科學圖書大庫

學習福斯程式語言

譯者　鄭新惠

徐氏基金會出版

譯　序

“福斯”語言在電腦界已經普遍應用，它是第四代電腦語言，它的優點在於使用的記憶區少，執行的速度快，和隨你的心意，可以編出你自己的指令而組成程式。家用電腦也極普遍，將來會和電視機一樣，家家都有一台。

有了一台家用電腦，就想玩玩電腦能做的工作、記帳、分析、存資料、計算題目，甚至當電動玩具，跟著就想自己編程式，興趣會愈來愈濃厚，用其他高階層語言，總會覺得儲存區不夠，64 K還嫌少，128 K甚至更多，於是小電腦不夠用，財力所及，買不起大電腦。

“福斯”語言就給你解決這個大問題，記憶區不很多，也一樣可以玩，8 K，16 K都足夠了。

要用“福斯”語言於家用電腦，就得學“福斯”語言的程式編製方法，本書介紹給你“福斯”的基本原理，進而至編“福斯”程式，即使你沒有電腦在手邊，在紙上繪製，也可以知道編的程式對不對，能不能在電腦上用，這一個方法可以教許多對編程式有興趣者着手進行，踏入“福斯”之門，進而成爲一位“福斯”程式師。

目　錄

第一章　敍述"福斯"

第一件事，你聽人家說的，"福斯"不是程式語言，它是宗教，"福斯"的程式師也會想它是神秘，推進像頻展示，用微小的胡言亂語，像魔術般支配機器工作。"福斯"眞是神秘的語言，它也可用眼覺察，所有驚嘆諸點，星散在低階層"福斯"程式中，暗示着語言操縱着程式師，有所誇張。

另一件事，你會聽說的：寫"福斯"程式很容易，但簡直不能想出它是怎樣進行的，有一個故事，講到一位程式師化費了六個月，寫出了150多幕的原始碼，但發現開始的十幕，自己都不知道做什麽用。他試試程式，却又沒有錯，他拿掉這十條，却又不行，他祇好放回它們，加了個註腳，像"改變這十幕是你自己在冒險"，然後進行他另一個計畫。實際，大多數"福斯"程式師能夠很容易讀他們自己的碼，去讀別人的"福斯"程式，比釋明自己的更困難。

你不久會發現，"福斯"的社區，有它的名字，且對每種符號，有它的正確發音，這些派定的邏輯，並非一致的，你會學到"福斯"裡的指令{!}讀若"STORE"（儲存），指令{R#}讀若"R-SHARP"，{UCASE}讀若"U-CASE"。

這些不會把你嚇倒，你也許聽到過關於"福斯"的可怕故事，使你不敢進一步去審查這語言，在這本書裡，"福斯"可以正確地用英文句子描述，沒有魔術似的拼法，假使某人不理會"福斯"老手的怪着，倒可以說他是頭腦清楚者。

"福斯"的起始

"福斯"在1970年光景由摩爾發明，這日期不能確定，因爲摩爾考慮和試驗"福斯"好多年，才確切組成，摩爾用的是一具首先能交互作用的IBM 1130電腦。

"福斯"這名字由於摩爾想像他的發明該是第四代電腦語言，IBM 1130祇能接受五個字符，摩爾把FOURTH（第四）縮短爲FORTH（福斯）。

在1979年舊金山舉行的"福斯"會議，摩爾發表演說，說明"福斯"是他在司丹福中心工作時用，他把許多不同的片段的語言，組成完整的"福斯"程式語言，在NRAO（National Radio Astronomy Observatory）使用。

在許多保護的年分，摩爾不斷地在NRAO爲"福斯"發展，在1973年摩爾和幾位同好脫離NRAO，自創"福斯公司"。摩爾說："今日我們選在這市場裡，世界上太少人能有新的遠見來支持這公司"。

1973年發展成"福斯公司"後，又組織了"福斯同好社"（FIG），在1979年摩爾估計已有1000位"福斯"程式師，而且每年在加倍，除此之外，摩爾和"福斯同好社"深信"福斯"現在差不多在每個中央處理機（CPU）裡已用到。

