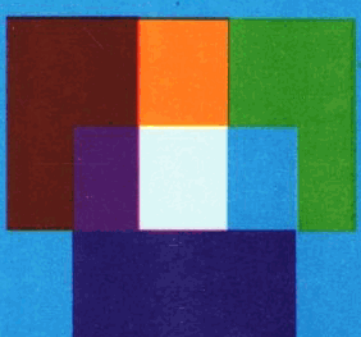
IBM個人電腦 COBOL 程式設計

劉文麒 朱明珍 編譯

Programming the IBM Personal Computer: COBOL

Neill Graham



復文書局

簡介

本書爲IBM個人電腦COBOL中的程式設計簡介（與IBM COBOL程式有關），而此種程式設計語言是爲寫出商用程式而設計的程式語言。

爲什麼要學習另外的程式設計語言呢？我們眞的需要大約175種程式語言嗎？當然我們能設計一種語言，使我們能表達電腦所能夠實現的指示。此種語言確實存在而它就叫做組合語言（或機器語言），但卻很難使用。

組合語言是很困難的，因爲它需要代替電腦內部工作的指示。換言之，諸如BASIC, PASCAL與COBOL之類的高階語言使我們能代替電腦所作職務的指示。既然電腦用在許多困難的領域中，且有其本身的目標與符號規定，故我們需要許多的程式設計語言，使在不同領域中的工作者能與電腦自然地溝通。

商用的程式設計對程式設計語言作出了特殊的需求。商業的程式處理了大量的資料，即所有顧客的記錄，或公司的所有雇員，例如，商業導向的語言應提供方便的方法來操作檔案與記錄。商業計算的結果通常總結在報表中，所以造出清晰的格式報表將是必要的。而數值應印在格式中，而有錢號、逗號、小數點在位置中，而且正負號也需要。在英文的字句中所描述的數學計算將是可能的，而此爲商業所使用的，它代替了較適合科學性程式設計的數學公式。

算術中所使用的數系系統特別地重要。許多的程式設計語言在二進位系統中作算術，而它是對電腦處理而言，是最有效的系統。不幸地，一些重要的商業算術運算，例如四捨五入，將代替熟悉的十進位系統的定義，而在二進位系統中實現。當結果有稍微地不精

確時，可在計算中捨去，例如預測、估計與模擬皆是，當要計算付款或收款的精確數量時，後面的計算通常爲正確的。所以，商業導向的程式設計語言應提供十進位的算術。

COBOL，即通用商業導向語言它是廣泛使用的程式語言，而在大型的電腦上，作商業的資料處理。大部分每個月所收到的帳單或帳務報表皆由COBOL程式所產出。許多個人電腦的商用程式是用BASIC語言所寫的 它比較不能提供諸如十進位算術，檔案與記錄之類的輕便操作，與像英文敍述之類的數學計算此種重要的結構。

IBM COBOL提供了標準COBOL中最有用的結構。IBM COBOL對與使用者介面，提供了標準COBOL之類富有威力的擴充。標準COBOL爲整批處理導向的， 即處理儲存在打卡、磁帶或磁碟上的大型檔案。微電腦需要經由鍵盤與顯示螢幕，來處理大型的檔案，與使用者溝通。IBM COBOL 擴展了標準COBOL的ACCEPT敍述， 來接受輸入，並在螢幕上的任何點顯示輸出。提示顯示了使用者輸入資料的適當格式，例如出現在小數點左右方的有多少位數。不在適當格式中的資料要自動地拒絕。而特殊的螢幕段落提供了方便的方法來特定每個顯示螢幕的裝置。

與COBOL編譯器有關的參考手册描述了每一個語言結構的技術性細節，但它並不會在COBOL中教你程式。本書可與撰寫的COBOL程式相對照，可在參考手册中討論使用不同的結構， 而製造出特殊職務的程式。既然本書並不能代替參考手册，故它將不會涵蓋著IBM COBOL 的所有結構，也不會討論到每一個結構的技術性細節。

