



§. 浮点数机内存储格式(IEEE 754)理解

基础知识：用于看懂float型数据的内部存储格式的程序如下：

注意：除了对黄底红字的具体值进行改动外，其余部分不要做改动，也暂时不需要弄懂为什么（需要第6章的知识才能弄懂）

```
#include <iostream>
using namespace std;
int main()
{
    float f = 123.456;
    unsigned char* p = (unsigned char*)&f;
    cout << hex << (int)(*p) << endl;
    cout << hex << (int)*(p+1) << endl;
    cout << hex << (int)*(p+2) << endl;
    cout << hex << (int)*(p+3) << endl;
    return 0;
}
```

//注：忽略本题出现的warning

Microsoft
79
e9
f6
42

上例解读：单精度浮点数123.456，在内存中占四个字节，四个字节的价值依次为0x42 0xf6 0xe9 0x79（按打印顺序逆向取）

转换为32bit则为：0100 0010 1111 0110 1110 1001 0111 1001

符号位

8位指数

23位尾数



§. 浮点数机内存储格式(IEEE 754)理解

基础知识：用于看懂double型数据的内部存储格式的程序如下：

注意：除了对黄底红字的具体值进行改动外，其余部分不要做改动，也暂时不需要弄懂为什么（需要第6章的知识才能弄懂）

```
#include <iostream>
using namespace std;
int main()
{
    double d = 1.23e4;
    unsigned char* p = (unsigned char*)&d;
    cout << hex << (int)(*p) << endl;
    cout << hex << (int)*(p+1) << endl;
    cout << hex << (int)*(p+2) << endl;
    cout << hex << (int)*(p+3) << endl;
    cout << hex << (int)*(p+4) << endl;
    cout << hex << (int)*(p+5) << endl;
    cout << hex << (int)*(p+6) << endl;
    cout << hex << (int)*(p+7) << endl;
    return 0;
}
```

Microsoft
0
0
0
0
0
6
c8
40

上例解读：双精度浮点数1.23e4，在内存中占八个字节，八个字节的值依次为0x40 0xc8 0x06 0x00 0x00 0x00 0x00 0x00(逆向)

转换为64bit则为：0100 0000 1100 1000 0000 0110 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

符号位

11位指数

52位尾数



§. 浮点数机内存储格式(IEEE 754)理解

自学内容：自行以“IEEE754” / “浮点数存储格式” / “浮点数存储原理” / “浮点数存储方式”等关键字，在网上搜索相关文档，读懂并了解浮点数的内部存储机制

学长们推荐的网址：

<https://baike.baidu.com/item/IEEE%20754/3869922?fr=aladdin>

<https://zhuanlan.zhihu.com/p/343033661>

[https://www.bilibili.com/video/BV1iW411d7hd?is_story_h5=false&p=4&share from=ugc&share medium=android&share plat=android&share session id=e12b54be-6ffa-4381-9582-9d5b53c50fb3&share source=QQ&share tag=s i×tamp=1662273598&unique k=AuouME0](https://www.bilibili.com/video/BV1iW411d7hd?is_story_h5=false&p=4&share_from=ugc&share_medium=android&share_plat=android&share_session_id=e12b54be-6ffa-4381-9582-9d5b53c50fb3&share_source=QQ&share_tag=s_i×tamp=1662273598&unique_k=AuouME0)

https://blog.csdn.net/gao_zhennan/article/details/120717424

<https://www.h-schmidt.net/FloatConverter/IEEE754.html>



§. 浮点数机内存储格式(IEEE 754)理解

例: float型数的机内表示

格式要求: 多字节时, 每8bit中间加一个空格或-(例: "11010100 00110001" 或 "11010100-00110001")

例1: 100.25

下面是float机内存储手工转十进制的方法:

(1) 得到的32bit的机内表示是: 0100 0010 1100 1000 1000 0000 0000 0000 (42 c8 80 00)

(2) 其中: 尾数的符号位是 0

指数是 1000 0101 (填32bit中的原始形式)

