

# C语言程序设计

The C Programming Language



---

## 第10章 文件

武汉光电国家研究中心

李春花



# 文件

---

- ◆ 存储在变量和数组中的数据是临时的，程序结束后都会消失
- ◆ 文件用来永久地保存大量数据
- ◆ 文件被存储在外存储设备中（如硬盘）
- ◆ 本章讨论怎样用C程序建立、更新、处理数据文件



# 主要内容

---

## ■ 文件的打开与关闭

- fopen、fclose、freopen函数

## ■ 文本文件的读写

- fgetc、fputc、fgets、fputs、fprintf、fscanf 等函数

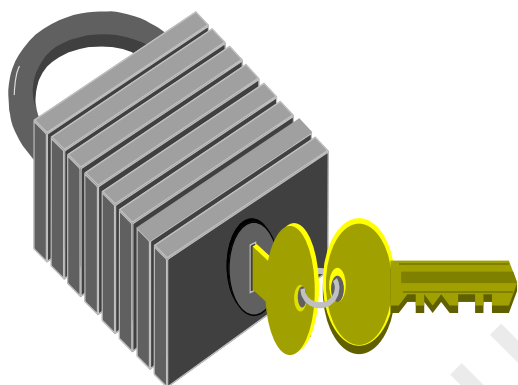
## ■ 二进制文件的读写

- fread、fwrite函数

## ■ 文件的随机读写

- fseek、rewind、ftell、fsetpos、fgetpos等文件指针定位函数

**问：**从键盘输入若干行字符，  
保存到d:\a.txt文件中，**该如何做？**



将输入写入  
文件中

# 程序

/\*从键盘输入若干行字符，保存到d:\a.txt文件中

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

```
int main( )
```

```
{
```

```
    FILE *fp;
```

```
    char ch;
```

```
    if ( (fp = fopen("d:\\a.txt","w") ) == NULL){
```

```
        printf("can't open the file!");
```

```
        return -1;
```

```
    }
```

```
    while((ch = getchar( ) ) != EOF)
```

```
        fputc(ch,fp);
```

```
    fclose(fp);
```

```
    return 0;
```

```
}
```

等价于: `ch = fgetc(stdin)`



# 字符读写函数fgetc和fputc

**int fgetc(FILE \*stream);**

从输入流stream当前位置读取一个字符，读写位置后移一个字符，返回读取的字符。到文件尾或读操作出错时返回EOF。

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

参数c转换为unsigned char类型然后写到输出流stream的当前位置处。返回值是被写字符；如果写操作出错或遇到文件尾返回EOF。

fgetc(stdin)	等价于	getchar()
fputc(c, stdout)	等价于	putchar()

# 读取文件中的内容

/\* 读d:\\a.txt文件中的内容，在屏幕上显示出来 \*/

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

```
int main( )
```

```
{
```

```
    FILE *fp;
```

```
    char ch;
```

```
    if ( (fp=fopen( "d:\\a.txt" , "r" ) ) == NULL) {
```

```
        printf("can't open the file!");
```

```
        return -1;
```

```
    }
```

```
    while((ch = fgetc (fp) ) != EOF)
```

```
        putchar(ch);
```

```
    return 0;
```

```
}
```

等价于: **fputc(ch, stdout);**

# 字符串读写函数

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

从stream流中读一行（行长度<n）或至多读n-1个字符到s指向的字符数组中。正常返回s，出错或遇文件尾返回NULL。

**char s[10];** // 输入: hust ✓

**fgets(s, 10, stdin);** // 换行符被读入

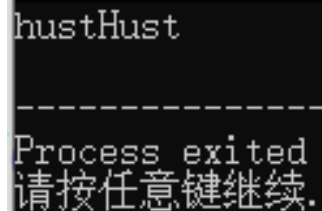
**gets(s);** // 换行符不被读入

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

将指针s指向的字符串写入stream流中，终结符不写。返回一个非负值；如写操作出错返回EOF。fputs函数与fgets配对使用。

**fputs("hust", stdout);** // 输出: hust

**puts("hust");** // 追加输出换行符: hust ✓



```
hustHust
-----
Process exited
请按任意键继续.
```





# 格式读写函数

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

将输出参数列表中的数据按指定的格式写入到stream流中。

写操作正常返回输出字符个数，写操作出错时返回负值。

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

从stream流中，按指定的格式读去数据，并赋值给相应的参数变量。函数返回已输入项数，如果读操作出错返回EOF。

printf("%d",x); 等价于 **fprintf(stdout, "%d", x);**

scanf("%d",&x); 等价于 **fscanf(stdin, "%d", x);**

# 文本文件的分解

- 将一个大的文本文件**以行为单位**分解成为若干个较小的文本文件，文件名和分解的行数都由用户从命令行输入。

```
int main(int argc, char *argv[ ]){ }
```