"福斯"的質

"福斯"有若干性能，沒有被認爲屬於電腦語言，它的特質

第一，"福斯"快速，語言的設計在速度方面很樂觀地有許多方法，它的快速並不驚奇，想到當初起源於操縱望遠鏡和其實際的裝置，必然快速。

第二，"福斯"小巧，需要的電腦可以小巧，大多數"福斯"系統祇要有8K到16K的RAM，它的缺點是，"福斯"缺乏內在的浮點算術，字串改竄，和攙雜的磁碟輸入／輸出（I/O）。雖然這些要求，

可以在“福斯”裡做到，但在基本的系統裡，它們沒有供應。

最後，“福斯”是可延性的，假使你不喜歡既存在標準“福斯”裡的指令，你可以創造你自己的，你可以把兩個指令合併成爲第三個指令，你可以把創新的指令，和既有的一個或幾個指令，合併成另一個指令，總之，“福斯”是砌磚式的語言。

用“福斯”編程式

“福斯”的結構與其他語言不同，所以編程式也和基本語言（BASIC）通用商業語言（COBOL）不同。

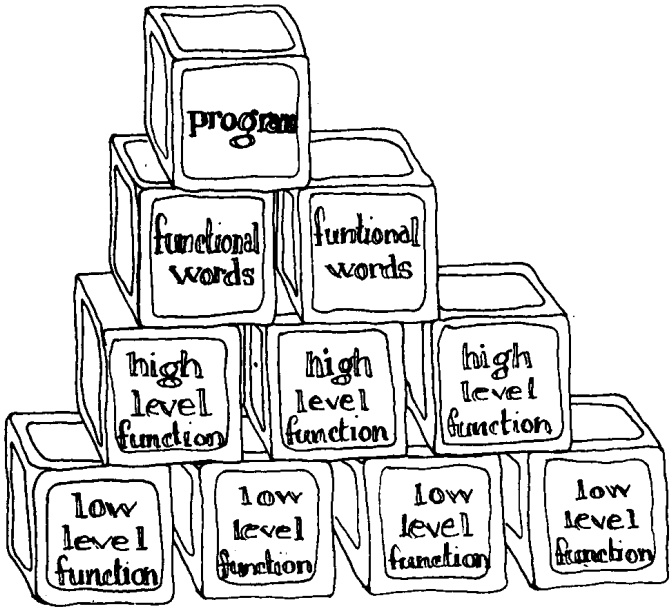
“福斯”不顧解碼或編譯的敍述，這語言包含了這兩種的成分，一方面你可以執行“福斯”，而馬上打入指令，祇要這指令，“福斯”識得，而可以照着做的，它就像解碼機，另一方面，你可以把原始碼打入編輯裡，去減少機械語言，然後編譯和執行，用“福斯”的解碼，你可以創造新功能，它們被編入詞典，用它的語言作爲解碼／編譯的組合體。

要完全瞭解用“福斯”編程式，你必須讀完這本書，目前編個大綱，把幾點基本的“福斯”來編程式指出。這僅是個粗枝大葉，你需要和“福斯”供應你的資料參照。

1. 輸入“福斯”解碼—用你的“福斯”磁碟及你的電腦系統，把“福斯”解碼輸入，好像你做其他機械語言程式一般。
2. 創造程式塊區—用“福斯”的性能，開始組合操作的塊區，直到最後祇成一句短句或一個字，成爲你需要的動作，做“福斯”程式，好像堆金字塔一般。

假使你要一個永久性的程式，那麼用下面兩個步驟：

3. 輸入編輯—利用編輯創造“福斯”原始碼的幕，主要是你想打入的編排譯本，去創造你的新機能。
4. 編譯說明—在標準的“福斯”系統，“福斯”去把資訊輸入電腦的動作，會編譯原始碼入可以執行的機能或程式，某些“福斯”譯本不用標準“福斯”，除了資訊塊區，那樣你得用預先規定的指令，像｛FLOAD｝去編譯“福斯”原始碼。



注意：你輸入資訊幕到“福斯”，可能含有直接命令，因此你可以達到其他語言不可能做的工作，像編譯程式後可以保護它自己。

5. 利用編譯機能—新機能的編譯碼，極像“福斯”解碼的一部分，你可以用你編譯的碼互相作用，或用它們來創造更大的機能塊區。

再說一遍，你必須要小心，上面所述的祇是個概況，解碼和編譯，這種名字很廣泛，以後你才能漸漸瞭解。

假使你以前做過電腦程式，你會覺得編“福斯”程式有點古怪，的確有許多跡像，證明編“福斯”程式很乖僻。資訊不一定要在合理的“福斯”程式存入變數，我們迎用疊層（基本上，疊層是記憶的區域，最後存入的，最先提出）就行。

