電脳技術叢書

微電脳連結程式

工業技術研究院 電子工業研究所 編印

序　言

電腦技術發展中心一般系統部所出版的技術叢書系列，目的在將本中心內研製計劃有關之技術資料公諸社會，其一方面可供各級學校利用，另一方面也可供工商研究機關參考。電腦技術發展中心叢書之出版，盼望能給目前電腦工業發展帶來直接助益。

本書爲叢書之一，目的在介紹微電腦連結程式結構，書中所採實例，則以目前本中心現有之微電腦連結程式爲主

本書在編寫期間，承蒙凌淑明小姐協助整理，在此致謝！

周　進　興

識於電子工業研究所電腦技術發展中心一般系統部

中華民國六十九年十月一日

目　錄

第一章　概論

原始程式經過組合程式（ Assembler ）或其他語言的轉譯程式（ Translator ）處理後，即產生此原始程式的目標程式（ Object Program ）；目標程式通常存放在磁碟或磁帶上，此時之目標程式尚不能在記憶體上執行，若要使其執行，則要經過下面四個步驟的處理：

1. 記憶體定位要求（ Memory Allocation ）—要求置定一可用記憶體空間，使該程式能進入主記憶體內執行。
2. 連結（ Linking ）—解決目標程式內之各目標模組間，副常式（ Subroutines ）和資料共用的問題。
3. 重新定址（ Relocation ）—目標程式內的定址，通常都自最低位址開始定址，此並非使用者所要求程式執行時的起始位址，故需重新定址。
4. 饋入主記憶體（ Loading ）—將該程式的機器碼（ Machine Code ）放置在主記憶體上。

經過上述四個處理步驟，一個可執行的程式已饋入主記憶體，此時則可開始執行此程式。

本書所要討論的微電腦連結程式（ CMC LINKER ），其功能與上述大抵相同，差別僅在連結程式並不將目標程式的機器碼，放置在主記憶體上，而是將整個目標程式的機器碼，建立一個檔案，存放在磁碟上；將來要執行時，才將此檔案饋入主記憶體執行；故連結程式的主要功能如下：

1. 連結目標程式內，各目標模組間之外部參考符號（ External Reference Symbol ）。
2. 重新定址。
3. 建立程序檔（ Procedure File ）—亦即機器碼檔。

而程序檔要執行時，才由作業系統（ Operating System ）做記憶體定位要求，並將程序檔饋入主記憶體，開始執行。以下分別說明連結程式之功能、特性、資料結構、程式結構及詳細流程圖。

