<https://blog.csdn.net/lyh__521/article/details/45958847>

1.先看64位下：

#include<stdio.h>

struct A

{

int a;

char b;

double c;

char d;

};

struct B

{

char a;

double b;

char c;

};

int main()

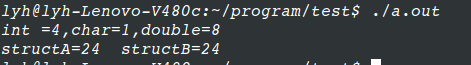
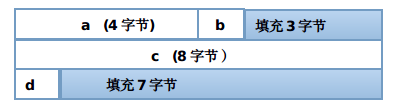
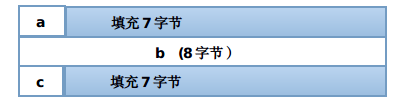
{

printf("int =%lu,char=%lu,double=%lu\n",sizeof(int),sizeof(char),sizeof(double));

printf("structA=%lu structB=%lu\n",sizeof(struct A),sizeof(struct B));

return 0;

}

输出结果：   
   
structA: 4+(1+3)+8+(1+7) = 24   
structB: (1+7)+8+(1+7) = 24   
计算结果与输出是一样的。   
这两个结构体在内存中存储应该是下面这样的：   
struct A： 整体按照8字节（double长度）对齐   
   
struct B :   


2.在32位下编译，gcc 加参数 -m32

#include<stdio.h>

struct A

{

int a;

char b;

double c;

char d;

};

struct B

{

char a;

double b;

char c;

};

int main()

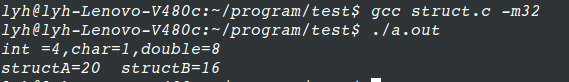
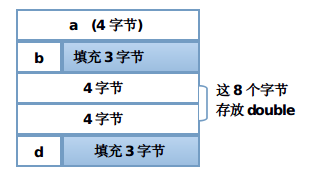
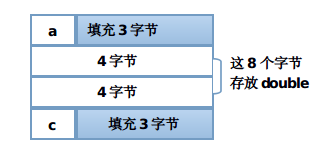
{

printf("int =%u,char=%u,double=%u\n",sizeof(int),sizeof(char),sizeof(double));

printf("structA=%u structB=%u\n",sizeof(struct A),sizeof(struct B));

return 0;

}

输出：   
   
结果和64位下完全不一样，很显然它没有按照最长成员double的8字节对齐。稍微想一下就明白了，因为32位只有4个字节，最长对齐模数只能按4个字节来对齐，double 是分成了2个4字节。上面两个结构体在内存中应该是这种形式。   
struct A：整体按照4字节对齐   
   
4+(1+3)+8+(1+3) = 20   
struct B :   
   
(1+3)+8+(1+3) = 16

内存空间实际上是连续的，上面分块的画法只是为了方便理解。

3.关于内存对齐的介绍   
<http://blog.csdn.net/csw_100/article/details/5495309>

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/lyh\_\_521/article/details/45958847