從TOOL KIT看APPLE系統

彭之瑞 著

聯星出版社 印行

從TOOL KIT看APPLE系統

譯　者：彭　之　瑞
出版社：聯星圖書公司
名　行：聯星圖書公司
地　址：香港洛克道168地下
印　刷：美華印刷廠
地　址：香港柴灣工業大厦二樓128號

序

只要是玩過 APPLE 的人必然都知道在 APPLE 主機板的後端有八個擴充插座（ expansion slot），只要在裏面插上擴充卡，就可以增加 APPLE的功能。譬如，只要插上漢卡就可以使 APPLE 變成中文電腦，具有產生中文字體的功能；而插上Z80卡則可使 APPLE 變成 CP／M 機器，可以執行CP／M 的所有功能。此外還有許多千奇百怪的擴充卡，一旦插入擴充插座都能提供一些額外的功能。類似的這種擴充卡，由於是硬體設備，我們看得見摸得著，因此大家便很能接受其擴充功能的觀念。但同樣的觀念，如果應用在軟體上，要了解並接受就不是一件容易的事了。因爲在一般人的觀念中，軟體並不像硬體能夠直接“插入”以獲得額外功能，軟體應該都是一套套的，功能都事先就定死的，所以講到擴充軟體，總沒有擴充硬體那麼好懂。本書就是要爲您說明這個觀念，藉著DOS TOOL KIT 中的幾個例子讓您了解軟體上的擴充，尤其是 APPLE 軟體的擴充有些什麼方法。

第一章主要是爲您介紹APPLE DOS 3.3檔案系統的擴充，

告訴你什麼是“可再定位檔”，同時藉著對 RBOOT 與 RLOAD 的介紹，引出一個電腦系統軟體中很重要的觀念—連結器（Linker）。

第二章則爲您介紹如何去擴充 APPLESOFT 的編輯功能，使 APPLESOFT 具有比 INTEGER 培基系統更靈活，更方便的程式編輯能力。而從這項介紹中，我們更希望您了解擴充的方法，進而自己動手爲 APPLESOFT 系統添加些您想要的功能。

第三章與第四章是一個單元，藉著DOS TOOL KIT 上的兩個程式HRCG與ANIMATRIX，我們將告訴您 APPLE 公司如何以軟體的方法來完成硬體的功能，這是個控制高解像度畫面的程式，雖然它只是個軟體，但感覺上就好像是硬體一般。

第五章則是藉著一個低解像度畫面控制程式，讓你了解控制畫面變化的技巧，這是爲了方便說明HRCG與ANIMATRIX 而設計的一個程式，當然你也可以利用第二章裏的方法，將它加入你的系統中。

以上是本書所要討論的主要內容。由這本書中，我們希望你能了解 APPLE 系統上軟體的擴充性與一貫性，同時我們也更希望你能了解軟體擴充在一個電腦系統中的重要性，最後謹祝各位開卷有益，日進有功！

著者 識

1984. 5. 1. 于左營

目錄

第一章

再定位載入程式
（RBOOT，RLOAD）

在APPLE 公司所出的DOS TOOL KIT 這一片磁碟上有兩個很奇怪的程式，一個是RBOOT ，另一個是RLOAD 。乍看之下，實在搞不清楚這兩個程式到底是幹啥用的，因爲在DOS中有兩個指令LOAD與BLOAD ，長得與RLOAD很像；而DOS的啓動過程英文爲BOOT ，似乎又與RBOOT 在名稱上極爲類似，常常使初用的人搞得迷迷糊糊的。偏偏APPLE公司所出的那本DOS TOOL KIT 的使用手册上又只有短短的一頁，寫得語焉不詳，只知道它是如何來使用，但到底怎麼回事，書上還是沒寫，讓人有“知其然，不知其所以然”的感覺。

這一章的目的，就是在說明這兩個程式的本末，從它們“何以會出現”，到如何靈活的使用，以至於這兩個程式的架構原理，我們都將有詳盡的介紹。末了，我將再介紹一些從RBOOT與RLOAD 的原理所延伸出來的一些重要觀念，假如你想深入了解電腦，而不只想玩一輩子的APPLE ，這些觀念對你將會非常有用的。

