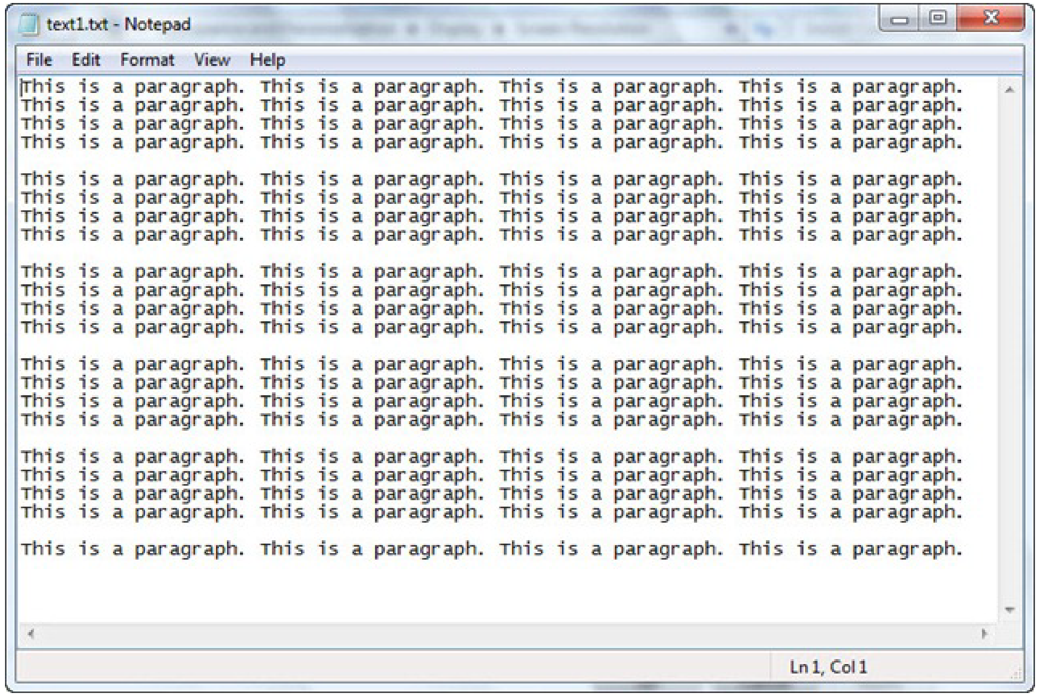
程序设计
Programming

Lecture 11: 文件

C语言文件

- 根据编码的方式可以分为
  - ✓文本文件：以字符ASCII码进行存储
  - ✓二进制文件：以二进制形式进行存储



This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.

This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.

This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.

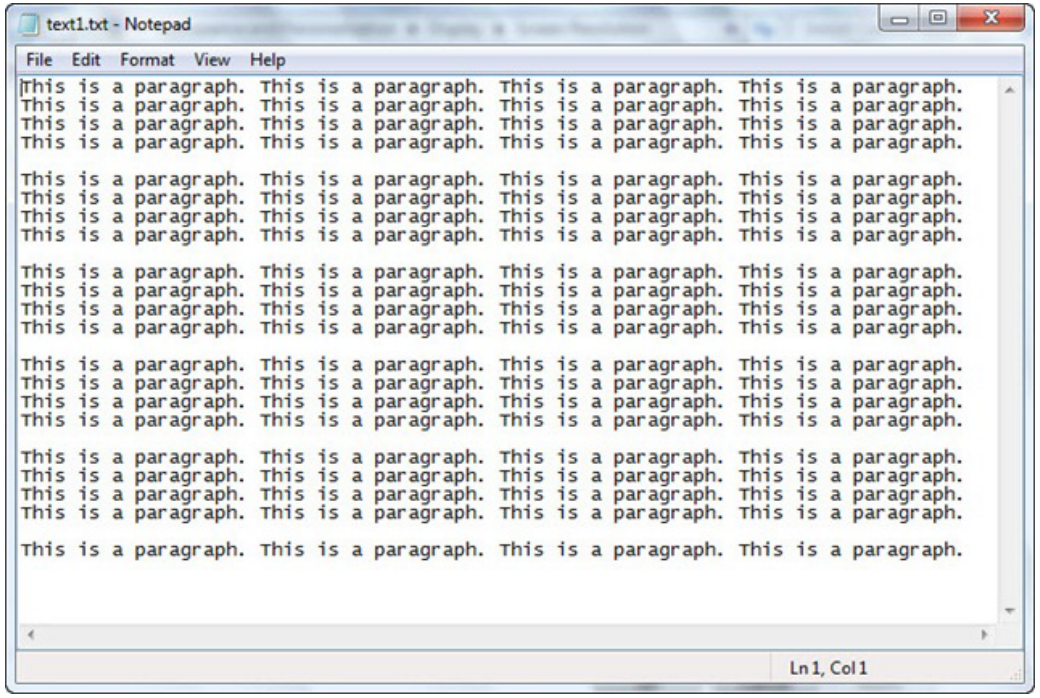
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.

