

C 程式語言



C

* 附 UNIX作業系統簡介

* 附 C Puzzle 習題

語言

涂永超 編著

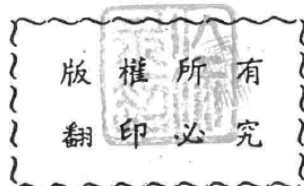
C 程 式 語 言

* 附 UNIX作業系統簡介

* 附 C Puzzle 習題

涂永超 編著

儒林圖書公司 印行



C 程式語言

編譯者：涂 永 超
發行人：楊 鏡 秋
出版者：儒 林 圖 書 有 限 公 司
地 址：台北市重慶南路一段 111 號
電 話：3812302 3110883 3140111
郵政劃撥：0106792-1 號

吉豐印刷廠有限公司承印
板橋市三民路二段正隆巷 46 弄 7 號
行政院新聞局局版台業字第 1492 號

香港地區出版權／發行權
有成書業有限公司
香港灣仔聖佛蘭士街秀華坊 23 號地下
* 在台港各地銷售任何盜印本，即屬違法

1984 年 11 月 10 日
定價新台幣 200 元正

序

C 語言之所以迅速地被廣泛採用，主要是因為它不但具備了高階語言方便、易學的特性，而且還具備了組合語言的優點——執行速度快且節省記憶體空間。此外，要精通 UNIX 作業系統，更必須由 C 語言入門。因為，幾乎所有的 UNIX 系統軟體均是利用 C 語言所撰寫而成的。

本書共分成十章，第一章至第八章是介紹有關 C 語言本身的特性，讀者若要用 C 語言來撰寫程式，解決一般的問題，那麼讀完一至八章已經綽綽有餘了。可是若要進一步認識 UNIX 系統本身，則第九、第十兩章可說是入門之鑰，非讀不可。

本書的特色是在各章後面均附有精簡習題及解答。讀者務必費心精研這些習題，以便探得 C 語言的特性。此外，在附錄 A 亦附有 UNIX 系統的操作方法，讀者可依照附錄 A 中的例題，逐一打進電腦練習。

編者利用課餘之暇，參考有關著作，編著此書。雖然力求完美，唯時間倉促且編者才疏學淺，漏誤之處在所難免，尚祈讀者及諸專家不吝賜正。

涂永超

民國 73 年 6 月於

美國，維基尼亞州

目 錄

序	I
第一章 C程式語言簡介	1
1.1 C的演進	1
1.2 編譯C程式	4
第二章 資料型態(Data Types)	15
2.1 八進位數字系統	15
2.2 變數名稱(Variable Name)	17
2.3 整數型態(Integer Type)	18
2.4 字元型態(Character Type)	21
2.5 溢出系列符號(Escape Sequence)	22
2.6 浮點數型態(Float Point Type)	24
2.7 倍精準數型態	26
2.8 設定初值(Initialization)	27
第三章 運算子(Operator)	29
3.1 算術運算子(Arithmetic Operator)	29
3.2 關係運算子(Relation Operator)	33
3.3 邏輯運算子(Logical Operator)	34
3.4 字元輸入和輸出	34
3.5 Define 型態和常數	36
3.6 型態轉換(Type Conversion)	37

3.7	遞增和遞減運算子 (Increment and Decrement Operator)	40
3.8	逐位元邏輯運算子 (Bitwise Logical Operator)	42
3.9	移位運算子 (Shift Operator)	45
3.10	指定運算子 (Assignment Operator)	47
3.11	條件運算式 (Conditional Expressions)	48
3.12	優先權和結合性 (Precedence and Associativity)	49
練習一	運算子	51
答案一	運算子	55
練習二	基本型態	57
答案二	基本型態	59
第四章	控制流程 (Flow Control)	61
4.1	敘述和區段 (Statements and Blocks)	61
4.2	If-Else	62
4.3	Else-If	66
4.4	While	69
4.5	Switch	70
4.6	For	73
4.7	逗點運算子 (Comma Operator)	79
4.8	Do-While	80
4.9	Break	80
4.10	Continue	82
4.11	Goto	84
練習三	控制流程	85
答案三	控制流程	90