1-1 R類磁碟檔簡介

只要是對APPLE Ⅱ的DOS 稍有了解或是使用過的人，一定都知道，在DOS 中有四種磁碟檔，分別是A類、B類、I類、與T類。幾乎所有DOS 中的指令都是用來處理這四類磁碟檔。比方說，BLOAD、BSAVE、BRUN 是專門用來處理B類磁碟檔；而LOAD、SAVE、RUN 則可用來處理A類或I類的磁碟檔。至於處理T類磁碟檔的指令，則有OPEN、CLOSE、READ、WRITE、APPEND、POSITION 等指令，感覺上DOS 中似乎應該就只有這四類的磁碟檔。但假如你夠仔細的話，我想你一定發現到當你CATALOG DOS TOOL KIT 這片磁碟時，其中有2個程式既非A類或I類，也不是B類或T類，出現在其檔案名稱之前的是“R”這個字母，換句話說，它們是屬於R類的磁碟檔。很顯然這是一種少見的磁碟檔，而且在DOS 中沒有一個指令是可以處理這種檔案（當然類似DELETE 這種指令是例外的），RBOOT 與RLOAD 這兩個程式就是專門用來處理R類檔。

聰明的讀者你，一定會問，到底什麼是R類磁碟檔？它與其他類型的檔案有何不同？爲什麼DOS 在設計之初不將這種磁碟檔考慮進去，而要在事後補上這兩個程式呢？在還沒有回答這一連串問題之前，讓我先介紹一些旁的資料，來幫助你認識R類檔。

根據APPLE母公司的資料，DOS 在設計之初應該可以存在8種類型的磁碟檔。除了上述所提到的5類之外，其他的3類

分爲S類、新A類、及新B類。而在其檔案的索引（Directory）中則以一個 byte 的代碼來區別這8類的檔案，這8類磁碟檔的代碼分爲：

$00－　TEXT檔
$01－　INTEGER BASIC檔
$02－　APPLESOFT BASIC檔
$04－　BINARY檔
$08－　S類檔
$10－　R類檔
$20－　新A類檔
$40－　新B類檔

另外如果該磁碟檔是被鎖定的（Locked），則只須將代碼加上$80即可。譬如$04爲未鎖定（Unlocked）之二進位檔，而$84則爲鎖定的二進位檔，又比方說$10與$90則分爲未鎖定與鎖定之R類檔。

雖然APPLE公司在DOS中規劃了這8種類型的磁碟檔，但是在實際上却因爲種種的限制使得只有其中的4種檔案類型能夠充份的被定義出來，並完成於DOS之中。另外四種只能略加定義，或者根本就予以放棄掉。R類檔就是在這種情形下被擺在一邊好像棄嬰一般，直到TOOL KIT這片磁碟問世之後才稍爲受到些微的眷顧。APPLE公司寫了兩個程式（實際上是三個，另外一個是Assembler）RBOOT與RLOAD來載入R類檔，但是當你深入了解後，你將會發覺這兩個程式並無法完全發揮R類檔的功能。它們就好像一條不夠長的破棉被一樣，只能蓋在嬰兒的身子上，嬰兒的脚仍然露在外面受凍受寒。

故事說完了，讓我們言歸正傳，到底什麼是R類檔呢？R類檔其全名爲Relocatable File，翻成中文就是“可再定位檔”的意思。基本上，它是B類磁碟檔的延伸，用來存放機器語言之程式，但與B類檔不同的是，B類檔在主記憶體中相對的位置是固定的，而R類檔在主記憶體中的位置却是可以變化的，也就是說“可以重新定位”。

只要是對於組合語言（Assembly Language）稍有認識的人一定都知道，一旦組合語言程式經過譯碼器（Assembler）譯成機器語言碼的型態之後，這個程式在主記憶體中的位置隨之固定，隨意的改變其位置的結果，必然是使程式無法執行。要想改變這段機器語言程式在記憶體中的位置，一般而言只有一個辦法，那就是先找到該程式的原始組合語言程式，改變其啓始位址（Start Address），然後再經譯碼器重新譯過，來得到新位置的機器碼。這一個程序說起來很簡單，可是當你眞的照著步驟一步步來時，你會發現實在很麻煩。譬如說你是使用TOOL KIT中的編輯／譯碼器（Editor／Assembler）系統，那麼你必須要，