因爲COBOL是相當大而複雜的語言，它似乎對初學者而言並不太適合。在拼音、頓號與打出行中導致錯誤上程式建構的適當定位。要對COBOL有感覺的最好方法是模仿正確的程式。在

電腦中打出每一個範例程式，並加以編譯與執行。要注意在開始時，只在正確的格式中打出COBOL程式即爲非常有用的習題。在你有了程式之後，試一試修改一點不同的職務，例如來處理不同種類的資料。在本章中，作出每一個程式之後，試一試作出本章的習題，它通常與本章所討論的程式很類似。此種步驟將幫助你對COBOL有一點點感應，而不只是要一些技術性細節而已。

在本書中的教材應用了使用磁碟片儲存，與使用固定磁碟儲存（例如IBM個人電腦XT型）的系統；它也應用了DOS 1.10與DOS 2.00。第一章首先描述了如何編譯COBOL程式，而使用磁碟片儲存與DOS 1.10，當使用固定磁碟儲存與DOS 2.00時，也給出了所需要的附加性考慮。在第二章至十一章中，“磁碟”可參照磁碟片或固定磁碟，而此程式在DOS 1.10與DOS 2.00中同樣地運轉。

最後感謝朱明珍作了校對的工作，劉文麒作了修改稿件的工作，而使本書的錯誤能儘可能地減少，但謬誤之處在所難免、尚望海內外先進，能提供參考意見。

林傑斌
陳奇麟
劉文麒

1985年10月於台北

目錄

第一章
使用 COBOL編譯器

在電腦能跟隨COBOL程式中的指示之前，此程式應編譯出來，而翻譯成容易處理的碼。要撰寫與編譯程式有一些步驟。你可以由打出原始程式（Source program）開始，則你所撰寫的COBOL程式會儲存在磁碟片上。這可由諸如EDLIN之類的文件編輯器（Text editor）的輔助來完成，而此文件編譯器提供在DOS（Disk Operating System）磁碟作業系統的磁碟片上。再來，你可使用COBOL編輯器把你的原始程式翻譯成目標程式（Object program）。目標程式並不完全，且並不包含許多COBOL程式庫的子程式（Subprograms）。你可使用連接器（Linker）來結合由程式庫中需要子程式的目標程式。此結果爲可執行的程式（executable program），它能在運轉時間模組（runtime module）的控制之下，運轉或執行。本章將專注於打出、編譯，連接與執行程式的細節。

1-1 準備磁蝶片 (Preparing the Diskettes)

要編譯與運轉COBOL程式的話，你將需要DOS磁碟片附加至兩個在COBOL編譯器套裝軟體中所供應的兩個磁碟片。你也將需要供應空白的磁碟片，來作出編譯器磁碟片的備用拷貝，並儲

存你所撰寫的COBOL程式中。

COBOL編譯器套裝軟體包含了兩個磁碟片，即COBOL 磁碟片（它包含了COBOL編譯器）與庫存磁碟片（它包含了COBOL庫存，連接器與運轉時間模組）。要預防磁碟外的意外損害的話，你應作出每一個磁碟片的備用拷貝，把原始的磁碟片，與備用拷貝放在安全的地方，並使用備用拷貝來編譯COBOL程式。

你將發現到，要創造COBOL磁碟片的版別是很方便的，它包含了DOS與通用的公用程式，例如FORMAT與DISKCOPY，與 COBOL 編譯器，你能使用此磁碟片來起始你的電腦，並執行常式與編譯COBOL程式。要作出此磁碟片的話，把你的DOS磁碟片放在A號驅動器中，並把空白的磁碟片放在B號驅動器中。用／S取向格式化空白的磁碟片，它導致了放在格式化磁碟片上的作業系統：

```
A>FORMAT B:/S
```

（當描述了DOS指令時，我通常將包含DOS提示行A>與B>，即使他們是被電腦打出，而不是被使用者打出。）再插入COBOL磁碟片在A驅動器中，並拷貝五種編譯器檔案至新的格式化磁碟片中。

```
A>COPY A:COBOL?.* B:
```