This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.

This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.

C语言文件

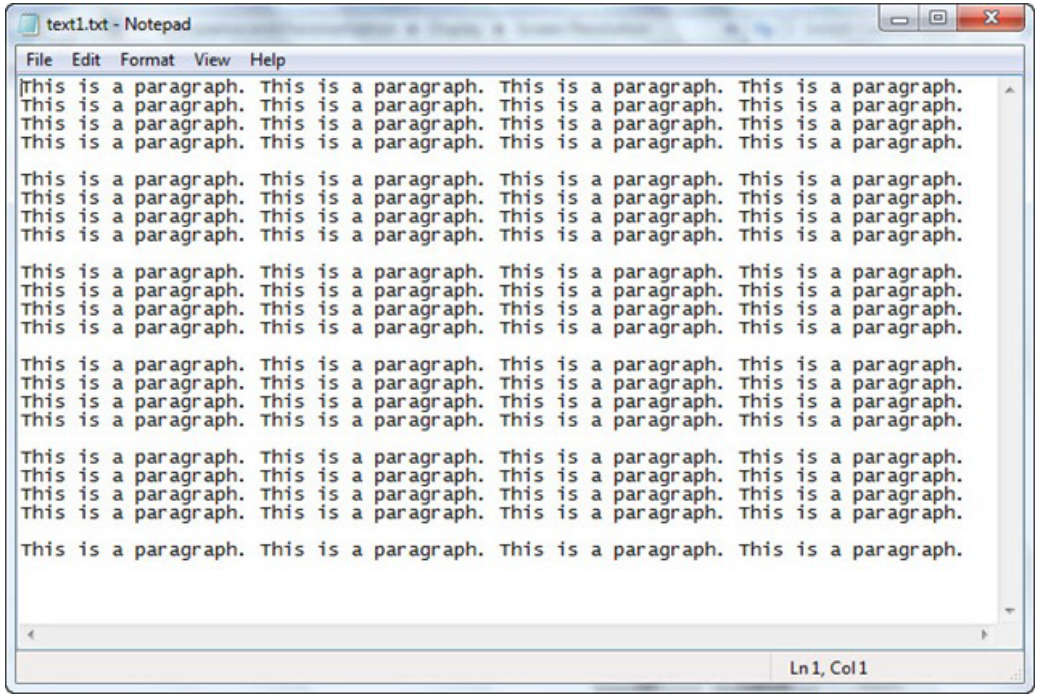
- 根据编码的方式可以分为
  - ✓文本文件：以字符ASCII码进行存储
  - ✓二进制文件：以二进制形式进行存储

