# 11.1 IO操作

I/O(input/output, 输入/输出)-计算机系统的重要组成部分，是指计算机与外部世界之间的信息交流过程。用于外部设备数据或外部设备写入数据。

  
常见的I/O操作：

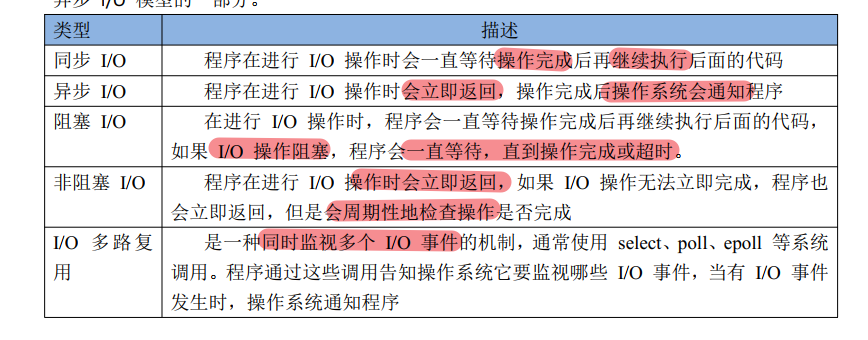
同步I/O：在进行 I/O 操作时， 程序会一直等待操作完成后再继续执行后面的代码。 如果 I/O 操作阻塞， 程序会一直等待， 直到操作完成或超时。

异步IO：在进行 I/O 操作时， 程序会立即返回， 而不必等待操作完成。 当操作完成后， 操作系统会通知程序。 这种方式可以允许程序在等待 I/O 操作完成的同时执行其他代码

阻塞IO：阻塞 I/O 是同步 I/O 的一种。

非阻塞IO：在进行 I/O 操作时， 程序会立即返回， 而不必等待操作完成。 如果 I/O 操作无法立即完成， 程序也会立即返回， 但是会周期性地检查操作是否完成。 非阻塞 I/O 是同步 I/O 的一种。

I/O多路复用：是一种同时监视多个 I/O 事件的机制， 通常使用 select、 poll、 epoll 等系统调用。 程序通过这些调用告知操作系统它要监视哪些 I/O 事件，当有 I/O 事件发生时， 操作系统通知程序， 并返回发生事件的描述符。 I/O 多路复用通常是异步 I/O 模型的一部分。



# 11.2 阻塞IO与非阻塞IO

阻塞和非阻塞的主要区别在于程序在进行 I/O 操作时是否会被阻塞。 阻塞模式下程序会一直等待 I/O 操作完成， 无法进行其他操作， 非阻塞模式下程序可以继续执行其他操作， 不必等待 I/O 操作完成。

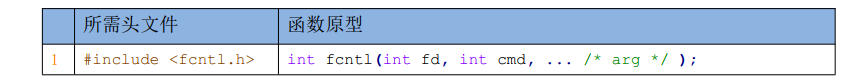
阻塞I/O会导致程序性能下降，造成资源浪费。

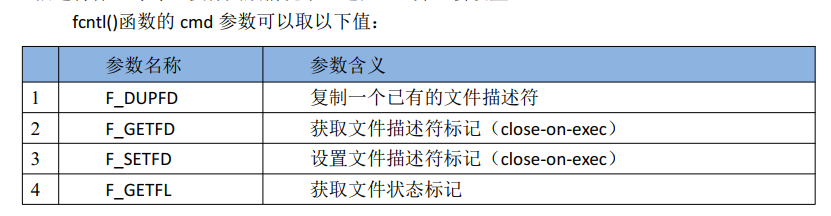
常用的scanf标准输入就属于阻塞IO。可不可以将其转化为非阻塞IO呢？

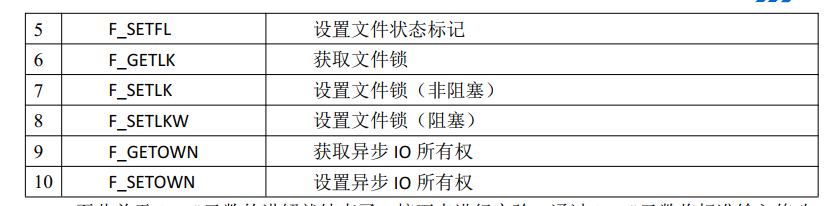
# 11.3 fcntl()函数

fcntl 函数是一个在 Linux 和 Unix 系统中使用的系统调用， 其主要用途是对已打开的文件描述符进行操作。其名称是file control的缩写。可用于对文件操作符进行控制操作，如：更改文件状态标志，获取或更改文件锁定。

