第二版序

本書第一版於 1999 年 9 月出版。我非常高興地發現，我終於寫了一本永遠不需要修改的書！這與我的第一本書形成鮮明的對比，後者是關於為 Microsoft Windows 編寫應用程式的書。那本書在短短十年內經歷了五個版本。我的第二本書是關於 OS/2 Presentation Manager（這是什麼？），很快就過時了。但我確信，本書將永遠存在。

我對本書的最初想法是，先從非常簡單的概念開始，再慢慢地對數位電腦（digital computer）的工作原理建構非常深刻的知識。透過知識的穩健發展，我將使用設計和製造電腦之工程師的語言和符號，儘量少用隱喻、類比和愚蠢的插圖。我還有一個非常聰明的伎倆：我會用古老的技術來呈現普遍的原則，假設這些古老的技術已經很古老了，而且永遠不會退流行（never get older）。就好像我在寫一本關於內燃機（internal combustion engine）的書，但卻是以福特 T 型車（Ford Model T）為基礎。

我仍然認為我的做法是正確的，但我在一些細節上是錯誤的。隨著歲月的流逝，本書開始呈現出老態。一些文化參考資料變得陳舊。電話和手指取代了鍵盤和滑鼠。網際網路（internet）在 1999 年當然存在，但它與現在的樣子完全不同了。Unicode——用於統一表示世界上所有語言和表情符號（emojis）的文字編碼——在第一版中只用了不到一頁的篇幅。而 JavaScript，這種在 web 上已經普遍存在的程式設計語言，根本就沒有被提及。

這些問題或許很容易解決，但第一版的另一個方面仍然困擾著我。我想呈現實際 CPU——中央處理單元（central processing unit），它是電腦的大腦、心臟和靈魂——的工作原理，但第一版並沒有完全做到。儘管我覺得已經接近這一關鍵突破，但後來我放棄了。讀者似乎沒有抱怨，但對我來說，這是一個明顯的缺陷。

碁峰 www.gotop.com.tw

這一缺陷在第二版中得到了糾正。這就是為什麼它多了大約 70 頁。是的，這是一段較長的旅程，但如果你能和我一起翻閱第二版的書頁，我們將更深入地瞭解 CPU 的內部結構。我不知道這對你來說是否會是一次更愉快的經歷。如果你覺得自己快要淹死了，請浮出水面上來透透氣。但如果你讀完了第 24 章，你應該感到非常自豪，並且你會很高興知道，這本書的其餘部分是輕而易舉的。

配套網站

本書的第一版在電路圖中使用紅色來表示電的流向。第二版也是如此，但這些電路的工作原理現在也同時在一個名為 CodeHiddenLanguage.com 的新網站上以更具圖形互動的方式進行了說明。

在本書的各頁中，你偶爾會想起這個網站，但我們還使用了一個特殊的圖示，你會在本段的空白處看到它。此後，每當你看到該圖示（通常伴隨著電路圖）時，你都可以在網站上探索電路的工作原理（對於那些想知道技術背景的人來說：我使用 HTML5 的 `canvas` 元素在 JavaScript 中編寫了這些 web 圖形）。

CodeHiddenLanguage.com 網站的使用是完全免費的。沒有付費牆（paywall），你所看到的唯一廣告是本書自己的。在一些例子中，該網站使用 cookie，但僅會在你的電腦上儲存一些資訊。該網站不會追蹤你或做任何邪惡的事情。

我還將使用該網站來澄清或更正書中的內容。

致謝

本書只有我一個人的名字出現在封面上；但協助過本書的其他人同樣不可或缺。

我特別想感謝執行主編（Executive Editor）海茲・亨伯特（Haze Humbert），她出人意料地向我提出第二版的可行性，而這恰好是我準備好這樣做的時刻。我於 2021 年 1 月開始工作，她巧妙地引導我們度過了這場磨難，即使本書已經過了截止日幾個月，而且我需要做出一些保證，表明我還沒有完全放棄。

碁峰 www.gotop.com.tw

第一章

最好的朋友

你今年 10 歲。你最好的朋友就住在街對面。你們臥室的窗戶剛好彼此相對。每天晚上，在你父母宣佈就寢時間後，你們仍然需要彼此交換想法、觀察、秘密、八卦、笑話和夢想。沒有人能責怪你。畢竟，交流的衝動是人的本性。

當你臥室裡的燈還亮著時，你和你最好的朋友可以從窗戶互相揮手，並使用廣泛的手勢和基本的肢體語言，傳達一兩個想法。但更複雜的交流似乎很困難，一旦父母下令「熄燈！」，就有必要採取更隱蔽的解決方案。

如何交流？如果你有幸在 10 歲的年紀擁有一支手機，或許可以試著打一通秘密電話或是傳一則無聲的簡訊。但如果你的父母習慣在睡前沒收手機，甚至關閉 Wi-Fi，該怎麼辦？沒有電子通訊的臥室確實是一個非常孤立的房間。

然而，你和你最好的朋友所擁有的是手電筒。每個人都知道，手電筒的發明是為了讓孩子在床罩下閱讀書籍；手電筒似乎也非常適合在天黑後進行交流。手電筒當然很安靜，而且光線具有很強的方向性，應該不會從臥室的門縫底下漏出來提醒多疑的人。

