系統程式

SYSTEMS PROGRAMMING FOR SMALL COMPUTERS

DANIEL H. MARCELLUS

吳建平・洪茂盛・林岳群　編譯

松崗電腦圖書資料有限公司

系統程式

吳建平
洪茂盛　編譯
林岳群

松崗電腦圖書資料有限公司　印行

系統程式

編譯者：吳建平・洪茂盛・林岳群
發行人：朱　小　珍
發行所：松崗電腦圖書資料有限公司
台北市敦化南路五九三號五樓
電　　話：(02) 7082125(代表號)
郵政劃撥：0109030-8
印刷者：建　發　印　刷　設　計　公　司
中華民國七十四年 八 月　初　版
中華民國七十六年 八 月　第三版

每本定價 280 元整

書號：4101029

本出版社經行政院新聞局核准登記，登記號碼為局版台業字第三一九六號

序　言

如果你認爲研究系統程式是爲了要使你能寫出好的系統程式，那麼這本書將給你很大的幫助。它能告訴你如何去寫小型計算機的編輯程式，直譯程式，編譯程式，作業系統等之系統程式。但是如果你認爲系統程式這一門是討論一些抽象機器的特性和語言翻譯程式的理論，那麼這本書將不太適合你，因爲這不是這本書的目的。

這本書適用於系統程式的入門課程，它強調了某些基本的觀念。我們嘗試把所有比較簡單的系統軟體的例子放在一塊。這些例子若不是專爲了這本書而設計的；便是從一般商業上的例子攫取來的。簡單的例子適合學習。只要讀者能花些時間去學，這本書應該能使你寫出像樣的系統軟體。我們希望能從讀者身上得到這樣的反應“喔！我也可以做到！”

這本書還有一個特色，它的系統軟體雖是爲了微計算機或是爲迷你計算機而寫的。而本書所採的方式是從微計算機入手。然而，不論小型或大型的計算機，軟體之一般性原則是有連續性的——這便是系統程式的心臟。但是從微計算機着手則顯得比較簡單。如果你手中握有一塊CPU，那麼對於去了解它究竟做什麼，你應該會較有信心。

這本書嘗試去尋找一種最適合教系統程式的方法。你先要熟悉一些簡單的指令；一旦熟悉這些指令之後，一層層具有相當能力的軟體再依序地加上去。問題在於這一層層的東西究竟是什麼？重要的核心觀念是什麼？

我們根據整個系統的複雜性來決定這一層層軟體的順序。令人驚

訝的是；在時間的演進上也吻合了其複雜性。因此，最內層是屬於最簡單，最老的系統軟體：ROM 監督程式的作業系統，組合程式和巨集處理程式與連結程式。再外一層則包括更複雜的控制軟體；例如單人磁碟作業系統，編輯程式，編譯程式，和直譯程式。最外一層則爲一個——多工作業系統。再往上，則超出這本書的範圍。

隨著這層層的軟體，我們也介紹了一些小型計算機的硬體，這樣子整個系統的觀念才能建立起來。

誰適合讀這本書呢？這本書適合大專系統程式的課程。它是傳統計算機科學三課題的第一課題

系統程式
編譯程式
作業系統

如果時間有限，則這本書也可用來作爲簡介作業系統，程式語言，及公用程式之課本。本書是作者教系統程式的心得。同時，除了關機、開機、與寫程式以外，對想要對計算機更深一層認識的人來說，這本書是適合他自修的。

有效率地使用這本書，需要什麼先決條件呢？讀者需先修過高階語言，組合語言，以及足夠使用計算機的經驗。因爲系統程式剛好位於硬體與軟體相接觸的部份；因此曾修過計算機硬體是有幫助的。除此以外，這本書也可用來做實驗課的補充。

在這本書內，有某些地方我們必須舉一些組合語言的例子。因此，我們選用了 Intel 8080 的組合語言。它並非是最好的微處理機，也不是最新的，我們選它主要是它最爲人所熟悉。如果你不熟悉，那麼你可以參考本書的附錄。

也許有些讀者會對於這本書把不同種類的軟體解剖開來的例子覺

得不太適應。但是我們認爲，在某種程度上，要了解軟體有如了解一輛車子。你必須先把它拆開來，曉得每一部份是怎麼工作，而後才能對整個系統有所了解。

本書將着重在實際，希望儘可能地對你寫系統軟體有更多的幫助。說到實際有個簡單的原則：只要你能寫，你就了解了。希望我們的努力是值得的。

本書完全依照讀者需要編寫，並實際於課堂教授使用，並可做爲自修參考材料。本書利用課餘時間編寫，疏漏之處在所難免；尙祈各位先進不吝指正，謝謝。

編者謹識

目錄

第一章
簡介

1-1 什麼是系統程式？

計算機系統是一能幫助人們來處理資料和解決計算上的問題的一組工具。這些工具是由機械硬體和電子電路組成的。除了計算機本身外；還有它的週邊裝置；像終端機、印表機、磁碟機，和通訊裝置。其它的工具便是軟體—系統程式（ System programs ）。這些程式將各部份連結起來，而使得使用者能立即着手解決他的問題，而不需要去了解計算機系統內部的工作情形。

系統程式有兩個目的：使一個非專家能夠簡單而容易地使用計算機，以及能夠有效率地充分使用系統的資源。當你買計算機時，軟體的系統程式通常已經包含在內。廠商所提供的程式通常包括：

- 輸入／輸出套裝副程式（ Input / Output Subroutine Packages ）
- 監督程式（ Monitors ）
- 作業系統（ Operating Systems ）
- 組合程式（ Assemblers ）
- 巨集處理程式（ Macroprocessors ）
- 直譯程式（ Interpreters ）
- 編譯程式（ Compilers ）
- 連結載入程式（ Linking Loaders ）

