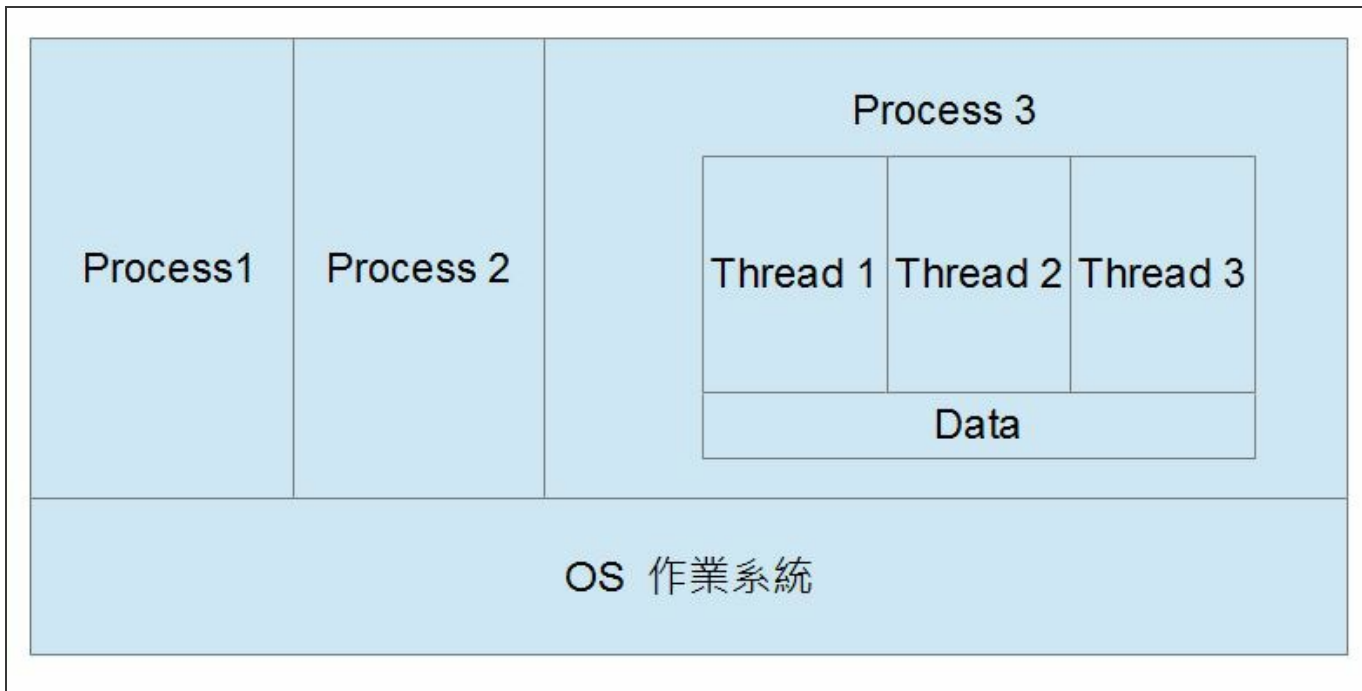
從 C# 看作業系統：(1) Thread 與 Deadlock 實作 - （作者：陳鍾誠）

很多資工系的學生都上過作業系統這門課，但是通常老師只有講理論，沒有說明如何實作，這讓很多同學都無法清楚的理解其中的概念。在本系列文章中，我們將使用 C# 語言來說明作業系統當中的一些關鍵概念，像是 Process (進程、行程、Task)、Thread (執行緒、線程)、Deadlock (死結)、Race Condition (競爭情況)、Semaphore (號誌、信號量) 等等。

Process 與 Thread

Thread 在台灣被稱為『執行緒』，但是在中國被稱為『線程』，作業系統教科書中通常會定義 Process 為：執行中的程式。因此假如您開了一個 Word 檔案，那就是有一個 Word 行程在執行，如果您又開了個命令列，那就是又有一個命令列行程在執行，如果又開第二個命令列，那就有兩個命令列行程在執行。