可以讓手電筒說話嗎？當然值得一試。你在一年級的時候就學會如何在紙上書寫字母和單字，所以將這些知識轉移到手電筒上似乎是合理的。你所要做的就是，站在你的窗戶前，用手電筒的光線畫出字母。字母 O 的話，你可以打開手電筒，在空中掃一圈，然後關掉手電筒。字母 I 的話，你可以劃一條垂直線。但你很快發現，這種方法是一場災難。當你看到你朋友的手電筒在空中畫出圈和線時，你發現要在你的頭腦中把多個筆畫組合在一起實在太難了。這些光線構成的圈圈和斜線實在是不夠精確。

碁峰 www.gotop.com.tw

也許你曾經看過一部電影，其中兩個水手在海上用閃爍的燈光互相示意。在另一部電影中，一個間諜擺動一面鏡子，將陽光反射到另一個間諜被囚禁的房間裡。也許這就是解決方案。因此，你首先要設計一個簡單的技巧：字母表（alphabet）中的每個字母對應手電筒的閃爍次數。字母 A 閃爍 1 次，字母 B 閃爍 2 次，字母 C 閃爍 3 次，依此類推，字母 Z 閃爍 26 次。而 BAD 這個單字則是閃爍 2 次、閃爍 1 次、閃爍 4 次，字母之間有些許停頓，所以你不會把這 7 次閃爍誤認為字母 G。你會在兩個單字之間停頓更長的時間。

這似乎很有希望。好消息是，你不再需要在空中揮舞手電筒；你需要做的就是，指向目標並快速按開關。壞消息是，你嘗試發送的第一則消息“How are you?”（你好嗎？）總共需要讓手電筒閃爍 131 次！此外，你忘記了標點符號，因此你不知道問號需要閃爍多少次。

但已經很接近了。你想，以前一定有人遇到過這個問題，你是絕對正確的。透過去圖書館或網路搜索，你會發現一個神奇的發明，稱為摩爾斯代碼（Morse code）。這正是你一直在尋找的，但你現在必須重新學習如何「書寫」字母表中的所有字母。

區別在於：在你發明的系統中，字母表中的每個字母都有一定的閃爍次數，從 A 的閃爍 1 次到 Z 的閃爍 26 次。在摩爾斯代碼中，你有兩種閃爍——短時間的閃爍（short blinks）和長時間的閃爍（long blinks）。當然，這使得摩爾斯代碼更加複雜，但在實際使用時，它的效率非常高。這句“How are you?”現在不需要閃爍 131 次，只需要閃爍 32 次（有些時間短，有些時間長），其中包括問號的摩爾斯代碼。

在討論摩爾斯代碼的工作原理時，人們不會說「短時間的閃爍」和「長時間的閃爍」。相反，他們會說「點」（dots）和「劃」（dashes），因為這是在印刷頁面上顯示摩爾斯代碼的便捷方式。在摩爾斯代碼中，字母表的每個字母都會對應著一串點和劃，如下表所示。

雖然摩爾斯代碼與電腦毫無關係，但熟悉代碼的本質是深入了解隱藏在電腦軟硬體底下之秘密和內部結構的必要前提。

本書中，代碼（*code*）這個詞通常是指在人與人之間、人與電腦之間或電腦內部傳輸資訊的一個系統。

碁峰 www.gotop.com.tw

A	•–
B	–•••
C	–•–•
D	–••
E	•
F	••–•
G	––•
H	••••
I	••

J	•–––
K	–•–
L	•–••
M	––
N	–•
O	–––
P	•––•
Q	––•–
R	•–•

S	•••
T	–
U	••–
V	•••–
W	•––
X	–••–
Y	–•––
Z	––••

代碼讓你得以進行交流。有時代碼是秘密的，但大多數代碼並不是。事實上，大多數代碼必須被充分瞭解，因為它們是人類交流的基礎。

我們用嘴所發出的聲音構成了一種代碼，任何能夠聽到我們的聲音並瞭解我們所說之語言的人，都可以瞭解這種代碼。我們將此種代碼稱為「口語」（spoken word）或「語音」（speech）。

在聾人群體中，各種手語（sign languages）使用手和手臂來形成動作（movements）和手勢（gestures），以傳達單字的個別字母或者整個單字和概念。北美最常見的兩個系統是美國手語（American Sign Language 或 ASL），它是 19 世紀初在美國聾人學校開發的，以及魁北克手語（*Langue des signes Québécoise* 或 LSQ），它是法國手語（French sign language）的變體。

我們用另一種代碼來表示紙上或其他媒體上的言辭（words），稱為「書面文字」（written word）或「文字」（text）。文字可以用手書寫（written）或鍵入（keyed），然後列印在報紙、雜誌和書籍上，或以數位方式顯示在各種裝置上。在許多語言中，語音和文字之間存在很強的對應關係。例如，在英語中，字母和字母串（或多或少）對應於口語聲音。

對於視力受損的人來說，可以用點字（Braille）來代替書面文字，點字使用了一個凸點（raised dots）系統，該系統與字母、字母串和整個單字相對應（我將在第 3 章中更詳細地討論點字）。

當口語必須非常快速地轉錄（transcribed）成文字時，速記（stenography）或簡寫（shorthand）就很有用。在法庭上或者為電視新聞或體育節目產生即時隱藏字幕

碁峰 www.gotop.com.tw

（closed captioning）時，速記員（stenographers）會使用帶有簡化鍵盤的速記機（stenotype machine），鍵盤上有與文字相對應的代碼。

