型分配的字节数。(我们在 2.2 节讨论 C 标准保证的字节数和典型的字节数之间的关系。)有些数据类型的确切字节数依赖于程序是如何被编译的。我们给出的是 32 位和 64 位程序的典型值。整数或者为*有符号的*，即可以表示负数、零和正数；或者为*无符号的*，即只能表示非负数。C 的数据类型 char 表示一个单独的字节。尽管“char”是由于它被用来存储文本串中的单个字符这一事实而得名，但它也能被用来存储整数值。数据类型 short、int 和 long 可以提供各种数据大小。即使是为 64 位系统编译，数据类型 int 通常也只有 4 个字节。数据类型 long 一般在 32 位程序中为 4 字节，在 64 位程序中则为 8 字节。

为了避免由于依赖“典型”大小和不同编译器设置带来的奇怪行为，ISO C99 引入了一类数据类型，其数据大小是固定的，不随编译器和机器设置而变化。其中就有数据类型 int32_t 和 int64_t，它们分别为 4 个字节和 8 个字节。使用确定大小的整数类型是程序员准确控制数据表示的最佳途径。

大部分数据类型都编码为有符号数值，除非有前缀关键字 unsigned 或对确定大小的数据类型使用了特定的无符号声明。数据类型 char 是一个例外。尽管大多数编译器和机器将它们视为有符号数，但 C 标准不保证这一点。相反，正如方括号指示的那样，程序员应该用有符号字符的声明来保证其为一个字节的有符号数值。不过，在很多情况下，程序行为对数据类型 char 是有符号的还是无符号的并不敏感。

对关键字的顺序以及包括还是省略可选关键字来说，C 语言允许存在多种形式。比如，下面所有的声明都是一个意思：

```
unsigned long
unsigned long int
long unsigned
long unsigned int
```

我们将始终使用图 2-3 给出的格式。

图 2-3 还展示了指针(例如一个被声明为类型为“char *”的变量)使用程序的全字长。大多数机器还支持两种不同的浮点数格式：单精度(在 C 中声明为 float)和双精度(在 C 中声明为 double)。这些格式分别使用 4 字节和 8 字节。

给 C 语言初学者　声明指针

对于任何数据类型 *T*，声明

```
T *p;
```

表明 p 是一个指针变量，指向一个类型为 *T* 的对象。例如，

```
char *p;
```

就将一个指针声明为指向一个 char 类型的对象。

程序员应该力图使他们的程序在不同的机器和编译器上可移植。可移植性的一个方面就是使程序对不同数据类型的确切大小不敏感。C 语言标准对不同数据类型的数字范围设置了下界(这点在后面还将讲到)，但是却没有上界。因为从 1980 年左右到 2010 年左右，32 位机器和 32 位程序是主流的组合，许多程序的编写都假设为图 2-3 中 32 位程序的字节分配。随着 64 位机器的日益普及，在将这些程序移植到新机器上时，许多隐藏的对字长的依赖性就会显现出来，成为错误。比如，许多程序员假设一个声明为 int 类型的程序对象能被用来存储一个指针。这在大多数 32 位的机器上能正常工作，但是在一台 64 位的机器上却会导致问题。