将文件abc.txt分解为a.txt, b.txt 和c.txt三个文件，分解的行数为10

**parts** abc. txt a. txt b. txt c. txt 10

假设程序名为parts

# 文本文件的分解

**parts** abc.txt a.txt b.txt c.txt 10

(1) len=命令行中分解的行数

atoi(argv[argc-1])

```
int main(int argc, char *argv[])
```

(2) 以读方式打开源文件

fin=fopen(argv[1], "r")

(3) 依次打开目标文件，从源文件读len行写入目标文件

for (i=2; i<argc-1; i++){

■ 以写方式打开文件 argv[i]

```
fout=fopen(argv[i], "w");
```

■ 从argv[1]读1行写入argv[i]直到写了len行或源文件到文件尾。

```
fgets(s, N, fin);  
fputs(s, fout);
```

■ 关闭 文件argv[i]

}

(4) 关闭源文件argv[1]

将一个大的文本文件以行为单位分解成为若干个较小的文本文件，文件名和分解的行数都由用户从命令行输入

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc,char *argv[])
```

```
{
```

```
FILE *fin,*fout;
```

parts abc.txt a.txt b.txt c.txt 10

```
int len = atoi(argv[argc-1]);/*将行数字符串转换为整数*/
```

```
int i,j;
```

```
char a[81];
```

```
if((fin = fopen(argv[1],"r")) == NULL){
```

```
    printf("can't open the %s file!\n",argv[1]);
```

```
    exit(-1);
```

```
}
```

```
for(i = 2; i < argc-1; i++){
```

```
    fout = fopen(argv[i],"w");/*打开argv[i]指定的文件进行写操作*/
```

```
    j=0;
```

```
    while((fgets(a,80,fin) != NULL) && j++<len)/*从fin中读一行到a中*/
```

```
        fputs(a,fout);/*将a中字符串写到fout中*/
```

```
    fclose(fout);/*写满len行后关闭文件*/
```

```
}
```

```
fclose(fin);
```

```
return 0;
```

```
}
```

# 数据采集和处理程序

从键盘输入商品信息（手工采集数据），采集商品名称、数量、单价，计算总金额

格式化读写函数

```
#include<stdio.h>
#include <stdlib.h>
void data_write(char *); /* 数据采集并存盘 */
void data_cal(char *, float ); /* 从文件读入数据并进行计算 */
int main(void)
{
    char a[20] = "d:\\goods.txt";
    data_write(a);
    data_cal(a);
    return 0;
}
```

# 输入数据并存盘

/\*从键盘输入商品名称、数量、单价，存于filename文件中 \*/

void data\_write(char \* filename)

{

FILE \*fout;

char name[20];

int number;

float price;

if((fout = fopen(filename,"w")) == NULL)

exit(-1);

printf("input name, number and price please!\n");

while( scanf("%s%d%f", name, &number, &price) != EOF ) {

fprintf(fout, "%s %d %f\n", name, number, price);

} 按指定格式向文件中写数据 fprintf()

fclose(out); 必须有空格, 便于能够正确进行读

}

# 从文件读数据并计算

/\*从filename文件中读入商品信息，计算商品的总金额 \*/

```
void data_cal(char * filename)
```

```
{
```

```
    FILE * in;
```

```
    char name[5];
```

```
    int number;
```

```
    float price;
```

```
    if((in = fopen(filename,"r")) == NULL)
```

```
        exit(-1);
```

从文件中按指定格式读取数据fscanf ()

```
    while( fscanf(in, "%s%d%f", name,&number,&price) != EOF)
```

```
        printf("%s\t%d\t%8.2f\n",name,number,price*number);
```

```
    fclose(in);
```

```
}
```



# 文件类型

---

文件按照数据格式分为**文本文件**和**二进制文件**两类。

## 文本文件（**ASCII文件**）

由字符的**ASCII**码组成的数据文件

## 二进制文件

内存表示的数据序列组成的数据文件



# 文本文件和二进制文件的存储

短整数 $x=128$ 以文本文件和二进制文件在磁盘上分别占多少字节？

内存:  $x$

0000 0000 1000 0000

文本文件

0011 0001 0011 0010 0111 0000

'1'

'2'

'8'

二进制文件

0000 0000 1000 0000



## 二进制文件的读写(数据块读写)

文件直接输入输出又称为文件成组输入输出。  
标准C为文件的直接输入输出提供了两个函数  
fread和fwrite, 适用于二进制形式文件的读写

