11

流与文件操作


E
SAVE


文件
开始
插入
设计
布局
引用
邮件
审阅
视图
告诉我您想要做什么...
剪切

11.1 概述

- **文件**：存储在外部介质上的数据的集合，是操作系统数据管理的单位

使用数据文件的目的

1、数据文件的改动不引起程序的改动——程序与数据分离

2、不同程序可以访问同一数据文件中的数据——数据共享

3、能长期保存程序运行的中间数据或结果数据

文件分类

按文件的逻辑结构：

- 记录文件：由具有一定结构的记录组成（定长和不定长）
- 流式文件：由一个个字符（字节）数据顺序组成

按存储介质：

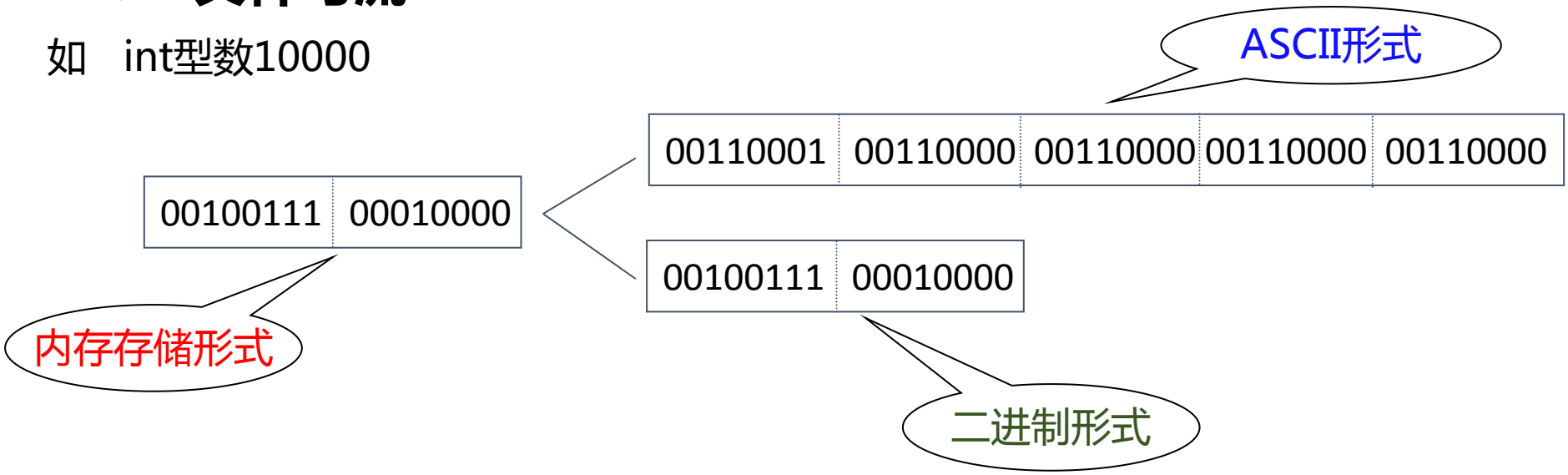
- 磁盘（普通）文件：存储介质文件（磁盘、磁带等）
- 设备文件：非存储介质（键盘、显示器、打印机等）

