年，美国国家标准学会下的一个工作组推出了 ANSI C 标准，对最初的贝尔实验室的 C 语言做了重大修改。ANSI C 与贝尔实验室的 C 有了很大的不同，尤其是函数声明的方式。Brian Kernighan 和 Dennis Ritchie 在著作的第 2 版[61]中描述了 ANSI C，这本书至今仍被公认为关于 C 语言最好的参考手册之一。

国际标准化组织接替了对 C 语言进行标准化的任务，在 1990 年推出了一个几乎和 ANSI C 一样的版本，称为“ISO C90”。该组织在 1999 年又对 C 语言做了更新，推出“ISO C99”。在这一版本中，引入了一些新的数据类型，对使用不符合英语语言字符的文本字符串提供了支持。更新的版本 2011 年得到批准，称为“ISO C11”，其中再次添加了更多的数据类型和特性。最近增加的大多数内容都可以向后兼容，这意味着根据早期标准(至少可以回溯到 ISO C90)编写的程序按新标准编译时会有同样的行为。

GNU 编译器套装(GNU Compiler Collection，GCC)可以基于不同的命令行选项，依照多个不同版本的 C 语言规则来编译程序，如图 2-1 所示。比如，根据 ISO C11 来编译程序 prog.c，我们就使用命令行：

```
linux> gcc -std=c11 prog.c
```

C 版本	GCC命令行选项
GNU 89	无，-std=gnu89
ANSI，ISO C90	-ansi，-std=c89
ISO C99	-std=c99
ISO C11	-std=c11

图 2-1　向 GCC 指定不同的 C 语言版本

编译选项-ansi 和-std=c89 的用法是一样的——会根据 ANSI 或者 ISO C90 标准来编译程序。(C90 有时也称为“C89”，这是因为它的标准化工作是从 1989 年开始的。)编译选项-std=c99 会让编译器按照 ISO C99 的规则进行编译。

本书中，没有指定任何编译选项时，程序会按照基于 ISO C90 的 C 语言版本进行编译，但是也包括一些 C99、C11 的特性，一些 C++ 的特性，还有一些是与 GCC 相关的特性。GNU 项目正在开发一个结合了 ISO C11 和其他一些特性的版本，可以通过命令行选项-std=gnu11 来指定。(目前，这个实现还未完成。)今后，这个版本会成为默认的版本。

2.1　信息存储

大多数计算机使用 8 位的块，或者字节(byte)，作为最小的可寻址的内存单位，而不是访问内存中单独的位。机器级程序将内存视为一个非常大的字节数组，称为虚拟内存(virtual memory)。内存的每个字节都由一个唯一的数字来标识，称为它的地址(address)，所有可能地址的集合就称为虚拟地址空间(virtual address space)。顾名思义，这个虚拟地址空间只是一个展现给机器级程序的概念性映像。实际的实现(见第 9 章)是将动态随机访问存储器(DRAM)、闪存、磁盘存储器、特殊硬件和操作系统软件结合起来，为程序提供一个看上去统一的字节数组。

在接下来的几章中，我们将讲述编译器和运行时系统是如何将存储器空间划分为更可管理的单元，来存放不同的程序对象(program object)，即程序数据、指令和控制信息。可以用各种机制来分配和管理程序不同部分的存储。这种管理完全是在虚拟地址空间里完成的。例如，C 语言中一个指针的值(无论它指向一个整数、一个结构或是某个其他程序对象)都是某个存储块的第一个字节的虚拟地址。C 编译器还把每个指针和类型信息联系起来，这样就可以根据指针值的类型，生成不同的机器级代码来访问存储在指针所指向位置处的值。尽管 C 编译器维护着这个类型信息，但是它生成的实际机器级程序并不包含关于数据类型的信息。每个程序对象可以简单地视为一个字节块，而程序本身就是一个字节序列。