最後再插入DOS磁碟片在A驅動器中，並拷貝在新磁碟片上所需要的公共程式，例如：

```
A>COPY A:CHKDSK.COM B:
A>COPY A:FORMAT.COM B:
A>COPY A:DISKCOPY.COM B:
A>COPY A:DISKCOMP.COM B:
```

由DOS磁碟片至庫存磁碟片中，拷貝COMMAND，COM，與DOS 指令解譯器程式。連接器在記憶體中重覆撰寫的程式，而

且你的COBOL程式通常也如此作。把COMMAND　COM放在LIBRARY磁碟片中，允許在連接器或COBOL　程式運轉之後，自動地重新負載。

格式化另外的磁碟片，將要參照抓取磁碟片（Scratch diskette）。此種抓取磁碟片將保留你的 COBOL 原始目標，與可執行的程式，與由編譯器與連接器所創造的暫時工作檔案。要拷貝文件編譯器的話，你將用來準備COBOL程式在抓取磁碟片上。如果你想要爲此目的使用EDLIN的話，則由DOS 磁碟片至抓取磁碟片拷貝EDLIN。

1-2　編輯，編譯，連接與運轉　(Editing, Compiling, Linking, and Runing)

現在讓我們看一看編輯（打出與改正），編譯、連接與運轉程式步驟的細節。我們將在圖 1－1 所示中，使用非常簡單的COBOL程式。

打出原始程式

整個編譯過程的起點爲COBOL原始程式，你應打入電腦中，並儲存在磁碟片上。程式應用輔助文件編輯器打出，它將會接受你所打出的程式，並允許在螢幕上顯示程式，並儲存在磁碟片檔案中的程式。你能夠使用文件編輯器EDLIN，它將在DOS 磁碟片中提供，你能夠購買進一步的文件編輯器，它可能更方便來撰寫COBOL程式。在下列的指示中，我將保證你正在使用EDLIN。

在開始的時候，你把插入抓取磁碟片（它包含了你的文件編輯器）插入B號磁碟片中，並使B號驅動器爲內定驅動器。

註：要確定你已有了此編譯器的更新版本。填補對供應商者同，而使此編譯器使用512 K或者更多的記憶體。

```
A>B:
B>_
```

我們想要由圖 1－1 中打出程式EXAMPLE。COBOL原始檔案有擴充的COB，所以我們將在檔案EXAMPLE，COB 中儲存程式EXAMPLE。我們創造出原始檔案EXAMPLE，COB 如下所示：

```
B>EDLIN EXAMPLE.COB
```

有了文件編輯器的輔助，你能在程式中打出，並改正你可能作出的任何打出錯誤。而使用文件編輯器的細節超過本書的範圍。而使用EDLIN的指令可在DOS 參考手册中找到。當你由文件編輯器打出時，你的COBOL原始程式將儲存在抓取磁碟片上的檔案EXAMPLE，COB中。

```
IDENTIFICATION DIVISION.
PRCGRAM-ID. EXAMPLE.

ENVIRONMENT DIVISION.

DATA DIVISION.

PROCEDURE DIVISION.
DISPLAY-MESSAGE.
    DISPLAY "You have succeeded in compiling and running".
    DISPLAY "a program in IBM Personal Computer COBOL".
    STOP RUN.
```

你能夠使用編譯器，編輯器與連接器的樣本COBOL 程式。所有的最後三行在第八行中開始，此即，每一個空七格。最後三行在12行中開始，每一個空了11格。

對圖 1－1 我們有一些話要說，所以你能正確地打出程式，下列爲其項目。

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EXAMPLE.
ENVIRONMENT DIVISION.
DATA DIVISION.
```

```
PROCEDURE DIVISION.
)ISPLAY-MESSAGE.
```

在線上的第八個字元位置開始。字元位置通常可參考行列（columns），既然在已知位置中的所有字元，在螢幕上的行列中排列起來或者印出。則會列出此項目，而所有的在8行中開始。每一個項目的前面有七個空格。此程式最後三行上的項目在12行中開始，而每一項接著十一個空格。

要用EDLIN打出此程式的第一行，敲擊空格鍵七次，並打出IDENTIFICATION DIVISION。剩下的項目在第8行中開始，而在IDENTIFICATION DIVISION之下。此三種空白行爲取向式的。第一項在12行中開始，壓下空格鍵11次，並打出此項目。剩下的項目在12行中開始，並在第一個下面。COBOL程式的格式將在第二章中更細節地描述。

編譯（Compiling）

使用編譯器

要編譯你所創造的來源檔的話，則可在A號驅動器中插入COBOL磁碟片，並打出下列的指令

