链滴

【读书笔记】《Computer System A Programmer'"'"'s Perspective》Writing TMin32 in C

作者：armstrong

原文链接：https://ld246.com/article/1357370795290

来源网站：链滴

Writing TMin32 in C (P105, second edition)

    C语言头文件<limits.h>中定义了TMin和TMax的取值：

<pre> /*
 * Maximum and minimum values for ints.
 */
 #define INT_MAX 2147483647
 #define INT_MIN (-INT_MAX-1)
 #define UINT_MAX 0xffffffff
 /* </pre>

   可以看到32位有符号整数取值范围为[-2147483648,2147483647]。书中提到了一有趣的问题：为什么最小值要写成-2147483647-1（-INT_MAX-1）？为什么不直接写成-214748368?

    答：由于Tow's Complement正负数的不对称性，直接写成-2147483648可会导致溢出。在某些编译器中，对于-X的计算，先取数值X再取负数。2147483648超过了32位有符整数的表示范围，造成了溢出，因此必须写成-2147483647-1。

<h2> Errata for
 Computer Systems: A Programmer's Perspective (CS:APP) </h2>

http://csapp.cs.cmu.edu/public/1e/public/errata.html

    p. 60, Problem 2.20:

      The value -2147483648 should be written -2147483647-1 i
 entries 1, 2, and 4, and as(-2147483647-1) in entries 3 and 5. See the <a href="h
tp://csapp.cs.cmu.edu/public/1e/public/tmin.html">Note on writing TMin in C for
a discussion.
      2nd & 3rd entries in table should have -2147483647 rather than&n
sp;-21474836487.
      4th & 5th entries in table should have 2147483647 rather than&nb
p;21474836487.

 <h2> A Note on TMin </h2>

 http://csapp.cs.cmu.edu/publ
c/1e/public/tmin.html

 <h1> Summary </h1> Due to one of the rules for processing integer constants in ANSI C, th

numeric constant -2147483648 is handled in a peculiar way on a 32-bit, two's c
mplement machine.
 The problem can be corrected by writing -2147483647-1, rather than -214
483648 in any C code.
<h1> Description of Problem </h1> The ANSI C standard requires that an integer constant t
o large to be represented as a signed integer be ``promoted'' to an unsigned value. When GC
encounters the value 2147483648, it gives a warning message: ``warning: decimal constant is
o large that it is unsigned.'' The result is the same as if the value had been written 214
483648U.
 The compiler processes an expression of the form -X by first
eading the expression X and then negating it. Thus, when the C c
mpiler encounters the constant -2147483648, it first processes 2147483648, yiel
ing 2147483648U, and then negates it. The unsigned negation of this value is also&nb
p;2147483648U. The bit pattern is correct, but the type is wrong!
<h1> Writing TMin in Code </h1> The ANSI C standard states that the maximum and minim
m integers should be declared as constants INT_MAX and INT_MIN
n the file limits.h. Looking at this file on an IA32 Linux machine (in the directory
usr/include), we find the following declarations:

<pre>/* Minimum and maximum values a `signed int' can hold. */#define INT_MAX2147483
47#define INT_MIN(-INT_MAX - 1)</pre>
 This method of declaring INT_MIN avoids accidental promotion and also a
oids any warning messages by the C compiler about integer overflow.
 The following are ways to write TMin_32 for a 32-bit machine that give the correct value
and type, and don't cause any error messages:

 -2147483647-1
 (int) 2147483648U
 1<<31
 The first method is preferred, since it indicates that the result will be a negative numbe
.