按数据的组织形式：

- 文本文件： ASCII文件，每个字节存放一个字符的ASCII码
- 二进制文件：数据按其在内存中的存储形式原样存放

11.2 文件与流

如　int型数10000



- 文本文件特点：存储量大、速度慢、便于对字符操作
- 二进制文件特点：存储量小、速度快、便于存放中间结果

文件分类

按文件的结构形式：

- 文本文件： ASCII文件，每个字节存放一个字符的ASCII码
- 二进制文件：数据按其在内存中的存储形式原样存放

按文件的读写形式：

- 顺序存取文件： 从文件头开始，以字符为单位读写数据
- 随机存取文件：随机从文件内指定的位置读写数据

按文件的存储介质：

- 磁盘（普通）文件：存储介质文件（磁盘、磁带等）
- 设备文件：非存储介质（键盘、显示器、打印机等）

按文件的处理方式：

- 缓冲文件系统：由具有一定结构的记录组成（定长和不定长）
- 非缓冲文件系统：由一个个字符（字节）数据顺序组成

按数据流动方向

- 输入文件：将数据从文件读入到内存中
- 输出文件：将数据从内存写入到文件中

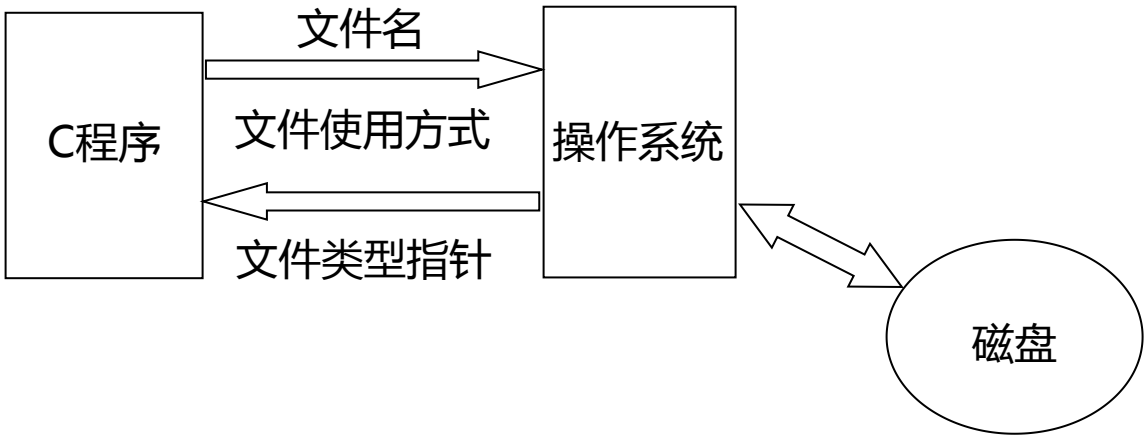
11.3 文件类型与文件类型指针指针

文件结构体FILE

- 缓冲文件系统为每个正使用的文件在
- 文件信息用系统定义的名为FILE的结
- FILE定义在stdio.h中
- 指针变量说明： FILE *fp;
- 用法：

```
typedef  struct{
    int  _fd;          //文件号
    int  _cleft;       //缓冲区中剩余的字符
    int  _model;       //文件操作模式
    int  *_nextc;      //下一个字符位置
    int  *_buff;       //文件缓冲区位置
}FILE;
```

- 文件打开时，系统自动建立文件结构体，并把指向它的指针返回来，程序通过这个指针获得文件信息,访问文件
- 文件关闭后，它的文件结构体被释放



11.4 文件的打开与关闭

- C文件操作用库函数实现,包含在stdio.h

文件使用方式	
“r/rb” (只读)	为输入打
“w/wb” (只写)	为输出打
“a/ab” (追加)	向文本/二
“r+/rb+” (读写)	为读/写打
“w+/wb+” (读写)	为读/写建立一个文本/二进制文件
“a+/ab+” (读写)	为读/写打开或建立一个文本/二进制文件

例 文件打开与测试

```
FILE *fp;
if((fp=fopen(“test.c”,“w”))==NULL)
{   printf(“File open error!\n”);
    exit(0);
}
```