```
B>A:COBOL EXAMPLE;
```

要注意分號接著EXAMPLE，它防止了編譯器提示你取向檔的名稱。如果你留下了分號，並作提行的話，只要壓下Enter鍵，而對每一個提示作出反應。一旦開始了，編譯器將自動地操作。當它終止了，則磁碟片將包含目標檔EXAMPLE，OBJ。

錯誤訊息（Error messages）

當打出程式時，如果你已作出了任何打出的誤差，則編譯器可能顯示出多種的錯誤訊息（error messages）。每一種錯誤訊息

得出了行數，而錯誤發生在錯誤的描述之後。可能的錯誤訊息與其解釋將在COBOL參考手册的附錄A中給出。

例如，圖1.2顯示了程式BADEXMPL，它除了在10 行中拼錯的字DISPLAY之外，與EXAMPLE 是一樣的。（要注意在數行數時，空白行也要計數。）當我們想要編譯 BADEXMPL 時，下列的錯誤訊息會出現在螢幕上：

```
IDENTIFICATION DIVISION.
PROGRAM-ID. BADEXMPL.

ENVIRONMENT DIVISION.

DATA DIVISION.

PROCEDURE DIVISION.
DISPLAY-MESSAGE.
    DISPLY "You have succeeded in compiling and running".
    DISPLAY "a program in IBM Personal Computer COBOL".
    STOP RUN.
```

圖1.2此程式包含了第三行中的錯誤，而DISPLAY字將拼錯

```
10:  UNRECOGNIZABLE ELEMENT IS IGNORED.  DISPLY
10:  UNRECOGNIZABLE ELEMENT IS IGNORED.  You have succeeeded in
```

第一個訊息告訴我們，編譯器並不認得誤拼的字DISPLAY，而且會忽略掉它。第二種訊息並不表現出錯誤，而且所有過份頻繁所發生的情況：錯誤改正了部分的程式。在此情況下，編譯器忽略了誤拼的DISPLAY 指令，而不知道要顯示的訊息是什麼。此訊息設計了來未承的元素，即使它並不包含錯誤。

如果你接到了任何錯誤訊息，則回轉至文件編譯器中，並用圖1.1檢驗問題中的行數，並作出任何需要的改正。要特別注意頓號（句點、分數與引數），它在COBOL中時常會有麻煩。當你已作出了改正，則由編譯器中出去，並執行COBOL 編譯器來編譯改正的程式。

要印出錯誤訊息有三種方法。打出Shift PrtSc 將印出螢幕上現行的錯誤訊息。如果你在螢幕上參與了更多的錯誤訊息，則

打出Crtrl PrtSc。在螢幕上所顯示的任何東西，其中包含錯誤訊息，也將送至印字機中。（可打出 Ctrl prtSc 來送至印字機中，停止顯示輸出。最後，你能有編譯器來製造列表（listing，它印出了有錯誤訊息的COBOL 程式文件印出。我們將在本章後面看到，如何查詢編譯器，而印出列表。

如果沒有發現到錯誤的話，則編譯器將由顯示訊息來完成。

```
No Errors or Warnings
```

連接

要連接目標程式的話，可由A驅動器移去COBOL磁碟片，並插入LIBRARY磁碟片。要執行連接器的話，可打出下列的指令行：

