嵌入式C语言之-

数据运算溢出的危害

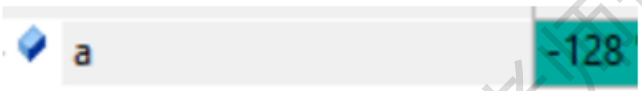
讲师：叶大鹏

什么是溢出

- **当计算机中要存储的数值超出了该类型可以表示的范围就会发生溢出；**
- 比如int8_t类型数据范围为[-128，127]，如果要存储的数为128，此时会发生上溢；要存储的数为-129，此时会发生下溢；
- 就像向杯子中倒水，水超出了杯子的容量，就会溢出。

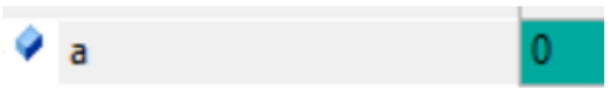
```
int8_t a = 127;
a++;
```

- 输出结果：




```
uint8_t a = 255;
a++;
```

- 输出结果：



什么是溢出

- 对于整型溢出，分为无符号整型溢出和有符号整型溢出：

- **对于unsigned整型溢出，C的规范是有定义的：**

“溢出后的数会以 **该类型最大值+1** 作模运算”，也就是说，如果一个uint8_t（1字节，8bits）溢出了，会把溢出的值与256求模。

- **对于signed整型的溢出，C的规范定义是“undefined behavior”**

也就是说，编译器爱怎么实现就怎么实现。

```
int8_t x = 0x7f;
int8_t y = 0x05;
int8_t r = x * y;
```

- 输出结果：

溢出的危害

```
void data_proc(char *s)
{
        for (uint8_t i = strlen(s) - 1; i >= 0; i--)
        {
                if (s[i] == 'a')
                {
                        s[i] = 'A';
                }
        }
        xxxx;
}
```

- 这段代码是我们经常用的从尾部遍历一个数组的for循环;
- strlen()返回值是一个unsigned int，如果strlen(s)是0，这个循环会成为个什么情况?
- 于是strlen(s) – 1 不会成为 -1，而是成为了(uint8_t)(-1)，一个正的最大数，导致数组越界访问，进而会影响函数后面的数据运算结果或者执行逻辑。

溢出的危害

```
#define MAX_LEN 200

int copy_something(char *buf, int8_t len)
{
    char mybuf[MAX_LEN];
    if(len > MAX_LEN)
    {
        return -1;
    }
    return memcpy(mybuf, buf, len);
}
```

- len是个signed类型，而memcpy需要一个unsigned 类型；
- 于是，len会被提升为unsigned，此时，如果我们给len传一个负数-1，会通过了if的检查，但在memcpy里会被提升为一个正数255，结果会导致数组越界访问，进而会影响函数后面的数据运算结果或者执行逻辑。

总结

- 溢出显然是一个BUG，我们应充分考虑各种数据的取值范围，使用合适的数据类型，减少失误；
- 对于溢出的问题，没有行之有效的方法论可以彻底杜绝，只能依赖工程师的经验以及责任心，在编写程序时对其保持敏感度。

THANK YOU!