第一節　連結程式之結構

連結程式之結構圖，如圖 1 — 1 所示：

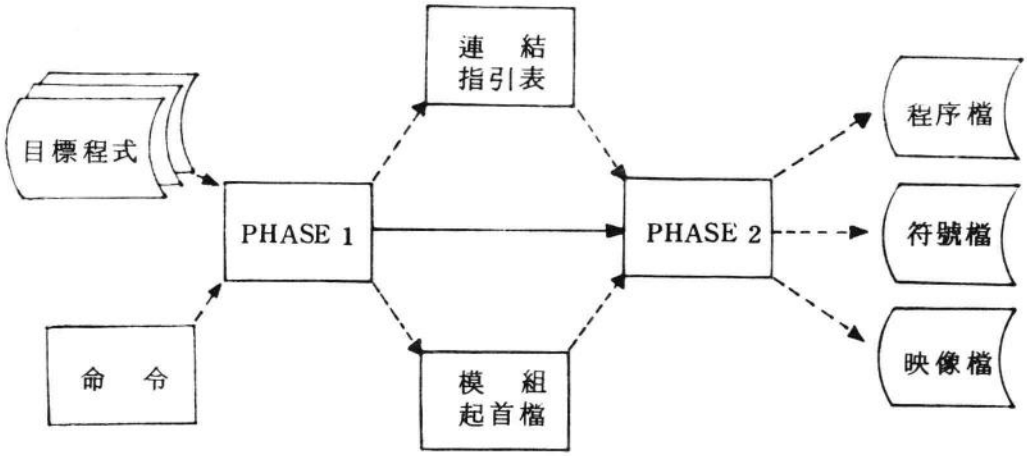


圖 1－1

連結程式可處理一個或一個以上的目標程式檔，其處理過程中，可分爲兩個 PHASE，在PHASE1 中，主要建立兩個表，即連結指引表（ Link Directory ）和模組起首表（ Module Header Table ），以供 PHASE 2 做重定址和外部符號 連結之用；PHASE 2 的主要工作爲連結（ Linking ），重新定址和建立三個檔案，即程序檔（ Procedure File ），符號表檔（Symbol File）及映像檔（ Map File ）；程序檔供將來執行用，符號表檔以備符號偵錯（ Symbolic Debug ）時用，映像檔供使用者參考用。連結程式解決目標程式重新定址的問題，並提供目標程式間副常式及資料共用的能力。

第二節　連結程式之特性

1. 程序檔的記錄長度（ Record Length ），可由用者自行定義爲 128，256，512 或 1024 個位元組（ Bytes ）。整個程序檔可分成一個以上，而不超過 16 個的段（ Segment ），每一個段可存放在可用記憶體空間（ Free Memory Space ）的任意地方；換句話說，程序檔是依段，分別饋入主記憶體內，因此其在記憶體內，不一定爲連續的。
2. 連結程式有能力將絕對位址目標程式（ Absolute Object Program）和可重新定址目標程式連結在一起，建成一個程序檔。
3. 用者可自行設定程序檔內，各目標程式的機器碼將來饋入記憶體內的起始位址。
4. 用者可自行設定程序檔的開始執行位址（ Entry Point ）。

5. 連結程式有能力檢查目標程式的總體符號（ Global Symbol ）是否重覆定義，及是否有無解的外部符號（ External Symbol ）—即在其他的目標程式內，找不到該外部符號所對應的總體符號，若有上述兩種錯誤，則將警告訊息（ Warning Message ）輸出在邏輯單元 2（ Logic Unit ）與映像檔上。
6. 提供僅連結（ Link Only ）的功能；當用者指定某一個目標程式爲僅連結時，連結程式只建立模組起首表及連結指引表，程序檔內，并不包含此目標程式檔的機器碼。
7. 建立映像檔，供用者參考。
8. 依用者的意思，決定是否要建立符號檔，此符號檔供符號偵錯用。

第三節　連結程式命令（ Command ）使用法

例一：LINK　*$*=5000　PRG1　PRG2　（ NOW NOM sy=SYMBOL ）

例二：LINK　*$*=5000　PRG1　*$*=*$*+1000　PRG2　（ RL=100 ST=100　P ）

例三：LINK　PRG1　PRG2　（ E=1000　LET　M=MAPFIL N=PROFIL ）

當用者在控制台（ Console ）輸入連結程式命令時，作業系統根據此命令，將程序檔自磁碟饋入主記憶體執行。連結程式的命令說明如下：

1. *$*=5000 及 *$*=*$*+1000

用者可自行設定各個目標程式連結時的起始位址；若未設定，則第一個目標程式自 0000 開始定址，以下的目標程式則依序定址。

2. E=1000 或 E=總體符號名稱（ Global Symbol Name ）

用者可自行設定程序檔的開始執行位址（ Entry Point ）；此開始執行位址，可爲一個十六進位的數值或爲一總體符號名稱。若用者未設定開始執行位址，則其位址被設定爲第一個目標程式的第一個機器碼。

3. LET

若連結程式在執行時，發生了錯誤，而此錯誤屬非致命（ Nonfatal ），則不執行 PHASE2，但若使用者在連結程式命令中有此參數，則强迫連結程式執行 PHASE2 。

4. M＝（ 檔名 ）　N＝（ 檔名 ）　SY＝（ 檔名 ）

用者可自行設定各個檔案的名稱。M 表示映像檔，N 表示爲程序檔，SY 表示

爲符號檔。若用者未設定各檔之檔名，則參考第一個目標程式的名稱。例如：第一個目標程式名稱爲 TEST .OBJ ，則程序檔名稱爲TEST，映像檔名稱爲 TEST .MAP，符號檔名稱 TEST .SYM 。若用者只定義了程序檔的檔名，依序加.MAP和SYM。