我們使用各種不同的代碼來相互交流，因為有些代碼比其他代碼更為方便。口語（spoken word）的代碼不能保存在紙上，因此使用書面文字（written word）的代碼。在黑暗中透過語音或紙張在遠處默默地交換資訊是不可能的。因此，摩爾斯代碼是一個方便的替代方案。如果一個代碼能達到其他代碼無法達到的目的，那麼該代碼就很有用。

正如我們將看到的，電腦中還使用了各種類型的代碼來保存和傳遞文字、數字、聲音、音樂、圖片和電影，以及電腦本身的指令。電腦無法輕鬆處理人類代碼（human codes），因為電腦無法精確複製人類使用眼睛、耳朵、嘴巴和手指的方式。教電腦說話很難，讓它們瞭解語音更難。

但已經取得了很大進展。電腦現在已能夠捕捉、保存、操作和呈現人類交流中使用之多種類型的資訊，包括視覺（文字和圖片）、聽覺（口語、聲音和音樂）或兩者的組合（動畫和電影）。所有這些類型的資訊都需要自己的代碼。

甚至你剛才看到的摩爾斯代碼列表本身也是一種代碼。該表列出的每個字母都是由一系列的點和劃來表示。然而，我們實際上不能發送點和劃。使用手電筒發送摩爾斯代碼時，這些點和劃跟閃爍相對應。

使用手電筒發送摩爾斯代碼時，需要打開手電筒開關，對於「點」，需要短時間打開手電筒開關，對於「劃」，需要打開手電筒開關較長的時間。例如，要發送 A，你需要短時間打開手電筒，然後打開手電筒開關較長的時間，接著在發送下一個字符之前停頓一下。按照慣例，「劃」的長度應約為「點」的三倍。接收端的人看到短閃爍和長閃爍，便知道這是 A。

摩爾斯代碼的「點」和「劃」之間的停頓時間至關重要。例如，當你發送 A 時，「點」和「劃」之間的停頓時間，大約等於一個「點」的長度。同一單字中的字母則由較長的停頓時間分隔，該停頓時間大約等於一個「劃」的長度。例如，下面是「hello」的摩爾斯代碼，它說明了字母之間的停頓情況：

碁峰 www.gotop.com.tw

單字之間由大約兩個「劃」的長度分隔。下面是「hi there」的代碼：

手電筒維持打開的時間長度並不固定。它們都與「點」的長度有關，這取決於手電筒開關被觸發的速度，以及摩爾斯代碼發送者記住特定字母之代碼的速度。一個快速發送者的「劃」可能與慢速發送者的「點」長度相同。這個小問題可能會讓閱讀摩爾斯代碼變得困難，但是在一兩個字母之後，接收端的人通常可以分辨出什麼是「點」，什麼是「劃」。

首先，摩爾斯代碼的定義——我所說的**定義**是指「點和劃的各種序列」與字母表中字母的對應關係——看起來就像電腦鍵盤的佈局一樣隨機。然而，仔細觀察，情況並非完全如此。較簡單和較短的代碼被分配給字母表中較常用的字母，例如 E 和 T。Scrabble（拼字遊戲）的玩家和 *Wheel of Fortune*（**命運之輪**）的粉絲可能會立即注意到這一點。不太常見的字母，如 Q 和 Z（在 Scrabble 中可讓你獲得 10 點，而且很少出現在 *Wheel of Fortune* 的謎題中）有較長的代碼。

幾乎每個人多少都知道一些摩爾斯代碼。三個「點」、三個「劃」和三個「點」代表國際求救信號 SOS。SOS 並非任何東西的縮寫——它只是一個易於記憶的摩爾斯代碼序列（Morse code sequence）。在第二次世界大戰期間，英國廣播公司的一些無線電廣播以貝多芬的第五交響曲片段——登登登，凳（BAH, BAH, BAH, BAHMMMMM）——作為節目的前奏，貝多芬在創作音樂當時並不知道它有一天會成為 V（代表勝利）的摩爾斯代碼。

摩爾斯代碼的一個缺點是，它不區分字母的大小寫。但除了表示字母之外，摩爾斯代碼還包括數字代碼（使用五個一組的「點」和「劃」序列）：

1	●▬▬▬▬	6	▬●●●●
2	●●▬▬▬	7	▬▬●●●
3	●●●▬▬	8	▬▬▬●●
4	●●●●▬	9	▬▬▬▬●
5	●●●●●	0	▬▬▬▬▬

碁峰 www.gotop.com.tw

這些數字代碼至少比字母代碼更有序一些。大多數標點符號使用五、六或七個一組的「點」和「劃」序列：

.	●▬●▬●▬	'	●▬▬▬▬●
,	▬▬●●▬▬	(	▬●▬▬●
?	●●▬▬●●	)	▬●▬▬●▬
:	▬▬▬●●●	=	▬●●●▬
;	▬●▬●▬●	+	●▬●▬●
-	▬●●●●▬	$	●●●▬●●▬
/	▬●●▬●	¶	●▬●▬●●
"	●▬●●▬●	_	●●▬▬●▬

其他代碼是為一些歐洲語言的重音字母和特殊用途的簡寫序列而定義的。SOS 代碼就是這樣一種簡寫序列（shorthand sequence）：它應該被連續發送，三個字母之間只有一個「點」的停頓。

