

**fgetc 函数说明：**

作用：从文件指针指向的文件流中读取一个字符，读取一个字节后，光标位置后移一个字节

格式：int fgetc(FILE \*stream);

返回值：返回所读取的一个字节，如果读到文件末尾或者读取出错时返回EOF(EOF是文件结束标识符,一般值为-1)

参数说明：

    \*stream：文件指针，从该文件指针指向的文件中读取一个字符，然后将光标后移一个字节

**fputc 函数说明：**

作用：将指定字符写到文件指针所指向的文件的当前写指针位置上

格式：int fputc (char c, File \*fp);

返回值：在正常调用情况下，函数返回写入文件的字符的ASCII码值，出错时，返回EOF

参数说明：

    c：需要写入的字符

    \*fp：文件指针，在当前文件指针所指向的文件的当前写指针位置上写入一个字符c，然后文件内部写指针会自动后移一个字节位置

**fgets 函数说明：**

作用：从文件结构体指针stream中读取数据，每次读取一行。读取的数据保存在buf指向的字符数组中，每次最多读取bufsize-1个字符（第bufsize个字符赋'\0'），如果文件中的该行，不足bufsize个字符，则读完该行就结束。如若该行（包括最后一个换行符）的字符数超过bufsize-1，则fgets只返回一个不完整的行，但是，缓冲区总是以NULL字符结尾，对fgets的下一次调用会继续读该行。

格式：char \*fgets(char \*buf, int bufsize, FILE \*stream);

返回值：函数成功将返回buf，失败或读到文件结尾返回NULL。因此我们不能直接通过fgets的返回值来判断函数是否是出错而终止的，应该借助feof函数或者ferror函数来判断。

参数说明：

    \*buf: 字符型指针，指向用来存储所得数据的地址。

    bufsize: 整型数据，指明存储数据的大小。

\*stream: 文件指针，将要读取的文件流。

**fputs 函数说明：**

作用：向指定的文件写入一个字符串（不自动写入字符串结束标记符‘\0’）

格式：int fputs(char \*str, FILE \*fp);

返回值：若成功返回0，失败返回EOF

参数说明：

    \*str: 需要写入的字符串

    \*fp: 文件指针，将要写入的文件流

**fprintf 函数说明：**

作用：格式化后输出到文件中

格式：int fprintf (FILE\*  stream, const char\*  format, [argument]);

返回值：若成功返回值是输出的字符数，当发生错误时返回一个负值

参数说明：

    \*stream：文件指针

    \*format：输出格式

    [argument]：附加参数列表

注：用法与printf函数类似，这里只是参数多了一个文件指针，将格式后的结果输出到文件中

**fscanf 函数说明：**

作用：从一个流中执行格式化输入

格式：int fscanf(FILE\* stream,constchar\* format,[argument]);

返回值：成功返回读入的参数的个数，失败返回EOF。

参数说明：

    \*stream：文件指针

    \*format：格式字符串

    [argument]：输入列表

注：用法与scanf函数类似

**fread 函数说明：**

作用：从一个文件流中读数据，最多读取count个元素，每个元素size字节

格式：size\_t fread ( void \*buffer, size\_t  size, size\_t  count, FILE \*stream) ;

返回值：如果调用成功返回实际读取到的元素个数，如果不成功或读到文件末尾返回0

参数说明：

    \*buffer：用于接收数据的内存地址

    size：要读的每个数据项的字节数，单位是字节

    count：要读count个数据项，每个数据项size个字节

    \*stream：文件指针  
注：这个函数以二进制形式对文件进行操作，不局限于文本文件

**fwrite 函数说明：**

作用：向文件写入一个数据块

格式：size\_t fwrite(const void\* buffer, size\_t  size, size\_t  count, FILE\* stream);

返回值：返回实际写入的数据块数目

参数说明：

    \*buffer：待写入的数据的内存地址

    size：要写入内容的单字节数

    count：要进行写入size字节的数据项的个数

    \*stream：文件指针

注：这个函数以二进制形式对文件进行操作，不局限于文本文件

**fseek 函数说明：**

作用：重定位流(数据流/文件)上的文件内部位置指针

格式：int fseek(FILE \*stream, long offset, int origin);

返回值：成功返回0，失败返回非0值

参数说明：

    \*stream：文件指针

    offset：偏移量，正数表示正向偏移，负数表示负向偏移。因为是long型数据，所以应在数字后面加一个字母L

    origin：设定从文件的哪里开始偏移,可取值为：SEEK\_CUR、 SEEK\_END 或 SEEK\_SET，详见下表：



**rewind 函数说明：**

作用：将文件内部的位置指针重新指向一个流（数据流/文件）的开头

格式：void rewind(FILE \*stream);

返回值：无

参数说明：

    \*stream：文件指针

**ftell 函数说明：**

作用：得到文件位置指针当前位置相对于文件首的偏移字节数（测定文件位置标记的当前位置）

格式：long ftell(FILE \*stream);