5. NOM

表示不建映像檔。

6. NOW

不需將警告訊息（ Warning Message ）輸出至邏輯單元 2 及映像檔。

7. RL = n

用者可自行設定程序檔的記錄長度（ Record Length ）爲 128，256，512 或 1028 個位元組，若未設定，則記錄長度爲 128 個位元組。

8. ST = n

用者可自行設定堆列（ stack ）暫存區的大小，若未設定則其大小爲 128 。

9. P

將映像資料由邏輯單元 3 顯示出來。

第二章　資料結構

連結程式的執行過程分爲兩個 PHASE ，各 PHASE 的主要工作如下：

PHASE1：

1. 建立模組起首表和連結指引表；此二表供 PHASE 2 做連結和重新定址之用。模組起首表主要記錄各目標程式的連結起始位址，及目標程式內機器碼的長度；連結指引表爲各目標程式的總體符號及外部符號（External Symbol）的集合，含該符號的值。
2. 在模組起首表及連結指引表內，設定各目標程式和總體符號的絕對位址。
3. 在連結指引表內，設定符號的值，即在連結指引表內，找出該外部符號所對應的總體符號，取出其值，存入外部符號之值欄。

PHASE2：

1. 將目標程式內的機器碼，重新定址：
2. 利用連結指引表，解決有外部符號參考的機器碼。
3. 建立程序檔，映像檔，符號表檔。

以下分別敍述各個表及檔案之格式（Format）

第一節　目標程式檔之格式

一個完整的目標程式檔，應具有四項資料如圖 2 — 1 所示

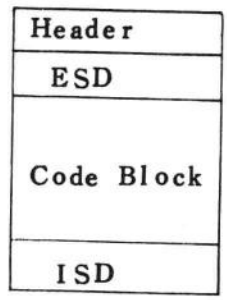


圖 2 — 1

1. 標頭：佔 16 個元組，各個位元組定義如下：

   位元組 0 — 80H 表爲絕對目標檔（Absolute Object File Mark），

   81H 表可可重定位目標檔（Relocatable Object File Mark）。

   位元組 1 — 未利用。

位元組 2 ，3 —存放此檔內目標碼的長度（單位爲位元組）。

位元組 4 ，5 —記錄 ESD（External Symbol Dictionary） 符號的個數。

位元組 6 ，7 —存放第一個碼塊（Code Block)的記錄編號。

位元組 8 ，9 —若此檔爲絕對目標檔，則此二位元組存放最低絕對位址（Lowest Absolute Address ）。

位元組 16,11 —未利用。

位元組 12,13 —記錄 ISD（Internal Symbol Dictionary）符號的個數。

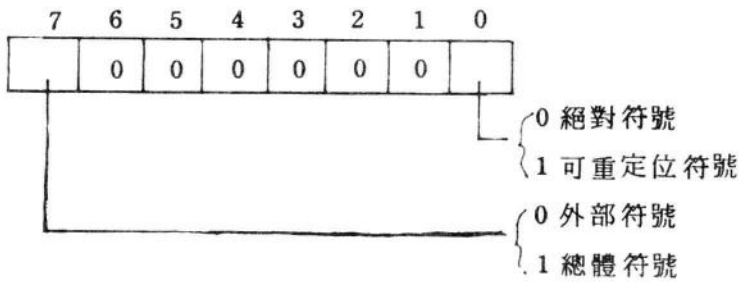
位元組 14,15 —存放第一個 ISD 組塊的記錄編號 。

2. E S D 塊

所佔的空間長度不固定，隨 E S D符號的多寡而定。E S D符號爲原始程式內宣布爲 GLOBAL 或 EXTERANL 的符號。

每一個 E S D符號的格式如下：

位元組 0 —屬性位元組（Attribute Byte）



位元組 1 —符號名稱的長度（單位爲位元組）。

位元組 2 ～ n － 3 —符號名稱 。

位元組 n － 2 ， n － 1 —符號值，若此符號，被宣告爲總體符號，則符號值爲定義此符號的位址；若此符號被宣告爲外部符號，則符號值爲其在 E S D符號表的相對位址。

