

第10章 结构体和共用体

——结构体所占内存的字节数

结构体所占内存的字节数

- 结构体类型占用内存字节数 = ?
- 是所有成员占内存的总和吗?

- 用 `sizeof` 获得结构体大小
`sizeof(变量或表达式)`
`sizeof(类型)`

```
typedef struct sample
{
    char m1;
    int m2;
    char m3;
} SAMPLE;
```

```
int main()
{
    SAMPLE s = {'a', 2, 'b'};
    printf("%d\n", sizeof(s));
    return 0;
}
```

```
printf("%d\n", sizeof(struct sample));
```

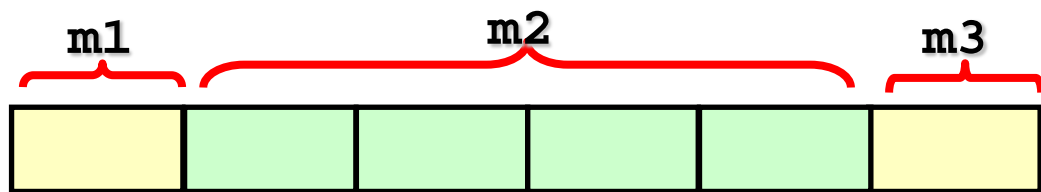
```
printf("%d\n", sizeof(SAMPLE));
```

12

Why?



结构体所占内存的字节数

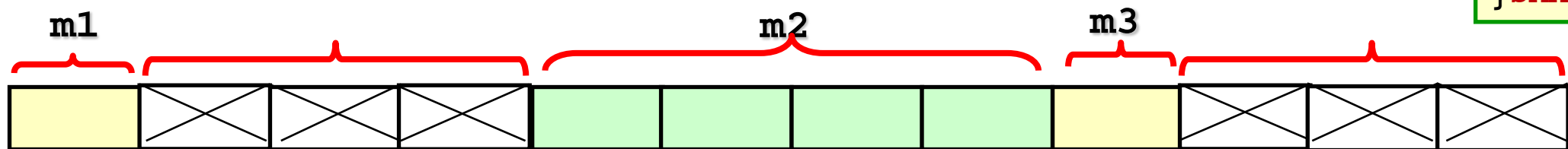


非所有成员占内存字节数的总和

```
printf("%d\n", sizeof(SAMPLE));
```

12

```
typedef struct
{
    char m1;
    int  m2;
    char m3;
} SAMPLE;
```



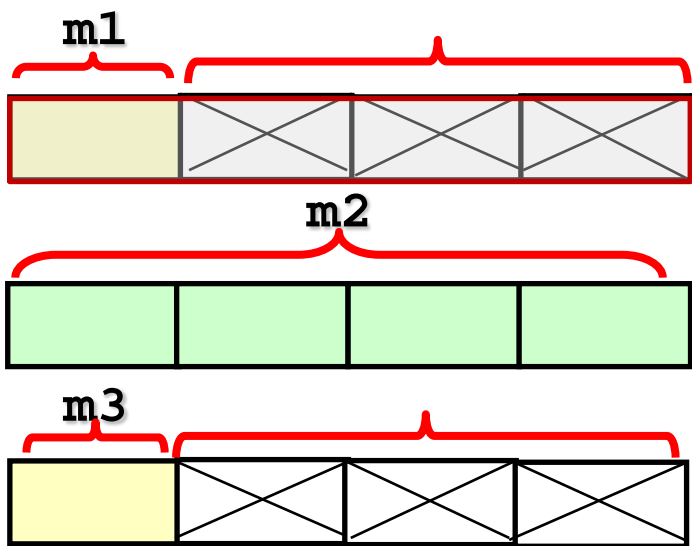
■ 内存对齐（Memory-alignment）

- 对于大多数计算机，数据项要求从某个数量字节的倍数开始存放
- `short`型数据从偶数地址开始存放，而`int`型数据则被对齐在4字节地址边界
- 为了满足内存地址对齐的要求，需要在较小的成员后加入补位

