

蜗牛小居

我要一步一步往上爬

C++博客 :: 首页 :: 新随笔 :: 联系 :: 聚合  :: 管理 posts - 1, comments - 18, trackbacks - 0, articles - 0

日历

< 2009年3月 >						
日	一	二	三	四	五	六
22	23	24	25	26	27	28
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	<u>16</u>	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4


常用链接

我的随笔
我的评论
我参与的随笔

留言簿(1)

给我留言
查看公开留言
查看私人留言

随笔分类

 C++(1)

随笔档案

2009年3月 (1)

相册

photo

搜索

最新评论

内存对齐的规则以及作用

Posted on 2009-03-16 09:36 蜗牛先生 阅读(31647) 评论(19) 编辑 收藏 引用 所属分类: C++ .

首先由一个程序引入话题:

```
1 //环境: vc6 + windows sp2
2 //程序1
3 #include <iostream>
4
5 using namespace std;
6
7 struct st1
8 {
9     char a ;
10    int b ;
11    short c ;
12 };
13
14 struct st2
15 {
16     short c ;
17     char a ;
18     int b ;
19 };
20
21 int main()
22 {
23     cout<<"sizeof(st1) is "<<sizeof(st1)<<endl;
24     cout<<"sizeof(st2) is "<<sizeof(st2)<<endl;
25     return 0 ;
26 }
27
```

程序的输出结果为:

```
sizeof(st1) is 12
sizeof(st2) is 8
```

问题出来了, 这两个一样的结构体, 为什么sizeof的时候大小不一样呢?

本文的主要目的就是解释明白这一问题。

内存对齐, 正是因为内存对齐的影响, 导致结果不同。

对于大多数的程序员来说, 内存对齐基本上是透明的, 这是编译器该干的活, 编译器为程序中的每个数据单元安排在合适的位置上, 从而导致了相同的变量, 不同声明顺序的结构体大小的不同。

1. re: 内存对齐的规则以及作用

非常感谢，看了这篇文章终于弄懂了，感谢

--素存

2. re: 内存对齐的规则以及作用

评论内容较长,点击标题查看

--芜蘅

3. re: 内存对齐的规则以及作用

清晰明了，感谢。

--ano

4. re: 内存对齐的规则以及作用

多谢，受益了。

--碧海一波

5. re: 内存对齐的规则以及作用

评论内容较长,点击标题查看

--wow

那么编译器为什么要进行内存对齐呢？程序1中结构体按常理来理解sizeof(st1)和sizeof(st2)结果都应该是7， $4(\text{int}) + 2(\text{short}) + 1(\text{char}) = 7$ 。经过内存对齐后，结构体的空间反而增大了。

在解释内存对齐的作用前，先来看下内存对齐的规则：

- 1、对于结构的各个成员，第一个成员位于偏移为0的位置，以后每个数据成员的偏移量必须是`min(#pragma pack()`指定的数，这个数据成员的自身长度)的倍数。
- 2、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。

`#pragma pack(n)` 表示设置为n字节对齐。VC6默认8字节对齐

以程序1为例解释对齐的规则：

St1：char 占一个字节，起始偏移为0，int 占4个字节， $\text{min}(\text{\#pragma pack}()$ 指定的数，这个数据成员的自身长度) = 4（VC6默认8字节对齐），所以int按4字节对齐，起始偏移必须为4的倍数，所以起始偏移为4，在char后编译器会添加3个字节的额外字节，不存放任意数据。short占2个字节，按2字节对齐，起始偏移为8，正好是2的倍数，无须添加额外字节。到此规则1的数据成员对齐结束，此时的内存状态为：

Oxxx		oooo		oo						
0	1	2	3	4	5	6	7	8	9	(地址)

(x表示额外添加的字节)

共占10个字节。还要继续进行结构本身的对齐，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，st1结构中最大数据成员长度为int，占4字节，而默认的`#pragma pack`指定的值为8，所以结果本身按照4字节对齐，结构总大小必须为4的倍数，需添加2个额外字节使结构的总大小为12。此时的内存状态为：

0xxx		oooo		ooxx								
0	1	2	3	4	5	6	7	8	9	a	b	(地址)

到此内存对齐结束。St1占用了12个字节而非7个字节。

St2 的对齐方法和st1相同，读者可自己完成。

内存对齐的主要作用是：

- 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：经过内存对齐后，CPU的内存访问速度大大提升。具体原因稍后解释。

图一：



这是普通程序员心目中的内存印象，由一个个的字节组成，而CPU并不是这么看待的。

图二：



CPU把内存当成是一块一块的，块的大小可以是2，4，8，16字节大小，因此CPU在读取内存时是一块一块进行读取的。块大小成为*memory access granularity*（粒度）本人把它翻译为“内存读取粒度”。