第五章 函式(Function)	91
5.1 函式.....	91
5.2 型態類別 (Type Specifier).....	98
5.3 函式引數 (Argument)	100
5.4 遞迴 (Recursion)	101
5.5 儲存系類 (Storage Class)	103
5.5.1 自動變數 (Automatic Variable).....	104
5.5.2 外部變數 (External Variable)	109
5.5.3 固定變數 (Static Variable)	116
5.5.4 暫存器變數 (Register Variable)	119
5.6 儲存系類的初值設定.....	120
練習四 儲存類別.....	123
答案四 儲存類別.....	127
第六章 C前置處理器(The C Preprocessor)	129
6.1 # define 敘述.....	129
6.2 巨集指令替換 (Macro Substitution).....	132
6.3 # include 敘述.....	134
6.4 條件編譯 (Conditional Compilation).....	135
練習五 前置處理器.....	138
答案五 前置處理器.....	140
第七章 指標和陣列(Pointer and Array)	141
7.1 指標和位址.....	141
7.2 指標和函式引數.....	143
7.3 指標和陣列.....	145
7.4 位址運算.....	150
7.5 字元指標和函式.....	154

7.6	多元陣列 (Multi-Dimensional Arrays)	158
7.7	指標陣列 (Pointer Arrays)	160
7.8	指標陣列和多元陣列比較	163
7.9	指令行引數 (Command-line Argument)	164
練習六	指標和陣列	166
答案六	指標和陣列	170
第八章	結構式 (Structure)	177
8.1	結構式的宣告	177
8.2	結構式和函式	180
8.3	結構式陣列 (Arrays of Structures)	183
8.4	結構式指標	188
8.5	連接串列 (Linked List)	189
8.6	表格查視 (Table-Lookup)	196
8.7	聯合 (Unions)	199
8.8	Typedef	200
練習七	結構式	203
答案七	結構式	206
第九章	輸入和輸出	221
9.1	標準輸入和輸出 (Getchar 和 Putchar)	221
9.2	Printf	227
9.3	Scanf	230
9.4	記憶體中的格式轉換 (Format Conversion)	232
9.5	檔案存取 (File Access)	233
9.6	Stderr 和 Exit	236
9.7	行輸入和輸出 (Line Input and Output)	238
9.8	其他各種函式	239

9.8.1	字元類別的測試和轉換	239
9.8.2	記憶體管理 (Storage Management)	240
9.8.3	系統呼叫 (System Call)	240
第十章	UNIX系統介面	241
10.1	檔案識別值 (File Descriptor)	241
10.2	低階 I/O — Read 和 Write	242
10.3	Open , Creat , Close , Unlink	245
10.4	隨機存取 (Random Access)— Seek 和 Lseek	248
10.5	實例— Fopen和 Getc	249
10.6	實例—列出目錄 (Listing Directory)	253
附錄A		259
一、引言		259
二、建立和維護檔案		264
三、UNIX 檔案系統 (The UNIX File System)		285
四、檔案操作		295
附錄B		305
參考書目		307

第一章

C 程式語言簡介

C 程式語言是由貝爾實驗室 (Bell Laboratories) 在 1972 年所發展出來的。整個語言的設計及撰寫都是由 Dennis Ritchie 個人獨自完成。他當時正和 Ken Thompson 共同發展 UNIX 作業系統 (Operation System)。

1.1 C 的演進

UNIX 作業系統是專門為軟體工程而設計的系統。C 語言可說是該系統下最基本的工具，UNIX 系統本身和其相關軟體均是以 C 語言撰寫而成。由於 C 語言可用來設計作業系統，所以又稱之為系統程式語言。其實 C 語言並非僅適用於某特定作業系統或特定機器，它同樣適用於數值運算、文字處理和資料庫 (data base) 等方面的程式設計。

