<https://blog.csdn.net/u014170207/article/details/53113901>

1、自然对齐：及默认结构体变量成员中最大的长度设置为对齐字节

如：

**[cpp]** [view plain](https://blog.csdn.net/u014170207/article/details/53113901) [copy](https://blog.csdn.net/u014170207/article/details/53113901)

1. **struct** node
2. {
3. **char** a;
4. **int** b;
5. **short** c;
6. };

默认以最大长度int类型4字节对齐。此时占用内存为12byte

2、指定对齐

及

#pragma pack(n)  设置以n字节对齐 超出n字节长度默认以超出字节长度对齐

#pragma pack () 取消指定

如：

**[cpp]** [view plain](https://blog.csdn.net/u014170207/article/details/53113901) [copy](https://blog.csdn.net/u014170207/article/details/53113901)

1. #pragma pack(2)
3. **struct** node1
4. {
5. **char** a;
6. **char** b;
7. **int** c;
8. };
10. #pragma pack()

此时sizeof（struct node1）为6byte;在比如

**[cpp]** [view plain](https://blog.csdn.net/u014170207/article/details/53113901) [copy](https://blog.csdn.net/u014170207/article/details/53113901)

1. **struct** node
2. {
3. **char** a;
4. **short** b;
5. **int** c;
7. **struct** node \*Last;
8. **struct** node \*Next;
9. };

由于默认内存对齐字节为int型四个字节，故：a占用一个字节,偏移量为1，b为short型，2字节，由于a偏移量1不是short型数据2字节的整数倍，故编译器在a后边填充一个空字

节，然后存储b,此时偏移量为4，为int型的整数倍，直接存储int数据c，最后两个由于为指针，所以每一个都为四字节，及一共16字节。此时应该看16字节是否为每一个成员的整

数倍，不足后边补充空字节填充。

 原则一：结构体中元素是按照定义顺序一个一个放到内存中去的，但并不是紧密排列的。从结构体存储的首地址开始，每一个元素放置到内存中时，它都会认为内存是以它自

己的大小来划分的，因此元素放置的位置一定会在自己宽度的整数倍上开始（以结构体变量首地址为0计算）。

原则二：在经过第一原则分析后，检查计算出的存储单元是否为所有元素中最宽的元素的长度的整数倍，是，则结束；若不是，则补齐为它的整数倍。

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/u014170207/article/details/53113901