你將發現，如果你有一個專門為此目的製作的手電筒，你和你的朋友發送摩爾斯代碼會容易得多。除了一般的滑動開關（slider switch），這些手電筒還包括一個按鈕開關（pushbutton switch），你只需按下並鬆開它，即可打開和關閉手電筒。透過一些練習，你或許能夠達到每分鐘 5 或 10 個單字的發送和接收速度——仍然比語音的速度（大約每分鐘 100 個單字）慢得多，但肯定夠了。

當你和朋友最終記住了摩爾斯代碼時（因為這是你能熟練地發送和接收摩爾斯代碼的唯一方法），你也可以用它來代替正常的語音。為了達到最大的速度，你將「點」發音為 *dih*（或 *dit* 代表字母的最後一個「點」），將「劃」發音為 *dah*，例如 *dih-dih-dih-dah*（滴 - 滴 - 滴 - 答）表示 V。就像摩爾斯代碼將書面語言簡化為「點」和「劃」一樣，口語版本的代碼將語音簡化為兩個元音（two vowel sounds）。

這裡的關鍵字是「兩」（*two*）。兩種類型的閃爍，兩種元音，兩種不同的東西，真的，可以透過適當的組合來傳達所有類型的資訊。

碁峰 www.gotop.com.tw

第二章

代碼與組合

摩爾斯代碼（Morse code）是 1837 年左右由撒母耳・芬利・布里斯・摩爾斯（Samuel Finley Breese Morse，1791-1872）發明的，我們將在本書後面適時地介紹它。後來有人對它做了進一步的發展，最著名的是阿爾弗雷德・維爾（Alfred Vail，1807-1859），並演變成了幾種不同的版本。本書中描述的系統更正式之名稱為國際摩爾斯代碼（International Morse code）。

摩爾斯代碼的發明與電報的發明密切相關，我將在本書後面更詳細地研究電報。正如摩爾斯代碼（Morse code）對代碼（codes）的本質提供了很好的介紹，電報包括了可以模擬電腦運作的硬體。

大多數人發現摩爾斯代碼的發送比接收更容易。即使你沒有記住摩爾斯代碼，你也可以輕易地使用你在上一章看到之按字母順序排列的表格：

A	•–
B	–•••
C	–•–•
D	–••
E	•
F	••–•
G	––•
H	••••
I	••

J	•–––
K	–•–
L	•–••
M	––
N	–•
O	–––
P	•––•
Q	––•–
R	•–•

S	•••
T	–
U	••–
V	•••–
W	•––
X	–••–
Y	–•––
Z	––••

碁峰 www.gotop.com.tw

接收摩爾斯代碼並將其翻譯回單字，比發送要困難得多，也更耗時，因為你必須反向工作才能找出對應於特定代碼（「點」和「劃」序列）的字母。如果你沒有記住代碼，並且收到「劃 - 點 - 劃 - 劃」（dash-dot-dash-dash），則必須逐字母地瀏覽表格，然後你最終才能發現它是字母 Y。

問題是，儘管我們有一個表格提供如下的轉換：

字母表中的字母 → 摩爾斯代碼的「點」和「劃」

但我們沒有一個可以反向轉換的表格：

摩爾斯代碼的「點」和「劃」 → 字母表中的字母

在學習摩爾斯代碼的早期階段，這樣的表格肯定會很方便。但我們如何構建它並不明顯。這些「點」和「劃」中沒有任何內容可以按字母順序排列。

所以讓我們忘記字母順序。也許組織代碼的更好方法是根據它們有多少個「點」和「劃」來對它們進行分組。例如，僅包含一個「點」或一個「劃」的摩爾斯代碼序列（Morse code sequence）只能表示兩個字母，即 E 和 T：

•	E
▬	T

兩個「點」或「劃」的組合提供了另外四個字母——I、A、N 和 M：

••	I
•▬	A

▬•	N
▬▬	M

三個「點」或「劃」的模式為我們提供了另外八個字母：

•••	S
••▬	U
•▬•	R
•▬▬	W

▬••	D
▬•▬	K
▬▬•	G
▬▬▬	O

碁峰 www.gotop.com.tw

最後（如果我們想在處理數字和標點符號之前停下來），四個點和劃的序列提供了另外 16 個字符：

●●●●	H	▬●●●	B
●●●▬	V	▬●●▬	X
●●▬●	F	▬●▬●	C
●●▬▬	Ü	▬●▬▬	Y
●▬●●	L	▬▬●●	Z
●▬●▬	Ä	▬▬●▬	Q
●▬▬●	P	▬▬▬●	Ö
●▬▬▬	J	▬▬▬▬	Ş

總之，這四張表包含 2 加 4 加 8 加 16 個代碼，總共 30 個字母，比拉丁字母表（Latin alphabet）中的 26 個字母多 4 個。因此，你會注意到最後一個表格中的 4 個代碼用於重音字母：三個具有上加變音符（umlauts），一個具有下加變音符（cedilla）。

當有人向你發送摩爾斯代碼時，這四張表當然會有所幫助。收到特定字母的代碼後，你就知道它有多少個「點」和「劃」，而你至少可以去正確的表格查找。每張表格都以有條不紊的方式組織，從左上角的全「點」代碼（all-dots code）開始，到右下角的全「劃」代碼（all-dashes code）結束。