大約在 1970 年代，UNIX 系統由貝爾實驗推廣出來，當時普遍使用在美國各大學中，但是不久它就被全世界廣泛的接受了。目前，C 語言在貝爾實驗室內已經成為一項“標準”語言，甚至成了它的註冊商標。但是值得強調的是 C 語言之所以受到熱烈的歡迎，在事前並沒有經過任何促銷廣告，而完全是由於使用者樂於接受的關係。到了 1980 年代，軟體設計公司相繼推出多

2 C 程式語言

種不同的 C 編譯器 (compiler)。因此，在許多非 UNIX 作業系統上，也開始提供 C 語言。如今，C 語言的使用已經是軟體發展的必然趨勢了。

C 語言是一種結構化程式語言 (structured language)。由於結構化程式語言的使用規則相當簡單、清楚，所以類似這種語言，在 1970 年代相繼被設計且推廣開來。其中 PASCAL 語言可說是衆所注目的焦點。但是由近幾年的發展情勢來看，已經有甚多 PASCAL 的喜好者，逐漸以 C 語言來替代 PASCAL 作為軟體發展的工具。因此，C 語言可說是後來居上，成為結構化程式語言的新寵兒了。

C 語言的逐代演變如下：

ALGOL 60 (1960 年問世)

↓

CPL (1963 年問世)

↓

BCPL (1967 年問世)

↓

B (1970 年，Ken Thompson，貝爾實驗室)

↓

C (1972 年，Dennis Ritchie，貝爾實驗室)

雖然 ALGOL 只比 FORTRAN 晚出道幾年，但是由於其嚴密的結構，對於後來的結構化程式設計，影響非常重大。但是不幸地，ALGOL 本身卻未被大家廣泛的接受。CPL (Combined Programming Language) 是延襲 ALGOL 而設計出來的另一種語言，但是由於該語言本身過於龐大且複雜，對於程式使用者來說，很難學習和撰寫。隨後的 BCPL (Basic Combined Programming Language) 和 B 語言，則是擷取前二者的部份精華所設計

出來的特殊用途語言。然而，這二種語言仍然無法解決一般常見的許多問題。後來經由 Dennis Ritchie 增添了一些特殊功能並經規劃整理後，C 語言才算是真正的問世了。

C 語言的撰寫有一項特性：它能由小而大，由簡單而趨複雜的堆積起來。因此，一個非常複雜的程式可以經由一些簡單的功能相互結合起來。舉例來說，就好像將電子及中子組成各種原子，再將各種原子任意組成各種物質的步驟一樣。利用此種功能，程式的設計就顯得方便且有彈性多了。

Dennis Ritchie 本人比較著重於系統軟體的發展；諸如電腦語言、作業系統和文件資料的處理。所以 C 語言是設計這一方面軟體最具成效的一種。雖然它具有處理一般問題的能力，但是得特別強調的是，C 語言並不是適合所有應用問題的語言。當然，使用者可以用 C 語言撰寫有關會計作業的軟體或是類似這方面的應用。但是，若採用其他語言來處理會計作業，可能會比 C 語言來得便利一些。

C 語言是一種著重於系統軟體的程式語言。其主要有二項優點。首先，它具有低階語言，如組合語言（Assembly Language）的性質。因此用它來設計的程式能夠節省記憶體空間及執行時間。其次，它具有一些高階語言的性質，使用者可以不須了解計算機硬體結構，即可著手撰寫程式，對程式的撰寫較具彈性且有效率。

舉例來說，若將 **FORTRAN** 中的一條敘述（statement）：

$$a = b + c$$

轉換成 8080 組合語言，表示如下：

```
LHLD .c  
PUSH H  
POP B  
LHLD .b  
DAD B  
SHLD .a
```

對於程式使用者來說，前者並不需要具備計算機硬體的知識就可以設計出來，但是後者則恰好相反。

由於 **FORTRAN**、**ALGOL** 等高階語言不適合用來設計和計算機硬體息息相關的系統軟體。因此，早期的系統軟體設計師還是必須使用組合語言來撰寫程式。但是，若使用 **C** 語言來設計系統軟體程式，不僅具有低階語言的功效，同時亦有高階語言的同等彈性。