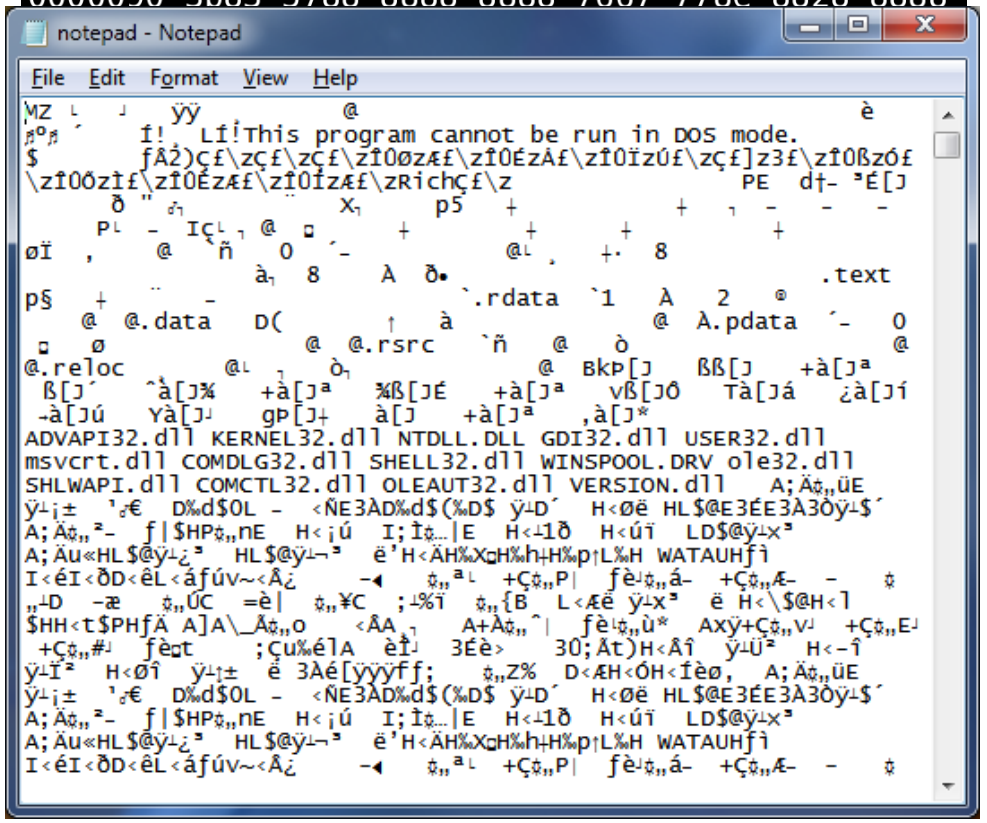


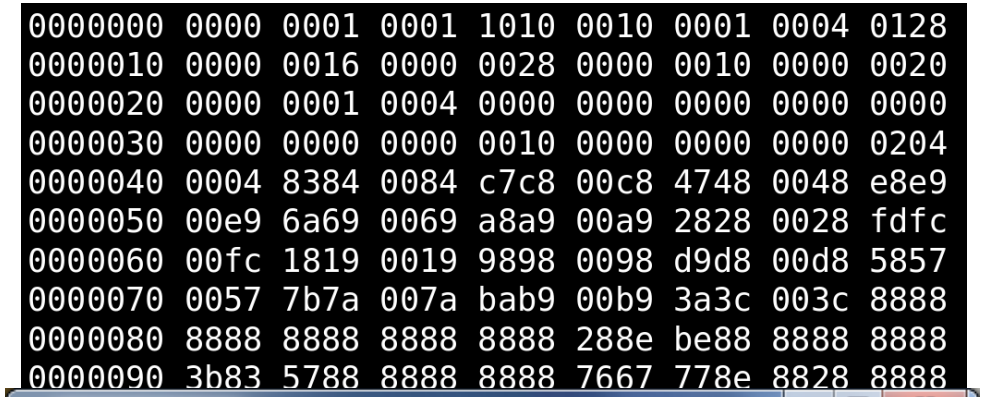

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

C语言文件

- 根据编码的方式可以分为
  - ✓文本文件：以字符ASCII码进行存储
  - ✓二进制文件：以二进制形式进行存储

