课程目标：掌握文件操作，库的制作和使用

课程特点：函数的使用

学习方法：函数手册 + 示例代码

**一、IO理论知识**

**1.1 基础知识**

**1.1.1 系统调用与库函数**

Linux系统下的文件操作有两大类函数

标准io --- C库提供 （库函数）

文件io --- linux系统提供 （系统调用）

文件io是linux内核对外提供的接口。用户空间的程序通过调用这些系统调用与内核交互，最终完成对文件（包括普通文件、设备文件、网络套接字等）的读写操作。

标准io由C库提供，是对系统调用的封装，主要目的是提供缓冲机制，减少系统调用次数，提升效率。

**1.1.2 错误解析**

Linux 使用一个全局变量 errno（类型为 int）存储最近一次系统调用或库函数出错时的错误码。它的定义在头文件 <errno.h>

为了将抽象的错误码（如 EACCES）转换为人类可读的描述，Linux 提供了以下工具函数：

perror() 、strerror()

**1.1.3 普通文件分类**

文本文件 (本质也是二进制数据，但这些二进制数据都是各种字符编码，可以通过文本编辑器查看)

二进制文件（存储的是原始的二进制数据）

**二、标准IO**

标准 I/O（Standard Input/Output）是 C 语言中用于处理输入输出操作的标准库（stdio.h），它提供了一套统一的接口，用于操作​​标准输入流​​、​​标准输出流​​和​​标准错误流​​，并支持对文件、终端等设备的读写。

其核心思想是将各种输入输出设备抽象为“流”（stream），通过统一的接口操作这些流。

**2.1 标准io相关函数**

fopen/fclose /freopen //打开与关闭

fgets/fputs //按行读写

fgetc/fputc //按字符读写

fread/fwrite //按块读写

fscanf/fprintf //从文件中格式化提取与打印

fseek、ftell、rewind //定位流

fflush //刷新流

feof //判断是否到文件末尾

**2.2 标准io缓存方式**

全缓存

当缓存满了

强制刷新或关闭文件时，执行io操作

典型例子：操作普通文件时

行缓存

当缓存满了，遇到换行符

强制刷新或关闭文件时，执行io操作

典型例子： stdout、stdin

无缓存

没有缓存，直接执行io操作

典型例子：stderr

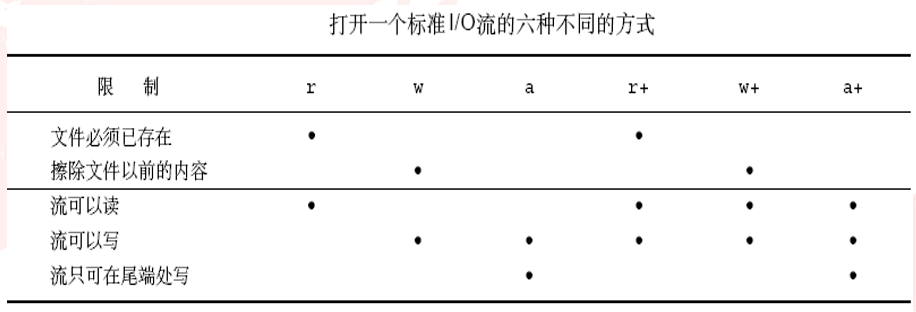
**2.3 stdin、stdout、stderr**

标准I/O预定义3个流，不需要用户手动打开，执行程序时默认打开，并关联到对应的设备

|  |  |
| --- | --- |
| stdin | 标准输入 |
| stdout | 标准输出 |
| stderr | 标准错误输出 |

**2.4 打开方式**

|  |  |  |
| --- | --- | --- |
| r | 只读 | 文件必须存在 |
| w | 只写 | 文件不存在则创建，文件存在则清空 |
| a | 追加写入 | 文件不存在则创建 |
| r+ | 在r方式上增加 写权限 | 文件必须存在 |
| w+ | 在w方式上增加 读权限 | 文件不存在则创建，文件存在则清空 |
| a+ | 在a方式上增加 读权限 | 文件不存在则创建 |



**2.5 文件的定位**

标准 I/O 提供了一组函数用于文件定位，核心包括 fseek、ftell、rewind。

fseek 函数用于将文件指针移动到指定位置，是最常用的定位函数。

ftell 函数用于获取文件指针的当前位置（相对于文件开头的偏移量，单位为字节）。

rewind 是一个便捷函数，等价于将文件指针移动到文件开头，并清除流的错误标志。

**2.6 示例代码**

以字符方式进行读写

#include <stdio.h>

int main(void)

{

    int ret;

    float val;

    char ch;

    FILE \*fp = fopen("1.txt", "r"); // 打开文件

    if (fp == NULL)

    {                          // 读方式

        perror("fopen 1.txt"); // 文件必须存在

        return -1;

    }

    // 读操作

    ch = fgetc(fp);         //读一个字符

    printf("ch:%c\n",ch);

#if 0

    char buf[256];

    fgets(buf,sizeof(buf),fp);  //读一行数据

    printf("buf:%s\n",buf);

#endif

#if 0

    fscanf(fp, "val:%f", &val); // 格式化提取

    printf("%.2f\n", val);

#endif

    fclose(fp); // 关闭文件

    return 0;

}

#include <stdio.h>

int main(void)

{

    int ret;

    float val = 465.3;

    FILE \*fp =  fopen("1.txt","w");    //打开文件

    if(fp==NULL){                      //写方式

        perror("fopen 1.txt");         //文件不存在 创建

        return -1;                     //文件存在   清空

    }

    //写操作

    //fputc('x',fp);       //写入一个字符

    //fputs("hello",fp);   //写入字符串

    fprintf(fp,"val:%.2f\n",val);  //格式化写入

    fclose(fp); //关闭文件

    return 0;