你能看出這四張表的大小有什麼規律嗎？每張表的代碼數量是其前張表的兩倍。這是有道理的：每張表包含，前張表中所有代碼前綴一個「點」，以及前張表中所有代碼前綴一個「劃」。

我們可以總結出一個有趣的規律：

點和劃的數量	代碼的數量
1	2
2	4
3	8
4	16

碁峰 www.gotop.com.tw

這四張表中，每一張表的代碼數量都是其前一張表的兩倍，因此，如果第一張表有 2 個代碼，則第二張表有 2×2 個代碼，而第三張表有 2×2×2 個代碼。所以總結的呈現方式變為：

點和劃的數量	代碼的數量
1	2
2	2 × 2
3	2 × 2 × 2
4	2 × 2 × 2 × 2

當我們看到一個自乘的數字，我們可以用指數（exponents）來表示次方數（powers）。例如，2×2×2×2 可以寫成 2^4（*2 的 4 次方*）。數字 2、4、8 和 16 都是 2 的次方，因為你可以透過將 2 自乘來計算它們。所以總結的呈現方式變為：

點和劃的數量	代碼的數量
1	2^1
2	2^2
3	2^3
4	2^4

這張表已經變得非常簡單。代碼的數量就是 2 的「點和劃數量」次方：

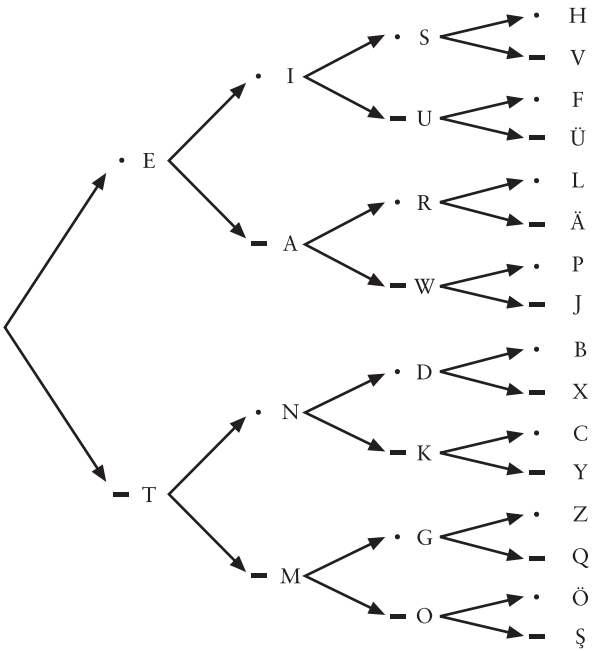
$$\text{代碼的數量} = 2^{\text{點和劃的數量}}$$

2 的次方（powers of 2）在代碼中經常出現，特別是在本書中。你會在下一章看到另一個例子。

為了使解碼摩爾斯代碼的過程更加容易，你可能需要繪製大型樹狀圖。

此圖顯示了由每個特定的連續「點」和「劃」序列所產生的字母。要解碼一個特定序列，請將你的視線按照箭頭從左到右移動。例如，假設你想知道哪個字母對應於「點 - 劃 - 點」（dot-dash-dot）代碼。從左邊開始，選擇「點」；然後繼續沿著箭頭向右移動，選擇「劃」，然後選擇另一個「點」。所以是字母 R，它就顯示在第三個「點」的旁邊。

碁峰 www.gotop.com.tw



仔細想想，構建這樣一張表對定義摩爾斯代碼來說可能是必要的。首先，它可以確保你不會犯將相同的代碼用於兩個不同字母的愚蠢錯誤！其次，你可以放心使用所有可能的代碼，而不會讓點和劃的序列變得過長。

冒著將此表擴展到超出印刷頁面之外的風險，我們可以繼續使用五個一組的「點」和「劃」代碼。恰好五個一組的「點」和「劃」序列為我們提供了 32（2×2×2×2×2 或 2^5）個額外的代碼。通常，對於在摩爾斯代碼中定義的 10 個數字和 16 個標點符號來說這已經夠了，儘管這些數字是用五個一組的「點」和「劃」編碼的，但事實上許多使用五個一組之「點」和「劃」序列編碼的其他代碼，所表示的是重音字母，而不是標點符號。

要包含所有標點符號，系統必須擴展為六個一組的「點」和「劃」，這為我們提供了 64（2×2×2×2×2×2 或 2^6）個額外的代碼，總計 2+4+8+16+32+64 或 126 個字符。這對摩爾斯代碼來說是多餘的，它留下了許多**未被定義**的較長代碼，在此環境中使用的代碼不代表任何東西。如果你正在接收摩爾斯代碼，並且你得到的是一個未定義的代碼，你可以非常確定，有人弄錯了。

碁峰 www.gotop.com.tw

因為我們夠聰明，可以發展出這個小公式，

$$\text{代碼的數量} = 2^{\text{點和劃的數量}}$$

我們可以繼續研究使用更長的序列能夠得到的代碼數量：

點和劃的數量	代碼的數量
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
4	$2^4 = 16$
5	$2^5 = 32$
6	$2^6 = 64$
7	$2^7 = 128$
8	$2^8 = 256$
9	$2^9 = 512$
10	$2^{10} = 1024$

幸運的是，我們不必實際寫出所有可能的代碼來確定會有多少個。我們所要做的就是一遍又一遍地將 2 乘以 2。

摩爾斯代碼被認為是**二進位的**（*binary*，字面意思是**二個一組**），因為代碼只由兩個東西組成——「點」和「劃」。這類似於硬幣，它只能落在正面或反面。投擲十次的硬幣可以有 1024 個不同的正面和反面的序列。