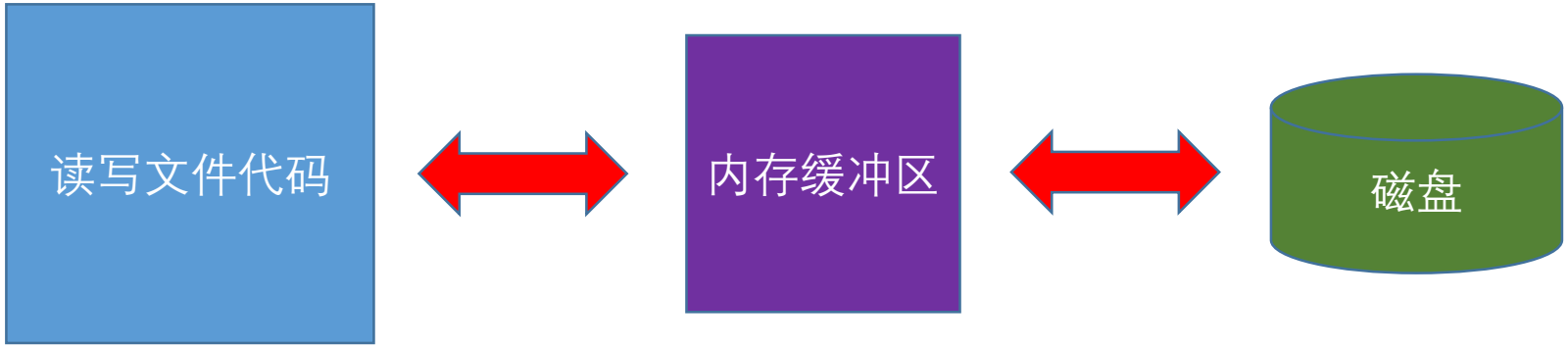






缓冲文件系统

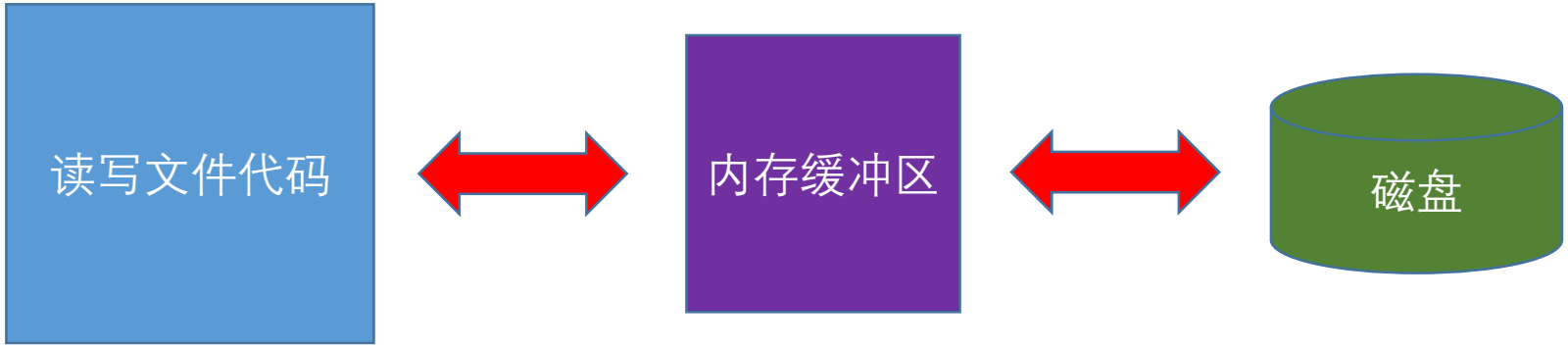
- 读写文件时，操作系统在内存开辟一个“缓冲区”（buffer）
  - ✓读文件时，先从磁盘读到buffer，buffer满后再读入到内存程序区
  - ✓写文件时，先写入buffer，buffer满后再写入文件



