内存对齐

内存对齐解释：

对齐规则是按照成员的声明顺序，依次安排内存，其偏移量为成员大小的整数倍，0看做任何成员的整数倍，最后结构体的大小为最大成员的整数倍

为什么要内存对齐?

1.平台原因(移植原因)：不是所有的硬件平台都能访问任意地址上的任意数据的；某些硬件平台只能在某些地址处取某些特定类型的数据，否则抛出硬件异常。

2.性能原因：数据结构(尤其是栈)应该尽可能地在自然边界上对齐。原因在于，为了访问未对齐的内存，处理器需要作两次内存访问；而对齐的内存访问仅需要一次访问。

解释二

原因有这么几点：

1、CPU 对内存的读取不是连续的，而是分成块读取的，块的大小只能是1、2、4、8、16 ... 字节；

2、当读取操作的数据未对齐，则需要两次总线周期来访问内存，因此性能会大打折扣；

3、某些硬件平台只能从规定的相对地址处读取特定类型的数据，否则会产生异常。

1.1、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(偏移)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

1.2、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。

1.3、结构体作为成员：

如果一个结构里有某些结构体成员，则结构体成员要从其内部最大元素大小的整数倍地址开始存储。

#pragma pack(n) 对齐系数

代码例子

#include

//#pragma pack(1)

using std::cout;

using std::endl;

struct xx { int a1; char b1; short c1; char d1; } MyStructxx;//12

```
struct x{ char a; int b; short c;  char d; }MyStruct1;//12
struct y{ int b; char a; char d; short c;}MyStruct2;//8
struct EE{  int a;  char b; short c;
  struct FF
  {  int a1; char b1;  short c1;  char d1;
  }MyStructFF;
  char d;
}MyStructEE;//24

struct DD
{
  int a; char b;  short c; int d;
struct FF
  { double a1;  char b1; short c1; char d1;
  }MyStructFF;
  char e;
}MyStructDD;//40

struct GG
{ char e[2];  short h;
  struct A
  {  int a;   double b;    float c;
  }MyStructA;
}MyStructGG;//32

struct SS
{  int a;//[0-3]
  char b;//[4]
  short c;//[6-7]
  int d;//12[8-11]
  struct FF
  {
    int a1;//16[12-15]
    char b1;//20[16]
```

```
    short c1;//20[18-19]
    char d1;//24[20]//21-23空出来,到这里21字节，不是struct里面最大的4的整数倍
  }MyStructFF;

#if 1
  char e;//[24],要4的整数倍,所以25-27为空
  //int e;
  //double ww;
#endif
}MyStructSS;//28

int main(int argc, char **argv)
{
  cout <<"sizeof(MyStructxx) = " << sizeof(MyStructxx) << endl;
  cout <<"sizeof(MyStruct1) = " << sizeof(MyStruct1) << endl;
  cout <<"sizeof(MyStruct2) = " << sizeof(MyStruct2) << endl;
  cout <<"sizeof(MyStructEE) = " << sizeof(MyStructEE) << endl;
  cout <<"sizeof(MyStructDD) = " << sizeof(MyStructDD) << endl;
  //cout <<"sizeof(GG) = " << sizeof(GG) << endl;
  cout <<"sizeof(MyStructGG) = " << sizeof(MyStructGG) << endl;
  cout <<"sizeof(MyStructSS) = " << sizeof(MyStructSS) << endl;
  return 0;
}
```