另一個“福斯”程式的怪僻，是所有操作有關數目字的，都用RPN（Reverse Polish Notation），那就是先給數目字，然後告訴它做些什麼。

怎樣用這本書

寫一本關於“福斯”的書，讓你明瞭，逆向語言的重要性，總之你要如何告訴“福斯”去工作，很幸運的，你可以省掉從後面讀到前面的極大困苦，或像程式師所謂的，“從底下上來”。

你必須明瞭“福斯”有兩個基本組織，詞典和疊層，以後分別會講到。

第四章講到用RPN方法做“福斯”算術，而且祇有整數。

跟着的三章ㄧ敍述“福斯”的特別形態，用變數和常數，控制機構、週邊裝置，以及記憶的操縱，這些都是本書的主要章節，可以放在電腦旁邊，隨時參考。

本書的最後部分，想讓你操作“福斯”的一貫性，成爲一個有效率的“福斯”程式師，這裡會討論你如何能夠編織“福斯”給你自己的需要，在“福斯”的環境裡，養成操作的習慣，而且明瞭更進一步的技巧，像用返回疊層和額外詞彙之類。

符號的習慣

爲了助你更能明瞭幾個習慣，在這書裡一直會用到的，每次用到“福斯”指令，我們用{ }符號來分別。

另一個符號是<>，那是表示你應該按那一個操作的鍵，像<ENTER>，<CR>之類。

最後，“福斯”祇是個一般的意義，雖然“福斯”已經介紹給社會大衆，它還是“福斯”公司的商標，這本書裡講到的“福斯”都根據“福斯”—79（一個標準的“福斯”），或者FIG—“福斯”（社會大衆的譯本），延展“福斯”的詞彙，“福斯”公司的出品，及其他出版的“福斯”語言。

現在你可以開始去發掘“福斯”了。

第二章 “福斯”詞典

這一章要教你基本的“福斯”結構，它周旋於所謂詞典，你也跟着學到進入詞典的所謂“福斯”指令。

查詞典

在其他語言，你告訴電腦要根據你的命令或敍述去做什麽，每一個命令經常祇是一個功能，在所有其他語言，這些命令都是事先由語言設計者決定的，你也祇能用他們的命令來操作，你不可以取消、替代，或變換它的功能。

“福斯”不同，在“福斯”每一個功能有一個指令，要用的指令會從記憶區的這處，移到另一處，像用｛CMOVE｝那樣，目前你祇要知道“福斯”的指令，就像其他語言裡的命令，它們各自完成一個功能，以後這種定義會修改，現在就這樣解釋好了。

就像英文那樣，“福斯”的指令在詞典裡下了定義，“福斯”的詞典裡把各自有定義的指令，按不同的長短，互相聯繫，要進入“福斯”詞典，需要下面幾個編排：

名字的長度（字符的多少）
名字（輸入詞典的）
與指標聯繫（指到第二個輸入）
字碼指標（指向指令的碼）
參數（變數等之值，或地址表）

這個大概的編排表示每個輸入詞典的，都保住指令的長度，這字符包含指令本身，與其他輸入詞典的相互聯繫，一個地址去尋找和指令聯想的

機械語言碼，和其他參數。

當“福斯”需要完成動作，你得打入有效的指令，它會在詞典裡去查，它要看你打入的是常數、變數或有效的指令，這差別在於字碼指標，指在什麼字碼上，和什麼資訊是存在詞典裡的那個參數區。

什麼是指令

大多數輸入詞典的“福斯”指令，有記憶區的地址，指向其他“福斯”指令，祇有幾個基本的“福斯”指令，僅實際含有機械語言碼，詞典參數區含有地址表，代表其他“福斯”指令，使“福斯”的執行，按着次序，這種地址表的實際效果，使“福斯”在記憶區互相聯繫，執行的指令，順着使用者特別規定的次序進行，這就是“福斯”所謂聯繫語言的特性。