3. 碼塊（Code Block）

長度隨程式的大小而定，其長度單位爲記錄，每一記錄佔 128 個位元組其格式如下：

位元組 0 —碼標（Code Mark ）。82H 表碼組塊記錄，83 H 表最後一個碼組塊記錄。

位元組 1 ，2 —存放在此碼組塊記錄內，第一個目標碼（Object Code） 的參考位址值。

位元組 3 —存放在此碼組塊記錄內，目標碼的長度，單位爲位元組。

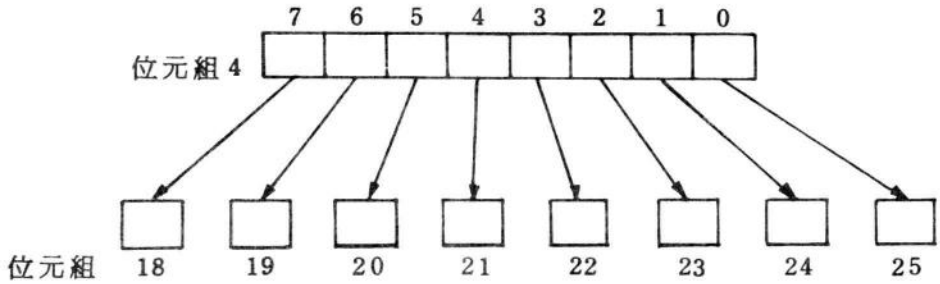
位元組 3　　—存放在此碼組塊記錄內，目標碼的長度，單位爲位元組。

位元組 4 … 17　—位元映像（Bit Map）。

位元組 18 … 126—存放目標碼（Object Code）。

位元組 127　　—此位元組爲檢核位元組（ Checksum Byte ）。將檢核位元組之前的位元組相加，再加此檢核位元組，其和爲零。

位元映像與目標碼的對應關係如下：

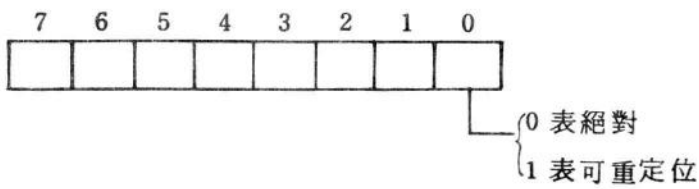


當位元映像的連續兩個位元爲 1 ，0 時，則表示對應的目標碼，於連結時可被重新定址。若連續兩個位元爲 1 、1 時，則表示對應的目標碼爲外部引用（External Reference）符號，接下去的兩個目標碼爲一個指標，指向該符號在 ESD 的位址。爲 0 、0 時，則表示爲絕對符號或機器碼，將來連結時，對應的目標碼不須重新定址。

4. I S D

所佔的空間長度不固定，隨 ISD 符號的個數而定。ISD 符號爲原始程式內，所有在址標欄的符號，扣除經宣告爲GLOBAL的符號，亦即該符號完全爲程式內參考之用。ISD符號的格式如下：

位元組 0 —屬性位元組



位元組 1 —存放符號名稱的長度，單位爲位元組

位元組 2 … n − 3 —存放符號名稱

位元組 n − 2 … n − 1 —存放符號值

第二節　模組起首表之格式

每一個目標程式在模組起首表內佔用10個位元組，其資料大多數由該目標程式的標頭（Header）塊而來，部份資料是由用者的命令而來，此10個位元組之資料，效分述如下：

