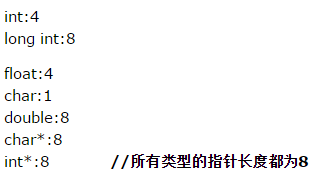
# C语言结构体字节对齐

**一、对齐方式**

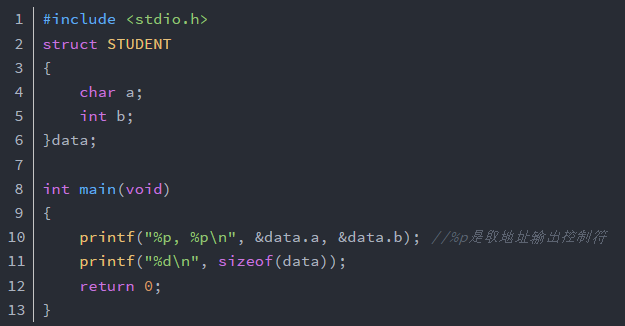
那么到底是怎么存储的呢？就是按字节对齐的方式存储的！**即以结构体成员中占内存最多的数据类型所占的字节数为标准，所有的成员在分配内存时都要与这个长度对齐。**

1. **Linux下各个数据类型的字节大小**



**三、常见例子**

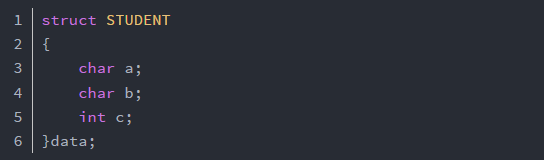
例子1：





总共占 8 字节。

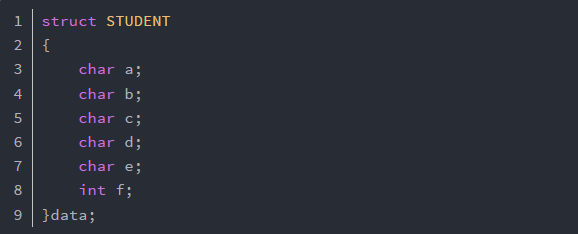
例子2：





总共占 8 字节。

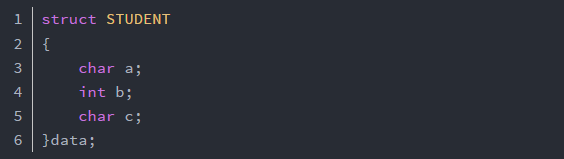
例子3：





总共占 12 字节。

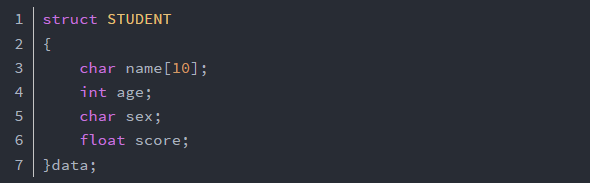
例子4：

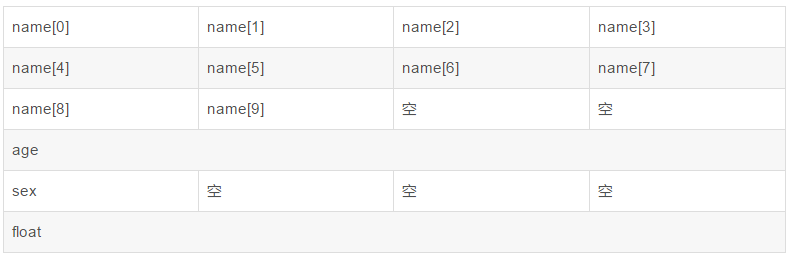




总共占 12 字节。

例子5：（特殊）



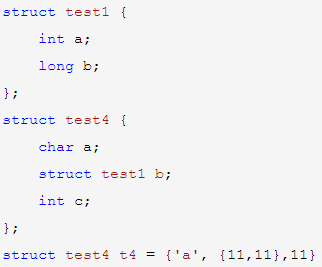


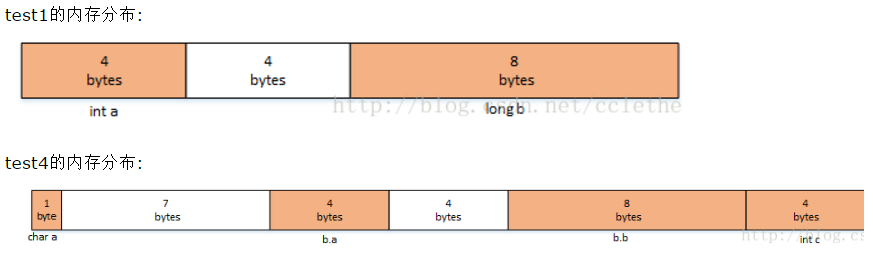
总共占 24 字节。

***特殊的：***

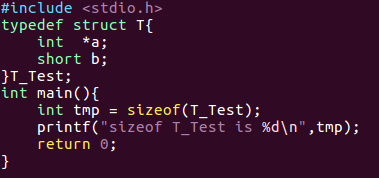
上面这个结构体变量 data 中有成员 char name[10]，长度最长，是 10，那是不是要以 10 对齐？不是，char a[10] 的本质是 10 个 char 变量，所以就把它当成 10 个 char 变量看就行了。**所以结构体变量 data 中成员最长类型占 4 字节，还是以 4 对齐。**

例子6：（内嵌结构体）





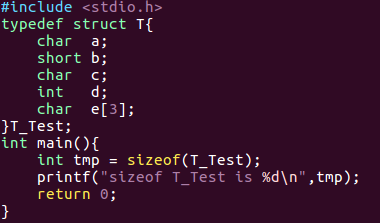
例子7：

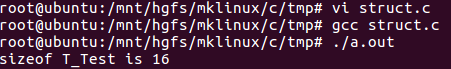


总共占 16 字节。

**Linux中指针为8字节**

例子8：





总共占用16字节。

结构体nums的最大成员int d占8个字节，故实际对齐字节是8字节。

1. char a放在结构体的起始地址;
2. short b占2个字节，2小于实际对齐字节8，故b的起始地址相对于a的起始地址的偏移量须为2的整数倍个字节;
3. int c占4个字节，4小于实际对齐字节8，故c 起始地址相对于a的起始地址的偏移量须为4的整数倍个字节；

所以T\_Test所占空间如下:

|  |  |  |  |
| --- | --- | --- | --- |
| a | 空 | b | |
| c | 空 | 空 | 空 |
| d | | | |
| e[1] | e[1] | e[1] | 空 |

小内存也要符合字节存储规则。

在Linux中2字节数据类型(例如short)的**地址**必须是2的倍数，而较大的数据类型(例如int，int \*，float和double)的**地址**必须是4的倍数。