二進位物件（例如硬幣）和二進位代碼（例如摩爾斯代碼）的組合總是由二的次方（powers of two）來描述。二（two）是本書中一個非常重要的數字。

碁峰 www.gotop.com.tw

第三章

點字與二進位代碼

撒母耳·摩爾斯（Samuel Morse）並不是第一個成功將書面語言（written language）的字母翻譯成可解譯代碼（interpretable code）的人。他也不是第一個「因為代碼的名稱而不是因為自己」被人們記住的人。事實上，這個榮譽應該還給一個法國盲人少年，他比摩爾斯晚 18 年出生，但成就卻早得多。人們對他的生平知之甚少，但所知道的是一個引人入勝的故事。

路易·布萊葉（Louis Braille）於 1809 年出生於法國庫夫賴（Coupvray, France），距巴黎以東僅 25 英里。他的父親是一名馬具製造商。在他三歲的時候——不應該在父親的工作室裡玩耍的年齡——不小心把一個尖銳的工具插進了眼睛。他的傷口被感染，而且擴散到他的另一隻眼睛，使他完全失明。在那個年代，大多數遭受這樣命運的人，註定要過著無知和貧窮的生活，但小路易的智慧和對學習的渴望很快就得到了認可。在村裡的牧師和一名教師的干預下，他先是和其他孩子一起在村裡上學，然後在 10 歲時被送到巴黎的皇家青少年盲人學校。

ullstein bild Dtl/Getty Images

盲童教育的主要障礙是他們無法獲得無障礙的閱讀材料。瓦朗坦·阿維（Valentin Haüy，1745-1822）是巴黎皇家青少年盲人學校的創辦人，他發明一種在紙上壓印字母的系統，字體大而圓，可以透過觸摸來閱讀。但這個系統非常難用，用這種方法製作的書籍也很少。

碁峰 www.gotop.com.tw

視力正常的阿維被困在一個模式中。對他來說，A 就是 A，字母 A 必須看起來（或感覺）像 A（如果給他一支手電筒來交流，他可能會嘗試在空中畫出字母，就像我們在發現效果不太好之前所做的那樣）。阿維可能沒有意識到，一種與壓印字母完全不同的代碼可能更適合視力不佳的人。

另一種代碼的起源，來自一個意想不到的源頭。1815 年，法國陸軍上尉查爾斯·巴比爾（Charles Barbier）設計了一種書寫系統，後來被稱為「夜曲」（*écriture nocturne*）或「夜間書寫」（night writing）。該系統在厚重的紙張上使用凸起的圓點圖案，目的是讓士兵們在需要噤聲時於黑暗中相互傳遞筆記。士兵們可以用類似錐子的尖筆把這些點戳進紙的背面。然後可以用手指讀取凸起的點。

路易·布萊葉在 12 歲時就熟悉了巴比爾的系統。他喜歡使用凸起的點，不僅因為手指易於閱讀，還因為它很容易書寫。教室裡配備了紙和尖筆的學生，實際上可以做筆記並把它讀回來。布萊葉孜孜不倦地努力改進這一系統，並在三年內（15 歲時）提出了自己的系統，其基本原理至今仍在使用。多年來，該系統僅在校內為人所知，但它逐漸進入世界其他地區。1835 年，路易·布萊葉感染了肺結核，最終在他 43 歲生日後不久，即 1852 年，死於肺結核。

今日，有各種版本的點字系統（Braille system）與有聲讀物競爭，為盲人提供書面文字，但點字仍然是一個無價的系統，也是盲人和聾啞人閱讀的唯一途徑。近幾十年來，隨著電梯和自動提款機使用點字，公眾對點字變得越來越熟悉。

本章中，我將要做的是剖析點字代碼（Braille code）並向你展示它的運作原理。你實際上不需要學習點字或記住任何東西。本練習的唯一目的是進一步瞭解代碼的本質。

在點字中，一般書面語言中使用的每個符號（特別是字母、數字和標點符號）會被編碼成 2×3 單元格中一或多個凸起的點。單元格中的點通常被編號成 1 到 6：

1 ○ ○ 4
2 ○ ○ 5
3 ○ ○ 6

碁峰 www.gotop.com.tw

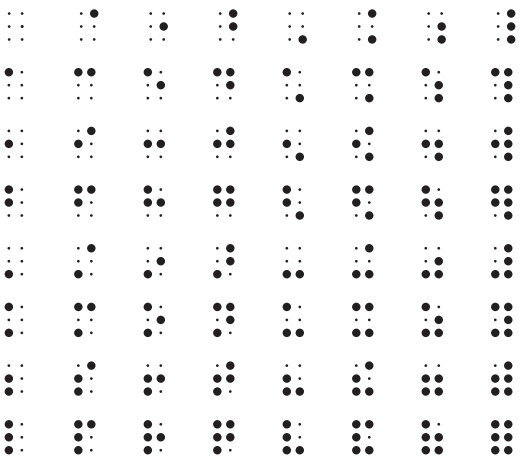
於是開發了特殊的打字機，將點字壓印到紙上，如今，可由電腦驅動的壓印機（embossers）來完成這項工作。

