

計 算 機 組 織 / 概 論 ( H T T P S : / / N O T F A L S E . N E T / C A T E G O R Y / C O M P U T E R - O R G A N I Z A T I O N )

# 位元組順序 (Byte Order or Endianness) — big-endian vs. little-endian

發表於 2017-02-12 (H T T P S : / / N O T F A L S E . N E T / 19 / B Y T E - O R D E R ) , 鄭中勝 (H T T P S : / / N O T F A L S E . N E T / A U T H O R / J A S O N )

我們時常會以 由左到右、由上到下 的方式書寫，

例: 一數『九千四百八十七』，

通常習慣寫成:『9487』，而不是『7849』，

(雖然，有些國家或族群的習慣可能是後者)

你 不能說他錯，因為這只是習慣的不同。

位元組順序 (Byte Order)，或稱 端序 (Endianness)，即是指 位元組 的排列順序，

同理，不同的硬體架構、網路協議... 其用法不盡相同，

沒有絕對的好壞，只有適合與否。



## 目錄 [隱藏]

端 (Endian)	
以儲存 0x1234ABCD 為例	
由來	
主機位元組順序 (Host Byte Order)	
big-endian	
little-endian	
檢測	
C	
Java	

C#
PHP
bi-endianness
網路位元組順序 (Network Byte Order)
轉換
總結
More

# 端 (Endian)

如前述的 『 9487 』 範例一般，

以 最高有效位元組 (Most Significant Byte, MSB) [\(/14/numeral-system-intro#msb\)](/14/numeral-system-intro#msb) <sup>[註1]</sup> 逐一儲存位元組者，稱為 大頭端 (big-endian)。

反之，如 『 7849 』 般，

以 最低有效位元組 (Least Significant Byte, LSB) [\(/14/numeral-system-intro#msb\)](/14/numeral-system-intro#msb) <sup>[註1]</sup> 逐一儲存位元組者，稱為 小頭端 (little-endian)。

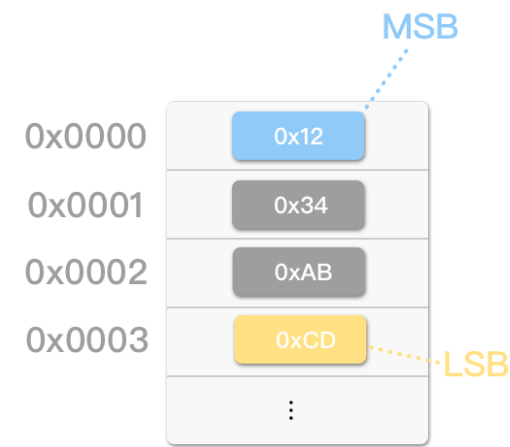
[註1]:

注意，連結內容為: 最高/低 有效位元，而非 位元組！

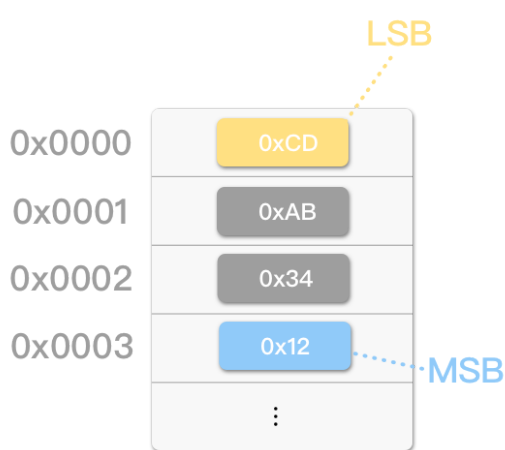
僅做理解名詞用。

## 以儲存 0x1234ABCD 為例

大頭端 (big-endian):



小頭端 (little-endian):



(另還有 Middle-endian，不再累述)

## 由來

端 (Endian) 的由來，相當有趣 🤔:

「*endian*」一詞，來源於十八世紀愛爾蘭作家喬納森·斯威夫特 ( *Jonathan Swift* ) 的小說《格列佛遊記》 ( *Gulliver's Travels* ) 。  
小說中，小人國為水煮蛋該從大的一端 ( *Big-End* ) 剝開還是小的一端 ( *Little-End* ) 剝開而爭論，  
爭論的雙方分別被稱為「大頭派」和「小頭派」。 — 維基百科

## 主機位元組順序 (Host Byte Order)

### big-endian

每台計算機因應其指令集架構 (Instruction Set Architecture, ISA) ，  
其 字組 (/16/basic-unit#word) 定址 (word addressing)、位元組順序 (Byte Order) 不盡相同，  
早期的 MIPS 架構 就是 大頭端 陣營！

big-endian 適合人類習慣，  
逐位元組 Memory dump 超方便閱讀 😊  
且在許多情況下 (如: 欲做數值排序、估計值、符號判斷...)，  
直接檢索 最高有效位元組，相當有用。

### little-endian

最廣為人知的小頭端 (little-endian) 就屬 — intel x86、x86-64 處理器啦！

但既然大頭端這麼直覺，為何還要小頭端呢? 🤔

一個常見的觀點是 — 相同位址 (same address)：

在小頭端 (little-endian) 中，一個值不論是用 8、16、32.. 位元的方式儲存，都可藉由相同的基底位址存取，簡化了硬體的設計，並做到向下兼容。

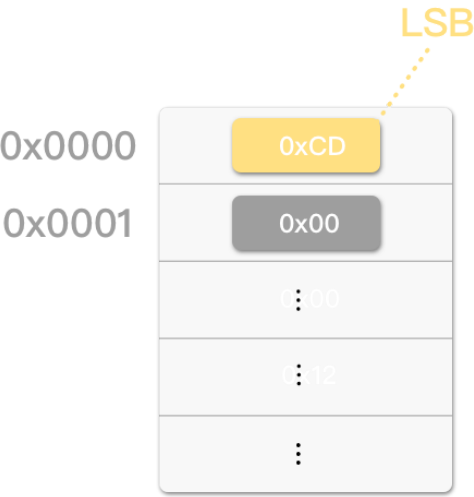
Ex:

8-bit data = 0xCD

換成 16-bit = 0x00CD

換成 32-bit = 0x000000CD

皆透過 相同位址 0x0000 取得



## 檢測

以下提供一些檢測本機「預設」位元組順序 (Byte Order) 的方法：

## C

C 語言，可藉由 指標轉型 與 間接運算子(\*) 達成：

程式碼風格：

hybrid

▼

```

1  #include <stdio.h>
2
3  int main() {
4      int i = 1;
5      char *c = (char *) &i;
6
7      if (*c)
8          printf("LITTLE_ENDIAN\n");
9      else
10         printf("BIG_ENDIAN\n");
11
12     return 0;
13 }

```

Copy

或者，藉由觀察位址變化:

```

1  #include <stdio.h>
2
3  int main() {
4      int num = 0x1234ABCD;
5
6      // 指標轉型: 將位址轉型為 指向 char
7      char *ptrNum = (char *) &num;
8
9      for (int i = 0; i < 4; i++)
10         printf("%p: %02x \n", (void *) ptrNum, (unsigned char) *ptrNum++);
11
12     return 0;
13 }

```

Copy

## Java

Java 能使用 `ByteOrder` 類別:

```

1  import java.nio.ByteOrder;
2
3  public class Main {
4
5      public static void main(String[] args) {
6          System.out.println(ByteOrder.nativeOrder());
7      }
8  }

```

Copy

## C#

C# 使用 `BitConverter` 類別:

```

Console.WriteLine( "IsLittleEndian: {0}",
    BitConverter.IsLittleEndian );

```

## PHP

PHP 則是使用 `pack` 方法:

```
1 <?php
2
3 $result = "BIG_ENDIAN";
4 $i = 0x12345678;
5
6 // 將 i 依本機 Byte Order 打包為 unsigned long
7 $uLong = pack('L', $i);
8
9 // 將 $uLong 依 little-endian 打包為 unsigned long
10 if ($i === current(unpack('V', $uLong))) {
11     $result = "LITTLE_ENDIAN";
12 }
13
14 echo $result;
```

Copy

我目前使用的 MacBook Pro (Retina, 13-inch, Early 2015) ·

執行結果就是:



— — LITTLE\_ENDIAN 😊

## bi-endianness

不是我少打一個 *g* (*big*) !!!