```
size_t fread(void *ptr, size_t size, size_t n, FILE *stream);
```

```
size_t fwrite(const void *ptr, size_t size, size_t n, FILE *stream);
```

```
typedef unsigned int size_t;
```



# fread函数

`size_t fread(void *ptr, size_t size, size_t n, FILE *stream);`

从stream指向的输入文件中读取至多n个大小为size的记录到指针ptr指向的内存单元中。

返回值是实际读取记录数。

文件尾测试函数`feof()`，如果到文件尾，函数返回非0值，否则返回0。

`ferror()`测试出错函数，如果出错，`ferror`函数返回非0值，否则返回0。



# fwrite函数

`size_t fwrite(const void *ptr, size_t size, size_t n, FILE *stream);`

从指针ptr指向的内存缓冲区中取n条、大小为size的记录写到stream指向的输出文件中。

返回值是写出的记录数，当返回值小于n时，这种情况只有在写操作出错时出现。

参数的含义同fread，不同的是stream指向的输出文件。

## 例：将int x[]={12,8,34,421}写到磁盘文件

```
size_t fwrite(const void *ptr, size_t size, size_t n, FILE *stream);
```

```
fp=fopen("d:\\a.dat","wb"); //二进制方式
```

```
(1) fwrite(x, sizeof(int), 4, fp);
```

```
(2) for(i=0;i<4;i++)
```

```
    fwrite(x+i, sizeof(int), 1, fp); // fprintf(fp,"%d", x[i]);
```

## 从磁盘文件（二进制）读数据到内存

```
size_t fread(void *ptr, size_t size, size_t n, FILE *stream);
```

```
int i=0, x[10];
```

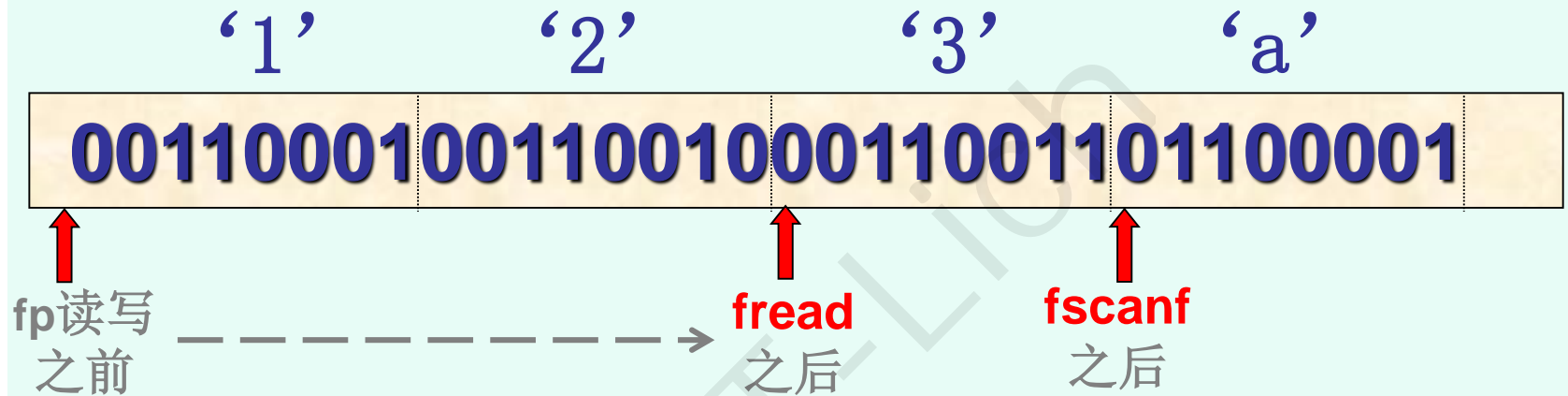
```
FILE *fp;
```

