<https://www.cnblogs.com/1zhk/articles/4971463.html>

[**32位和64位系统区别及字节数**](https://www.cnblogs.com/1zhk/articles/4971463.html)

一）64位系统和32位有什么区别？

1、64bit CPU拥有更大的寻址能力，最大支持到4T内存，而32bit只支持4G内存

2、64位CPU一次可提取64位数据，比32位提高了一倍，理论上性能会提升1倍。但这是建立在64bit操作系统，64bit软件的基础上的。

什么是64位处理器？

之所以叫做“64位处理器”，是因为电脑内部都是实行2进制运算，处理器（CPU）一次处理数据的能力也是2的倍数。8位处理器、16位处理器、32位处理器和64位处理器，其计数都是2的倍数。一次处理的数据越大，该电脑处理信息的能力越来越大；因此64位处理在先天就比32位处理器具有快速的能力。那为什么不用更高级的128位处理器呢？因为位数越高，处理器芯片的设计也就越复杂，目前的技术水平暂时无法制造这么复杂的芯片。

64位处理器之失

※硬件———缺乏驱动程序，很多现有硬件无法使用

※软件———操作系统不是问题，但是软件出现不兼容难题

64位处理器之得

※硬件———更快的执行速度，更大的内存管理

※软件———最新的尖端软件首先出现在64位平台

（二）数据类型对应字节数

程序运行平台  
      不同的平台上对不同数据类型分配的字节数是不同的。  
      个人对平台的理解是CPU+OS+Compiler，是因为：   
      1、64位机器也可以装32位系统（x64装XP）；   
      2、32位机器上可以有16/32位的编译器（XP上有tc是16位的，其他常见的是32位的）；   
      3、即使是32位的编译器也可以弄出64位的integer来（int64）。   
      以上这些是基于常见的wintel平台，加上我们可能很少机会接触的其它平台（其它的CPU和OS），所以个人认为所谓平台的概念是三者的组合。   
      虽然三者的长度可以不一样，但显然相互配合（即长度相等，32位的CPU+32位的OS+32位的Compiler）发挥的能量最大。   
      理论上来讲 我觉得数据类型的字节数应该是由CPU决定的，但是实际上主要由编译器决定(占多少位由编译器在编译期间说了算)。

常用数据类型对应字节数  
  可用如sizeof（char),sizeof(char\*)等得出

 32位编译器：

      char ：1个字节  
      char\*（即指针变量）: 4个字节（32位的寻址空间是2^32, 即32个bit，也就是4个字节。同理64位编译器）  
      short int : 2个字节  
      int：  4个字节  
      unsigned int : 4个字节  
      float:  4个字节  
      double:   8个字节  
      long:   4个字节  
      long long:  8个字节  
      unsigned long:  4个字节

  64位编译器：

      char ：1个字节  
      char\*(即指针变量): 8个字节  
      short int : 2个字节  
      int：  4个字节  
      unsigned int : 4个字节  
      float:  4个字节  
      double:   8个字节  
      long:   8个字节  
      long long:  8个字节  
      unsigned long:  8个字节

16位操作系统中，int 占16位；在32位操作系统中，int 占32位。但是现在人们已经习惯了 int 占32位，因此在[64位操作系统](https://www.baidu.com/s?wd=64%E4%BD%8D%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9PyRzrj6zuADsrAcduHnk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1nzPHmznHmkr0" \t "_blank)中，int 仍为32位。64位整型用 long long 或者 \_\_int64