Thread 在作業系統中通常被定義為輕量級行程 (Light Weight Process)，一個 Process 可以包含很多個 Thread，如下圖所示：

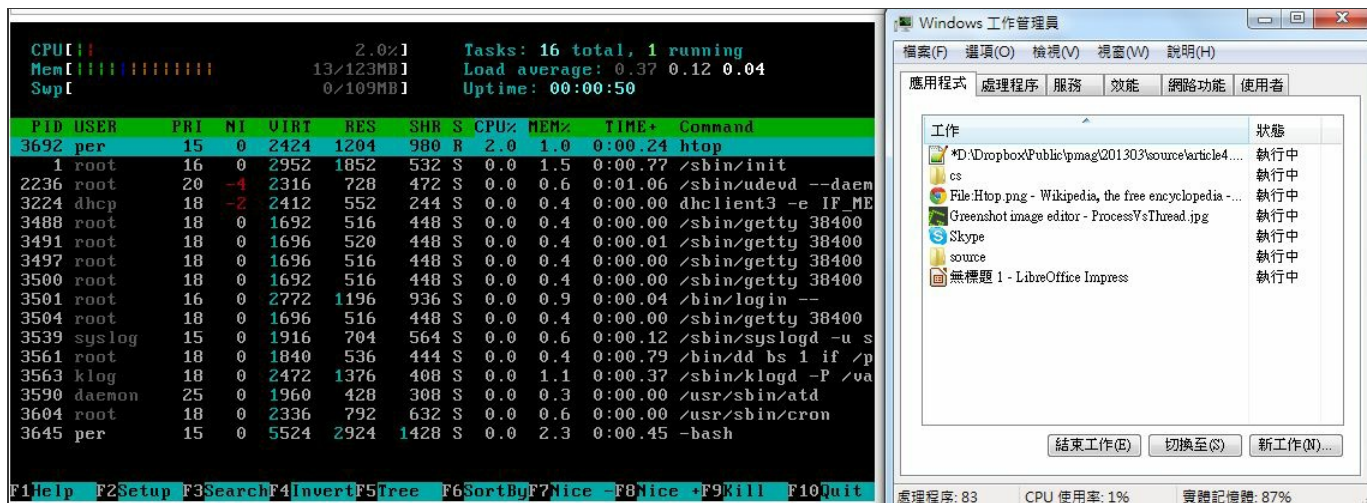


圖、Process 與 Thread 的關係

每個 Process 與 Thread 都會執行，而且執行到一半很可能就會因為進行輸出入或佔用 CPU 過久而被作業系統切換出去，改換另一個 Process 或 Thread 執行，這種概念稱為 **Multitasking** (多工)。

在 Windows 當中，我們可以按下 **Ctrl-Alt-Del** 鍵以顯示出系統的行程資訊，而在 Linux 中則可以用 **ps**

(Process Status) 這個指令顯示行程資訊，以下是這兩個作業系統中的行程資訊範例。



Linux Process Status Output:

```
CPU[ ] 2.0% Tasks: 16 total, 1 running
Mem[ ] 13/123MB Load average: 0.37 0.12 0.04
Sup[ ] 0/109MB Uptime: 00:00:50
```

PID	USER	PRI	NI	UIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
3692	per	15	0	2424	1204	980	R	2.0	1.0	0:00.24	htop
1	root	16	0	2952	1852	532	S	0.0	1.5	0:00.77	/sbin/init
2236	root	20	-4	2316	728	472	S	0.0	0.6	0:01.06	/sbin/udevd --daemon
3224	dhcp	18	-2	2412	552	244	S	0.0	0.4	0:00.00	dhclient3 -e IF_ME
3488	root	18	0	1692	516	448	S	0.0	0.4	0:00.00	/sbin/getty 38400
3491	root	18	0	1696	520	448	S	0.0	0.4	0:00.01	/sbin/getty 38400
3497	root	18	0	1696	516	448	S	0.0	0.4	0:00.00	/sbin/getty 38400
3500	root	18	0	1692	516	448	S	0.0	0.4	0:00.00	/sbin/getty 38400
3501	root	16	0	2772	1196	936	S	0.0	0.9	0:00.04	/bin/login --
3504	root	18	0	1696	516	448	S	0.0	0.4	0:00.00	/sbin/getty 38400
3539	syslog	15	0	1916	704	564	S	0.0	0.6	0:00.12	/sbin/syslogd -u s
3561	root	18	0	1840	536	444	S	0.0	0.4	0:00.79	/bin/dd bs 1 if /p
3563	klog	18	0	2472	1376	408	S	0.0	1.1	0:00.37	/sbin/klogd -P /va
3590	daemon	25	0	1960	428	308	S	0.0	0.3	0:00.00	/usr/sbin/atd
3604	root	18	0	2336	792	632	S	0.0	0.6	0:00.00	/usr/sbin/cron
3645	per	15	0	5524	2924	1428	S	0.0	2.3	0:00.45	-bash