這裡舉個例子，假使你有三個機械語言的程式，都很短的，但要完成下面三個特定的功能

1. 在記憶區裡提出一個儲存着的值。
2. 從這值減去1。
3. 看看結果是不是0。

在“福斯”這三個功能可以合併成一個叫{DTZ}（Decrement and Test for Zero），{DTZ}並沒有任何機械碼在有關的地址裡，但有三個地址在不同的路線裡。

組合一個指令用一連串其他指令是非常活用的。

這是“福斯”的一項特性，用既有的指令，你可以創造新指令，“福斯”確實有三條不同的路線爲上面的例子。

1. {@}—從記憶區提取一個值。
2. {1-}—從提取得的值，減去1。
3. {0=}—檢查結果是不是0。

所以你可以創造{DTZ}，祇要每次做{DTZ}時，告訴“福斯”，順序執行{@}、{1-}和{0=}。

建立塊區

假使你懂得什麼是指令，你馬上會想到“福斯”是堆積塊區的語言，，你用標準的“福斯”指令堆積成複雜的指令，你可以用複雜的指令，堆積成更複雜的指令。

編“福斯”程式就像堆金字塔，在最下一層，你可以用許多既有的“福斯”指令，每上一層就像過濾了一層，等於把下一層的組合起來，最上的一層僅是一個指令，它會順着次序指揮所有其他的指令。

這很簡單吧，其實即使小小的程式，也像堆金字塔，大的程式好像把小金字塔互相堆積起來，這種金字塔的想法很奇妙，因爲它讓你認識“福斯”程式重重壓在最下一層。

“福斯”詞典

上面釋明“福斯”詞典裡的許多指令，在結構上非常活用，基本的好處在於你自己要把一個指令的定義，隨心所欲地變化，而適用於你的程式，你要成爲一個好的“福斯”程式師，對指令的多種定義，還得細心研究，目前你還沒有學怎樣來做“福斯”，你剛學到“福斯”怎樣結構來創造程式。

你得知道下面的基本原理：

- 在“福斯”，你打入指令來召呼功能，還有，有很多“福斯”指令，可以召呼其他指令來一同執行。
- “福斯”指令在詞典裡下了定義，並且有指令聯繫的表。
- 詞典輸入的指令要包括，名字的長度、名字、前一指令的地址、機械碼的地址、其他參數。
- 指令僅是要進行的說明表而已。
- 你可以組合許多指令，創造一個新指令。
- 做“福斯”程式不過是給指令下定義而已。
- 簡單的“福斯”就像堆金字塔，一層一層堆上去。
- 複式詞典是可能的，它可用來更改定義。

第三章　疊　　層

本章解釋“福斯”用不尋常的方法，去臨時儲存資訊，這叫疊層，你會學到“福斯”怎樣允許你去操縱資訊堆，且是逆向的格局。

“福斯”和疊層

執行“福斯”程式時需要順着許多細節進行，這可以說是順着你打入的字符去進行，或者順着程式在內部做了計算的結果而進行。

傳統上，許多電腦語言用變數來儲存程式需要的資訊，變數是電腦保留的記憶區，由你規定它的名字。

“福斯”允許用變數，但大多有效率的“福斯”程式，利用疊層作暫存資訊之用，疊層像堆盆子，你總是把後來者放在上面，如果你要拿盆子，你得從上面開始一個一個地拿走。

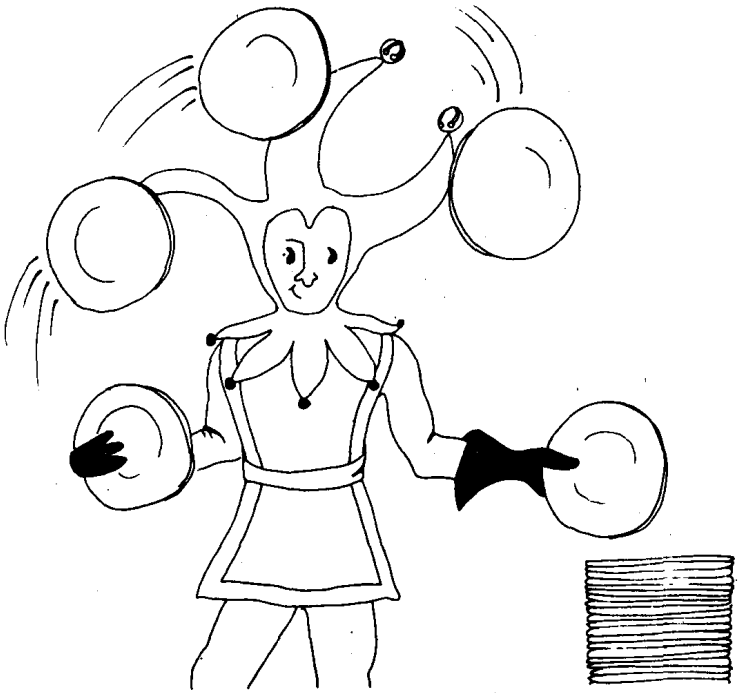
“福斯”疊層是一堆資訊，你把資訊一個一個放上去，當你要拿出來時，你得從上面一個一個地拿走，這就是所謂後進先出（LIFO＝Last In, First Out）的規定。

