

程式扎記

This is a blog to track what I had learned and share knowledge with all who can take advantage of them

[英文學習]	[計算機概論]	[深入雲計算]	[雜七雜八]	[Algorithm in Java]	[Data Structures with Java]	[IR Class]	[Java 文章收集]								
[Java 套件]	[JVM 應用]	[LFD Note]	[MangoDB]	[Math CC]	[MongoDB]	[MySQL 小學堂]	[Python 考題]	[Python 常見問題]							
[心得扎記]	[網路教學]	[C 常見考題]	[C 範例代碼]	[C/C++ 範例代碼]	[Intro Alg]	[Java 代碼範本]	[Java 套件]	[Linux 小技巧]	[Linux 常見問題]						
[ML In Action]	[ML]	[MLP]	[Postgres]	[Python 學習筆記]	[Quick Python]	[Software Engineering]	[The python tutorial]	工具收集							
ActiveMQ In Action	AI	Algorithm	Android	Ansible	AWS	Big Data 研究	C/C++	C++	CCDH	CI/CD	Coursera				
Design Pattern	Device Driver Programming		Docker	Docker 工具		Docker Practice	Eclipse	English Writing	ExtJS 3.x	FP	FreeBSD				
GCC	Git	Git Pro	GNU	Golang	Gradle	Groovy	Hadoop	Hadoop. Hadoop Ecosystem		Java	Java Framework	Java 8			
JavaScript	Jenkins	JFreeChart	Kaggle	Kali/Metasploit		Keras	KVM	Learn Spark	LeetCode	Linux	Lucene	Math			
MPI	Nachos	Network	NLP	node.js	OO	OpenCL	OpenMP	OSC	OSGi	Pandas	Perl	PostgreSQL	Python		
Python 自製工具		Python Std Library		Python tools		QEMU	R	Real Python		RIA	RTC	Ruby	Ruby Packages	Scala	Scipy
TensorFlow		Tools	UML	Unix	Verilog	Vmware	Windows 技巧		wxPython						

2011年5月25日 星期三

[C 文章收集] Bitwise Operation

轉載自 [這裡](#)
前言：
歡迎來到二進位的世界。電腦資料都是以二進位儲存，想當然程式語言的變數也都是以二進位儲存。在 C/C++ 當中有幾個位元運算子： << SHIFT LEFT 、 >> SHIFT RIGHT 、 & AND 、 | OR 對變數進行位元運算，接下來要介紹位元運算的一些用途。

Bitwise operator 介紹：
*** << SHIFT LEFT , >> SHIFT RIGHT**

這兩個運算子的功能主要是移動一個變數中的所有位元，位元向左 / 向右移動之後，最高位 / 最低位的位元會消失，最低位 / 最高位的位元補 0，運算範例如下：

```
5 << 1 = 10 // 00101 的全部位元向左移動一位數變成 01010.
5 << 2 = 20 // 00101 的全部位元向左移動兩位數變成 10100.
5 >> 1 = 2 // 00101 的全部位元向右移動一位數變成 00010.
5 >> 2 = 1 // 00101 的全部位元向右移動一位數變成 00001.
```

在十進位當中，當全部位數向左移動一位時，數值大小會變成原來的十倍，向左移動兩位時，會變成原來的百倍。這種情形在二進位也是成立的，當全部位元向左移動一位時，會變成原來的兩倍，向; 倍。至於往右移動也是類似道理，變成了除法而已。由於電腦進行位元運算比乘法、除法運算快上許多，所以有很多專業的程式設計師，會利用位元運算來取代乘法、除法運算，優點是程式執行效率增; 低。範例如下：

view plain copy to clipboard print ?

```
01. int n = 5;
02. n = n >> 1; // 即是 n = n / 2 之意。
03. /* 該式子也可寫成 n >>= 1 或 n /= 2。 */
```

*** & AND**

```
0 & 0 = 0
0 & 1 = 0
1 & 0 = 0
1 & 1 = 1
```

& 的功能是將兩個變數對應的位元進行 AND 邏輯運算，然後產生新變數。& 的特色，就是可以判斷出位元是不是 1，例如我們想要數一個變數有幾個位元是 1，則可以如下操作：

view plain copy to clipboard print ?

```
01. int n = 19; // 待測數
02. int digit = sizeof(n) * 8; // 待測數為幾位元。
03. int c = 0; // Counter
04. for(int i=0; i
05. {
06.     if(n & (1<
07. }
08. printf("Result : %d\n", c);
```