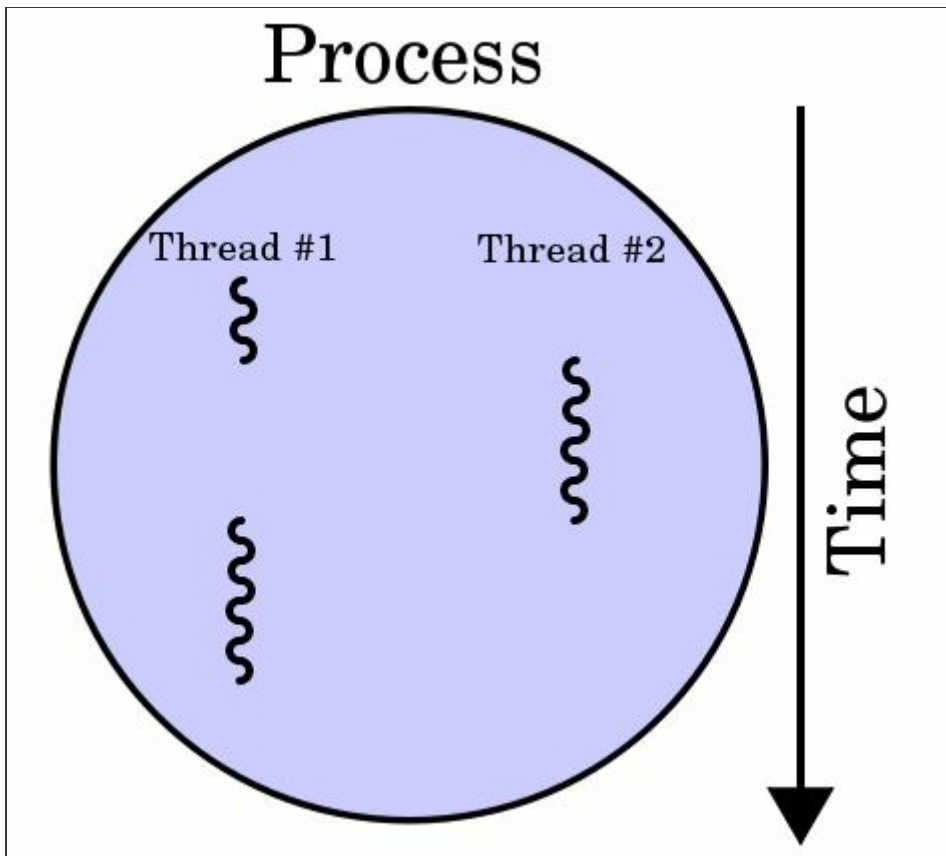
Windows Task Manager:

工作	狀態
*D:\Dropbox\Public\pmag\201303\source\article4...	執行中
cs	執行中
File:Htop.png - Wikipedia, the free encyclopedia - ...	執行中
Greenshot image editor - ProcessVsThread.jpg	執行中
Skype	執行中
source	執行中
無標題 1 - LibreOffice Impress	執行中

處理程序: 83 CPU 使用率: 1% 實體記憶體: 87%

圖、Linux 與 Windows 中的 Process

Thread 交替執行的這種概念可以用下圖表示。(Proces 也是如此，只是將圖中的 Thread 改為 Process 而已)



圖、Thread 的概念

C# 中的 Thread 概念

在現代的作業系統當中，如果我們將一個程式重複執行兩次，將會產生兩個 **Process**，那麼這兩個程式將是毫不相關的。任何一個程式都不需要知道另一個程式是否存在，通常也不會與另一個程式進行溝通。

