c++98(学校学习版本)-->c++03-->c++11(超级大版本)-->c++14(小版本)

0x-->十六位二进制 0-->八位二进制

对文件的操作函数:

打开文件:fopen(“文件路径”,打开方式[w/r/d])

关闭文件:fclose(“”)

stdin 标准输入文件描述符为0

stdout 标准输出 文件描述符为1

stderr 标准错误 文件描述符为2

接口

**open(“文件路径”,打开方式[w/r/d])** -->系统调用打开文件

返回值是文件描述符fd(每个文件都不一样,较小且递增), stdin stdout stderr都是一个文件,文件描述符分别是0 1 2

进程PCB task\_struct结构体中的文件描述符表指针指向一个文件描述符表(数组),数组中的每个元素是一个struct file\*,能够对文件属性进行描述,因此,文件描述符就是文件描述符表的下标,文件描述符表是有上限的(可配置)(ulimit -n 配置),因此如果不用文件了,就要及时关闭如果一个文件一直打开不关闭 就会造成文件描述符泄露,从而无法正确打开新的文件.

**close([文件描述符])**  -->系统调用关闭文件

文件描述符分配规则:从0开始查找,找到第一个空闲的位置,作为新文件的对应表项,并且把这个下标当作描述符返回.

**重定向系统调用函数: dup2(old\_fd,new\_fd)**

往new里写,实际写到了old中

**FILE**

linux中的缓冲机制有三种：

1没有缓冲

2.行缓冲**(**把数据输出到标准输出上**)**  何时刷新缓冲区？？

a)遇到\n b)缓冲区满了c)手动调用fflush d)进程终止

3.全缓冲**(**把数据输出到文件中**)** 何时刷新缓冲？？？

a)缓冲区满了也会刷新缓冲区 b)手动调用fflush也会刷新缓冲区 c)进程终止

**inode 在磁盘上给文件名编号的节点编号**

**ln -s 创建出一个文件的软连接（**符号链接）相当于快捷方式

软连接文件其实是一个新文件，但与原文件内容相同，且原文件改变软连接文件内容也会自动改变即（写多个项目需用到相同一个文件时可使用）

stat [文件名] 可以看到文件更详细的信息

readlink [文件名] 读文件的软连接

**ln 创建文件的硬链接（windows中不支持硬连接）**

硬链接和原来的文件之间其实是同一个 inode 硬连接只是创建了另外一个文件名 能够和原来的文件名引用到同一个文件

当多个文件名对应到一个inode的时候 把有多少个文件对应到该inode这样的数字称为“引用计数”解决盘块以及inode何时释放的问题

静态库:把若干个 .o 文件打包在一起（隐藏文件）

lib[库名].a： --->静态库