# ****文件描述符****

**在 Linux 的世界里，一切设备皆文件**。

打开现存文件或新建文件时，系统（内核）会返回一个文件描述符，**这个文件描述符相当于这个已打开文件的标号**

程序运行起来后（每个进程）都有一张文件描述符的表，标准输入、标准输出、标准错误输出设备文件在程序运行时默认打开，对应的文件描述符 0、1、2 记录在表中。

#define STDIN\_FILENO  0 //标准输入的文件描述符

#define STDOUT\_FILENO 1 //标准输出的文件描述符

#define STDERR\_FILENO 2 //标准错误的文件描述符

Linux 中一个进程最多只能打开 NR\_OPEN\_DEFAULT （即1024）个文件，故当文件不再使用时应及时调用 close() 函数关闭文件。

# 常用 I/0 函数

需要的头文件：

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

**打开文件**

**int open(const char \*pathname, int flags, mode\_t mode);**

功能：

打开文件，如果文件不存在则创建。

参数：

pathname: 文件的路径及文件名。

flags: 打开文件的行为标志



mode: 这个参数，只有在文件不存在时有效，指新建文件时指定文件的权限



返回值：

成功：成功返回打开的文件描述符

失败：-1

**关闭文件**

**int close(int fd);**

功能：

关闭已打开的文件

参数：

fd: 文件描述符，open()的返回值

返回值：

成功：0

失败：-1

**写入文件**

**ssize\_t write(int fd, const void \*addr, size\_t count);**

功能：

把指定数目的数据写到文件（fd）

参数：

fd: 文件描述符

addr: 数据首地址

count: 写入数据的长度（字节），**一般情况下，数据有多少，就往文件里写多少，不能多也不能少**

返回值：

成功：实际写入数据的字节个数

失败：-1

**读取文件**

**ssize\_t read(int fd, void \*addr, size\_t count);**

功能：

把指定数目的数据读到内存（缓冲区）

参数：

fd: 文件描述符

addr: 内存首地址

count: 读取的字节个数

返回值：

成功：实际读取到的字节个数

失败：-1

# 实战示例

使用 open() 打开源文件，使用 read() 从文件读数据，使用 write() 向目的文件写数据，示例代码如下：

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

int main(int argc, char \*argv[])

{

    if((argc == 3) && (strcmp(argv[1], argv[2]) != 0))

    {// 保证有 3 个参数，而且源文件和目的文件名字不能一样

        int fd\_src, fd\_dest, ret;

        //只读方式打开源文件

        fd\_src = open(argv[1], O\_RDONLY);

        if(fd\_src < 0)

        {

            perror("open argv[1]");

            return -1;

        }

        // 新建目的文件

        fd\_dest = open(argv[2], O\_WRONLY|O\_CREAT, 0755);

        if(fd\_dest < 0)

        {

            close(fd\_src);

            perror("open argv[2]");

            return -1;

        }

        do

        {

            char buf[1024] = {0};

            // 从源文件读取数据

            ret = read(fd\_src, buf, sizeof(buf));

            // 把数据写到目的文件，注意最后一个参数，有多少写多少

            write(fd\_dest, buf, ret);

        }while(ret > 0);

        // 关闭已打开的文件

        close(fd\_src);

        close(fd\_dest);

    }

    return 0;

}