但是，如果我們希望兩個程式能夠互相分享某些變數，但是卻又同時執行，此時就可以利用 **Thread** 的機制。對於程式設計師而言，**Thread** 就像一個可以單獨執行的函數，這個函數與其他程式 (包含主程式) 同時執行，感覺上好像互相獨立，但是又可以共用某些變數。以下是一個 C# 的 **Thread** 範例：

```
using System;
using System.Threading;

class SimpleThread
{
    String name;

    public static void Main(String[] args)
    {
        SimpleThread a = new SimpleThread("A");
        SimpleThread b = new SimpleThread("B");
        Thread athread = new Thread(new ThreadStart(a.run));
```

```
Thread bthread = new Thread(new ThreadStart(b.run));  
athread.Start();  
bthread.Start();  
athread.Join();  
bthread.Join();  
}
```

```
SimpleThread(String pName)
```

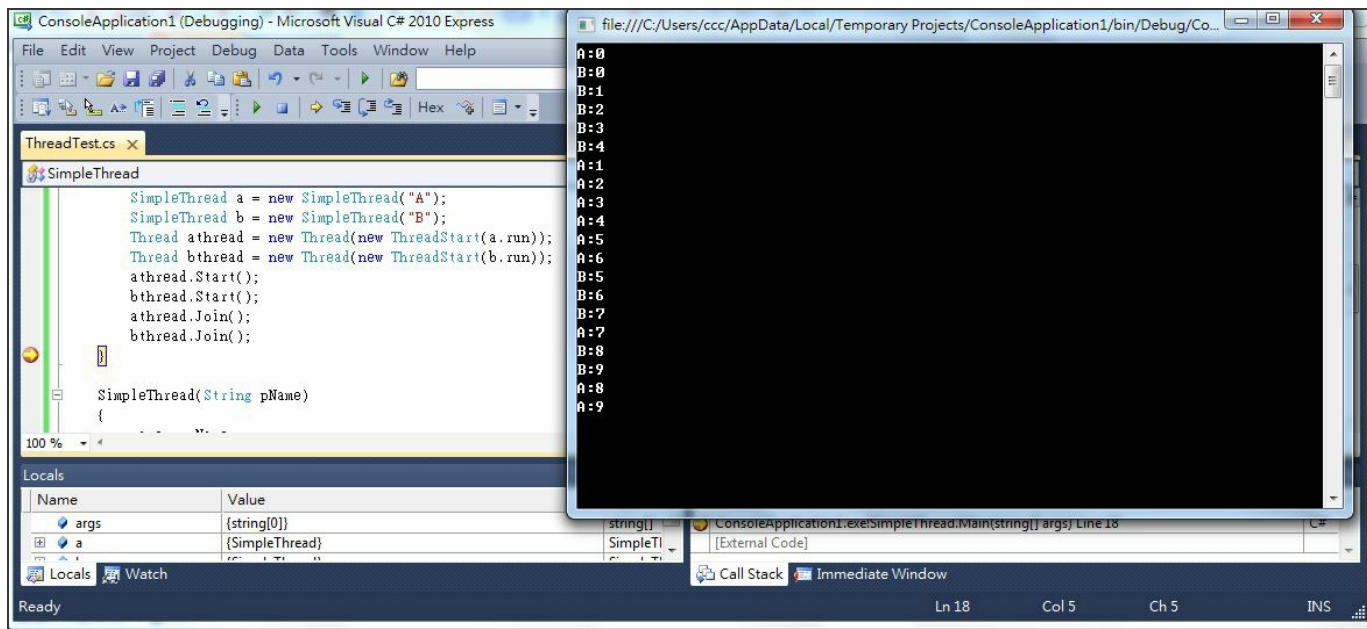
```
{  
    name = pName;  
}
```

```
public void run()
```

```
{  
    for (int i = 0; i < 10; i++)  
    {  
        String line = name + ":" + i;  
        Console.WriteLine(line);  
        // Thread.Sleep(10);  
    }  
}
```

```
}
```