```
fp=fopen("d:\\a.dat","rb");
```

```
while( fread(x+i, sizeof(int), 1, fp) ==1) i++;
```

```
也可使用： fscanf(fp,"%d",&x[i]);
```

# 二进制文件数据不需要间隔符



```
short x;
```

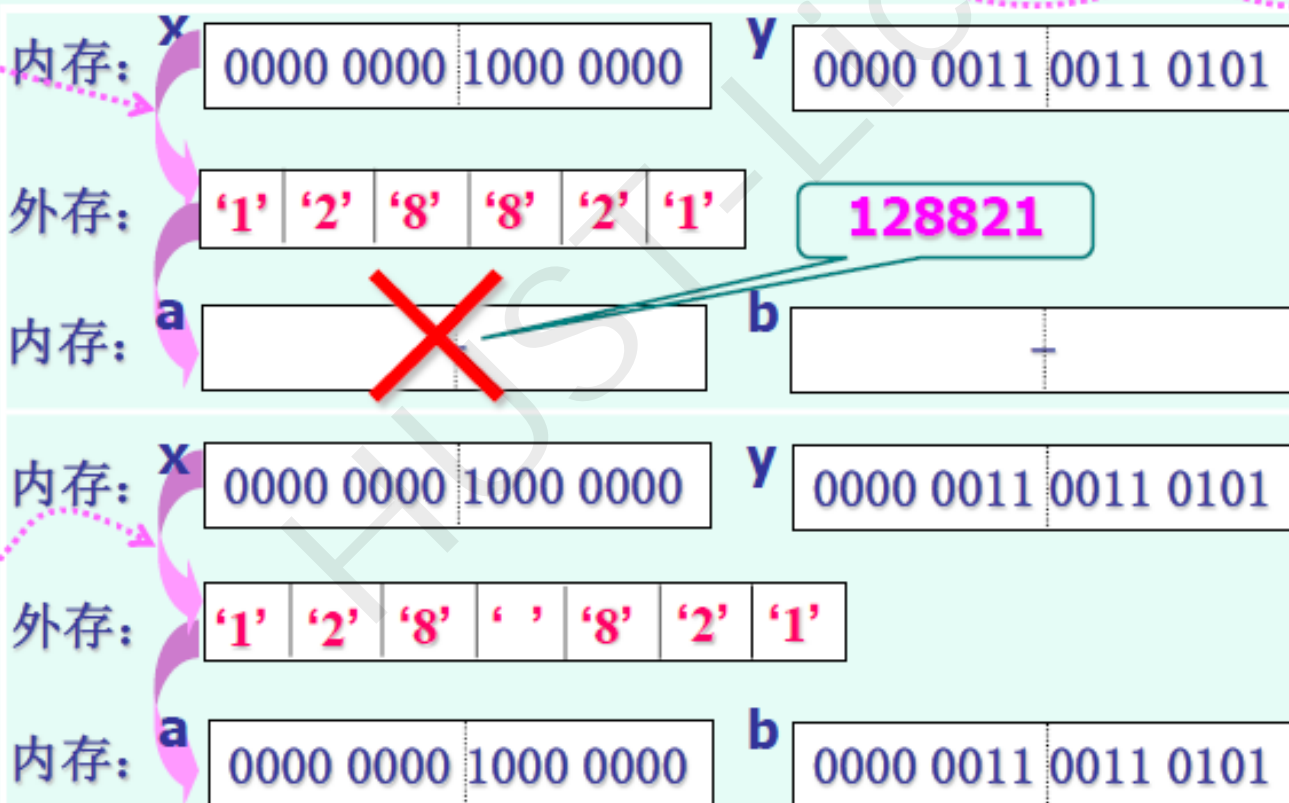
```
fread(&x, sizeof(short), 1, fp); //按二进制读入1个short数  
// x=0x3231
```

```
fscanf(fp, "%hd", &x); //按文本格式读入1个short数  
// x=123
```

# 二进制文件数据不需要间隔符

如果当将内存多个数据写入文本文件的时候，特别注意的是：在两个数据之间增加必要的**间隔符**。目的在于读文本文件时，能够保证获得数据转换的正确性。

例如，`short x=128,y=821,a,b`；格式：“%hd%hd”或“%hd %hd”



# 结构化数据的读写

## 将例9.5中的物品信息写入磁盘文件

```
#include<stdio.h>
#include<stdlib.h>
/* 声明物品信息结构类型struct goods */
struct goods {
    long code;           /* 货物编码 */
    char name[20];       /* 名称 */
    float price;         /* 价格 */
};

void data_write(const char *filename);
void data_read(const char *filename);

int main(void)
{
    char fname[]="goods_table.dat";
    data_write(fname);
    data_read(fname);
    return 0;
}
```



# 输入商品信息写入磁盘文件