為了提高性能、兼容性...

現今許多架構 (如: PowerPC、MIPS、ARM、IA-64...) 皆可透過『切換』的方式 ·

來支援 big、little 兩種順序 · 也就是 雙端 (bi-endianness) 啦!

這也是為何 · 上方說的是:

檢測『預設』位元組順序 (許多個人電腦 · 皆預設為 little-endian) 。

## 網路位元組順序 (Network Byte Order)

由於 · 不同的主機架構端序不盡相同 · 彼此在網路上傳輸 · 就需有順序規範:

使 *IP* 位址, *Port*, 封包... 能夠通用

也就是 網路位元組順序 (Network Byte Order) 啦！

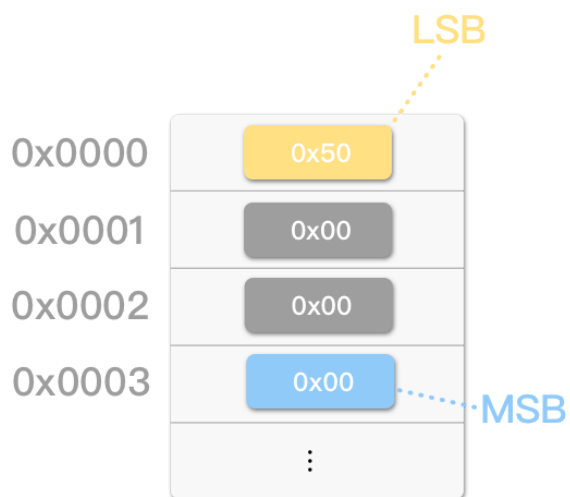
大部分網路協定 (ex: TCP、UDP、IPv4、IPv6...)，

皆是使用 大頭端 (big-endian)，因此兩者通常被視為等價。

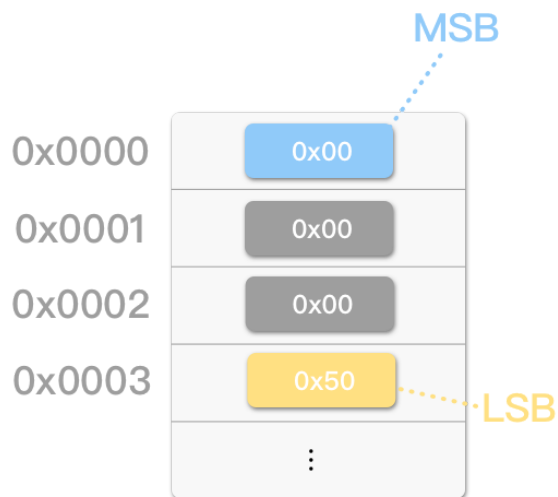
## 轉換

像是，十進位的數字 80 (= 50<sub>16</sub>)，

儲存在 little-endian 中，可能長這樣：



在 big-endian，則變成：



也就是 little-endian 中的 1342177280<sub>10</sub> ☹️...

幸好，Berkeley Socket 定義了一組函式 (通常是 巨集)，

以使 主機位元組順序 與 網路位元組順序 能夠互相轉換。

也就是鼎鼎大名的:

### 1. htons

Return host\_uint16 converted to network byte order

### 2. htonl

Return host\_uint32 converted to network byte order

### 3. ntohs

Return net\_uint16 converted to host byte order

### 4. ntohl

Return net\_uint32 converted to host byte order

[註]：

(h 指的是 host (主機) · n 是 network · u 是 unsigned (無號) ·

s 是 short integer (短整數) · l 是 long integer (長整數) [註2])

[註2]:

需要小心的是，這是函數「原型」的命名方式:

早期多數系統，*short integer* 與 *long integer*，分別是 16 位元 與 32 位元。

然而，現今的 *long integer* 通常已不再是 32 位元！

許多 socket program，都看的到這幾個函式的蹤影，

來看個使用範例:

(礙於篇幅，不加入 socket 部分，

待撰寫 socket 的使用時，再補充說明)



```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <arpa/inet.h>
4  #include <memory.h>
5
6  int main(int argc, char **argv) {
7      struct sockaddr_in svaddr;
8
9      char *address = "127.0.0.1";
10     int port_num = 9527;
11
12     /* Clear structure */
13     memset(&svaddr, 0, sizeof(struct sockaddr_in));
14
15     svaddr.sin_family = AF_INET;
16
17     // 將 port 由 本機位元組順序 轉換為 網路位元組順序
18     svaddr.sin_port = htons(port_num);
19
20     printf("------(1)-----\n");
21     printf("欲轉換 port: %i\n", port_num);
22     printf("-----Result-----\n");
23     printf("htons: %i\n", svaddr.sin_port);
24
25     printf("\n\n");
26
27     // 將 address 由 本機位元組順序 轉換為 網路位元組順序
28     if (inet_pton(AF_INET, address, &svaddr.sin_addr) <= 0) {
29         printf("inet_pton failed for address %s\n", address);
30         exit(EXIT_FAILURE);
31     }
32
33     printf("------(2)-----\n");
34     printf("欲轉換位址: %s\n", address);
35     printf("-----Result-----\n");
36     printf("inet_pton: %p\n", svaddr.sin_addr);
37
38     return 0;
39 }
40
41
42 /*
43  * Result:
44  *
45  * -----(1)-----
46  * 欲轉換 port: 9527
47  * -----Result-----
48  * htons: 14117
49  *
50  * -----(2)-----
51  * 欲轉換位址: 127.0.0.1
52  * -----Result-----
53  * inet_pton: 0x100007f
54  */

```

Copy

中間有另一個重要的函式 — `inet_pton`，

用以取代傳統的 `inet_aton`、`inet_addr...`。

可以將 IPv4、IPv6 位址，轉換為 網路位元組順序，

其中 p 是 presentation (表示式)，n 是 network。

**Q: 如果本機是 big-endian，是否就不需用這些函式?**

是的，但考量到程式的可攜性，

仍應使用這些函式，避免不必要的問題。


# 總結


除了 IP 與 埠號，資料格式本身，也必須定義位元組順序、編組或序列化格式...，  
如：訊框 (frame), 遠端程序呼叫 (RPC) 的 外部資料表示方式 (XDR)、XML...。


因此，不論是跨網路 或是 儲存裝置 的資料傳輸，  
位元組順序 (Byte Order)，時常是 coding 需注意的細節，  
使用了錯誤的順序，便會造成檔案損毀或例外錯誤。


範例原始檔 📄📄📄 (<https://github.com/JS-Zheng/blog/tree/master/19.%20Byte-Order>).

分享此文：

 (<https://notfalse.net/19/byte-order?share=twitter&nb=1>)

 (<https://notfalse.net/19/byte-order?share=facebook&nb=1>)

 (<https://notfalse.net/19/byte-order?share=google-plus-1&nb=1>)

 (<https://notfalse.net/19/byte-order?share=pocket&nb=1>)

More



<https://notfalse.net/20/signed-number-representations>

有號數字表示法 -- 2 的補數、1 的補數 與 符號大小 (<https://notfalse.net/20/signed-number-representations>)

2017-02-25



<https://notfalse.net/27/tcp-error-control>

TCP 錯誤控制 (Error Control) (<https://notfalse.net/27/tcp-error-control>)

2017-03-23



<https://notfalse.net/24/tcp-flow-control>

TCP 流量控制 (Flow Control) (<https://notfalse.net/24/tcp-flow-control>)

