<https://blog.csdn.net/yuhai8702/article/details/5864550>

对内存对齐的理解：

   （struct）内存对齐的规则是：

       1、对于结构的各个成员，第一个成员位于偏移为0的位置，以后每个数据的偏移量必须是Min(编译器被指定的对齐字节数，该数据成员的自身长度)的倍数。

GCC中，Min(系统默认（4）, 数据成员自身长度（char/short/int/double）)。

        2、在数据成员完各自对齐之后，结构体（或联合体）本身也要进行对齐，对齐将按照编译器被指定的对齐字节数和结构体（或联合体）最大数据成员长度中，比较小的那个进行。

GCC中，Min(系统默认（4）, struct成员中最大的数据类型（char/short/int/double）)。

如：

1、

#include <stdio.h>

struct abcd{

    char a;

    char b;

    short c;

    char d;

};

int main()

{

    printf("sizeof(abcd)=%d/n",sizeof(struct abcd));// 运行结果：sizeof(struct abcd)=6

    return 0;

}

      在struct  abcd 中，最大的数据类型是short型。所以在数据成员各自对齐后，整个结构体（或联合体）在进行对齐的时候，按照short型（长度为2）对齐，因为min( 4, short(2) )=2;

再看另一个代码：

2、

/\*2010-09-03\*/

#include <stdio.h>

/\*32位机器访问起始地址是4的倍数处的内存比较方便

 min(系统默认(4),数据成员自身长度)\*/

struct node1{  //sizeof(struct node1)=8;

    char a1;  //1

    short b1; //2

    int c1;   //4

};

struct node2{  //sizeof(struct node2)=12;

    char a2;  //1

    int c2;   //4

    short b2; //2

};

struct id\_all\_char{//

    char id3[3]; //3

    char ver;    //1

    char revision;//1

    char flag;    //1

    char size[4]; //4

};

struct idid{

    char id3[3];

    char ver;

    char revision;

    char flag;

    int size;

}\_\_attribute\_\_((packed));//\_\_attribute\_\_((packed))用于告诉编译器不要对其进行内存对齐！

struct all\_short{//sizeof(struct all\_short)=6;而不是8！因为a,b,c三个都是short，所以a,b,c最大的数据类型是short,所以min(4,2)=2(short),所以最后整个struct按照short型（长度为2）对齐

    short a;//2

    short b;//2

    short c;//2

};

 int main(void)

{

    printf("sizeof(struct node1)=%d/nsizeof(struct node2)=%d/n",sizeof(struct node1),sizeof(struct node2));

    printf("sizeof(struct all\_short)=%d/n",sizeof(struct all\_short));

    printf("sizeof(struct id\_all\_char)=%d/n",sizeof(struct id\_all\_char));

    printf("sizeof(struct idid)=%d/n",sizeof(struct idid));

    return 0;

}

以下部分，来自网上，这些内容给了我很大的帮助，所以也粘进来了：

**二、C++内存对齐规则**

每个特定平台上的编译器都有自己的默认“对齐系数”(也叫对齐模数)。程序员可以通过预编译命令#pragma pack(n)，n=1,2,4,8,16来改变这一系数，其中的n就是你要指定的“对齐系数”。

对齐规则：

1、数据成员对齐规则：结构(struct)(或联合(union))的数据成员，第一个数据成员放在offset为0的地方，以后每个数据成员的对齐按照 #pragma pack指定的数值和这个数据成员自身长度中，比较小的那个进行。

2、结构(或联合)的整体对齐规则：在数据成员完成各自对齐之后，结构(或联合)本身也要进行对齐，对齐将按照#pragma pack指定的数值和结构(或联合)最大数据成员长度中，比较小的那个进行。

3、结合1、2推断：当#pragma pack的n值等于或超过所有数据成员长度的时候，这个n值的大小将不产生任何效果。

4.各成员变量存放的起始地址相对于结构的起始地址的偏移量必须为该变量的类型所占用的字节数的倍数。

5.各成员变量在存放的时候根据在结构中出现的顺序依次申请空间，同时按照上面的对齐方式调整位置，空缺的字节自动填充。

6.同时为了确保结构的大小为结构的字节边界数（即该结构中占用最大空间的类型所占用的字节数）的倍数，所以在为最后一个成员变量申请空间后，还会根据需要自动填充空缺的字节。