

C Programming I

HW0205

Answer

作者: 吳文元 (jw910731)

日期: 2021/03/20

```
1 #include <stdio.h>
2 #include <stdint.h>
3
4 int main()
5 {
6     int32_t number = 0;
7
8     scanf( "%d", & number );
9
10    int32_t bit = 1;
11    bit = bit << 31;
12
13    for( int i = 0 ; i < 32 ; i++ ){
14        if( bit & number )
15            printf( "1" );
16        else
17            printf( "0" );
18        bit = bit >> 1;
19    }
20    return 0;
21 }
```

Listing 1: 題敘中的程式

其中第11行中的左移在規格書中是未定義行為！根據規格書ISO/IEC 9899:201x中的第 95 頁指出

The result of $E1 \ll E2$ is $E1$ left-shifted $E2$ bit positions; vacated bits are filled with zeros. If $E1$ has an unsigned type, the value of the result is $E1 \times 2^{E2}$, reduced modulo one more than the maximum value representable in the result type. If $E1$ has a signed type and nonnegative value, and $E1 \times 2^{E2}$ is representable in the result type, then that is the resulting value; otherwise, the behavior is undefined.

而我們可以發現程式中第11行，裡面表達的 $1 \ll 31$ 所指的 1×2^{31} 已超出 32 位有號整數所能儲存的 $[2^{31} - 1, -2^{31}]$ 範圍了，使得這個 expression 的行為將會是 platform specific 的。

這隻程式應該使用union，將有號整數與無號整數共用同個空間，並使用無號整數做安全的位元運算！我修改後如下：

```

1 | #include <stdio.h>
2 | #include <stdint.h>
3 |
4 | union soviet{
5 |     int32_t num;
6 |     uint32_t usgn_num;
7 | };
8 |
9 | int main()
10| {
11|     union soviet number;
12|
13|     scanf( "%d", & number.num );
14|
15|     int32_t bit = 1;
16|     bit = bit << 31;
17|
18|     for( int i = 0 ; i < 32 ; i++ ){
19|         if( bit & number.usgn_num )
20|             printf( "1" );
21|         else
22|             printf( "0" );
23|         bit = bit >> 1;
24|     }
25|     return 0;
26| }

```

Listing 2: 修改後的程式

這樣就可以避免掉不安全且不可移植的行為了！