```
A:LINK EXAMPLE;
```

作爲編譯器而言，指令行的末端中的分號避免了連接器提示取向的檔案名稱。如同編譯器一樣，連接器一旦開始就自動地操作。當連接器完成時，磁碟片將包含可執行的檔案EXAMPLA、EXE。連接器錯誤訊息很少，而且通常對應了明顯的景觀，例如如果要連接的目標檔並不出現在磁碟片上。

運轉（Running）

一旦編譯好了COBOL程式之後，我們由可以打出程式的名稱，而反映至DOS程式行中，而能夠運轉。如果我們打出了指令（不移動LIBRARY磁碟片，或磨擦磁碟片）。

```
B>EXAMPLE
```

程式EXAMPLE將會運轉，而且在兩個DISPLAY敘述中

的訊息，將出現在螢幕上；

```
B>EXAMPLE
You have succeeded in compiling and running
a program in IBM Personal Computer COBOL

B>_
```

編譯COBOL程式再運轉時間模組COBRUN。EXE 的控制之下執行，他編譯了由編譯器所產生的指示碼。你並不需要煩惱負載與執行COBRUN的問題。當你執行你的COBOL 程式時，這可以自動完成。無論如何，COBRUN應可用在磁碟片上。系統首先在內定驅動器中的磁碟片上尋找COBRUN；如果COBRUN 沒有找到的話，系統將在A號碼動器中尋找此磁碟片。

當你運轉了你已編譯的程式之後，則將使用LIBRARY的磁碟上COBURN的拷貝。當你滿意了COBOL 程式的績效時，你通常在每天的使用中把磨擦磁碟片拷貝可執行的程式到其他的磁碟片中。要確知所需要的運轉時間模組是可以用的，由包含COBOL程式的LIBRARY 磁碟片拷貝COBRUN到此磁碟片中。要確知程式在運轉時程式磁碟片在A號驅動器中或者在內定驅動器中。

如同以前所提及的，COBOL程式通常過份寫出了DOS指令編譯器，COMMAND，COM 。所以，當COBOL 程式在運轉時，COMMAND，COM的拷貝，應在A號驅動器的磁碟上，（否則的話，在COBOL程式已完成之後 此系統將在A號驅動器中提示你插入DOS磁碟片中。）

1-3 使用整批檔 (Using a Batch File)

我們能夠由使用整批檔（Using a Batch File），來專注指令，而簡化編譯的程序，此種指令與文件編譯器、編譯器、連接器，與可執行的程式有關。整批檔與程式有關，例如在本段中我們

將寫出一個整批檔，RUN．BAT（所有的整批檔都有擴充BAT）來編譯COBOL程式，如果要編譯和執行程式example的話，我們用指令行來發動整批檔。

```
B >RUN EXAMPLE
```

（注意：在EXAMPLE後並沒有分號EXAMPLE 爲能夠傳送至指令中，而組成整批檔的指令行參數，（command-line parameter。當我們寫出整批檔時，我們使用"/„ 1來表示第一個指令行參數，"/„2來表示第二個指令行參數等等。在每一個指令執行之前，第一個參數將被"/„1所取代，第二個參數將被"/„2所取代等等。所以當RUN被參數EXAMPLE 所啓動時EXAMPLE被整批檔中每„ 1個所取代。

整批檔是用"/„1，"/„2的DOS 指令列表，而且參數能夠被取代。你能夠用你的文件編輯器來創造整批檔。交替的，你能使用COPY指令而由控制台中拷貝 ，此卽，鍵盤至所想要的整批檔中，讓我們使用此方法來創造整批檔RUN．BAT並把它貯存在磁碟片上：（見原書第七頁程式）

```
B>COPY CON: RUN.BAT
EDLIN %1.COB
PAUSE Insert the COBOL diskette in drive A
A:COBOL %1;
PAUSE Insert the LIBRARY diskette in drive A
A:LINK %1;
%1
^Z
```

^Z，能由壓下功能FG來打出，它是檔案終端的信號，在COPY指令與^Z之間的六行儲存在整批檔中。當整批檔啓動時，此種指令一個接一個執行，而用指令行參數來取代"/„1。

例如，假設我們用指令行來啓動RUN。