缓冲文件系统

printf("%d", BUFSIZ)查看缓冲区大小

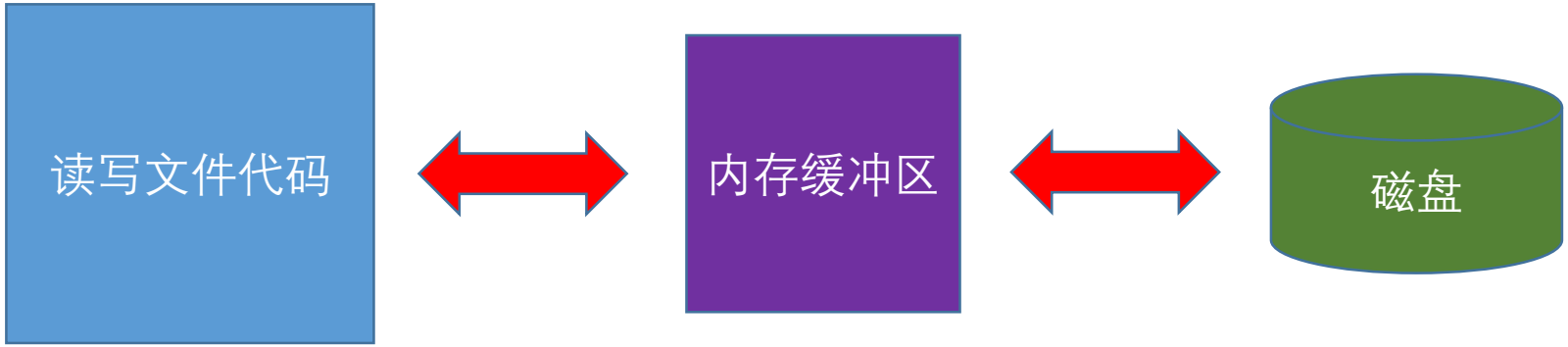
- 读写文件时，操作系统在内存开辟一个“缓冲区”（buffer）
  - ✓读文件时，先从磁盘读到buffer，buffer满后再读入到内存程序区
  - ✓写文件时，先写入buffer，buffer满后再写入文件



缓冲文件系统

printf("%d", BUFSIZ)查看缓冲区大小

- 读写文件时，操作系统在内存开辟一个“缓冲区”（buffer）
  - ✓读文件时，先从磁盘读到buffer，buffer满后再读入到内存程序区
  - ✓写文件时，先写入buffer，buffer满后再写入文件

