 1.**文件I/O又称为低级磁盘I/O**，遵循POSIX（可移植操作系统接口）标准。任何兼容POSIX标准的操作系统都支持文件I/O。**标准I/O又称为高级磁盘I/O**，遵循ANSI C相关标准。只要开发环境有标准C库，标准I/O就可以使用。

在Linux系统中使用GLIBC标准，它是标准C库的超集，既支持ANSI C中定义的函数又支持POSIX中定义的函数。因此Linux下既可以使用标准I/O，也可以使用文件I/O。

    2.通过文件I/O读写文件时，每次操作都会执行相关**系统调用**。这样的好处是直接读写实际文件，坏处是频繁的系统调用会增加系统开销。标准I/O在文件I/O的基础上封装了缓冲机制，每次先**操作缓冲区**，必要时再访问文件，从而**减少了系统调用的次数**。

3.文件I/O使用文件描述符打开操作一个文件，可以**访问不同类型的文件**（例如普通文件、设备文件和管道文件等）。而标准I/O使用FILE指针来表示一个打开的文件，通常**只能访问普通文件**。

<https://blog.csdn.net/nan_lei/article/details/81460798>

介于底层和应用层的开发（中间层），系统级别的开发。

文件IO/系统调用IO

Fd是在文件IO中贯穿始终的类型

文件描述符的概念

（整形数，数组下标，文件描述符有线使用当前可用范围内最小的）

文件IO操作：open, close, read, write, lseek

open()函数是使用变参实现的，和C++里面的重载不一样，如何判断一个函数是通过变参还是函数重载实现的，可以测试：传入多个参数，如果报出语法错误，就说明是函数重载实现的，如果是报出警告错误，就说明是通过变参实现的。在gcc里面可以使用-Wall打印出编译过程中的警告，建议在以后编程的时候，排除任何你无法解释的警告。

文件IO与标准IO的区别

文件IO的响应速度快；标准IO的吞吐量大

面试：如何使一个程序变快？

通常说的是吞吐量。

提醒：标准IO和文件IO不可以混用。

转换：fileno, fdopen

(strace 可执行文件)----->显示文件执行内容

(time 可执行文件)------>显示文件执行的时间

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main()

{

putchar(‘a’);

write(1, “b”, 1);

putchar(‘a’);

write(1, “b”, 1);

putchar(‘a’);

write(1, “b”, 1);

exit(0);

}

IO的效率问题

文件共享

原子操作：不可分割的操作。

原子：不可分割的最小单位。

原子操作的作用：解决竞争和冲突。

如tmpnam

程序中的重定向：dup, dup2

同步：sync, fsync, fdatasync

Fcntl();

Ioctl();

/dev/fd/目录：虚目录，显示的是当前进程的文件描述符信息