- 返值：正常打开，为指向文件结构体的指针，打开失败，为NULL

例

```
FILE *fp;
fp= fopen (''test.dat'',''r'');
```

例

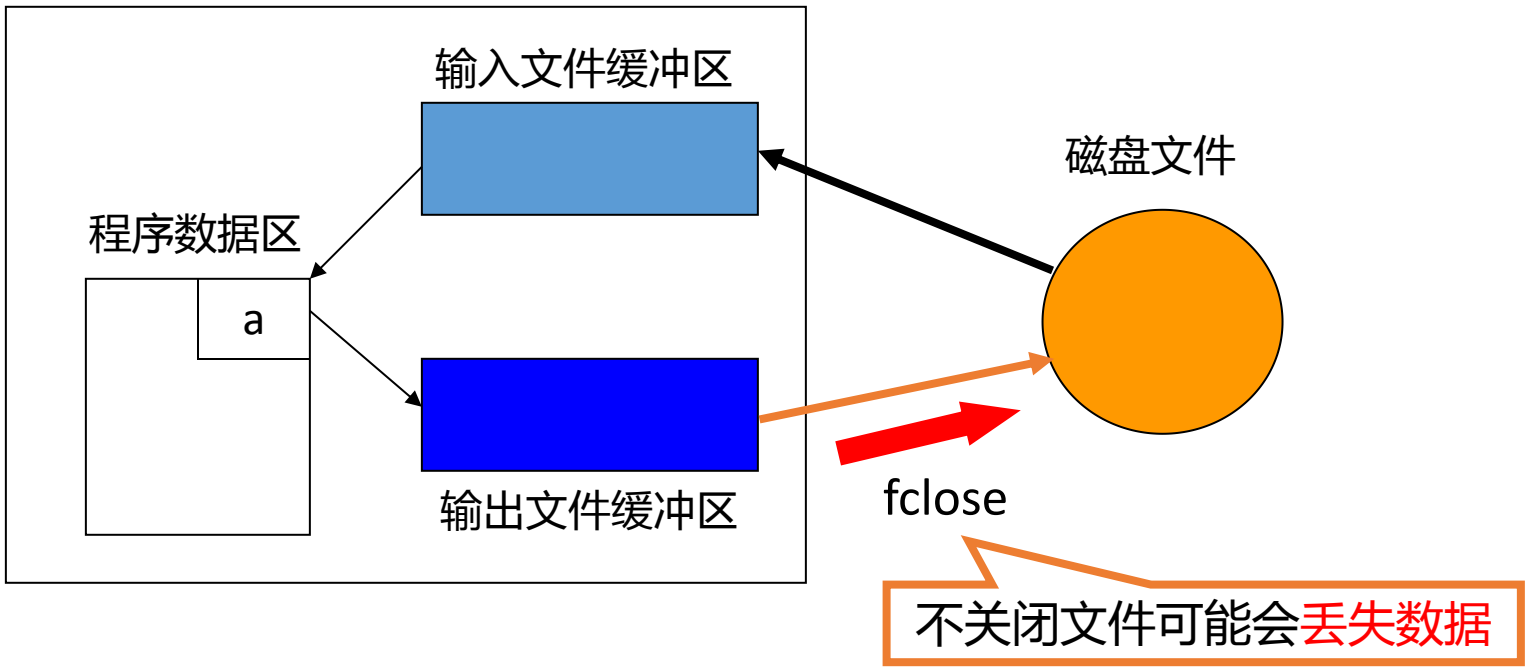
```
FILE *fp;
char *filename=''test.dat'';
fp= fopen(filename, ''r'');
```

文件关闭fclose

- 作用:使文件指针变量与文件“脱钩”，释放文件结构体和文件指针
- 函数原型：int fclose(FILE *fp)
- 功能：关闭fp指向的文件
- 返值：正常关闭为0;出错时,非0

文件打开时返回的文件类型指针

缓冲文件系统：



不关闭文件可能会丢失数据

11.5

格式化

- 函数

- 返值

例 从键

```
#include <stdio.h>
#include<stdlib.h>
main()
{ char s[80],c[80];
  int a,b;
  FILE *fp;
  if((fp=fopen("test.txt","w"))==NULL)
  {  printf("can't open file\n");   exit(0) ;   }
  fscanf(stdin,"%s%d",s,&a);  /*read from keaboard*/
  fprintf(fp,"%s  %d",s,a);  /*write to file*/
  fclose(fp);
  if((fp=fopen("test.txt","r"))==NULL)
  {  printf("can't open file\n"); exit(0);   }
  fscanf(fp,"%s%d",c,&b);  /*read from file*/
  fprintf(stdout,"%s %d",c,b);  /*print to screen*/
  fclose(fp);
}
```