返回值：成功返回当前文件位置，失败返回-1L

参数说明：

    \*stream：文件指针

注：因为ftell返回long型，根据long型的取值范围-2^31~2^31-1（-2147483648～2147483647），故对大于2.1G的文件进行操作时出错。

stdin & stdout & stderr

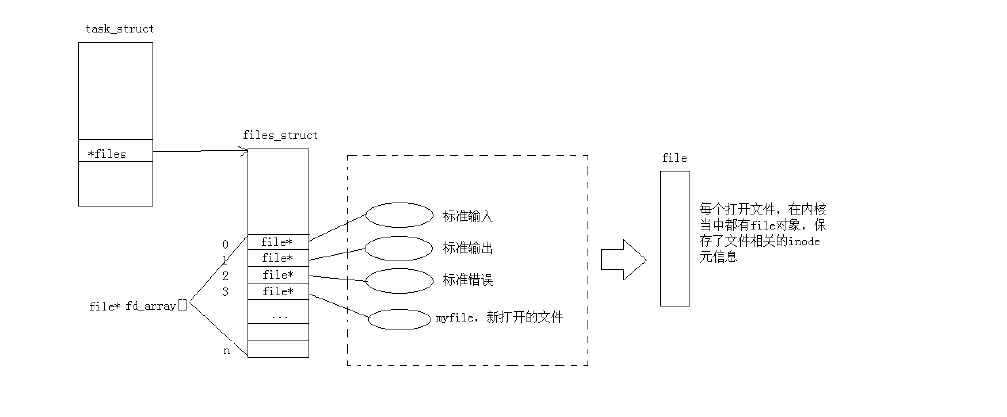
C默认会打开三个输入输出流，分别是stdin, stdout, stderr

仔细观察发现，这三个流的类型都是FILE\*, fopen返回值类型，文件指针

1.文件描述符：

①单个进程 查看：ulimit -n 一般情况下最大1024

②系统最大打开文件描述符数：/proc/sys/fs/file-max



2.重定向：

2.1使用 dup2 系统调用

#include <unistd.h>

int dup2(int oldfd, int newfd);

3. 查看所有进程打开的文件数量：

lsof |wc -l

查看某个进程打开的文件数：

lsof -p pid|wc -l *# pid为进程id*

4.FILE

因为IO相关函数与系统调用接口对应，并且库函数封装系统调用，所以本质上，访问文件都是通过fd访问的。

所以C库当中的FILE结构体内部，必定封装了fd。

-rw-r--r--. 1 root root 654 "9月 13 14:56" test.c

模式

硬链接数

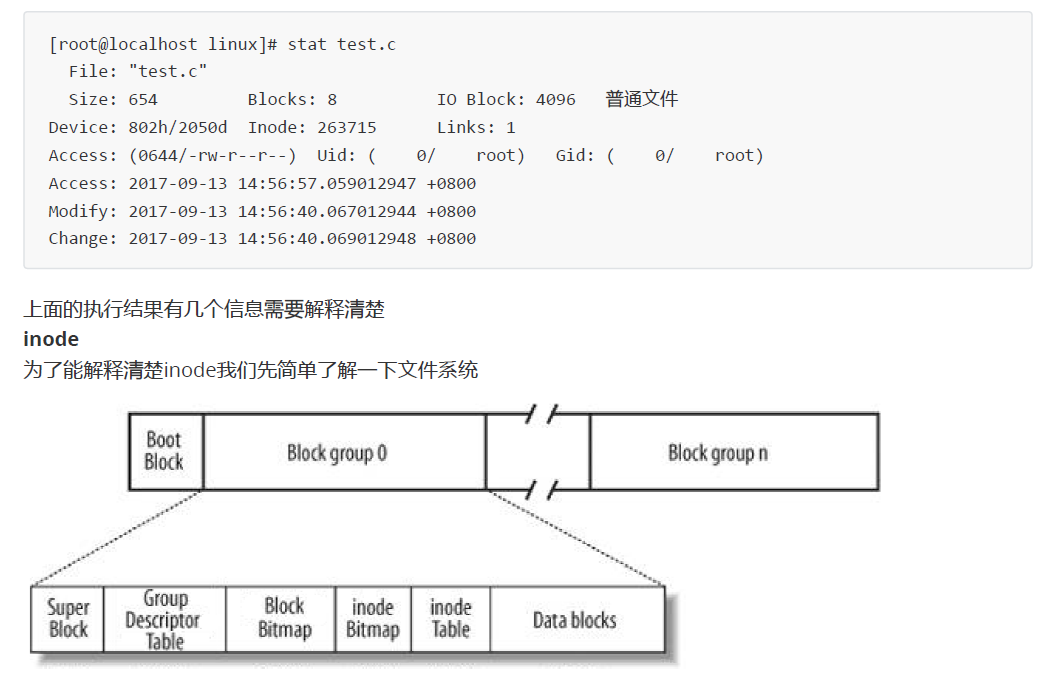
文件所有者

组

大小

最后修改时间

文件名



mke2fs的-b选项可以设定block大小为1024、2048或4096字节。而上图中启动块（Boot Block）的大小是确定的，

①Block Group：ext2文件系统会根据分区的大小划分为数个Block Group。而每个Block Group都有着相同的结构组成。政府管理各区的例子

②超级块（Super Block）：存放文件系统本身的结构信息。记录的信息主要有：bolck 和 inode的总量，未使用的block和inode的数量，一个block和inode的大小，最近一次挂载的时间，最近一次写入数据的时间，最近一次检验磁盘的时间等其他文件系统的相关信息。Super Block的信息被破坏，可以说整个文件系统结构就被破坏了

③GDT，Group Descriptor Table：块组描述符，描述块组属性信息，有兴趣的同学可以在了解一下

④块位图（Block Bitmap）：Block Bitmap中记录着Data Block中哪个数据块已经被占用，哪个数据块没有被占用

⑤inode位图（inode Bitmap）：每个bit表示一个inode是否空闲可用。

⑥i节点表:存放文件属性 如 文件大小，所有者，最近修改时间等

⑦数据区：存放文件内容

生成静态库：ar -rc libmymath.a add.o sub.o

查看：ar -tv libmymath.a