2017-03-08

ARRAY ( <a href="https://notfalse.net/tag/array">https://notfalse.net/tag/array</a> )	0	BYTEORDER ( <a href="https://notfalse.net/tag/byteorder">https://notfalse.net/tag/byteorder</a> )	0	HTONS ( <a href="https://notfalse.net/tag/htons">https://notfalse.net/tag/htons</a> )	0
MEMORY DUMP ( <a href="https://notfalse.net/tag/memory-dump">https://notfalse.net/tag/memory-dump</a> )	0	MIPS ( <a href="https://notfalse.net/tag/mips">https://notfalse.net/tag/mips</a> )	0	SOCKET ( <a href="https://notfalse.net/tag/socket">https://notfalse.net/tag/socket</a> )	0
WORD ( <a href="https://notfalse.net/tag/word">https://notfalse.net/tag/word</a> )	0	位元 ( <a href="https://notfalse.net/tag/%E4%BD%8D%E5%85%B3">https://notfalse.net/tag/%E4%BD%8D%E5%85%B3</a> )	0	進制 ( <a href="https://notfalse.net/tag/%E9%80%B2%E5%88%B6">https://notfalse.net/tag/%E9%80%B2%E5%88%B6</a> )	

作者: 鄭中勝  
喜愛音樂，但不知為何總在打程式？期許能重新審視、整理自身所學，  
幫助有需要的人。

匿名訪客表示:  
2020-02-20 15:36:24 (HTTPS://NOTFALSE.NET/19/BYTE-ORDER#COMMENT-369)

Hi 請問 這文章能借轉到臉書私人社團嗎?

回覆

發表迴響

在此輸入你的回應...

搜尋...

適用電子郵件訂閱網站

輸入你的電子郵件地址訂閱網站的新文章，使用電子郵件接收新通知。

電子郵件位址

訂閱

分類

選取分類

程式碼風格

hybrid

Popular		Recent	
	<b>進制轉換 (二進制、八進制、十進制、十六進制) (https://notfalse.net/17/positional-numeral-systems-conversion)</b> 2月 2, 2017		
	<b>中繼器 (Repeater)、集線器 (Hub)、橋接器 (Bridge)、交換器 (Switch) 原理與介紹 (https://notfalse.net/66/repeater-hub-bridge-switch)</b> 6月 12, 2018		
	<b>XMLHttpRequest — JavaScript 發送 HTTP 請求 (I) (https://notfalse.net/29/xmlhttprequest)</b> 5月 31, 2017		
	<b>控制反轉 (IoC) 與 依賴注入 (DI) (https://notfalse.net/3/ioc-di)</b> 11月 23, 2016		
(https://n			
<b>引數 (Argument) vs. 參數 (Parameter) (https://notfalse.net/6/arg-vs-param)</b>			

