

中国慕课网：计算机系统基础(一)：程序的表示、转换与链接

袁春风

例1：
ISO C90标准下，在32位系统上
以下C表达式的结果是什么？

$-2147483648 < 2147483647$

结果：**false**（与事实不符）！**Why?**

以下关系表达式结果呢？

`int i = -2147483648;`

`i < 2147483647`

结果：**true**！**Why?**

$-2147483647-1 < 2147483647$ ，**结果怎样？**

理解该问题需要知道：
编译器如何处理字面量
高级语言中运算规则
高级语言与指令之间的对应
机器指令的执行过程
机器级数据的表示和运算
.....

例2 :

```
sum(int a[], unsigned len)
{
    int i, sum = 0;
    for (i = 0; i <= len-1; i++)
        sum += a[i];
    return sum;
}
```

当用len=0调用sum函数时，其返回值应该是多少？

当参数len为0时，返回值应该是0，但是在机器上执行时，却发生访存异常。但当len为int型时则正常。Why?



例3：

若x和y为int型，当x=65535时， $y=x*x$ ；y的值为多少？

$y=-131071$ 。Why？

现实世界中， $x^2 \geq 0$ ，但在计算机世界并不一定成立。

对于任何int型变量x和y， $(x > y) == (-x < -y)$ 总成立吗？

当x=-2147483648，y任意（除-2147483648外）时不成立

Why？

在现实世界中成立，

但在计算机世界中并不一定成立。

例4：

main.c

```
int d=100;
int x=200;
int main()
{
    p1();
    printf ( "d=%d, x=%d\n" , d, x );
    return 0;
}
```

p1.c

```
double d;

void p1( )
{
    d=1.0;
}
```

打印结果是什么？

d=0 , x=1 072 693 248

Why ?

例5：

```
/* 复制数组到堆中，count为数组元素个数 */  
int copy_array(int *array, int count) {  
    int i;  
    /* 在堆区申请一块内存 */  
    int *myarray = (int *) malloc(count*sizeof(int));  
    if (myarray == NULL)  
        return -1;  
    for (i = 0; i < count; i++)  
        myarray[i] = array[i];  
    return count;  
}
```

当 $\text{count}=2^{30}+1$ 时，
程序会发生什么情况？

当参数count很大时，则
 $\text{count} \times \text{sizeof}(\text{int})$ 会溢出。
如 $\text{count}=2^{30}+1$ 时，
 $\text{count} \times \text{sizeof}(\text{int})=4$ 。



堆 (heap) 中大量
数据被破坏！

例6：

代码段一：

```
int a = 0x80000000;
```

```
int b = a / -1;
```

```
printf("%d\n", b);
```

运行结果为-2147483648

objdump

反汇编代码,
得知除以 -1
被优化成取
负指令neg,
故未发生除
法溢出

代码段二：

```
int a = 0x80000000;
```

```
int b = -1;
```

```
int c = a / b;
```

```
printf("%d\n", c);
```

运行结果为“Floating point exception”，显然CPU检测到了溢出异常

a/b用除法指令IDIV实现，但它不生成OF
标志，那么如何判断溢出异常的呢？

实际上是“除法错”异常#DE（类型0）
Linux中，对#DE类型发SIGFPE信号