其執行結果如下圖所示：



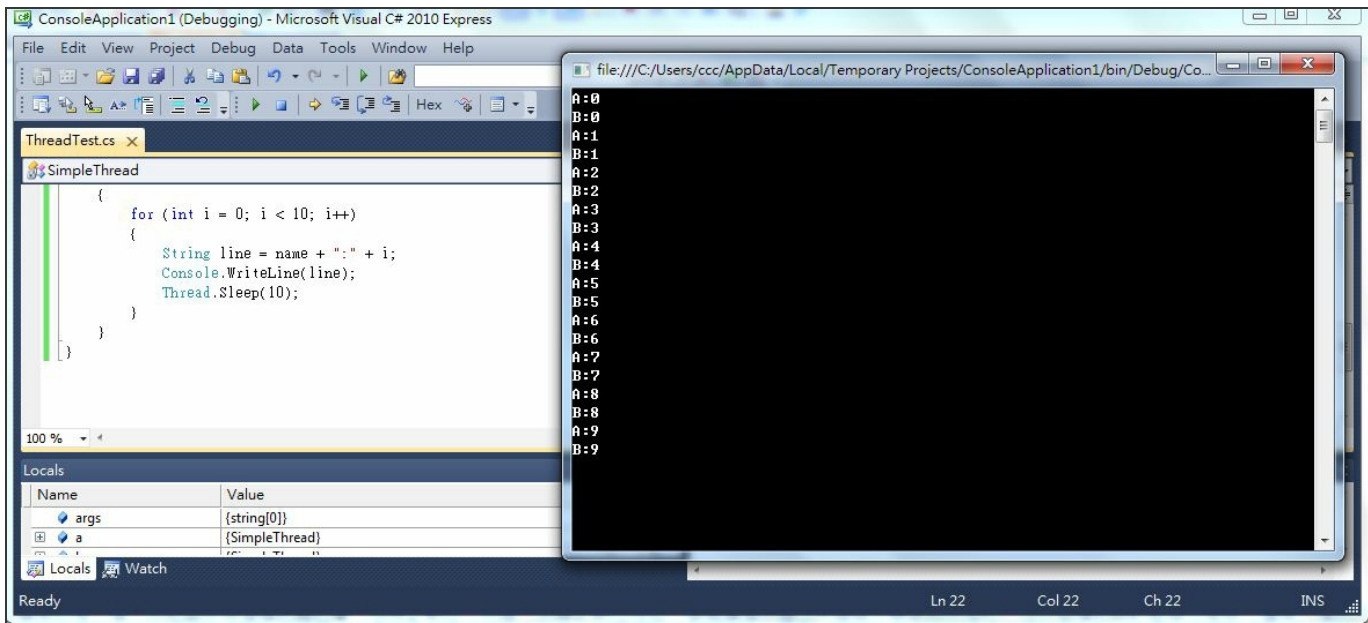
圖、Thread 的執行結果 – 沒有 Sleep 的情況

對於剛開始接觸 Thread 的程式人員而言，會感覺到相當的詭異。因為『兩個 Thread 同時執行』是一個相當難以理解的概念。事實上，對於只有一個 CPU 的程式而言，並非兩個程式真的會「同時」執行，而只不過是「交錯」執行而已。但是這個交錯方式是由作業系統決定的，而非由程式設計師自行安排。而對於多 CPU 或多核心的處理器而言，就真的會「同時」執行，而不是只有「交錯」執行而已。

通常，程式人員對於這種不能由自己操控決定的情況會有不安的感覺，但是當您多寫幾個程式之後，這種

疑慮就會消除了，畢竟，程式人員本來就相當依賴作業系統，只是自己通常感覺不到而已。

當然，如果我們想要稍微控制一下 **Thread** 的執行順序，那麼就可以要求目前的 **Thread** 去休息睡覺，像是上述程式中的 **Thread.Sleep(10)** 這行程是本來是被註解掉的，但是如果我們將這行程式的註解拿掉，那麼將得到下列執行結果。