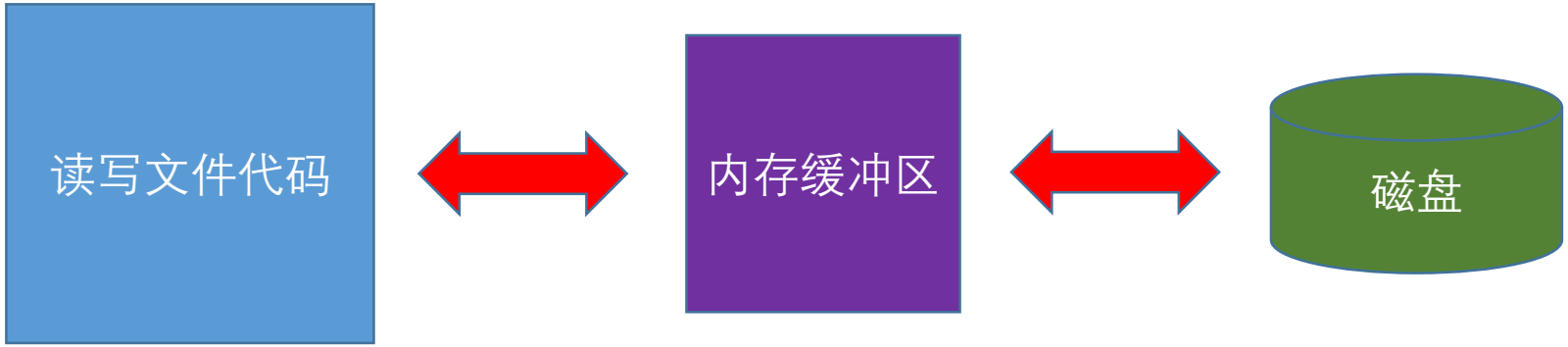


- 非缓冲文件系统
  - ✓由程序员编写程序手动开辟缓冲区
  - ✓一般专门用来读写二进制文件，效率高、速度快
  - ✓ANSI C规定只采用缓冲文件系统

缓冲文件系统

printf("%d", BUFSIZ)查看缓冲区大小

- 读写文件时，操作系统在内存开辟一个“缓冲区”（buffer）
  - ✓读文件时，先从磁盘读到buffer，buffer满后再读入到内存程序区
  - ✓写文件时，先写入buffer，buffer满后再写入文件



- 非缓冲文件系统

C语言处理文件，实际是和内存缓冲区进行交互；内存缓冲区和磁盘交互由操作系统自动完成

  - ✓ANSI C规定只采用缓冲文件系统

C语言文件读写步骤

结构指针（书本p301查看FILE结构成员）

- 定义文件指针
  - ✓FILE *fp;
- 打开文件：文件指针指向buffer
  - ✓fp = fopen(“文件名”, “打开方式”);
- 读写操作
  - ✓fgetc(), fgets(), fscanf(), fread()
  - ✓fputc(), fputs(), fprintf(), fwrite()
  - ✓rewind(), fseek(), ftell(), feof(), ferror(), clearerr()
- 关闭文件
  - ✓fclose(文件指针)

文件指针

- 指向文件结构类型FILE的指针
- FILE结构类型包含了文件缓冲区、缓冲区工作指针、文件状态标志等成员，用来操作文件
- C语言用FILE指针来传递FILE类型的参数，进行文件操作
  - ✓FILE *fp;
  - ✓每次文件操作后，fp会自行移动到相应的位置

打开文件

- fopen("/config/workspace/myFile.txt", "r"); （Windows下： E:/data/myFile.txt）
  - ✓以只读方式打开/config/workspace文件夹下的myFile.txt
  - ✓文件打开成功则返回文件指针
  - ✓否则返回NULL

- 一般需要对文件打开的结果进行判断（不论哪种方式打开）

```
FILE *fp
if((fp = fopen("/config/workspace/myFile.txt", "r")) == NULL)
{
        printf("Fail to open the file!");
        exit(1);
}
```

以只读方式打开文件

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE *fp;
    if((fp = fopen("/config/workspace/myFile.txt", "r")) == NULL)
    {
        printf("Fail to open file \"myFile.txt\"!\n");
        exit(1);
    }
    return 0;
}
```

打开文件的方式

文本文件		二进制文件
打开方式	含义	相应的打开方式
r	只读方式打开（如不存在则报错）	rb
w	新建文本并只写（如存在则覆盖）	wb
a	打开文本并追加写（如不存在则新建）	ab
r+	读/写方式打开（如不存在则报错）	rb+
w+	新建文本并读/写（如存在则覆盖）	wb+
a+	打开文本并读/写/追加（如不存在则新建）	ab+

关闭文件

- fclose(文件指针);
  - ✓若成功则返回0
  - ✓否则表示无法正常关闭

```
if(fclose(fp))
{
    printf("Fail to close the file!");
    exit(0);
}
```

- 文件读写结束后一定要关闭（否则部分内容可能留在了缓冲区）
  - ✓在一开始就成对写fopen和fclose

成对编写文件打开和关闭

```c
FILE *fp;
if((fp = fopen("/config/workspace/myFile.txt", "r")) == NULL)
{
    printf("Fail to open file \"myFile.txt\"!\n");
    exit(1);
}