爲什麼用疊層

既然“福斯”允許用變數，爲什麼還要用疊層來臨時儲存資訊？就有許多可能的答案。

1. 速度—疊層給予兩個速度的好處，第一，在於你用“福斯”作解碼型式，第二在於“福斯”固有的設計。

   當作爲解碼時，“福斯”找出變數的地址，然後第二步把找到的變



數裡的資訊替代變數的地址，利用疊層祇要一個步驟——步放入資訊，一步就可以拿出資訊。

固有的“福斯”設計也有效於速度，有許多標準“福斯”指令，直接可以和疊層裡的數目字運算，如果用變數，當然也可以，但就多了幾個額外不必要的碼在程式裡。

同時，祇因爲你提用了變數裡儲存的值，而新的計算出來的結果，像其他語言般，自動儲存在它的位置，額外的時間用來查對變數裡的資訊，是不是最新的，多費時間，自然與程式速度有關。

2. 方便—有時資訊僅是短暫性的，它不需要給它個名字，而儲存在記憶區，譬如你的程式需要三個裡選擇一個來執行，這程式會按照你打入的字符，而選擇該用的字碼來進行，之後這程式就不再用這字符，假使你把字符放在疊層，而不儲存在變數裡，它很容易取得，

而且用過後就消除。

3. 連續進行—有時許多資訊要按次序進行，你已知道“福斯”的疊層必須按逆向的連續次序進行。

給你個簡單例子，假如你由1打入到10，它們都按次序進入了疊層，如果你想取消最後兩個數目，你很容易把它們拿去。

利用疊層來存資訊，另有個好處，把資訊重現就是好處之一，所謂重現，基本上是把已有的資訊，一次、再次地重現。

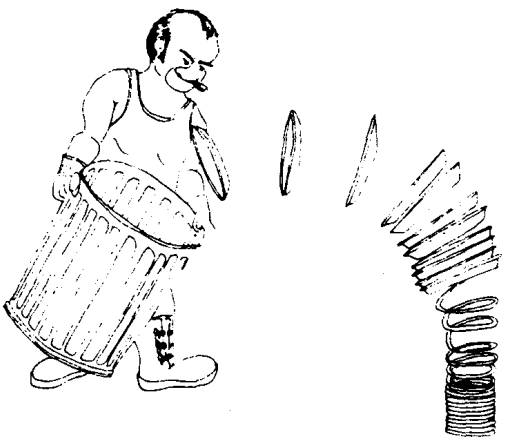
舉個簡單的例子，譬如說要求2的乘冪（$2\times2=4$，$2\times4=8$，$2\times8=16$……），你可以給個指令{POWERS}

```
[ : POWERS DUP 2 * ; ]
```

這指令把疊層最上面的數目乘以2，讓結果存在疊層上。

再說明一下，利用疊層和利用變數儲存資訊不同的地方，像一個簡單的題目，去利用變數，會被稱爲“垃圾收集者”。這個不雅的名稱，表示變數裡的資訊，祇用一次以後，再也不用，那又何必佔用了記憶區的位置，豈不是存了一大堆的垃圾。