结构体所占内存的字节数

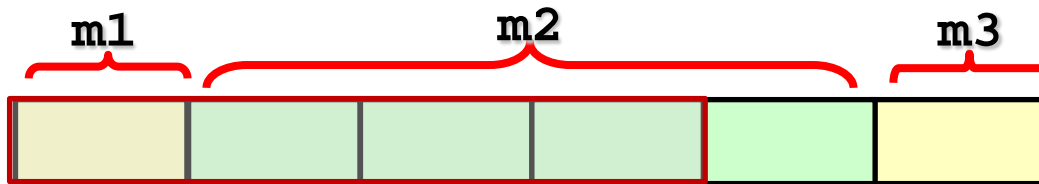
- 问题：为什么要求内存地址对齐呢？

- 提高内存寻址效率



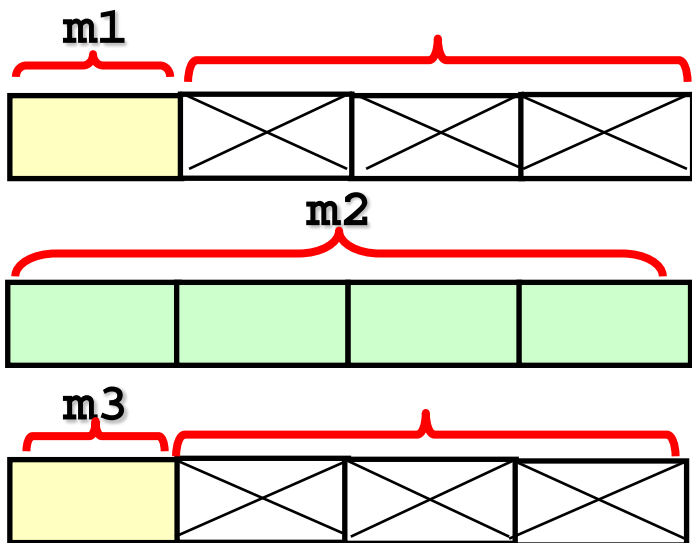
```
typedef struct  
{  
    char m1;  
    int  m2;  
    char m3;  
} SAMPLE;
```

- 32位体系结构中，int值被对齐在4字节地址边界
- 读写一个4字节int型数据，只需一次内存访问操作

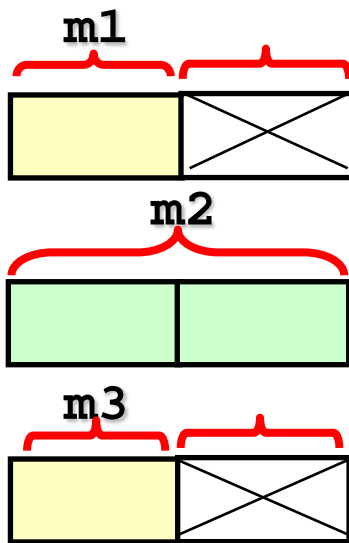


结构体所占内存的字节数

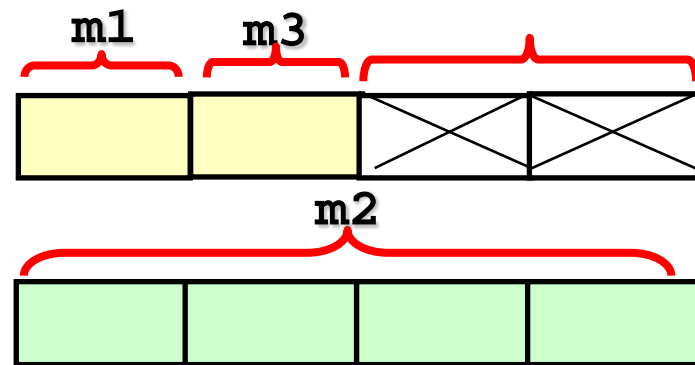
```
typedef struct sample
{
    char m1;
    int m2;
    char m3;
}SAMPLE;
```



```
typedef struct sample
{
    char m1;
    short m2;
    char m3;
}SAMPLE;
```



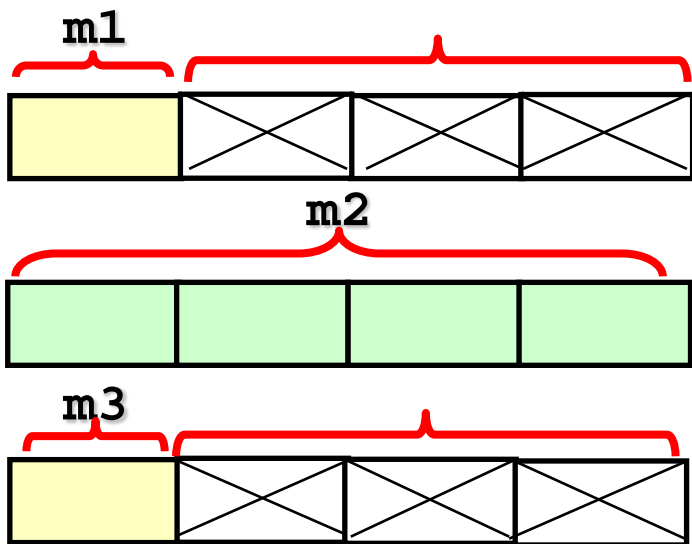
```
typedef struct sample
{
    char m1;
    char m3;
    int m2;
}SAMPLE;
```



- 结构体在内存中所占的字节数与所定义的结构体类型有关

结构体所占内存的字节数

```
typedef struct sample
{
    char m1;
    int  m2;
    char m3;
}SAMPLE;
```



- 结构体在内存中所占的字节数不仅与所定义的结构体类型有关，还与计算机系统本身有关
- 不同的系统和编译器，内存对齐方式可能会不同，是机器相关的
 - 在DOS下的Turbo C2.0中运行结果为：4
- 计算结构体所占内存的字节数时，一定要使用sizeof运算符

```
printf("%d\n", sizeof(SAMPLE));
```

讨论

- 是否可能会在结构体的开始处有补位？猜猜有可能是什么原因？
- 为什么C语言不支持使用==来判定两个结构体是否相等？这个和本节介绍的什么内容有可能有关？