数据块（二进制形式）读写：fread 与 fwrite

- 函数原型：

  size_t fread(void *buffer , size_t size , size_t count , FILE *fp)

  size_t fwrite(void *buffer , size_t size , size_t count , FILE *fp)

- 功能：读/写数据块
- 返值：成功，返回读/写的块数；出错或文件尾，返回0
- 说明：
  - typedef unsigned size_t;
  - buffer: 指向要输入/输出数据块的首地址的指针
  - size: 每个要读/写的数据块的大小（字节数）
  - count: 要读/写的数据块的个数
  - fp: 已打开文件的地址
  - fread与fwrite一般用于二进制文件的输入/输出

例 float f[2];
FILE *fp;
fp=fopen(“aa.dat”,“rb”);
fread(f,sizeof(float),2,fp);

例 for(i=0;i<2;i++)
fread(&f[i],sizeof(float),1,fp);

例 struct student
{ int num;
char name[20];
char sex;
int age;
float score[3];
}stud[10]={ ... ... };
for(i=0;i<10;i++)
fwrite(&stud[i],sizeof(struct student),1,fp);

例 输入3位同学的姓名、学号和住址，将数据存盘再讲数据读出并显示。

```c
#include <stdio.h>
#include<stdlib.h>
struct student_type
{     char name[10];
      int num;
      char addr[15];
}stud[3];
void main()
{
      void save(void);
      void display(void);
      int i;
      for(i=0;i<3;i++){
         printf("第%d个学生姓名: ",i+1);
         scanf("%s",stud[i].name);
         printf("第%d个学生学号: ",i+1);
         scanf("%d",&stud[i].num);
         printf("第%d个学生籍贯: ",i+1);
         scanf("%s",stud[i].addr);
      }
      save();
      display();
}
```

```c
void save()
{    FILE *fp;
     int  i;
     if((fp=fopen("student_1.dat","w"))==NULL)
     { printf("cannot open file\n");
       exit(0);
     }
     for(i=0;i<3;i++)
     if(fwrite(&stud[i],sizeof(struct student_type),1,fp)!=1)
         printf("file write error\n");
      fclose(fp);
}
```

```c
void display()
{    FILE *fp;
     int  i;
     if((fp=fopen("student_1.dat","r"))==NULL)
     { printf("cannot open file\n");
       exit(0);
     }
     for(i=0;i<3;i++)
     { fread(&stud[i],sizeof(struct student_type),1,fp);
       printf("%s\t%d\t%s\n",stud[i].name,stud[i].num,stud[i].addr);
     }
     fclose(fp);
}
```

不使用全局数组

例 输入3位同学的姓名、学号和住址，将数据存盘再讲数据读出并显示。

```c
#include <stdio.h>
#include<stdlib.h>
struct student_type{
     char name[10];
     int num;
     char addr[15];
    };
main()
{
     void save(struct student_type *q);
     void display(struct student_type *q);
     struct student_type stud[3];
     int i;
     struct student_type *p=stud;
     for(i=0;i<3;i++,p++){
        printf("第%d个学生姓名: ",i+1);
        scanf("%s",p->name);
        printf("第%d个学生学号: ",i+1);
        scanf("%d",&p->num);
        printf("第%d个学生籍贯: ",i+1);
        scanf("%s",p->addr);
     }
     p=stud;
     save(p);
     display(p);
}
```

```c
void save(struct student_type *q)
{    FILE *fp;
     int  i;
     if((fp=fopen("student_2.dat","w"))==NULL)
     { printf("cannot open file,save failed\n");
       exit(0);
     }
     for(i=0;i<3;i++,q++)
     if(fwrite(q,sizeof(struct student_type),1,fp)!=1)
         printf("file write error\n");
     printf("保存成功! \n");
     fclose(fp);
}
```

```c
void display(struct student_type *q)
{    FILE *fp;
     int  i;
     if((fp=fopen("student_2.dat","r"))==NULL)
     { printf("cannot open file,display failed\n");
       exit(0);
     }
     for(i=0;i<3;i++,q++)
     { fread(q,sizeof(struct student_type),1,fp);
       printf("%s\t%d\t%s\n",q->name,q->num,q->addr);
     }
     fclose(fp);
}
```