1.2 編譯C程式

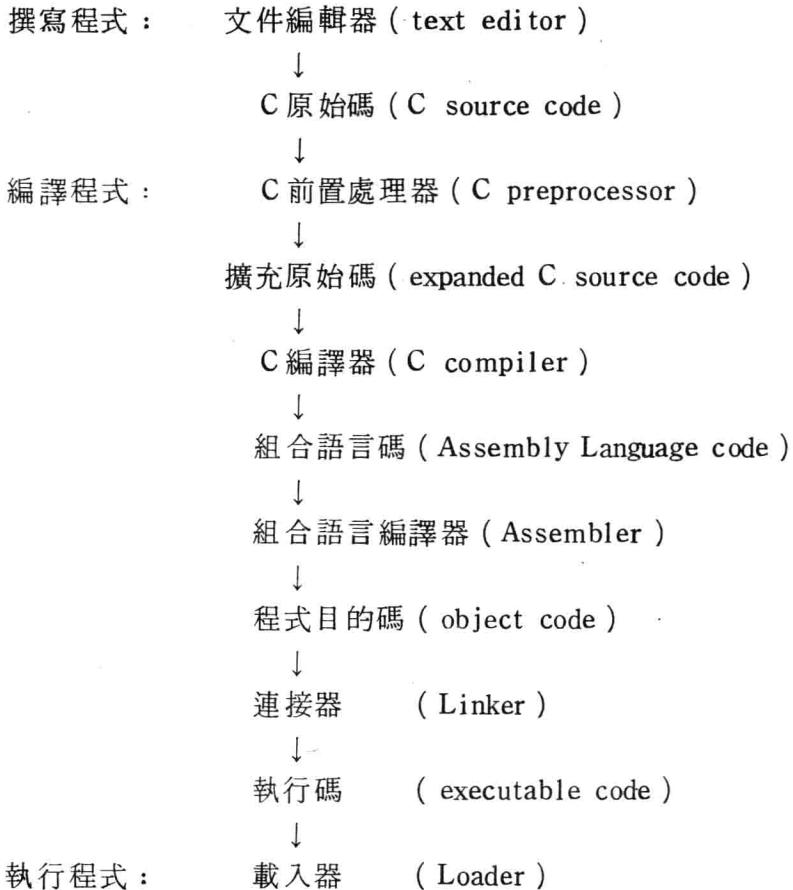
計算機無法直接執行由 **C** 語言所撰寫的程式。因此，為了執行 **C** 語言程式，首先必須藉著另一個軟體程式，將 **C** 程式轉換成計算機能夠認識的語言。這個軟體程式稱為編譯器。

以 **C** 語言所撰寫的程式稱為原始碼 (source code)。編譯器的主要功能即是將原始碼翻譯成計算機能夠了解及執行的指令。由於各種計算機的結構均有所不同，因此在不同的機器上，隨著機器語言的不同，就需要不同的 **C** 語言編譯器。原始碼經過編譯後所產生的結果稱為執行碼 (executable code)。由上述可以知道，相同的原始碼，隨著機器的不同將會產生不同的執行碼。

由原始碼到執行碼的產生，中間需要經過多次的處理步驟。首先，使用者撰寫好的程式必須經由系統的文件編輯器 (text

editor) 予以輸入。然後將該程式存檔起來，稱為原始碼檔案 (source code file)。再打入編譯的指令 (UNIX 系統中，編譯指令是 C C) 來進行編譯該原始碼。

下面是整個步驟的流程說明：



C 前置處理器將原始碼中的縮寫型式敘述轉換成完整的 C 程式原始碼，再經過 C 編譯器的編譯後，原始碼即被轉換成計算機的組合語言程式碼，組合語言程式碼再經過組合語言編譯器的編

譯後就產生了目的碼，目的碼並不可以直接執行，必須再經過連接器的幫助，將和該程式相關的其他程式或是庫存函式（library function）連接起來。經過連接器處理過的即是執行碼。執行碼的執行必須再經過系統的載入器才能載入記憶體內執行。

由上述的步驟來看，程式由原始碼到能執行為止，其過程似乎是極其漫長。幸好，上面的絕大步驟都不需要使用者操心，UNIX 系統會自動處理這些煩雜的程序。通常使用者只要打入 CC 指令，編譯的程序即可完成。下面舉例說明。