*** | OR**

```

0 | 0 = 0
0 | 1 = 1
1 | 0 = 1
1 | 1 = 1

```

| 的功能是將兩個變數對應的位元進行 OR 邏輯運算，然後產生新變數。其特色，就是把位元強制標記成 1。例如我們想要把五位數標成 1：

```

view plain copy to clipboard print ?
01. int mark_5th_bit(int n)
02. {
03.     return n | (1 << 4);
04. }

```

* ^ XOR

```

0 ^ 0 = 0
0 ^ 1 = 1
1 ^ 0 = 1
1 ^ 1 = 0

```

^ 的功能是將兩個變數對應的位元進行 XOR 邏輯運算，然後產生新變數。其特色，就是把位元的 0 和 1 顛倒。例如我們想要顛倒第五位數：

```

view plain copy to clipboard print ?
01. int reverse_5th_bit(int n)
02. {
03.     return n ^ (1 << 4);
04. }

```

* ~ NOT

```

~ 0 = 1
~ 1 = 0

```

~ 的功能是顛倒一個變數每一個位元的 0 和 1。

Bitwise 應用範例：

- 整數加一與減一

```

view plain copy to clipboard print ?
01. // 注意：比直接加一和減一還要慢。
02. int add_one(int x)
03. {
04.     return ~x; // ++x
05. }
06.
07. int sub_one(int x)
08. {
09.     return ~x; // --x
10. }

```

- 整數變號

```

view plain copy to clipboard print ?
01. int negative(int x)
02. {
03.     return ~x + 1; // -x;
04. }
05.
06. int negative(int x)
07. {
08.     return (x ^ -1) + 1; // -x;
09. }

```

- 判斷一整數是偶數還是奇數

```

view plain copy to clipboard print ?
01. // 若回傳1則為奇數，回傳0則為偶數。
02. int is_odd(int x)
03. {
04.     return x & 1; // x % 2;
05. }

```

- 整數取絕對值 (32 位元整數)

```

view plain copy to clipboard print ?
01. int abs(int x)
02. {
03.     // x < 0 ? -x : x;
04.     // x >> 31 = 111...111 (如果x是負數) or 000...000 (如果x是正數)
05.     // x ^ (x>>31) => 如果 x 為負數則將 x 的 0轉1, 1轉0, 如果 x 為正數, 則保持x 不變。
06.     // (x ^ (x >> 31)) - (x >> 31) => 如果 x 為正數則 x-0=x, 如果 x 為負數則 ~x-(-1) = -x
07.     return (x ^ (x >> 31)) - (x >> 31);

```

```
08. | }
```

- 最低位的位元 1

view plain copy to clipboard print ?

```
01. int lowest_bit_1(int x)
02. {
03.     return x & -x;
04. }
```

- 判斷一個整數是不是 2 的次方

view plain copy to clipboard print ?

```
01. bool is_power_of_2(int x)
02. {
03.     return (x & -x) == x;
04. }
```

- 交換兩個 int 變數

view plain copy to clipboard print ?

```
01. void swap(int& x, int& y)
02. {
03.     x = x ^ y; // x' = x ^ y
04.     y = x ^ y; // y' = x' ^ y = x ^ y ^ y = x
05.     x = x ^ y; // x = x' ^ y' = x ^ y ^ x = y
06. }
```

- 計算有幾個位元是 1 (32 位元整數)

view plain copy to clipboard print ?

```
01. int count_bits(int x)
02. {
03.     x = (x & 0x55555555) + ((x & 0xaaaaaaaa) >> 1);
04.     x = (x & 0x33333333) + ((x & 0xcccccccc) >> 2);
05.     x = (x & 0x0f0f0f0f) + ((x & 0xf0f0f0f0) >> 4);
06.     x = (x & 0x00ff00ff) + ((x & 0xff00ff00) >> 8);
07.     x = (x & 0x0000ffff) + ((x & 0xffff0000) >> 16);
08.     return x;
09. }
10.
11. int count_bits2(unsigned int n) {
12.     int i=0;
13.     for ( ; n != 0; n >>= 1)
14.         if (n & 1)
15.             ++i;
16.     return i;
17. }
```

- 顛倒位元順序 (32 位元整數)

view plain copy to clipboard print ?

```
01. int reverse_bits(int x)
02. {
03.     x = ((x >> 1) & 0x55555555) | ((x << 1) & 0xaaaaaaaa);
04.     x = ((x >> 2) & 0x33333333) | ((x << 2) & 0xcccccccc);
05.     x = ((x >> 4) & 0x0f0f0f0f) | ((x << 4) & 0xf0f0f0f0);
06.     x = ((x >> 8) & 0x00ff00ff) | ((x << 8) & 0xff00ff00);
07.     x = ((x >> 16) & 0x0000ffff) | ((x << 16) & 0xffff0000);
08.     return x;
09. }
```

補充說明：

* [Bit Twiddling Hacks](#)

於 5月25, 2011

標籤：C/C++

1 則留言:




Unknown 2018年7月4日 晚上8:03

國中學這個好難

回覆

輸入您的留言...