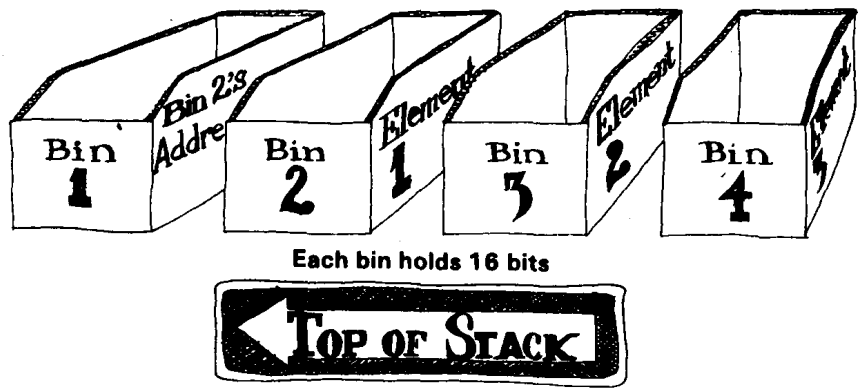
疊層的想像，好似它自己有隻垃圾桶，當你用了疊層裡的資訊，它自動丟入垃圾桶，消除了，留出空位，這有二個好處，第一是速度快（在別的語言要消除用過後，不需要保存的資訊，得另外用些命令，自然



要多費時間），第二是方便（你不會把用過的資訊，仍舊留在記憶區而造成混亂）。

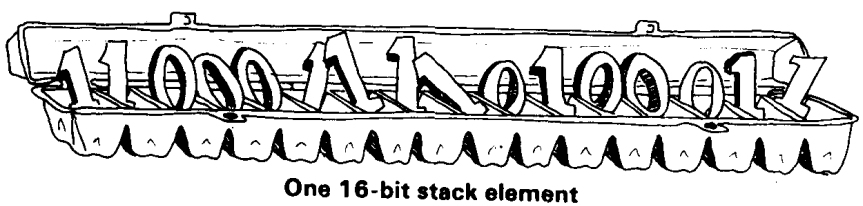
疊層裡是些什麼

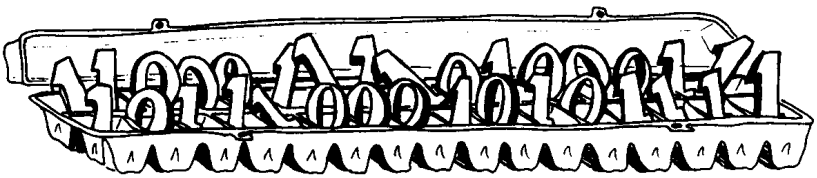
所有“福斯”的譯本，疊層的每個元素有兩個字元，換句話說每個元素有16個位元，有兩個方法把資訊存入單疊層的元素，一個是存入16位元整數（有正負號的），一個是存入16位元正整數（沒有正負號的）。



以後你會學到，整數和正整數的分別，但基本的分別，在於數目的範圍。

整數（15位元＋正負號）	範圍－32768到32767
正整數（16位元）	範圍0到65535





One 32-bit stack element

這些範圍是絕對限止的，另一種存資訊的有用雙疊層，它含有兩個16位元（即32位元）的元素，它們也有整數和正整數之別

整數（31位元+正負號） 範圍－2147483648到2147483647

正整數（32位元） 範圍0到4294967296

“福斯”並不爲了用雙疊層而改變疊層的大小，雙疊層祇是用兩個單疊層當作一個來用，你必須弄清楚，那些數目要用單疊層，那些要用雙疊層，才能用適當的功能去操縱。

另外也有用半個元素的，那就是用8個位元（即一個字元）。

簡言之，“福斯”疊層是含有16位元的元素，但自有方法讓你用雙元素或半元素，不幸的是某些有用的“福斯”指令，不能用在字元資訊，“福斯”祇可以直接和疊層頂端的字元交往（你也可以創造你自己的方法，去用疊層頂端下面的第二個字元，若是你眞的必要如此的話）。

疊層之表示

“福斯”裡的疊層是關鍵之所在，你需要確實知道在本書裡疊層怎樣被操縱。

當你打進第一個數目（不是“福斯”指令），它就進入疊層的元素#1，你再打進第二個時，它進入元素#1，而元素#1裡原有的，被擠入元素#2，你打進第三個時，元素#1被它佔用，而元素#1的被擠入元素#2，元素#2裡的被擠入元美#3，餘類推。

如果你提取元素#1的，提取後，元素#2的推入元素#1，元素#3的推入元素#2，餘類推。