}

读写结构体

//部分代码

struct Student{

    int id;

    char name[32];

    double score;

};

typedef struct Student Stu;

//写结构体对象 到文件中

Stu stu;

    printf("输入学生信息(id name score)");

    scanf("%d %s %lf",&stu.id,stu.name,&stu.score);

    fwrite(&stu,sizeof(stu),1,fp);

//从文件中读 结构体数据

    Stu tmp;

    ret = fread(&tmp,sizeof(tmp),1,fp);

    printf("%d %s %.2f\n",tmp.id,tmp.name,tmp.score);

行缓存示例

#include <stdio.h>

//stdout : 标准输出流， 行缓存

int main(void)

{

    while(1){

        fprintf(stdout,"abc");   //通过循环写入 填满一行  执行io操作

    }

    //fprintf(stdout,"abc\n");   //遇到换行符 执行io操作

#if 0

    fprintf(stdout,"abc");      //当文件关闭时 执行io操作

    fclose(stdout);

#endif

#if 0

    fprintf(stdout,"abc");      //通过fflush刷新 触发io操作

    fflush(stdout);

    while(1);

#endif

    return 0;

}

**三、函数备注**

FILE \*fopen(const char \*path, const char \*mode);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 打开文件，并关联到流指针

\* Input: path：路径名，mode：打开的方式，例如常用的方式：

r:只读，文件必须存在

w:只写，文件不存在则创建，文件存在则清空

a:只写，追加方式，文件不存在则创建，文件存在则打开

\* Output:

\* Return: 成功返回流指针，失败返回NULL

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int fclose(FILE \*fp);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 关闭文件

\* Input: fp：流指针

\* Output:

\* Return: 成功返回0，失败返回EOF

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int fgetc(FILE \*stream);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 从流中读取一个字符

\* Args: stream：流指针

\* Return: 成功返回字符ascii编码值，出错或读到末尾返回EOF(-1)

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int fputc(int c, FILE \*stream);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 向流中写一个字符

\* Args: c:字符，stream：流指针

\* Return: 成功返回c，出错返回EOF

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char \*fgets(char \*s, int size, FILE \*stream);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 从流中读取一行

\* Input: size:希望读取的字节数，stream：流指针

\* Output:   s:内存空间（存储读取的数据）

\* Return: 成功返回s，出错或到达末尾，返回NULL

\* Others:   fgets，必须指定内存空间的大小，此函数一直读到换行符或末尾，

但是不超过指定的大小。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int fputs(const char \*s, FILE \*stream);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 向流中写字符串

\* Args: s:待写入的数据地址，stream：流指针

\* Return: 成功返回非负值，出错返回EOF

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

size\_t fread(void \*ptr, size\_t size, size\_t nmemb, FILE \*stream);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 从流中读一片数据（按对象读）

\* Input: size:对象的大小，nmemb:多少个对象，stream：流指针

\* Output:   ptr:内存空间（存储读取的数据）

\* Return: 成功返回读取到的对象个数。到达末尾或出错返回0。

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

size\_t fwrite(const void \*ptr, size\_t size, size\_t nmemb,FILE \*stream);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 向流中写一片数据

\* Input: ptr:待写入的数据地址，size:对象大小，nmemb:对象个数，

stream：流指针

\* Output:

\* Return: 成功返回写了多少对象，出错或到达末尾返回值小于nmemb。

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int fprintf(FILE \*stream, const char \*format, ...);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 往文件中打印

\* Input: format：格式控制符，...：不定参数

\* Output:

\* Return: 成功返回：成功打印的字节数

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int fscanf(FILE \*stream, const char \*format, ...);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 从文件中进行格式化提取

\* Input: format：格式控制符，...：不定参数

\* Output:

\* Return: 成功返回：提取成功的数据项

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void perror(const char \*s);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 打印错误提示

\* Args: s:标签

\* Return:

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char \*strerror(int errnum);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 根据错误码返回错误提示

\* Args: errnum：错误码

\* Return: 成功返回错误提示

\* Others:   /usr/include/asm-generic/errno-base.h

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int fseek(FILE \*stream, long offset, int whence);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 流的定位

\* Args: stream：流指针，offset:偏移量（正数向后，负数向前），

whence:基准位置

SEEK\_SET:文件头

SEEK\_CUR:当前位置

SEEK\_END:文件尾

\* Return: 成功返回0，出错返回-1。

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

long ftell(FILE \*stream);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 用于取得当前的文件位置

\* Args: stream：流指针

\* Return: 成功返回当前位置指示，出错返回-1。

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void rewind(FILE \*stream);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 重置文件指针到开头

\* Args: stream：流指针

\* Return:

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int fflush(FILE \*stream);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 强制刷新

\* Args: stream：流指针

\*   对于写入操作（输出），执行刷新会立即写入

对于读取操作（输入），执行刷新会丢掉数据

\* Return: 成功返回0，出错返回-1。

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int feof(FILE \*stream);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description： 判断是否达到末尾

\* Args: stream：流指针

\*

\* Return: 非零值代表文件结束（真：到达文件末尾）

\* Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**作业**

7月2日：

1.实现一个简单的文件加密

7月3日：

1.间隔1s将时间写入文件并带上行号，ctrl+c 结束后，再次执行可以继续写入并且序号能续上。

(缓存问题，获取文件行号，追加写入)

2.采用文件存储用户信息，实现用户信息的输入与查询

(读写操作，追加写入)

3.实现bmp图片的拷贝，并尝试对图片进行修改

（二进制拷贝）