首先必須聲明的是，本書的例子均是以 PDP-11 計算機及 UNIX 作業系統作為依據。若使用者採用的不是 UNIX 作業系統，請參考該系統的操作手冊並比較其中的差異。

第一個簡單的程式是要印出下列文字：

```
hello, world
```

首先利用文件編輯器將下面的程式輸入：

```
/*                      C                      */
/* This C program prints a message on the screen */
main ( )
{
    printf ( "hello, world\n" );
}
```

假設該程式存檔在 hello.c 檔案中。接著要求系統予以編譯及執行。整個過程如下所示：

```
$ cc ohello.c
```

```
$ a.out
```

```
hello, world
```

\$ 是 **UNIX** 系統所顯示的符號稱為謝耳提示 (shell prompt)。使用者必須待終端機螢幕上出現此符號後，始能打入任何指令。因此，在 **\$** 符號之後打入 **CC** 編譯指令：

```
$ cc hello.c
```

此處必須強調的是，原始碼的檔案名稱一定要以 **.c** 做為結尾。例如 **hello.c**。

假設整個編譯過程進行得很順利，**\$** 符號會再度顯示。此時打入 **a.out** 指令要求執行該程式：

```
$ a.out
```

系統即會立刻執行該程式，並將結果顯示在終端機上：

```
hello, world
```

若使用者熟悉 **UNIX** 系統 (請參考附錄 A)，此時檢查自己的檔案目錄 (file directory)，將會發現其中多了一個 **a.out** 檔案名稱。因為 **C** 編譯器每次編譯後均會將編譯結果存檔到 **a.out** 檔案名稱中。

上面的例子非常簡單。事實上一般 **C** 語言的程式要比該程式複雜得多，而且可能分別存檔在數個不同的檔案中。這些程式必須經過個別編譯，直到使用者將他們組合成一個完整而可執行的程式。例如若將三個函式存檔在三個檔案內，其檔案名稱分別是 **file1.c**、**file2.c** 和 **file3.c**，使用指令：

```
$ cc file1.c file2.c file3.c
```

編譯這三個檔案，並把編譯後產生的目的碼存檔在 **file1.o**、**file2.o** 和 **file3.o** 中，最後將這些目的碼載入 **a.out** 的可執行檔案。

若在編譯過程發生錯誤，例如 **file1.c** 有錯誤，該檔案可重

新編譯，並將結果和先前沒有錯誤的目的檔案一起載入。使用下面的指令：

```
$ cc file1.c file2.o file3.o
```

cc 指令會根據檔案命名方式，以 “.c” 和 “.o” 來區分原始檔案和目的檔案。

利用上述的編譯程序就可以將一個複雜的程式分割成數個較小的程式，分別予以撰寫、編譯和測試，直到最後再組合起來。

現在僅對該程式作個簡單說明。前面二行：

```
/*                                C                                */
/* This C program prints a message on the screen */
```

即是所謂的註解 (comment)，註解可以任意自由使用，使得程式令人容易了解。任何空格 (blank) 和跳行字元 (newline) 可以出現的地方，都可以加入註解。註解部份必須置於 /* 和 */ 二組註解符號之間。

不管程式的大小，C 程式至少可包含一個或數個函式 (function)，這些函式都具有特定的功能。C 語言中的函式和 **FORTRAN** 中的副常式 (subroutine) 或 **PLI**、**PASCAL** 中的程序 (procedure) 類似。在上例中 main 即是一個函式。一般而言，函式名稱可以任意命名。然而，main 是其中比較特殊的一個函式，程式被執行時將由 main 函式的開端執行起。換句話說，每一個 C 程式一定要有一個名稱爲 main 的函式。通常 main 將呼叫 (call) 其他的函式以執行其工作。函式呼叫時可以呼叫在同一程式中的其他函式，也可以呼叫系統中的庫存函式。

函式間資料轉換的方法之一是利用引數 (argument) 方式來傳遞引數串列 (argument list)。引數必須放在函式名稱後