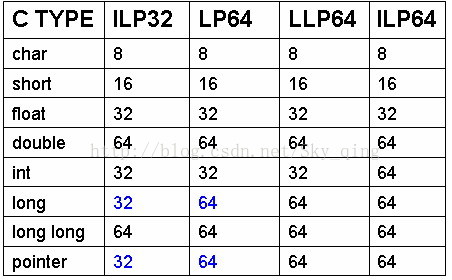
**32位与64位下各类型长度对比**

2013年09月13日 16:28:52 [Joe\_vv](https://me.csdn.net/Sky_qing) 阅读数：129810

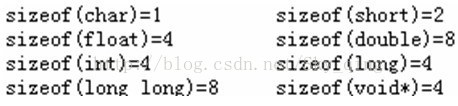
64 位的优点：64 位的应用程序可以直接访问 4EB 的内存和文件大小最大达到4 EB（2 的 63 次幂）；可以访问大型数据库。本文介绍的是64位下C语言开发程序注意事项。

**1. 32 位和 64 位C数据类型**

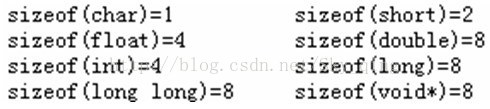
32和64位C语言内置数据类型，如下表所示：



上表中第一行的大写字母和数字含义如下所示：  
I表示：int类型  
L表示：long类型  
P表示：pointer指针类型  
32表示：32位系统  
64表示：64位系统  
如：LP64表示，在64位系统下的long类型和pointer类型长度为64位。  
64位Linux 使用了 LP64 标准，即：long类型和pointer类型长度为64位，其他类型的长度和32位系统下相同类型的长度相同，32位和64位下类型的长度比较见上图的蓝色部分。  
下图为在32和64位linux系统下使用sizeof检测出的数据类型的长度。  
32位平台下结果：



64位平台下结果：



**2. 64系统下开发注意事项：**

**2.1 格式化字符串：long使用%ld，指针使用%p，例如：**

1. char \*ptr = &something;
2. printf (%x\n", ptr);

上面的代码在 64 位系统上不正确，只显示低 4 字节的内容。正确的方法是：使用 %p，如下：

1. char \*ptr = &something;
2. printf (%p\n", ptr);

**2.2 数字常量：常量要加L**  
例1，常数 0xFFFFFFFF 是一个有符号的 long 类型。在 32 位系统上，这会将所有位都置位（每位全为 1），但是在 64 位系统上，只有低 32 位被置位了，结果是这个值是 0x00000000FFFFFFFF。  
例2，在下面的代码中，a 的最大值可以是 31。这是因为 1 << a 是 int 类型的。

long l = 1 << a;

要在 64 位系统上进行位移，应使用 1L，如下所示：

long l = 1L << a;

**2.3 符号扩展：避免有符号数与无符号数运算，例如：**

1. int i = -2;
2. unsigned int j = 1;
3. long l = i + j;
4. printf("Answer: %ld\n",l);

32位下是-1，在64位下是4294967295。原因在于表达式(i+j)是一个unsigned int表达式，但把它赋值给k时，符号位没有被扩展。要解决这个问题，两端的操作数只要均为signed或均为unsigned就可。  
  
**2.4 转换截断：**  
转换截断发生在把long转换成int时，如下例：

int length = (int) strlen(str);

strlen返回size\_t（它在LP64中是unsigned long），当赋值给一个int时，截断是必然发生的。而通常，截断只会在str的长度大于2GB时才会发生，这种情况在程序中一般不会出现。虽然如此，也应该尽量使用适当的多态类型（如size\_t、uintptr\_t等等）。

**2.5 赋值：**  
不要交换使用 int 和 long 类型，例如：

1. int i;
2. time\_t l;
3. i = l;

不要使用 int 类型来存储指针，例如：

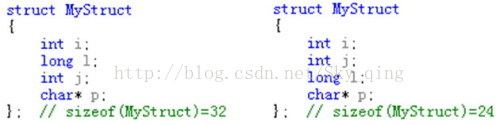
1. unsigned int i, \*ptr;
2. i = (unsigned) ptr;

不要使用指针来存放 int 类型的值。例如：

1. int \*ptr;
2. int i;
3. ptr = (int \*) i;

**2.6 移植倒64位环境下的性能：**

移植到64位平台后，性能实际上降低了。原因是64位中的指针长度和数据大小有关，并由此引发的缓存命中率降低、数据对齐等问题。通过改变结构中数据排列的先后顺序，会因为少了填充数据，存储空间也随之减少。如：



**2.7 程序中链接到的库要使用64位的库。**

**由上可见所有的问题都是由long和指针长度改变引起，在开发过程中只有牢记long和指针类型的长度。**

**转自：**[**http://blog.sina.com.cn/s/blog\_57421ff80100c7g1.html**](http://blog.sina.com.cn/s/blog_57421ff80100c7g1.html)