将建好的链表存入文件

```
void save(struct person *head)
{
    struct person *p=head;
    FILE *fp;
    char fname[50];
    printf("\n\n请输入要存储数据的文件名:");
    fflush(stdin);
    gets(fname);
    if((fp=fopen(fname,"wb"))==NULL){
        printf("文件无法打开，存储失败!\n\n");
        exit(0);
    }
    while(p!=NULL){
        fwrite(p,sizeof(struct person),1,fp);
        p=p->next;
    }
    fclose(fp);
    printf("存储成功! \n\n");
}
```

从文件中读取数据建立链表

```c
struct person *load()
{
    struct person *p,*s,*head=NULL;
    FILE *fp;
    char fname[50];
    printf("\n\n请输入要读取数据的文件名:");
    fflush(stdin);
    gets(fname);
    if((fp=fopen(fname,"rb"))==NULL){
        printf("文件无法打开，读取失败！\n\n");
        return;
    }
    while(!feof(fp)){
        if(head==NULL){
            p=(struct person *)malloc(sizeof(struct person));
            fread(p,sizeof(struct person),1,fp);
            if(feof(fp))//读取结束跳出循环
            {
                printf("文件为空！\n\n");
                return NULL;
            }
            head=s=p;
            continue;
        }
        p=(struct person *)malloc(sizeof(struct person));
        fread(p,sizeof(struct person),1,fp);
        if(feof(fp))//读取结束跳出循环
        {
            s->next = NULL;
            break;
        }
        s->next=p;
        s=p;
        p->next=NULL;
    }
    fclose(fp);
    printf("读取成功！\n\n");
    return(head);
}
```

feof

- 函数原型： int feof(FILE *fp)
- 功能：判断文件是否结束
- 返值：文件结束，返回真（非0）；文件未结束，返回0
- 判断二进制文件是否结束

```
while(!feof(fp))
{   c=fgetc(fp);
    ……
}
```

```c
#include<stdio.h>
main()
{
    FILE *fp;
    if((fp=fopen("test.dat","r"))==NULL)
    { printf("cannot open file\n");
      exit(0);
    }
    if (feof(fp))
    {
        printf("文件为空。。");
    }
    else
    {
        printf("文件不为空。。");
    }

}
```

字符读写：fgetc 与 fputc

- 函数原型：

  int fgetc(FILE *fp)

  int fputc(int c , FILE *fp)

- 功能：从fp指向的文件中读取一字节代码/把一字节代码c写入fp指向的文件中
- 返值：成功，返回读取/写入字符的ASCII码值，出错返回-1（EOF）

文件I/O与终端I/O	
putchar(c)	fputc(c,stdout)
getchar( )	fgetc(stdin)

例11.4 从键盘输入字符，逐个存到磁盘文件中，直到输入'#'为止

```
#include <stdio.h>
main()
{  FILE *fp;
   char ch;
   if((fp=fopen("out.txt","w"))==NULL)
   {   printf("cannot open file\n");
       exit(0);
   }
   printf("input string:");
   while((ch=getchar())!='#')
   {   fputc(ch,fp);
       putchar(ch);
   }
   fclose(fp);
}
```

例 读文本文件内容，并显示

```
#include <stdio.h>
main()
{  FILE *fp;
   char ch;
   if((fp=fopen("out.txt", "r"))==NULL)
   {   printf("cannot open file\n");
       exit(0);
   }
   while((ch=fgetc(fp))!=EOF)
      putchar(ch);
   fclose(fp);
}
```