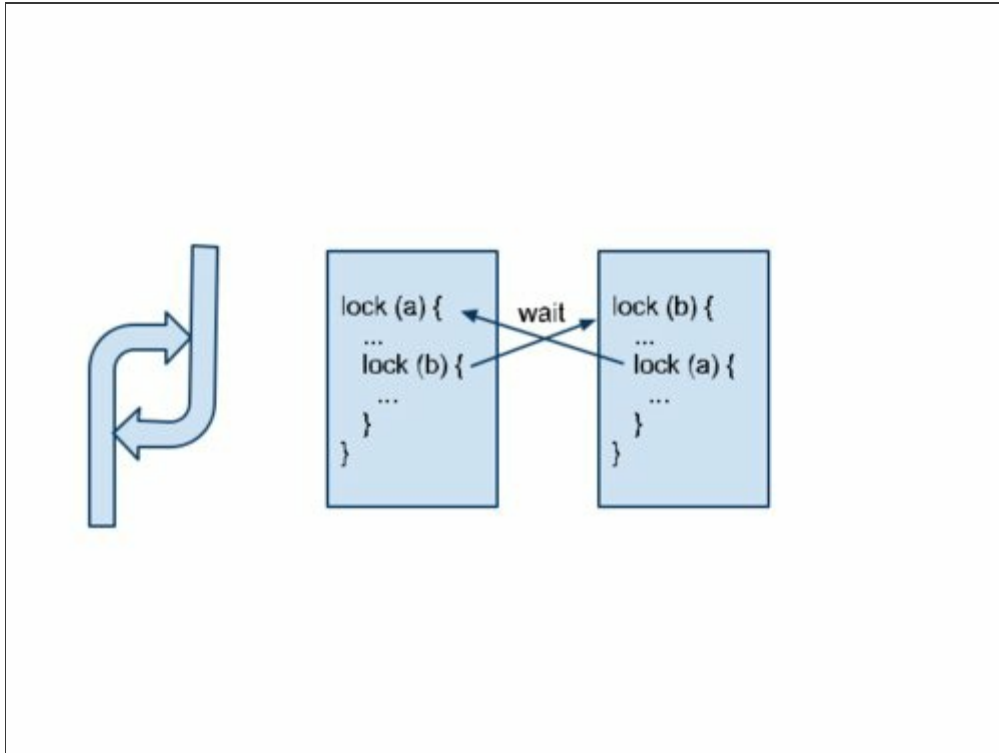


圖、Thread 的執行結果 2 – 有 Sleep 的情況

從上面兩個圖中，您可以看到還沒加入 `Thread.Sleep(10)` 之前，兩個 Thread 的交錯方是很隨興，基本上是由作業系統任意安排的，但是在加入 `Thread.Sleep(10)` 之後，因為兩個 Thread 在印一次後就會禮讓給對方，所以就成了嚴格交互的 A, B, A, B 之情形了。

以 C# 體驗 Deadlock 死結

在作業系統的課程當中我們會學到『死結』這個問題，當程式 1 抓住資源 A，卻又在等程式 2 釋放資源 B，而程式 2 則抓住資源 B，卻又在等程式 1 釋放資源 A 的時候，就會進入死結狀態。這就像兩台很長的火車，互相卡住對方一般，下圖顯示了死結情況的示意圖。



圖、死結的示意圖

在程式設計中我們真的會遇到死結嗎？如果真的有死結，能否寫一個會造成死結的程式呢？

這並不難，只要用執行緒 (Thread) 與鎖定 (lock) 機制，我們很容易就可以造出會導致死結的程式，以下是我們用 C# 撰寫的一段死結程式，請參考。

```
using System;
using System.Threading;
using System.Text;

class ThreadTest
{
    static StringBuilder A = new StringBuilder("A");
    static StringBuilder B = new StringBuilder("B");

    public static void Main(String[] args)
    {
        Thread thread1 = new Thread(new ThreadStart(AB));
        Thread thread2 = new Thread(new ThreadStart(BA));
        thread1.Start();
        thread2.Start();
    }
}
```

```
thread1.Join();  
thread2.Join();  
}
```