/*****文件操作*****/

if(fclose(fp))
{
    printf("Fail to close the file!");
    exit(2);
}
```

以字符方式进行文件读写

- ch=fgetc(fp): 每次读入一个字符
  - ✓成功则返回字符；失败则返回EOF
- fputc(ch, fp): 每次向文件写一个字符ch
  - ✓成功则返回字符；失败则返回EOF

- fgetc() Vs. getchar()
- fputc() Vs. putchar()

以字符方式进行文件读写

- ch=fgetc(fp): 每次读入一个字符
  - ✓成功则返回字符；失败则返回EOF
- fputc(ch, fp): 每次向文件写一个字符ch
  - ✓成功则返回字符；失败则返回EOF

- fgetc() Vs. getchar()
- fputc() Vs. putchar()

逐字符从一个文件读入到另一个文件

```
char ch;
while((ch = fgetc(fp)) != EOF)
    fputc(ch, fp2);
```

以字符串方式进行文件读写

- fgets(s, n, fp): 每次最多读入一个长度为n-1的字符串，并存入s中
  - ✓s为字符指针或者字符数组
  - ✓成功则返回字符串；失败则返回NULL
- fputs(s, fp): 每次向文件写一个字符串s
  - ✓s为字符指针、字符数组或字符串常量
  - ✓成功则返回最后一个字符；失败则返回EOF
- fgets() Vs. gets()（已经弃用）
- fputs() Vs. puts()

- fgets中字符串s形成条件
  - ✓ 读取指定字符数
  - ✓ 接收到换行符
  - ✓ 读到文件末尾
  - ✓ 换行符将被读入到s中
- 使用fgets(s, n, stdin)可以从键盘读取字符串！
- 使用fputs(s, stdout)可以在终端打印字符串！

以字符串方式进行文件读写

- fgets(s, n, fp): 每次最多读入一个长度为n-1的字符串，并存入s中
  - ✓s为字符指针或者字符数组
  - ✓成功则返回字符串；失败则返回NULL
- fputs(s, fp): 每次向文件写一个字符串s
  - ✓s为字符指针、字符数组或字符串常量
  - ✓成功则返回最后一个字符；失败则返回EOF
- fgets() Vs. gets()（已经弃用）
- fputs() Vs. puts()

从一个文件读入最大长度为length-1的字符串，写入到另一个文件

```
char str[100];
int length = 8;
fgets(str, length, fp);
fputs(str, fp2);
```

以格式化方式进行文件读写

- fscanf(fp, “%d”, &n): 从文件按照十进制整型格式读入到变量n；失败则返回EOF
- fprintf(fp, “%d”, n): 向文件按照十进制整型格式写入变量n；失败则返回一个负数

- fscanf() Vs. scanf()
- fprintf() Vs. printf()

以数据块方式进行文件读写（了解）

- 一般用于读写二进制文件，打开方式中加“b”
- fread(buffer, size, count, fp): 从fp对应的文件中将count*size个字节读入到buffer（size表示每次读入的数据块字节数）；返回值为实际读入的数据块数
- fwrite(buffer, size, count, fp): 将buffer中count*size个字节写到fp对应的文件中；返回值为写到文件中的数据块数
- 一般在了解文件格式的情况下使用（p314，例12-5）

其他文件操作函数（了解，自行尝试）

- rewind(fp)：重定位文件首，使fp回到文件打开时的位置
- fseek(fp, offset, from)：指针移动，使fp从from处移动offset个字节
  - ✓from为常量，取值为0, 1或2，分别表示文件开头，当前位置和文件末尾
- ftell(fp)：获取fp当前位置（相对于文件开头偏移字节数）
- feof(fp)：检测是否已到文件末尾
  - ✓一般文本文件用EOF判断结束，二进制文件用feof(fp)判断结束
- ferror(fp)：读写错误检查，如未出错则返回0
- clearerr(fp)：清除出错标志

小结

- 文件读写基本概念
  - ✓C语言文件读写的类型
  - ✓缓冲区的概念
- 文件读写的步骤
  - ✓打开
  - ✓读写：常用文件操作函数
  - ✓关闭
- 其他常用文件操作