位元組0、1—存放該目標程式外部參考符號塊（ESD Symbol Block）在連結指引表的起始位址。

位元組2、3—存放該目標程式的連結起始位址。

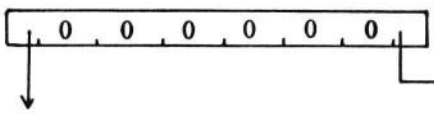
位元組4、5—存放該目標程式內碼段（Code Block）的第一個記錄之編號，其中位元組5之位元6表示該目標程式檔是否有ISD塊，位元組5之位元7，表示該目標程式檔是否爲絕對位址的目標程式檔（Absolute Object Program）。

位元組6、7—存放該目標程式檔內，ESD之符號個數。

位元組8、9—存放該目標程式檔，碼段內機器碼之長度，單位爲位元組。

第三節　連結指引表之格式

連結程式組合各個目標程式檔內之ESD塊，建立連結指引表，每一符號之值，皆已轉換成絕對位址（Absolute Address），每一個符號之長度不固定，其格式如下所示。

位元組0 —— 

位元組1　——存放符號名稱長度

位元組2～n－3—存放符號名稱

位元組n－2　n－1—存放符號值

第四節　符號檔之格式

連結程式組合各個目標程式檔的名稱，總體符號及ISD塊，建立符號檔，以供符號偵錯（Symbolic Debug）時使用。各個符號之長度不一致，其格式如下所示.

目標程式檔名稱之格式：

位元組 0 — 04H 表絕對位址程式，

05H 表可重定位程式 。

位元組 1 —存放符號名稱之長度 。

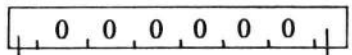
位元組 2 ～ n－5 —存放符號名稱 。

位元組 n－4 ～ n－3 —存放符號值 。

位元組 n－2 ～ n－1 －0000

總體符號與內部符號（ Internal Symbol ）之格式：

位元組 0 —



位元組 1 —存放符號名稱之長度

位元組 2 ～ n－3 —存放符號名稱

位元組 n－2 ～ n－1 —存放符號值

第五節　映像檔之格式

現舉例說明映像檔之格式如下：

```
LINK 1.6
POSSIBLE CODE ONERLAY AT 5000 IN PROG2

LOAD MAP
MODULE          ORIGIN          LENGTH

PROG1           4400            1121
PROG2           5000            0200
PROG3           6000            0345


GLOBAL          ADDRESS         MODULE

BELL            4831            PROG1
IGRD            5133            PROG2
SYM             6245            PROG3
PROGRAM         PROG1 —— 1F80 BYTES
ENTRY : 4400
```

映像檔之內容包含有：

1. 警告訊息。
2. 各目標程式之起始位址及長度。
3. 各目標程式內，總體符號的位址和定義此總體符號的目標模組名稱。
4. 程序檔的名稱及程序檔的長度。
5. 程序檔之開始執行位址。

第六節　程序檔之格式

程序檔內之資料，完全是可執行的機器碼，可直接饋入主記憶體內執行；其饋入主記憶體是分段饋入的，因此程序檔內的機器碼是有段落性的。程序檔內有一描述記錄（ Descriptor Record ），用以記錄程序檔內各個分段的起始位址和長度；作業系統依據此描述記錄，將程序檔內的資料饋入主記憶體內執行。

程序檔之描述記錄佔128個位元組，其格式如下：

位元組 0～3 —未用。

位元組 4～5 —存放檔案定義號碼。

位元組 6～7 —指向目錄磁區（ Directory Sector ），該磁區包含此檔之進入點（ Entry Point ）。

位元組 8～9 —指向此檔案的第一個記錄。

位元組 10～11 —指向此檔案的最後一個記錄。

位元組　12　—存放檔案之類別。

位元組 13～14 —存放檔案內記錄的數目。

位元組 15～16 —存放記錄內位元組之數目。

位元組 17～18 —存放塊之長度。目前未使用，其值同於位元組 15～16 。

位元組　19　—存放此檔之保護性（ Protection Property ）。

位元組 20～21 —存放程序檔之開始執行位址。

位元組 22～23 —存放檔內最後一個記錄所含位元組的數目。

位元組 24～31 —存放此檔建立之日期。

位元組 32～39 —存放更新（ Update ）此檔之最近日期。

位元組 40～40+4×n−1 —1≤n≤16，n 爲分段描述（ Segment Descriptor ）的數目，每一個分段描述佔四個位元組，前兩個位元組存放此分段的起始位址；後兩個位元組存放此分段