假设CPU要读取一个int型4字节大小的数据到寄存器中，分两种情况讨论：

- 1、数据从0字节开始
- 2、数据从1字节开始

再次假设内存读取粒度为4。

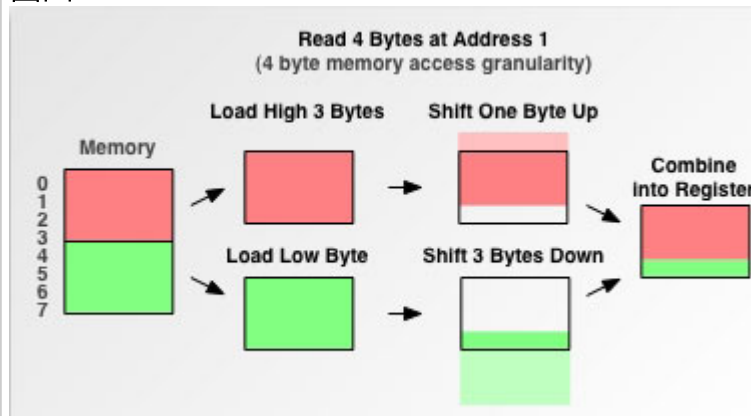
图三：



当该数据是从0字节开始时，很CPU只需读取内存一次即可把这4字节的数据完全读取到寄存器中。

当该数据是从1字节开始时，问题变的有些复杂，此时该int型数据不是位于内存读取边界上，这就是一类内存未对齐的数据。

图四：



此时CPU先访问一次内存，读取0—3字节的数据进寄存器，并再次读取4—5字节的数据进寄存器，接着把0字节和6，7，8字节的数据剔除，最后合并1，2，3，4字节的数据进寄存器。对一个内存未对齐的数据进行了这么多额外的操作，大大降低了CPU性能。

这还属于乐观情况了，上文提到内存对齐的作用之一为平台的移植原因，因为以上操作只有有部分CPU肯干，其他一部分CPU遇到未对齐边界就直接罢工了。

图片来自：[Data alignment: Straighten up and fly right](#)

如大家对内存对齐对性能的具体影响情况，可以参考上文。

Feedback

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)
2009-11-09 22:32 by fynn

(◕v◕)嗯，终于有点开窍了，呵呵

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)
2010-04-21 17:26 by jamshulo

恩，很谢谢您的写作，明白了不少东西。

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)
2010-06-18 22:38 by yond

谢谢,最近在看这个，从这里才看明白了

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)
2010-10-20 18:17 by 谢谢

看了好几遍，总算看懂了，谢谢分享。

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)
2010-10-26 16:34 by don

非常感谢！

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)
2011-07-19 11:34 by 敬相少爷

应该是剔除0字节和5，6，7字节，不是6，7，8字节

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)
2011-09-16 10:52 by 游客

谢谢楼主分享啊，说的这么仔细！

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)
2011-10-03 13:03 by 杨捷

谢谢 终于明白了？

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)
2012-04-06 20:32 by linrulei

非常感谢啊，终于看懂了，下午看了头都大了，也没明白为什么.....

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)

2012-05-28 15:42 by caicai

St2为什么不说不下呢，如果按你上面提到的原则应该是 00 0 0000 X (X为补全字节)，而实际中我测试的是 00 00 0000. 能解释一下么，3ks

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)

2012-05-29 09:46 by caicai

@caicai

2了

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)

2012-07-22 17:01 by dirver

楼主说的很详细，看完之后终于理解了。

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)

2012-08-02 18:33 by tww

因为int是4字节，所以起始地址要为4的倍数，所以char之后要补X...实际应该为00 0x 0000@caicai

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)

2012-11-20 22:26 by wow

假设CPU要读取一个int型4字节大小的数据到寄存器中，分两种情况讨论：

1、数据从0字节开始

2、数据从1字节开始

看了博主的假设内存颗粒为4的时候，读取内存时指针应是4的倍数啊，所以应该有数据从2字节，3字节开始的可能性，对从2字节开始的数据，首先指针指向0地址读取4字节到数据寄存器a，然后指向4地址读取4字节到数据寄存器b,最后剔除a中的0,1字节，b中的后2个字节，最后数据合并到数据寄存器c,这个思路是对的不，博主？

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)

2013-06-03 23:35 by 碧海一波

多谢，受益了。

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)

2014-02-15 21:45 by ano

清晰明了，感谢。

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)

2016-06-07 17:57 by 茈蘅

@caicai

St2 应该是 00 0x 0000。前两个0是short, char 1字节对齐, 无需补, int 4字节对齐, 前面只有三个0, 因此补一位x, 整体是4字节的倍数, 因此就是00 0x 0000

re: 内存对齐的规则以及作用 [回复](#) [更多评论](#)

2016-08-17 20:35 by 素存

非常感谢, 看了这篇文章终于弄懂了, 感谢

[刷新评论列表](#)

只有注册用户登录后才能发表评论。

【推荐】 [超50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库](#)

网站导航: [博客园](#) [IT新闻](#) [BlogJava](#) [知识库](#) [博问](#) [管理](#)

Powered by:

[C++博客](#)

Copyright © 蜗牛先生