```
void data_write(const char *filenameme)
{
    struct goods g; /*声明 struct goods 类型结构变量 */
    FILE *out;

    if((out = fopen(filenameme,"wb")) == NULL)
        exit(-1);
    printf("输入货物编码、名称、价格\n");
    while(scanf("%ld%s%f",&g.code,g.name,&g.price)==3) {
        fwrite(&g,sizeof(struct goods),1,out);
    }
    fclose(out);
}
```

# 从磁盘读取商品信息显示到屏幕

```
void data_read(const char *filenamme)
{
    struct goods g; /*声明 struct goods 类型结构变量 */
    FILE *in;
    if((in = fopen(filenamme,"rb")) == NULL)
        exit(-1);
    printf("货物编码\t名称\t价格\n");
    fread(&g,sizeof(struct goods),1,in);
    while(!feof(in)){
        printf("%ld\t%s\t%f\n", g.code,g.name,g.price);
        fread(&g,sizeof(struct goods),1,in);
    }
    fclose(in);
}
```



# 文件尾测试函数feof()

文件尾测试函数feof()，如果到文件尾，函数返回非0值，否则返回0。

```
#define _IOEOF 0x0010
```

```
#define feof(_stream) ((_stream) ->_flag & _IOEOF)
```

## 文件尾测试函数feof( )      #define \_IOEOF 0x0010

文件尾测试函数feof(), 如果到文件尾, 函数返回非0值, 否则返回0。

从二进制文件读数据显示到屏幕

```
int x;  
FILE *fp;  
fp=fopen("a.dat","rb");  
fread(&x, sizeof(int), 1, fp);  
while(!feof(fp)){  
    printf("%d", x);  
    fread(&x, sizeof(int), 1, fp);  
}
```

只有当文件位置指针到了文件末尾, 然后再发生读/写操作时, 标志位(fp->\_flag)才会被置为含有\_IOEOF

所以, 要先读, 再 feof

```
size_t fread(void *ptr, size_t size, size_t n, FILE *stream);
```

文件尾测试函数feof( )      #define \_IOEOF 0x0010

文件尾测试函数feof(), 如果到文件尾, 函数返回非0值, 否则返回0。

从二进制文件读数据显示到屏幕

```
int x;  
FILE *fp;  
fp=fopen("a.dat","rb");  
//fread(&x, sizeof(int), 1, fp);  
while(!feof(fp)){      最后1个数显示2次  
    fread(&x, sizeof(int), 1, fp);  
    printf("%d", x);  
}
```

```
size_t fread(void *ptr, size_t size, size_t n, FILE *stream);
```

文件尾测试函数feof( )     #define \_IOEOF 0x0010

文件尾测试函数feof(), 如果到文件尾, 函数返回非0值, 否则返回0。

从二进制文件读数据显示到屏幕

```
int x, j=0;
FILE *fp;
fp=fopen("a.dat","rb");
while(!feof(fp)){
    j=fread(&x, sizeof(int), 1, fp);
    if(j==1) printf("%d\n", x);
}
fclose(fp);
```

```
size_t fread(void *ptr, size_t size, size_t n, FILE *stream);
```

## 3.6 文件的随机读/写

实现随机读写的关键是要按要求移动位置指针，这称为文件的定位。

### 文件的定位函数