(1)進入編輯器（Editor）中，並將原始程式讀入。

(2)修改原始程式，然後將程式存起來。

(3)進入譯碼器中，將原始程式譯成機器碼。

(4)重新載入新的機器碼。

小程式還好，修改一次不會花太多的時間。但如果碰到大程式的話，你就累了（事實上，應該說“磁碟機就累了”），因爲除了編輯／譯碼系統的載入外，你還要做五次的磁碟讀寫的動作。而其中有三次的讀寫是與原始程式有關，有經驗的人一定知道，原

始程式檔是屬於T類檔，也就是那種體積最龐大，操作起來最費時費力的檔案類型。保守的估計，如果你的機器碼有1K byte的大小的話，那麼修改一次起碼花上你五分鐘的時間。而一般像樣一點，能眞正做些事的程式，至少都有10K 的大小。

於是我們不禁要問，是不是眞的只有這個辦法才能改變一個程式在記憶中的位置，沒有其他較簡單的方法嗎？答案是有，只要我們在譯碼時，能夠附帶的加上些資料，標明了程式中有那些地方是當程式被移動位置後，會發生錯誤而需要加以修正的，就可以了。如此我們只需要在載入程式時，稍加注意並適當的修正就可以了。譬如下例，

```
*800L

0800-   A9 00      LDA   #$00
0802-   A2 05      LDX   #$05
0804-   CA         DEX
0805-   30 07      BMI   $080E
0807-   18         CLC
0808-   7D 12 08   ADC   $0812,X
080B-   4C 04 08   JMP   $0804
080E-   8D 17 08   STA   $0817
0811-   60         RTS
0812-   01 02      ORA   ($02,X)
0814-   03         ???
0815-   04         ???
0816-   05 00      ORA   $00
0818-   00         BRK
0819-   00         BRK
081A-   00         BRK
081B-   00         BRK
081C-   00         BRK
081D-   00         BRK
O
*
```

這個程式原來是在$ 0800 到$ 0817 中，如果我們將之移到$ 0900則程式執必當掉，當掉的原因是因爲在$ 0808，$ 080B及$ 080E三個地方，有三個絕對位址，要使程式能正確執行，我們必須將$ 080A，$ 080D，$ 0810 三個位址內的$ 08 改成$ 09 ，成以下之情形，

```
*900L

0900-   A9 00      LDA   #$00
0902-   A2 05      LDX   #$05
0904-   CA         DEX
0905-   30 07      BMI   $090E
0907-   18         CLC
0908-   7D 12 09   ADC   $0912,X
090B-   4C 04 09   JMP   $0904
090E-   8D 17 09   STA   $0917
0911-   60         RTS
0912-   01 02      ORA   ($02,X)
0914-   03         ???
0915-   04         ???
0916-   05 00      ORA   $00
0918-   00         BRK
0919-   00         BRK
091A-   00         BRK
091B-   00         BRK
091C-   00         BRK
091D-   00         BRK
091E-   00         BRK
*
```

因此，我們在譯碼時只要標明這三個位址內的資料可能會有問題，如此則不論程式被搬到那裏將都不會產生問題。

R類檔的原理就是這樣，但APPLE公司的做法則略微複雜些。由於並非所有的程式都需要具有這種“隨遇而安”的特性，

所以譯碼器並不需要在所有的機器語言碼的後面都加上這些資料。APPLE 公司的做法是，如果你需要一個程式具備這種可以移來移去的特性，那麼你在翻譯原始程式時，必須先通知譯碼器，讓譯碼器在產生機器碼之外，還附上我們所需要的資料；但假如你不需這種特性，那麼你就無需知會譯碼器，譯碼器會自動依其初值，只產生“不可再定位”的機器碼。在這種做法之下，譯碼器將會產生兩種機器碼，前面的那種就是R類檔，而後者則是B類檔。

B類檔中程式的載入比較簡單，因爲其位置是固定的，所以我們只要直接從磁碟中讀出，然後將之放到其特定的位置上即可。至於R類檔中程式的載入，顯然就要複雜得多了。首先，由於DOS 無法載入R類檔，所以必須另外用一個載入程式來達成這個功能。但使用程式來載入另一個程式，却會產生許多問題，