因為壓印本書這幾頁內容的點字非常昂貴，所以我使用了一種在印刷頁面上顯示點字的常用表示法。在此表示法中，單元格中的六個點都會被顯示出來。大圓點表示單元格中紙張凸起的部分。小圓點表示單元格中沒有凸起的部分。例如，在如下的點字中

圓點 1、3 和 5 是凸起的，圓點 2、4 和 6 則沒有凸起。

此刻，我們應該感興趣的是，這些圓點是二進位的。一個特定的圓點要嘛是未凸起，要嘛是凸起的。這意味著，我們可以將所學到之關於摩爾斯代碼與二進位組合的知識應用到點字上。我們知道有六個圓點，每個圓點可以是未凸起的，也可以是凸起的，所以六個未凸起和凸起之圓點的組合總數為 $2\times2\times2\times2\times2\times2$，或 2^6，或 64。

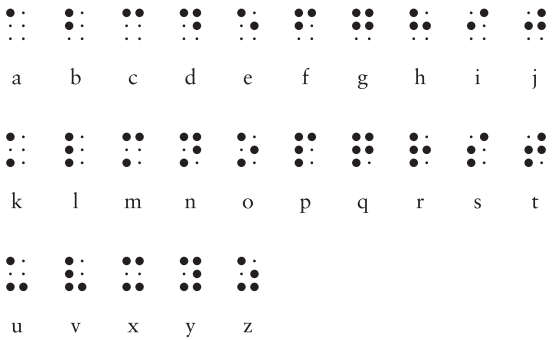
因此，點字系統能夠表示 64 個獨一無二的代碼。如下所示——全部 64 個：



沒有必要在點字中使用所有 64 個代碼，但 64 絕對是六點模式（six-dot pattern）的上限。

碁峰 www.gotop.com.tw

現在開始剖析點字的代碼，讓我們看一下基本的小寫字母表：



例如，點字中的片語“you and me”如下所示：

請注意，單字中每個字母的單元格由一點空間隔開；單字之間則使用較大的空間（基本上是一個沒有凸起圓點的單元格）隔開。

這是路易·布萊葉所設計之點字的基礎，或者至少適用於拉丁字母。路易·布萊葉還為帶有重音符號的字母設計了代碼，這在法語中很常見。請注意，沒有 *w* 的代碼，古典法語中並未使用 *w*（別擔心，這個字母最終會出現的）。此刻，64 個可能的代碼中只有 25 個被考慮在內。

仔細檢查後，你將在 25 個小寫字母的點字代碼（Braille codes）中發現一個模式。第一列（字母 *a* 到 *j*）僅使用單元格中的前四個圓點——點 1、2、4 和 5。第二列（字母 *k* 到 *t*）複製了第一列，只是圓點 3 被凸起了。第三列（*u* 到 *z*）是相同的，只是圓點 3 和 6 被凸起了。

路易·布萊葉最初將他的系統設計為用手打孔。他知道這可能不是很精確，因此他以一種減少歧義的方式，巧妙地定義了 25 個小寫字母。例如，在 64 個可能的點字代碼中，有 6 個代碼具有一個凸起的圓點。但其中只有一個用於小寫字母 *a*。64 個代碼中有四個代碼具有兩個垂直相鄰的圓點，但同樣只使用一個用於字母 *b*。有三個代碼具有兩個水平相鄰的圓點，但只有一個用於 *c*。

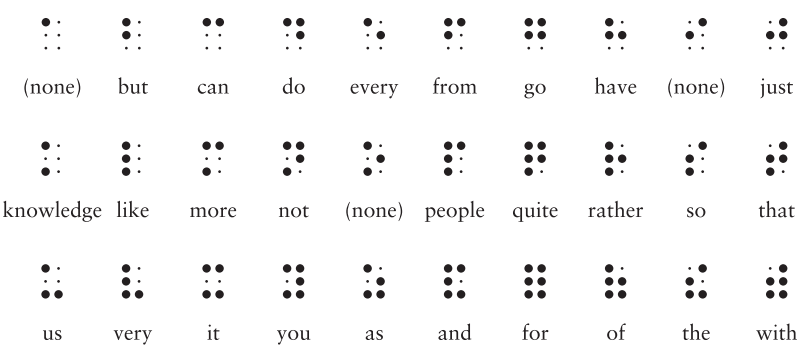
碁峰 www.gotop.com.tw

路易·布萊葉實際定義的是一組獨特的**形狀**，這些**形狀**可以在頁面上稍微移動，但仍然具有相同的含義。*a* 是一個凸起的圓點，*b* 是垂直相鄰的兩個圓點，*c* 是水平相鄰的兩個圓點，依此類推。

代碼通常容易出錯。編寫代碼時（例如，當使用點字的學生在紙上標記圓點時）發生錯誤稱為**編碼**錯誤（*encoding* error）。讀取代碼時出現錯誤稱為**解碼**錯誤（*decoding* error）。此外，還可能存在**傳輸**錯誤（*transmission* errors）——例如，當包含點字的頁面以某種方式損壞時。

更複雜的代碼通常包含各種類型的內置糾錯功能（built-in error correction）。從這個意義上說，最初由路易·布萊葉定義的點字是一種複雜的編碼（coding）系統：它使用冗餘（redundancy）來允許圓點的打孔和讀取可以稍微不精確。