- 編輯程式（ Editors ）
- 偵錯程式（ Debuggers ）
- 資料庫管理程式（ Data Base Managers ）
- 通訊軟體（ Communications Software ）

我們常把程式區分爲系統程式與應用程式（ Application programs）。應用程式是使用者寫來解決問題的程式，也是他最初買計算機的目的。而系統程式則提供資源給應用程式。

一典型的大計算機中，機械的外圍是由一層層的系統程式包圍着，最外面才是使用者。圖 1-1 表明這種現象，在圖中，外圍的使用者具有相當高的生產力；因爲他只需要很少的命令便能使機器做一些有用的事。這是我們所謂軟體的生產力。在另一方面，外圍的使用者也不必知道這一層層的軟體在做什麼，這是系統程式設計者的範圍。

1-2 這種軟體是如何演進？

讓我們回憶整個計算機與系統程式的演進，然後我們將嘗試去了解小型計算機在其中所站的位置。近代計算機的發展主要是由於第二次世界大戰時，許多工程計算上的需要。最早的計算機可把它看成是機械式計算器的延伸。

在那段期間，兩種交互影響的計算觀念發生，第一種觀念是嘗試去抓住那一連串按鈕的順序以及如何把這一連串的順序以電的形式存入某種機器中。一但抓住之後，這“程式”便能一再重複地運用在不同的資料上。

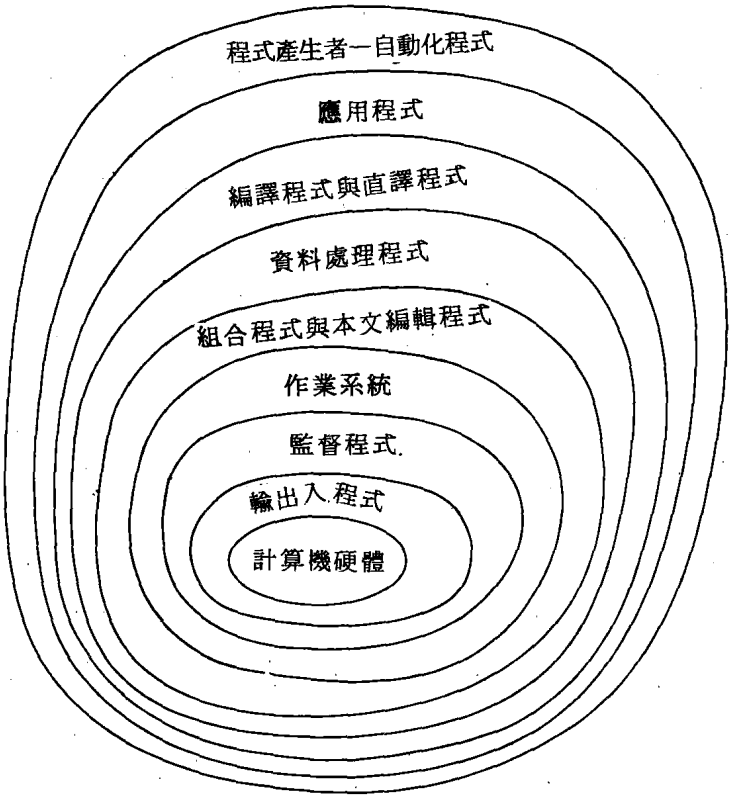

圖 1-1　系統軟體的階等，使用者與機器間隔着好幾層系統軟體

這簡單的觀念有另外一種含意。如果指令能存於機器上，則一種新的計算形態就成爲可能。重覆地計算變得相當容易，因爲你不再需要從鍵盤上重覆地打進指令。這種把指令存在機器裏面的能力使重覆

計算顯得很方便。

在1939年，在哈佛大學，由Howard Aiken領導；而由IBM 資助的一群人開始從事於第一部程式儲存式計算機（ Stored program computer）的製造。Mark I ，（完成於1944年）能以電的形式存指令，進行重覆計算。但它有兩個缺點。它的資料儲存於速度很慢的繼電器上。更嚴重的是，每次問題改變時，這機器便須重新接線。由於指令是以硬體的形式存放着，每次重新接線常花上好幾個星期。

現代計算機的另一重要觀念便是硬體不需要重新接線。它是固定的，但程式不是。資料與指令則以數據碼存放於機器的記憶體裏面。基本上，資料與指令的形式並沒有任何差異。John Von Neuman 在1940 年，從事於ENIAC的計劃時，構思出此一觀念。具有這種結構的機器能很從容不迫地處理新的問題。問題的解決已簡化爲如何寫出程式——一連串的數據碼來控制這機器。現今的計算機，基本上，其形態仍追隨Von Neuman 。只是速度比較快，體積比較大而已。

在計算機的啓始階段；程式的產生是一件很累的工作。機器指令的存入，完全靠手來調面板上的開關，以逐句存入。後來，程式則打在紙帶上。一簡短的載入程式（ Loader program）可以藉着板動開關存入。然後執行此載入程式，此時在紙帶上較複雜的程式便可輸入。

在那時，寫一程式是需要很長一段時間。修改錯誤更是麻煩。錯誤很容易發生，因爲機器碼實在很難辨認。即使，你已經發現了錯誤，仍然需要很長的時間去修改。

這種情形，實在無法令人滿意。後來，人們發現了從計算機的本身；可獲得許多幫助。這應該是一個具有革命性的觀念：讓計算機本身來幫助你寫自己的程式。至少有兩種重要的系統程式，在這階段發展出來：組合程式與小型計算機上ROM 監督程式（Monitor）的前身。