执行失败情况下， 返回-1， 并且会设置 errno。



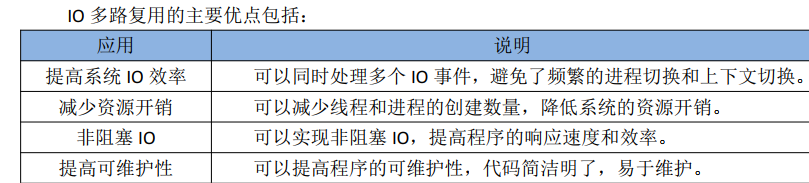




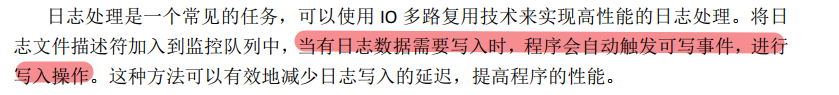
# 11.4 I/O多路复用

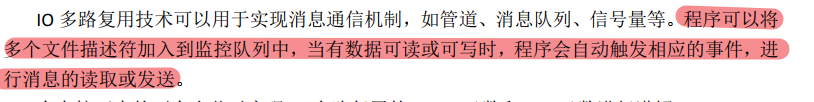
## 11.4.1 I.O多路复用介绍

是异步IO的一种，是一种同时监视多个文件描述符的技术，可以通过单个进程来处理多个IO操作，从而提高系统的IO效率。监视中的多个文件描述符号，若其中任意一个文件描述符准备就绪，就会通知进程进行读写操作。