[AJAX \(https://notfalse.net/tag/ajax\)](https://notfalse.net/tag/ajax) [Array \(https://notfalse.net/tag/array\)](https://notfalse.net/tag/array) [Async/Await \(https://notfalse.net/tag/asyncawait\)](https://notfalse.net/tag/asyncawait) [Checksum \(https://notfalse.net/tag/checksum\)](https://notfalse.net/tag/checksum) [CORS \(https://notfalse.net/tag/cors\)](https://notfalse.net/tag/cors) [GoF \(https://notfalse.net/tag/gof\)](https://notfalse.net/tag/gof) [IoC/DI \(https://notfalse.net/tag/iocdi\)](https://notfalse.net/tag/iocdi) [jQuery \(https://notfalse.net/tag/jquery\)](https://notfalse.net/tag/jquery) [Liskov 替代原則 \(https://notfalse.net/tag/liskov-%e6%9b%bf%e4%bb%a3%e5%8e%9f%e5%89%87\)](https://notfalse.net/tag/liskov) [Memory dump \(https://notfalse.net/tag/memory-dump\)](https://notfalse.net/tag/memory-dump) [MSS \(https://notfalse.net/tag/mss\)](https://notfalse.net/tag/mss) [mtu \(https://notfalse.net/tag/mtu\)](https://notfalse.net/tag/mtu) [Pub/Sub \(https://notfalse.net/tag/pubsub\)](https://notfalse.net/tag/pubsub) [rwnd \(https://notfalse.net/tag/rwnd\)](https://notfalse.net/tag/rwnd) [SACK \(https://notfalse.net/tag/sack\)](https://notfalse.net/tag/sack) [Socket \(https://notfalse.net/tag/socket\)](https://notfalse.net/tag/socket) [SOLID \(https://notfalse.net/tag/solid\)](https://notfalse.net/tag/solid) [stack \(https://notfalse.net/tag/stack\)](https://notfalse.net/tag/stack) [sync vs. async \(https://notfalse.net/tag/sync-vs-async\)](https://notfalse.net/tag/sync-vs-async) [Vue \(https://notfalse.net/tag/vue\)](https://notfalse.net/tag/vue) [word \(https://notfalse.net/tag/word\)](https://notfalse.net/tag/word) 一的補數 (https://notfalse.net/tag/%e4%b8%80%e7%9a%84%e8%a3%9c%e6%95%b8) 介面導向程式設計 (https://notfalse.net/tag/%e4%bb%8b%e9%9d%a2%e5%b0%8e%e5%90%91%e7%a8%8b%e5%bc%8f%e8%a8%ad%e8%a1%e5%bc%8f) 位元 (https://notfalse.net/tag/%e4%bd%8d%e5%85%83) 依賴倒置原則 (https://notfalse.net/tag/%e4%be%9d%e8%b3%b4%e5%80%92%e7%bd%ae%e5%8e%9f%e5%89%87) 函式呼叫 (https://notfalse.net/tag/%e5%87%bd%e5%bc%8f%e5%91%bc%e5%8f%ab) 同源政策 (https://notfalse.net/tag/%e5%90%8c%e6%ba%90%e6%94%bf%e7%ad%96) 單一職責原則 (https://notfalse.net/tag/%e5%96%ae%e4%b8%80%e8%81%b7%e8%b2%ac%e5%8e%9f%e5%89%87) 回調函式 (https://notfalse.net/tag/%e5%9b%9e%e8%aa%bf%e5%87%bd%e5%bc%8f) 多型 (https://notfalse.net/tag/%e5%a4%9a%e5%9e%8b) 多載 (https://notfalse.net/tag/%e5%a4%9a%e8%bc%89) 存取範圍 (https://notfalse.net/tag/%e5%ad%98%e5%8f%96%e7%af%84%e5%9c%8d) 工廠方法模式 (https://notfalse.net/tag/%e5%b7%a5%e5%bb%a0%e6%96%b9%e6%b3%95%e6%a8%a1%e5%bc%8f) 延遲確認 (https://notfalse.net/tag/%e5%bb%b6%e9%81%b2%e7%a2%ba%e8%aa%8d) 快速重送 (https://notfalse.net/tag/%e5%bf%ab%e9%80%9f%e9%87%8d%e9%80%81) 滑動視窗 (https://notfalse.net/tag/%e6%bb%91%e5%8b%95%e8%a6%96%e7%aa%97) 箭頭函式 (https://notfalse.net/tag/%e7%ae%ad%e9%a0%ad%e5%87%bd%e5%bc%8f) 覆寫 (https://notfalse.net/tag/%e8%a6%86%e5%af%ab) 觀察者模式 (https://notfalse.net/tag/%e8%a7%80%e5%af%9f%e8%80%85%e6%a8%a1%e5%bc%8f) 記憶體 (https://notfalse.net/tag/%e8%a8%98%e6%86%b6%e9%ab%94) 設計原則 (https://notfalse.net/tag/%e8%a8%ad%e8%a8%88%e5%8e%9f%e5%89%87) 設計模式 (https://notfalse.net/tag/%e8%a8%ad%e8%a8%88%e6%a8%a1%e5%bc%8f) 進制 (https://notfalse.net/tag/%e9%80%b2%e5%88%b6) 遞迴 (https://notfalse.net/tag/%e9%81%9e%e8%bf%b4) 開閉原則 (https://notfalse.net/tag/%e9%96%8b%e9%96%89%e5%8e%9f%e5%89%87)