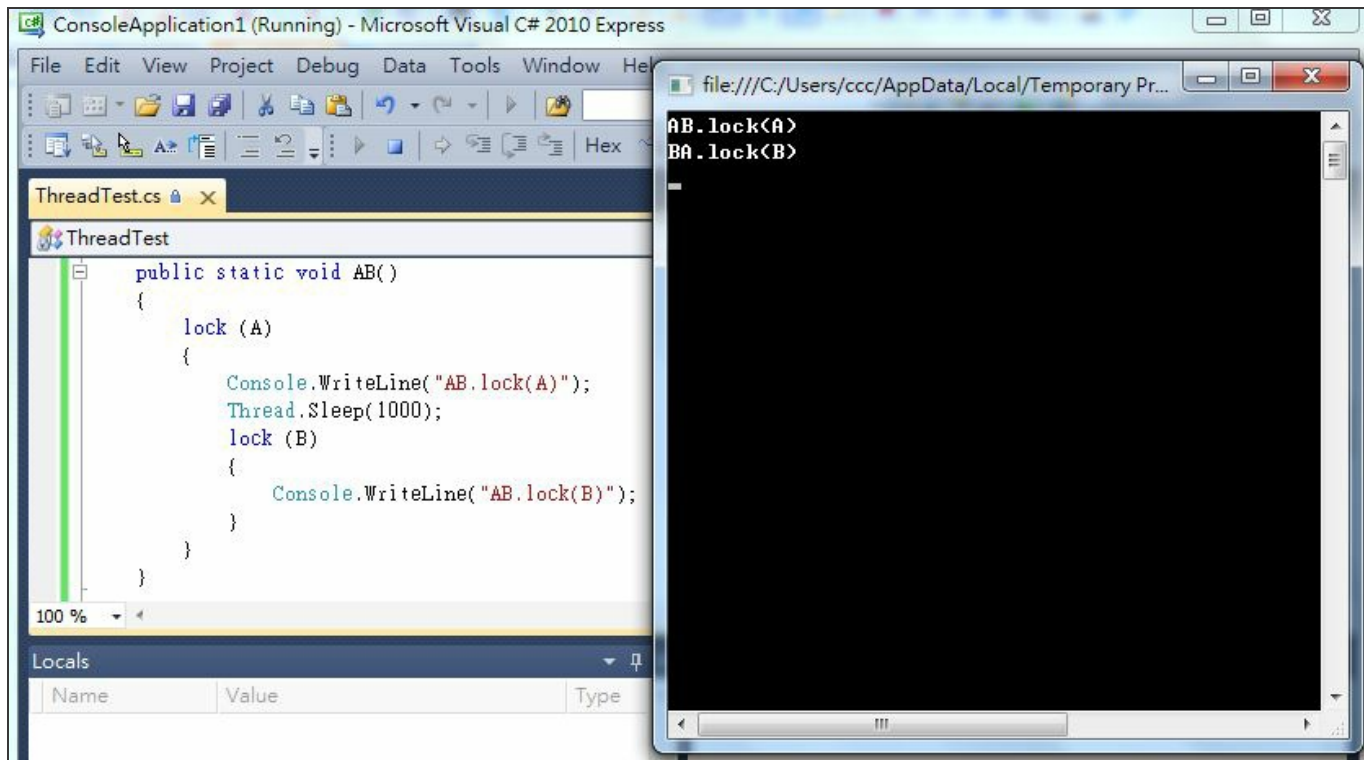
```
public static void AB()  
{  
    lock (A)  
    {  
        Console.WriteLine("AB.lock(A)");  
        Thread.Sleep(1000);  
        lock (B)  
        {  
            Console.WriteLine("AB.lock(B)");  
        }  
    }  
}
```

```
public static void BA()  
{  
    lock (B)  
    {  

```

```
Console.WriteLine("BA.lock(B)");  
Thread.Sleep(1000);  
lock (A)  
{  
    Console.WriteLine("BA.lock(A)");  
}  
}  
}  
}
```

上述程式的執行結果如下圖所示，當程式跑到 `BA.lock(B)` 之後就進入了死結，再也無法跑下去了，因此我們不會看到 `BA.lock(A)` 與 `AB.lock(B)` 這兩行輸出的結果，程式已經進入了死結狀態，再也出不來了。



圖、Deadlock.cs 程式的執行結果

至此，我們已經用 C# 實作了作業系統中的 Thread 與 Deadlock 這兩種概念，以便讓讀者能透過實作真正的去感受作業系統，希望這樣的說明方式對讀者會有所幫助。

在下一期當中，我們將繼續利用 C# 說明作業系統中的 Race Condition (競爭情況) 概念與利用 lock 及 Semaphore 機制避免 Race Condition 的方法，我們下期見！

