(86 条消息) 2、进入缓冲区（缓存）的世界，破解一切与输入输出有关的疑难杂症（转载自 C 语言中文网，仅作学习笔记）_wyf0312 的博客-CSDN 博客

缓冲区（Buffer）又称为缓存（Cache），是内存空间的一部分。也就是说，计算机在内存中预留了一定的存储空间，用来暂时保存输入或输出的数据，这部分预留的空间就叫做缓冲区（缓存）。

有时候，从键盘输入的内容，或者将要输出到显示器上的内容，会暂时进入缓冲区，待时机成熟，再一股脑将缓冲区中的所有内容“倒出”，我们才能看到变量的值被刷新，或者屏幕产生变化。

有时候，用户希望得到最及时的反馈，输入输出的内容就不能进入缓冲区。缓冲区是输入输出的“名门”所在，只有学习缓冲区，才能解开前几节遇到的问题。缓冲区是治愈与输入输出有关的大部分疑难杂症的良药，它能使你对输入输出的认识上升到一个更高的层次。

为什么要引入缓冲区（缓存）

缓冲区是为了让低速的输入输出设备和高速的用户程序能够协调工作，并降低输入输出设备的读写次数。

> 用户程序的执行速度可以看做 CPU 的运行速度，如果没有各种硬件的阻碍，理论上它们是同步的。例如，我们都知道硬盘的速度要远低于 CPU，它们之间有好几个数量级的差距，当向硬盘写入数据时，程序需要等待，不能做任何事情，就好像卡顿了一样，用户体验非常差。计算机上绝大多数应用程序都需要和硬件打交道，例如读写硬盘、向显示器输出、从键盘输入等，如果每个程序都等待硬件，那么整台计算机也将变得卡顿。

但是有了缓冲区，就可以将数据先放入缓冲区中（内存的读写速度也远高于硬盘），然后程序可以继续往下执行，等所有的数据都准备好了，再将缓冲区中的所有数据一次性地写入硬盘，这样程序就减少了等待的次数，变得流畅起来。

缓冲区的另外一个好处是可以减少硬件设备的读写次数。其实我们的程序并不能直接读写硬件，它必须告诉操作系统，让操作系统内核（Kernel）去调用驱动程序，只有驱动程序才能真正的操作硬件。从用户程序到硬件设备要经过好几层的转换，每一层的转换都有时间和空间的开销，而且开销不一定小；一旦用户程序需要密集的输入输出操作，这种开销将变得非常大，会成为制约程序性能的瓶颈。

这个时候，分配缓冲区就是必不可少的。每次调用读写函数，先将数据放入缓冲区，等数据都准备好了再进行真正的读写操作，这就大大减少了转换的次数。实践证明，合理的缓冲区设置能成倍提高程序性能。缓冲区其实就是一块内存空间，它用在硬件设备和用户程序之间，用来缓存数据，目的是让快速的 CPU 不必等待慢速的输入输出设备，同时减少操作硬件的次数。

缓冲区的类型

根据不同的标准，缓冲区可以有不同的分类。

根据缓冲区对应的是输入设备还是输出设备，可以分为输入缓冲区和输出缓冲区。

根据数据刷新（也可以称为清空缓冲区，就是将缓冲区中的数据“倒出”）的时机，可以分为全缓冲、行缓冲、不带缓冲。这种分类才本节要重点讲解的内容。

1) 全缓冲

在这种情况下，当缓冲区被填满以后才进行真正的输入输出操作。缓冲区的大小都有限制的，比如 1KB、4MB 等，数据量达到最大值时就清空缓冲区。

在实际开发中，将数据写入文件后，打开文件并不能立即看到内容，只有清空缓冲区，或者关闭文件，或者关闭程序后，才能在文件中看到内容。这种现象，就是缓冲区在作怪。

2) 行缓冲

在这种情况下，当在输入或者输出的过程中遇到换行符时，才执行真正的输入输出操作。行缓冲的典型代表就是标准输入设备（也即键盘）和标准输出设备（也即显示器）。

C语言标准的模棱两可

C语言标准规定，输入输出缓冲区要具有以下特征：

- 当且仅当输入输出不涉及交互设备时，它们才可以是全缓冲的。
- 错误显示设备不能带有缓冲区。

> 现代计算机已经没有了专门的错误显示设备，所有的信息都显示到一个屏幕上，这里的错误显示设备只能是计算机的显示器。上面提到的 perror() 其实就是向错误显示设备上输出信息，但是现代计算机已经把显示器作为了错误显示设备，所以 perror() 也是向显示器上输出内容。

所谓交互设备，就是现代计算机上的显示器和键盘。C标准虽然规定它们不能是全缓冲的，但并没有规定它们到底是行缓冲还是不带缓冲，这就导致不同的平台有不同的实现。

1) 输入设备

scanf()、getchar()、gets() 就是从输入设备（键盘）上读取内容。对于输入设备，没有缓冲区将导致非常奇怪的行为，比如，我们本来想输入一个整数 947，没有缓冲区的话，输入 9 就立即读取了，根本没有机会输入 47，所以，没有输入缓冲区是不能接受的。Windows、Linux、Mac OS 在实现时都给输入设备带上了行缓冲，所以 scanf()、getchar()、gets() 在每个平台下的表现都一致。

但是在某些特殊情况下，我们又希望程序能够立即响应用户按键，例如在游戏中，用户按下方向键人物要立即转向，而且越快越好，这肯定就不能带有缓冲区了。Windows 下特有的 getche() 和 getch() 就是为这种特殊需求而设计的，它们都不带缓冲区。

2) 输出设备

printf()、puts()、putchar() 就是向输出设备（显示器）上显示内容。对于输出设备，有没有缓冲区其实影响没有那么大，顶多是晚一会看到内容，不会有功能性的障碍，所以 Windows 和 Linux、Mac OS 采用了不同的方案：

- Windows 平台下，输出设备是不带缓冲区的；
- Linux 和 Mac OS 平台下，输出设备带有行缓冲区。

缓冲区的刷新（清空）

所谓刷新缓冲区，就是将缓冲区中的内容送达到目的地。缓冲区的刷新遵循以下的规则：

- 不管是行缓冲还是全缓冲，缓冲区满时会自动刷新；
- 行缓冲遇到换行符\n时会刷新；
- 关闭文件时会刷新缓冲区；
- 程序关闭时一般也会刷新缓冲区，这个是由标准库来保障的；
- 使用特定的函数也可以手动刷新缓冲区

总结

缓冲区位于用户程序和硬件设备之间，用来缓存数据，目的是让快速的 CPU 不必等待慢速的输入输出设备，同时减少操作硬件的次数。对于 IO 密集型的网络应用程序，比如网站、数据库、DNS、CDN 等，缓冲区的设计至关重要，它能十倍甚至一百倍得提高程序性能。