自路易·布萊葉的時代以來，點字代碼以各種方式得到了擴展，包括用於記錄數學和音樂的系統。目前，在出版的英文文本（English text）中最常用的系統稱為 2 級點字（Grade 2 Braille）。2 級點字使用了許多縮寫，以減少紙張的使用並加快閱讀速度。例如，如果字母代碼單獨出現，則代表常用單字。以下三列（包括「已完成的」第三列）可以看到這些單字代碼：



因此，片語“you and me”若用 2 級點字會被寫成這樣：

到目前為止，我已經描述了 31 個代碼——單字之間的無凸點空間（no-raised-dots space）、以及字母和單字的三列（每列十個）代碼。我們仍然沒有接近理論上可用的 64 個代碼。正如我們將看到的，在 2 級點字中，沒有浪費任何東西。

碁峰 www.gotop.com.tw

字母 *a* 到 *j* 的代碼可以與「有凸點的」（raised dot）6 組合在一起。這些代碼主要用於單字中字母的縮寫，還包括 *w* 和另一個單字縮寫：

ch gh sh th wh ed er ou ow w（或「will」）

例如，單字“about”若用 2 級點字會被寫成這樣：

下一步將引入一些潛在的模糊性，這在路易·布萊葉的原始構想中是不存在的。字母 *a* 到 *j* 的代碼也可以有效地降低為僅使用圓點 2、3、5 和 6。這些代碼代表一些標點符號和縮寫，具體取決於前後文：

ea bb cc dis en to gg his in was
, ; : . ! () “ ”

這些代碼中的前四個分別是逗號、分號、冒號和句點。請注意，左括號和右括號使用相同的代碼，但左引號和右引號使用的是兩個不同的代碼。因為這些代碼可能會被誤認為是 *a* 到 *j* 的字母，所以它們只有在較長的前後文中與其他字母一起才有意義。

到目前為止，我們已經有了 51 個代碼。以下六個代碼使用了圓點 3、4、5 和 6 的各種未使用的組合，來表示縮寫和一些額外的標點符號：

st ing ble ar ' com
/ # -

“ble”的代碼非常重要，因為當它不是單字的一部分時，這意味著後面的代碼應該被解釋為數字。這些數字代碼與字母 *a* 到 *j* 的代碼相同：

1 2 3 4 5 6 7 8 9 0

碁峰 www.gotop.com.tw

因此，這個代碼序列

代表數字 256。

最後，再七個代碼，我們才會達到 64 的上限。它們分別是：

第一個代碼（圓點 4 是凸起的）被用作重音指示符（accent indicator）。其他的代碼用作某些縮寫的前綴，也用於其他一些目的：當圓點 4 和 6 是凸起的時（第五個代碼），它是一個數字的小數點（numeric decimal point）或強調指示符（emphasis indicator），具體取決於前後文（context）。當圓點 5 和 6 是凸起的時（第六個代碼），它是一個字母指示符（letter indicator），用於平衡數字指示符（number indicator）。

最後（如果你一直想知道點字是如何對大寫字母進行編碼的），我們還有凸起的圓點 6——大寫指示符（capital indicator）——可用。這表示後面的字母是大寫的。例如，我們可以將此系統之原始創建者的名稱寫為

此序列以一個大寫字母指示符開頭，後面跟著字母 *l*，縮寫 *ou*，字母 *i* 和 *s*，一個空格，另一個大寫指示符，以及字母 *b*、*r*、*a*、*i*、*l*、*l* 和 *e*（在實際使用中，這個名字可以透過取消最後兩個不發音的字母，或者透過將其拼成“brl”，而變得更加縮略）。

總之，我們已經看到六個二進位元素（圓點）如何產生 64 個可能的代碼。碰巧的是，這 64 個代碼中的許多代碼會根據其前後文執行雙重任務。特別有趣的是，數字指示符（number indicator）以及撤銷（undoes）數字指示符的字母指示符（letter indicator）。這些代碼改變了它們後面之代碼的含義——從字母到數字，從數字回到字母。諸如此類的代碼通常稱為**優先**代碼（*precedence* codes）或**移位**代碼（*shift* codes）。它們會更改所有後續代碼的含義，直到移位被撤銷。

碁峰 www.gotop.com.tw

移位代碼類似於按住電腦鍵盤上的 Shift 鍵，之所以這樣命名，是因為老式打字機上的等效鍵會移動機械裝置以鍵入大寫字母。

點字大寫字母指示符（Braille capital indicator）意味著隨後的字母（而且只有隨後的字母）應該是大寫而不是小寫。諸如此類代碼稱為**轉義**代碼（*escape* code）。轉義代碼可以讓你「規避」（escape）代碼的正常解釋，並以不同的方式解釋它。當書面語言（written languages）用二進位代碼（binary codes）表示時，移位代碼和轉義代碼很常見，但它們可能會帶來複雜性，因為如果不知道之前出現哪些代碼，就無法自行解釋單個代碼。

早在 1855 年，一些點字的倡導者就開始以另一排的兩個圓點來擴展這個系統。八圓點點字（Eight-dot Braille）已被用於一些特殊用途，例如音樂、速記和日語漢字字符。因為它將具唯一性之代碼（unique codes）的數量增加到 2^8 或 256，所以它在某些電腦應用程式中也很方便，使得小寫和大寫字母、數字和標點符號皆擁有具唯一性的代碼，而無須擔心移位和轉義代碼。

碁峰 www.gotop.com.tw