的長度（單位爲位元組）。

位元組122～123—存放最低位址分段之起始位址。

位元組124～125—存放最高位址分段之結束位址。

位元組126～127—存放堆列（stack）暫存區之大小。

第三章　連結程式之系統結構

本章將以一具體之實例，逐步分析，說明微電腦連結程式之結構。

第一節　連結程式之功能

連結程式的主要工作，包括有下列十一項：

1. 保存返回作業系統之位址。
2. 在邏輯單元號碼爲 2 之裝置（通常定義爲控制台）顯示字串 'LINK 1.6' 。
3. 要求作業系統配置（ Allocate ）所有的可用記憶體空間。
4. 依據使用者所輸入的連結程式命令資料，設定命令旗標（ CMDFLG ）值，以供日後程式執行之依據。
5. 設定連結指引表暫存區之位址，並清除之。
6. 建立模組起首表及連結指引表，此時僅連結指引表內之總體符號被定址。
7. 對連結指引表內的外部符號做定址工作。
8. 重新定址目標程式內之所有指令運算元之位址；並建立程序檔，映像檔及符號檔。
9. 將錯誤數目訊息顯示在邏輯單元號碼爲 2 之裝置上；並關閉映像檔然後將連結程式完結的訊息顯示出來。
10. 對作業系統的可用記憶體空間做去位（ Deallocate ）工作。
11. 取出作業系統的返回位址，並依此位址返回作業系統。

連結程式在處理過程中，隨時會遭遇到用者之命令資料或所要連結之目標程式有錯誤的情況；這些錯大致可分爲兩類，一爲致命的錯誤（ Fatal Error ）和非致命錯誤（ Non-Fatal Error ）；致命的錯誤情況有下列八種：

錯誤編號	錯誤訊息	錯誤原因
ERROR1	INVALID OPTION : ×××	用者輸入之命令資料有錯誤的參數
ERROR2	LINK DIRECTORY OVERFLOW	可用記憶體空間等於零或太小，不足以做爲連結指引表之暫存區
ERROR3	INVALID FORMAT :（檔名）	目標程式檔的資料型態不是 Binary 型態

ERROR4	INVALID DATA：（檔名）	所要連結的目標程式檔內資料不合理；例如符號名名稱之長度爲零
ERROR5	TOO MANY SEGMENTS	程序檔之分段超過16個
ERROR6	PROGRAM TOO BIG	某一分段或某一指令所欲連結之位址，大於記憶體之最高位址0FFFFH
ERROR7	INVALID FILE：（檔名）	所要連結之目標程式檔名稱有不合法的字串存在
ERROR8	FILE NOT FOUND：（檔名）	在磁碟裏找不到該目標程式檔

在致命的錯誤情況下，依錯誤的類別，跳至不同的錯誤處理程式執行，這些程式計有ERROR1 ～ ERROR8八個程式，所處理的工作大致相同：

1. 在控制台上顯示出適當的錯誤訊息。
2. 關閉已打開（Open）的檔案。
3. 做可用記憶體空間去位之工作，然後返回作業系統。

非致命的錯誤情況有兩種：

1. 在兩個目標程式檔內，定義同一個總體符號名稱，此時產生之錯誤訊息爲：

MULTIPLY-DEFINED GLOBAL IN MODULE：

（符號名稱）　（檔名）

⋮　⋮

2. 目標程式所用之外部符號，在其他目標程式內，沒有定義爲總體符號，此種錯誤之訊息爲：

UNRESOLVED EXTERNAL IN MODULE：

（符號名稱）　（檔名）

⋮　⋮

非致命的錯誤情況，連結程式會將錯誤訊息顯示在控制台上，並將此錯誤訊息輸出至映像檔；當連結程式偵測到有非致命的錯誤情況發生，則不進入PHASE2執