(1)我們要安排一個適當的位置來放這個載入程式，使得R類檔中程式的載入不受影響，而且，

(2)爲了能夠在程式執行中載入一個R類檔，這個載入程式必須要能不佔用原執行中程式的位置。

這是兩個很頭痛的問題，尤其是第2個問題，因爲我們無法預估一個培基程式的大小，所以載入程式位置的決定勢必困難。讓我們來看看 APPLE 公司是如何克服這個問題的。

1-2 如何載入R類磁碟檔

在DOS TOOL KIT 的使用手册中，簡單的介紹了利用 RBOOT 與 RLOAD 這兩個程式來載入R類檔的標準程序，首先

如果我們要載入一個稱爲MYMODULE的R類磁碟檔，那麼我們只要在程式中加入下面一個程式片斷，

```
100  ADRS = 0
110  PRINT CHR$(4);"BLOAD RBOOT":
     CALL 520
120  ADRS = USR(0),"MYMODULE"
```

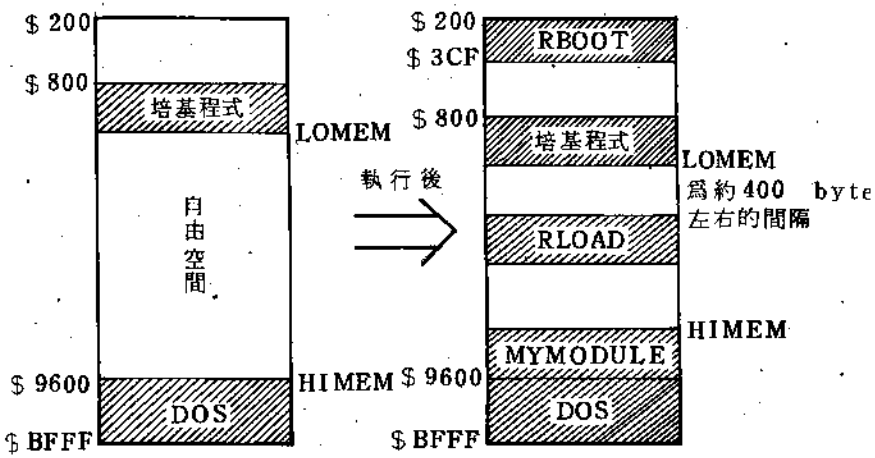
在這段程式執行過後，APPLE會將MYMODULE載入主記憶體中自由記憶（Free Memory）的最頂端，然後利用修改HIMEM的方法將MYMODULE給藏起來，最後將MYMODULE之啓始點傳給ADRS。

我們先解釋一下這三行程式都在幹些什麼，行號100很簡單，將ADRS設定爲0，

行號110，將RBOOT載入主記憶體中，位置是在$208到$3CF的一段記憶中，再利用CALL 520（$208）來執行此程式，故此行改爲PRINT CHR$(4);"BRUN RBOOT"亦可。RBOOT執行結果是RLOAD會被載入記憶體中，載入之位置則根據系統中原存在之培基程式的大小而變動，如果培基程式很大，則RLOAD會被放在後面一點的地方；反之如很小，則會被放在前面一點的地方（如何達成，後面會說明）。載入後RLOAD的啓始點會被放入（$0B，$0C）之中，這也就正是USR函數的啓始點。

行號120，利用USR函數來執行剛載入的RLOAD。執行的結果爲MYMODULE被讀到記憶之中，位置是在系統原先HIMEM的前方。當MYMODULE被載入後，它的確實開始位址（Start Address）會被當做USR函數的值給傳回去，於是

ADRS中將會保有此值，另外系統的HIMEM則會被修改到ADRS－1，以便將MYMODULE給藏起來。



上圖是這段程式執行前與後，主記憶中各個程式的分佈情形，看了之後，你應該略有概念了吧！

從以上說明，我們可以看出，眞正將MYMODULE載入的是RLOAD而不是RBOOT，因此RLOAD才是我們在前一節中所提到的“載入程式”。爲了克服前一節中所列出的兩個困難，APPLE公司的辦法是，讓RLOAD也可以“到處放”，也就是可再定位，使RLOAD永遠與主記憶體中的培基程式保持一小段安全距離，約在256 byte到512 byte之間，這樣既可降低其對MYMODULE載入之影響，又可以不佔到原有培基程式的空間。而爲了使RLOAD可以具有隨遇而安的特性，因此APPLE公司特別寫了RBOOT來達成這項功能。