可用于实现高效的网络服务器、实现异步事件处理、实现高性能的日志处理以及实现消息通信。

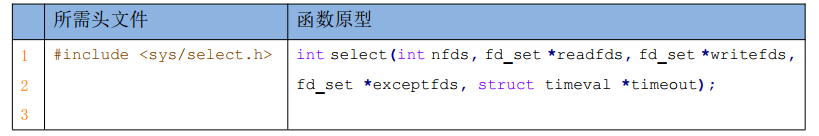


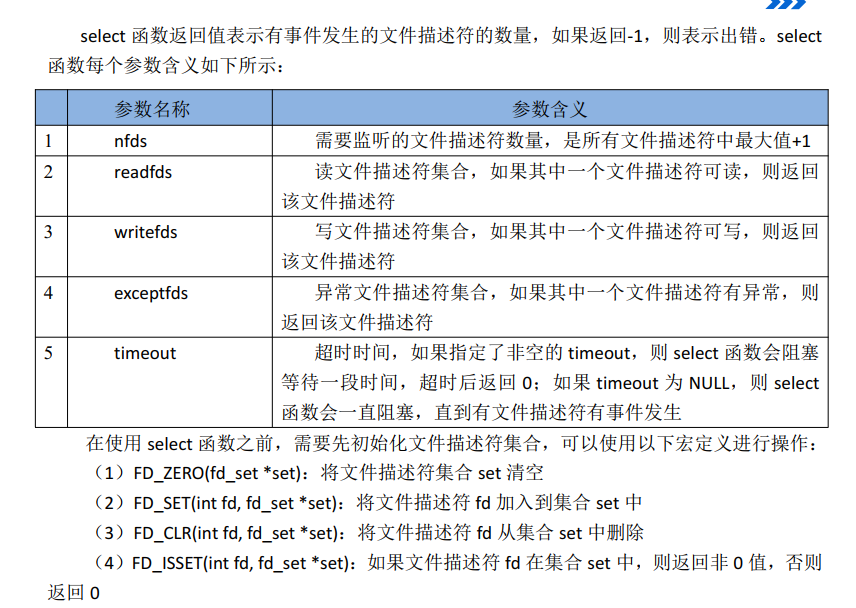


## 11.4.2 select函数

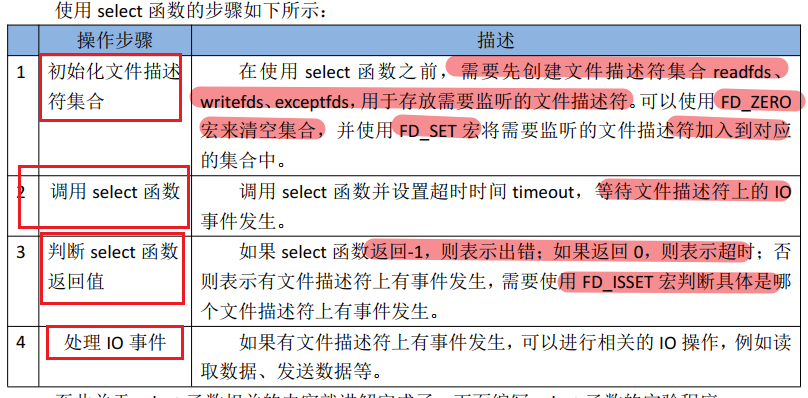
一种IO多路复用机制，可以同时监听多个文件描述符上的 IO 事件， 并在其中一个或多个文件描述符上发生 IO 事件时进行处理。

该函数常用于网络编程中，可以实现同时监听多个客户端连接请求，处理网络数据收发操作。



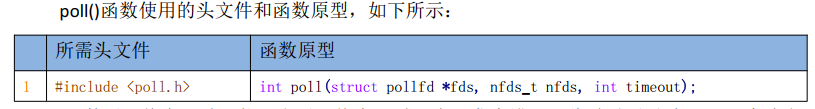


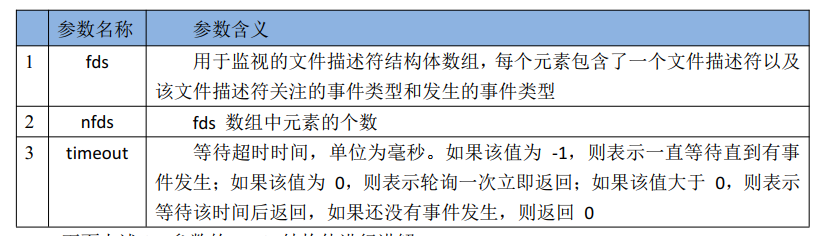
**使用select（）函数的步骤：**



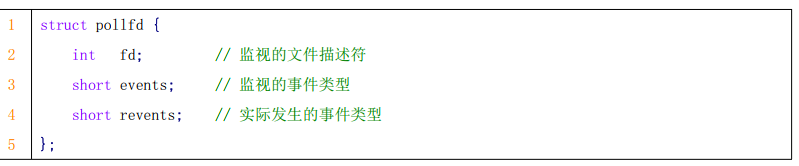
## 11.4.3 poll函数

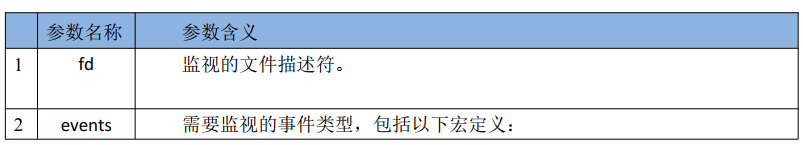
用于多路复用 I/O的一个系统调用函数，用于检测一组文件描述符中是否有可读、可写或异常等事件发生，从而实现异步I/O，它可以监视多个文件描述符（socket、标准输入输出、 文件等） 的状态， 一旦其中有任意一个描述符就绪（可读、 可写或有异常） ，则 poll() 函数返回， 并填充一个数组来表示就绪的文件描述符。

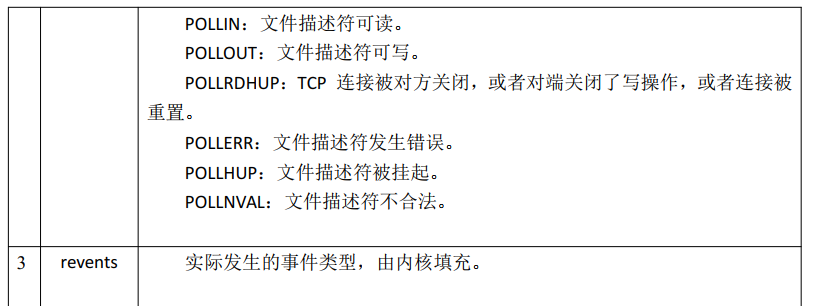




对pollfd结构体详解：







Poll函数的优点在于，能够同时监听多个文件描述符号，并不需要想select函数那样每次调用都需要将文件描述符重新添加到监视集合中。