指数转换为十进制形式是 133 (32bit中的原始形式按二进制原码形式转换)

指数表示的十进制形式是 6 (32bit中的原始形式按IEEE754的规则转换)

1000 0101

- 0111 1111

= 0000 0110 (0x06 = 6)

尾数是 100 1000 1000 0000 0000 0000 (填32bit中的原始形式)

尾数转换为十进制小数形式是 0.56640625 (32bit中的原始形式按二进制原码形式转换)

尾数表示的十进制小数形式是 1.56640625 (加整数部分的1后)

100 1000 1000 0000 0000 0000 = $2^0 + 2^{-1} + 2^{-4} + 2^{-8}$

= 0.5 + 0.0625 + 0.00390625 = 0.56640625 => 加1 => 1.56640625

1.56640625 x 2^6 = 100.25 (此处未体现出误差)

下面是十进制手工转float机内存储的方法:

100 = 0110 0100 (整数部分转二进制为7位, 最前面的0只是为了8位对齐, 可不要)

0.25 = 01 (小数部分转二进制为2位)

100.25 = 0110 0100.01 = 1.1001 0001 x 2^6 (确保整数部分为1, 移6位)

符号位: 0

阶码: 6 + 127 = 133 = 1000 0101

尾数(舍1): 1001 0001 => 1001 0001 0000 0000 0000 000 (补齐23位, 后面补14个蓝色的0)

100 1000 1000 0000 0000 0000 (从低位开始四位一组, 共23位)



§. 浮点数机内存储格式(IEEE 754)理解

例: float型数的机内表示

格式要求: 多字节时, 每8bit中间加一个空格或-(例: "11010100 00110001" 或 "11010100-00110001")

例2: 1.2

下面是float机内存储手工转十进制的方法:

(1) 得到的32bit的机内表示是: 0011 1111 1001 1001 1001 1001 1010 (3f 99 99 9a)

(2) 其中: 尾数的符号位是 0

指数是 0111 1111 (填32bit中的原始形式)

指数转换为十进制形式是 127 (32bit中的原始形式按二进制原码形式转换)

指数表示的十进制形式是 0 (32bit中的原始形式按IEEE754的规则转换)

0111 1111

- 0111 1111

= 0000 0000 (0x0 = 0)

尾数是 001 1001 1001 1001 1001 1010 (填32bit中的原始形式)

尾数转换为十进制小数形式是 0.2000000476837158203125 (32bit中的原始形式按二进制原码形式转换)

尾数表示的十进制小数形式是 1.2000000476837158203125 (加整数部分的1后)

001 1001 1001 1001 1001 1010 = $2^{-3} + 2^{-4} + 2^{-7} + 2^{-8} + 2^{-11} + 2^{-12} + 2^{-15} + 2^{-16} + 2^{-19} + 2^{-20} + 2^{-22}$

= 0.125 + ... + 0.0000002384185791015625 (详见右侧蓝色) = 0.2000000476837158203125

=> 加1 = 1.2000000476837158203125 (此处已体现出误差)

下面是十进制手工转float机内存储的方法:

1 = 1 (整数部分转二进制为1位)

0.2 = 0011 0011 0011 0011 0011 0011 (小数部分无限循环, 转为二进制的24位)

=> 0011 0011 0011 0011 0011 010 (四舍五入为23位, 此处体现出误差)

1.2 = 1.0011 0011 0011 0011 0011 010 = 1.0011 0011 0011 0011 0011 010 x 2^0 (确保整数部分为1, 移0位)

符号位: 0

阶码: 0 + 127 = 127 = 0111 1111

尾数(舍1): 0011 0011 0011 0011 0011 010 (共23位)

001 1001 1001 1001 1001 1010 (从低位开始四位一组, 共23位)

0.125 +
0.0625 +
0.0078125 +
0.00390625 +
0.00048828125 +
0.000244140625 +
0.000030517578125 +
0.0000152587890625 +
0.0000019073486328125 +
0.00000095367431640625 +
0.0000002384185791015625

0.2000000476837158203125