```
int fseek(FILE *stream, long offset, int origin);  
long ftell(FILE *stream);  
int fgetpos(FILE *stream, fpos_t *pos);  
int fsetpos(FILE *stream, const fpos_t *pos);  
void rewind(FILE *stream);
```

对二进制格式文件和文本格式文件，均可使用定位函数。

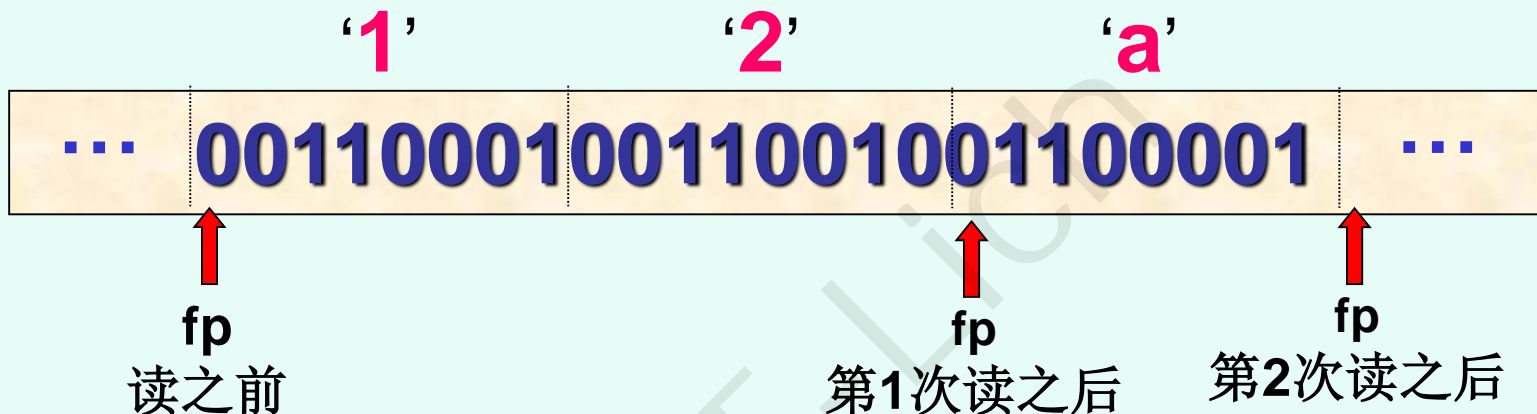


# 文件的读写方式

- ◆ 文件的读写方式有两种：**顺序读写**和**随机读写**。
- ◆ 打开文件时，读写指针指向文件头；读写一个“数据”后，读写指针自动指向下一个“数据”。
- ◆ **顺序读写**：从文件头到文件尾顺序读写数据。
- ◆ **随机读写**：对文件的读写可以从文件内指定的位置进行，而不必每次从头顺序开始。



# 顺序读写



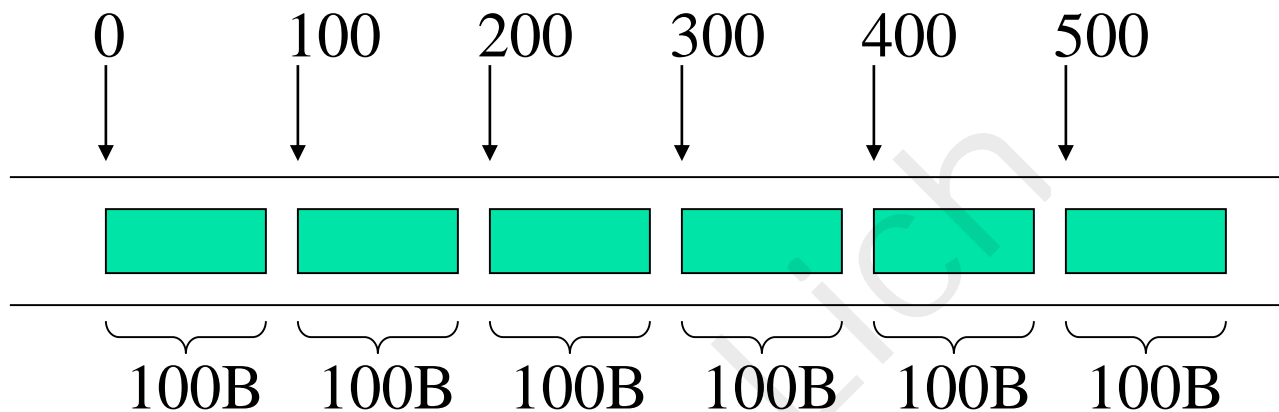
按文本格式，读入1个short 数据和1个char 数据

```
fscanf(fp,"%hd",&x); // x:12
```

```
fscanf(fp,"%c",&x); // x= 'a'
```

文本文件只能顺序读写，因为其数据不是定长的。

# 随机读写



具有定长记录的随机存取文件

- 二进制文件数据具有同样的长度，**二进制文件能顺序读写，也能随机读写。**
- 利用文件的定位函数和文件的读写函数，即可实现文件的随机读写。



# 文件定位函数

---

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

**long ftell(FILE \*stream);**

**int fgetpos(FILE \*stream, fpos\_t \*pos);**

**int fsetpos(FILE \*stream, const fpos\_t \*pos);**

**void rewind(FILE \*stream);**



# 其它文件操作函数

---

**int fflush(FILE \*stream);**

**int setvbuf(FILE \*stream, char \*buf, int mode, size\_t size);**

**void setbuf(FILE \*stream, char \*buf);**

**int remove(const char \* filename);**

**int rename(const char \* oldname, const char \* newname);**

**FILE \* tmpfile(void);**

**char \* tmpnam(char \*s);**

**void clearerr(FILE \*stream);**

**int ferror(FILE \*stream);**

**void perror(const char \*s);**