 發表留言的身分：

kuo5564@gm: ▼

發布

預覽

較新的文章

首頁

訂閱： [張貼留言 \(Atom\)](#)

[Python 文章收集] Pydon't - Pass-by-value, reference, and assignment

Source From [Here](#) Preface When you call a function in Python and give it some arguments... Are they passed by value? No! By reference? No!...



[Verilog Tutorial] 行為模型的敘述: always, if/else, case 與 for loop

Preface: 在這個階層中，我們只需考慮電路模組的功能，而不需考慮其硬體的詳細內容. Verilog 的時序控制為以事件為基礎的時序控制: * 接線或暫存器的值被改變。 * 模塊...

[Linux 命令] du : 顯示目錄或是檔案的大小

屬性：系統相關 - 檔案與目錄 語法： du [參數] [檔案] 參數 | 功能 -a | 顯示目錄中個別檔案的大小 -b | 以bytes為單位顯示 -c | 顯示個別檔案大小與總和 -D | 顯示符號鏈結的來源檔大小

[C 文章收集] Bitwise Operation

轉載自 [這裡](#) 前言：歡迎來到二進位的世界。電腦資料都是以二進位儲存，想當然程式語言的變數也都是以二進位儲存。在 C/C++ 當中有幾個位元運算子： << SHIFT LEFT 、 >> SHII

搜尋此網誌

首頁

關於我自己



John

Where there is a will, there is a way!

[檢視我的完整簡介](#)

網誌存檔

- 2021 (90)

► 2020 (130)

► 2019 (106)

► 2018 (144)

► 2017 (248)

► 2016 (273)

► 2015 (276)

► 2014 (245)

► 2013 (112)

► 2012 (197)

▼ 2011 (265)

► 十二月 (10)

► 十一月 (14)

► 十月 (15)

► 九月 (13)

► 八月 (11)

► 七月 (14)

► 六月 (13)

▼ 五月 (26)

[ExtJS2.0 開發指南] CH4 : 常用表單 - 表單及表單元素 (3)

[Java 代碼範本] 取得系統記憶體, CPU 相關使用資訊
- puremonkey2010.blogspot.com/2011/05/c-bitwise-operation.html
- 4/5

- [ExtJS2.0 開發指南] CH4 : 常用表單 - 表單及表單元素 (2)
 - [C 文章收集] Bitwise Operation
 - [ExtJS2.0 開發指南] CH4 : 常用表單 - 表單及表單元素 (1)
 - [ExtJS2.0 開發指南] CH3 : ExtJS 的基本功能 - 實現工具欄與菜單欄
 - [NP in MS] Scalable Winsock Applications (Part1 ...
 - [ExtJS2.0 開發指南] CH3 : ExtJS 的基本功能 - 進度條組件介紹
 - [MSDN 文章收集] I/O Concepts : I/O Completion Ports
 - [ExtJS2.0 開發指南] CH3 : ExtJS 的基本功能 - 訊息提示框組件基礎
 - MBTI 職業性格測試報告
 - [Data Structures with Java] Section 26.1 : Repre...
 - [ExtJS2.0 開發指南] CH3 : ExtJS 的基本功能 - 組件配置說明
 - [Data Structures with Java] Section 25.6 : Minim...
 - [Data Structures with Java] Section 25.5 : Dijks...
 - [HF Software Dev] Chap6 : Version Control - Defe...
 - [HF Software Dev] Chap4 : User stories and tasks...
 - [NP in MS] Socket Options and ioctl (Part1 : So...
 - [Data Structures with Java] Section 25.4 : Short...
 - [Data Structures with Java] Section 25.2 : Stron...
 - [Data Structures with Java] Section 25.2 : Stron...
 - [NP in MS] Winsock I/O Methods (Part2 : Socket I...
 - [C++ 文章收集] 教學 : typedef 知多少?
 - [NP in MS] Winsock I/O Methods (Part2 : Socket I...
 - [Data Structures with Java] Section 25.1 : Topol...
 - [Data Structures with Java] Section 24.3 : Graph...
- 四月 (35)
 - 三月 (42)
 - 二月 (29)
 - 一月 (43)
 - 2010 (264)

總網頁瀏覽量



檢舉濫用情形

學習筆記

- 鳥哥 Linux 私房菜
- Perl Tutorial
- 良葛哥學習筆記

頂尖企業主題. 技術提供 : Blogger.