雜誌訊息

- [讀者訂閱](#)
- [投稿須知](#)
- [參與編輯](#)
- [公益資訊](#)

讀者訂閱

程式人雜誌是一個結合「開放原始碼與公益捐款活動」的雜誌，簡稱「開放公益雜誌」。開放公益雜誌本著「讀書做善事、寫書做公益」的精神，我們非常歡迎程式人認養專欄、或者捐出您的網誌，如果您願意成為本雜誌的專欄作家，請加入 [程式人雜誌社團](#) 一同共襄盛舉。

我們透過發行這本雜誌，希望讓大家可以讀到想讀的書，學到想學的技術，同時也讓寫作的朋友的作品能產生良好價值 – 那就是讓讀者根據雜誌的價值捐款給慈善團體。讀雜誌做公益也不需要壓力，您不需要每讀一本就急著去捐款，您可以讀了十本再捐，或者使用固定的月捐款方式，當成是雜誌訂閱費，或者是季捐款、一年捐一次等都 OK！甚至是單純當個讀者我們也都很歡迎！本雜誌每期參考價：NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體。例如可捐贈給「羅慧夫顱顏基金會 彰化銀行(009) 帳號：5234-01-41778-800」。(若匯款要加註可用「程式人雜誌」五個字)

想訂閱本雜誌的讀者，請按 [雜誌訂閱](#) 連結並填寫表單，我們會在每一期雜誌出刊時寄送通知與下載網址到您的信箱。

投稿須知

給專欄寫作者： 做公益不需要有壓力。如果您願意撰寫專欄，您可以輕鬆的寫，如果當月的稿件出不來，我們會安排其他稿件上場。

給網誌捐贈者： 如果您沒時間寫專欄或投稿，沒關係，只要將您的網誌以 [創作共用的「姓名標示、非商業性、相同方式分享」授權] 並通知我們，我們會自動從中選取需要的文章進行編輯，放入適當的雜誌當中出刊。

給文章投稿者： 程式人雜誌非常歡迎您加入作者的行列，如果您想撰寫任何文章或投稿，請用 markdown 或 LibreOffice 編輯好您的稿件，並於每個月 25 日前投稿到[程式人雜誌社團](#) 的檔案區，我們會盡可能將稿件編入隔月1號出版程式人雜誌當中，也歡迎您到社團中與我們一同討論。

如果您要投稿給程式人雜誌，請盡可能使用以下兩種方式：

1. 使用 markdown 格式：markdown 的撰寫格式請參考 [markdown樣版](#)
2. 使用 LibreOffice odt 格式：odt 格式請下載 [odt樣版](#)。

關於以上兩種格式的寫法請下載 [template.zip](#) 以便瞭解寫作的方式與細節，請下載後閱讀 submit.md 檔案。

我們目前的編輯流程是用 pandoc 軟體將 markdown 轉換成 htm 與 epub 檔，然後再用 calibre 軟體中的 ebook-conver 指令將 epub 轉換為 pdf 與 mobi 檔案，如此就可以得到 htm, epub, pdf, mobi 等四種版本，而針對 pdf 檔我們還進一步區分為 A4 版與 ipad 版，以方便讀者根據需求自行取用。

參與編輯

您也可以擔任程式人雜誌的編輯，甚至創造一個全新的公益雜誌，我們誠摯的邀請您加入「開放公益出版」的行列，如果您想擔任編輯或創造新雜誌，也歡迎到 [程式人雜誌社團](#) 來與我們討論相關事宜。

公益資訊

公益團體	聯絡資訊	服務對象	捐款帳號
財團法人羅慧夫顱顏基金會	http://www.nncf.org/ lynn@nncf.org 02-27190408分機 232	顱顏患者 (如唇顎裂、小耳症或其他罕見顱顏缺陷)	銀行：009彰化銀行民生分行 帳號：5234-01-41778-800
社團法人台灣省兒	http://www.cyga.org/	單親、隔代教養、弱勢及一	銀行：新光銀行

童少年成長協會	cyga99@gmail.com 04-23058005	般家庭之兒童青少年	戶名：台灣省兒童少年成長協會 帳號：103-0912-10-000212-0
---------	---	-----------	---