例 11.5 实现在DOS环境下运行的文件拷贝命令　copy　1.txt　2.txt

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc,char **argv)
{
   void filecopy(FILE *fp1,FILE *fp2);
   FILE *fp1,*fp2;
   if(argc>1)
   {
      if((fp1=fopen(*++argv,"r"))==NULL){
         printf("can\'t open %s\n",*argv);
         exit(0);}
      if((fp2=fopen(*++argv,"w"))==NULL){
         printf("can\'t open %s\n",*argv);
         exit(0);}
   }
   filecopy(fp1,fp2);
}
void filecopy(FILE *fp1,FILE *fp2)
{  char c;
   while((c=fgetc(fp1))!=EOF)
      fputc(c,fp2);
}
```

复制文本文件

复制二进制文件

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc,char **argv)
{
   void filecopy(FILE *fp1,FILE *fp2);
   FILE *fp1,*fp2;
   if(argc>1)
   {
      if((fp1=fopen(*++argv,"rb"))==NULL){
         printf("can\'t open %s\n",*argv);
         exit(0);}
      if((fp2=fopen(*++argv,"wb"))==NULL){
         printf("can\'t open %s\n",*argv);
         exit(0);}
   }
   filecopy(fp1,fp2);
}
void filecopy(FILE *fp1,FILE *fp2)
{  char c;
   while(!feof(fp1)){
      c=fgetc(fp1);
      fputc(c,fp2);
}
```

字符串读写：fgets 与 fputs

- 函数原型：

```
char  *fgets(char  *s,int n,FILE  *fp)
int   fputs(char  *s,FILE  *fp)
```

- 功能：fgets从fp所指文件读n-1个字符送入s指向的内存区,并在最后加一个'\0'（若读入n-1个字符前遇换行符或文件尾(EOF)即结束）

  fputs把s指向的字符串写入fp指向的文件

- 返值：fgets正常返回读取字符串的首地址；出错或文件尾，返回NULL

  fputs正常返回非负整数；出错为EOF

例　从键盘读入字符串存入文件，再从文件读回显示

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
main()
{   FILE  *fp;
    char  string[81];
    if((fp=fopen("file.txt","w"))==NULL)
    {  printf("can't open file");exit(0); }
    while(strlen(gets(string))>0)
    {  fputs(string,fp);
       fputs("\n",fp);}
    fclose(fp);
    if((fp=fopen("file.txt","r"))==NULL)
    {  printf("can't open file");exit(0); }
    while(fgets(string,100,fp)!=NULL)
       fputs(string,stdout);
    fclose(fp);
 }
```

11.5 文件的定位与随机读写

- 文件位置指针-----指向当前读写位置的指针
- 读写方式
  - 顺序读写：位置指针
  - 随机读写：位置指针

rewind函数

- 函数原型 void rewind
- 功能：重置文件位置指

例 对一个磁盘文件进行显

```
#include <stdio.h>
main()
{   FILE *fp1,*fp2;
    fp1=fopen("test_1.c","r");
    fp2=fopen("test_2.c","w");
    while(!feof(fp1))  putchar(fgetc(fp1));
    rewind(fp1);
    while(!feof(fp1))  fputc(fgetc(fp1),fp2);
    fclose(fp1);
    fclose(fp2);
}
```

fseek函数

- 函数原型 int fseek(FILE *fp,long offset,int origin)
- 功能：定位文件指针的位置
- offset：位移量（从起始点移动的字节数）正数向后移动，负数向前移动
- origin：起始点设定　文件开始　SEEK_SET　0
  文件当前位置　SEEK_CUR　1
  文件末尾　SEEK_END　2

ftell函数

- 函数原型 long ftell(FILE *fp)
- 功能：返回位置指针当前位置(用相对文件开头的位移量表示)
- 返值：成功，返回当前位置指针位置；失败，返回-1

11.5 文件出错检测

ferror函数

- 函数原型： int ferror(FILE *fp)
- 功能：测试文件是否出现错误
- 返值：未出错，0；出错，非0
- 说明
  - 每次调用文件输入输出函数，均产生一个新的ferror函数值，所以应及时测试
  - fopen打开文件时，ferror函数初值自动置为0

clearerr函数

- 函数原型： int clearerr(FILE *fp)
- 功能：使文件错误标志置为0
- 说明：出错后，错误标志一直保留，直到对同一文件调clearerr(fp)或rewind或任何其它一个输入输出函数