CHƯƠNG 3 : NHẬP VÀ XUẤT DỮ LIỆU

§1. KHÁI NIỆM CHUNG

1. Khái niệm :Trước đây chúng ta đã xét việc nhập dữ liệu từ bàn phím. Trong nhiều trường hợp thực tế , để thuận lợi , chúng ta phải nhập dữ liệu từ các tập tin trên đĩa . Các hàm thư viện của C cho phép truy cập tập tin và chia là 2 cấp khác nhau :

- các hàm cấp 1 là các hàm ở cấp thấp nhất , truy cập trực tiếp đến các tập tin trên đĩa.C không cung cấp vùng nhớ đệm cho các hàm này
- các hàm cấp 2 là các hàm truy xuất tập tin cao hơn , do chúng được C cung cấp vùng nhớ đệm

Đối với các hàm cấp 1 , tập tin được xem là khối các byte liên tục do đó khi muốn truy cập mẫu tin cụ thể thì phải tính toán địa chỉ của mẫu tin và như vậy công việc vất vả hơn . Ngoài ra phải cung cấp vùng nhớ đệm cho kiểu đọc ghi này. Đối với các hàm cấp hai công việc nhẹ nhàng hơn do :

- trình biên dịch tự động cung cấp vùng kí ức đệm cho chúng
- có thể truy xuất các mẫu tin mà không gặp khó khăn như với các hàm cấp 1

Trong C , các thông tin cần thiết cho các hàm xuất nhập cấp 2 được đặt trong tập tin stdio.h còn các thông tin về hàm nhập xuất cấp 1 thì ở trong tập tin io.h

2. Stream và các tập tin : Ta phải phân biệt hai thuật ngữ là stream và file .Hệ thống xuất nhập của C cung cấp một không gian tưởng tượng giữa người lập trình và các thiết bị được dùng . Cấp trung gian tưởng tượng này gọi là stream và thiết bị cụ thể là tập tin .

a. Các streams : Trong máy tính ta dùng 2 loại stream : văn bản và nhị phân . Một stream văn bản là một loạt kí tự được tổ chức thành dòng mà mỗi dòng được kết thúc bằng kí tự xuống dòng newline("\n") . Khi ghi , một kí tự chuyển dòng LF(mã 10) đucợ chuyển thành 2 kí tự CR(mã 13) và LF . Khi đọc 2 kí tự liên tiếp CR và LF trên tập tin chỉ cho ta một kí tự LF .

Một stream nhị phân là một loạt các byte .

a. Các tập tin : Trong C ,một tập tin là một khái niệm logic mà hệ thống có thể áp dụng cho mọi thứ từ các tập tin trên đĩa cho đến các terminal . Khi bắt đầu thực hiện chương trình , máy tính mở 3 stream văn bản đã được định nghĩa trước là stdin , stdout và stderr . Đối với hầu hết các hệ thống , các thiết bị này là console

§2. NHẬP XUẤT CHUẨN

1. Nhập xuất kí tự , chuỗi kí tự , định dạng và bản ghi : Nhập xuất cấp 2(nhập xuất chuẩn) cung cấp 4 cách đọc và ghi dữ liệu khác nhau (ngược lại nhập xuất cập1 chỉ dùng 1 trong 4 cách này) .

Trước hết dữ liệu có thể đọc ghi mỗi lần một kí tự , tương tự như cách làm việc của putchar() và getche() để đọc dữ liệu từ bàn phím và hiển thị lên màn hình .

Thứ hai , dữ liệu có thể nhập xuất theo chuỗi bằng các dùng các hàm gets() và puts()

Thứ ba , dữ liệu có thể được nhập và xuất theo khuôn dạng bằng các hàm fprintf() và fscanf()

Thứ tư , dữ liệu được đọc và ghi theo khối có chiều dài cố định thường dùng lưu trữ mảng hay cấu trúc bằng các hàm fread() và fwrite() . Tóm lại :

Các hàm dùng chung cho hai kiểu nhị phân và văn bản

fopen : dùng mở tập tin

fclose : đóng tập tin
fclose : đóng tất cả các tập tin
fflush : dùng làm sạch vùng đệm của tập tin
flushall : dùng làm sạch vùng đệm của tất cả tập tin
ferror : cho biết có lỗi (khác không) hay không có lỗi (bằng 0)
perror : thong báo lỗi trên màn hình
foef : cho biết cuối tập tin hay chưa
unlink và remove : dùng để loại tập tin trên đĩa
fseek : di chuyển con trỏ đến vị trí bất kì trên tập tin
ftell : cho biết vị trí hiện tại của con trỏ

Các hàm nhập xuất kí tự

putc và fputc : nhập kí tự vào tập tin
getc và fgetc : đọc kí tự từ tập tin
fprintf : dùng ghi dữ liệu định dạng lên tập tin
fscanf : dùng đọc dữ liệu định dạng từ tập tin
fputs : dùng ghi chuỗi lên tập tin
fgets : dùng đọc chuỗi từ tập tin

Các hàm dùng cho kiểu xuất nhập nhị phân

putw : dùng ghi một số nguyên hai byte lên tập tin
gets : dùng đọc một số nguyên hai byte từ tập tin
fwrite : dùng ghi một mẫu tin lên tập tin
fread : dùng đọc một mẫu tin từ tập tin

2.Dạng văn bản và dạng nhị phân : Cách khác để phân loại các thao tác nhập xuất tập tin là nó được mở theo kiểu văn bản hay nhị phân . Điểm khác biệt giữa hai loại này là kí tự newline và end of line . Điểm thứ hai để phân biệt hai kiểu tập tin là là cách lưu trữ các số vào đĩa . Đối với dạng văn bản thì các số được lưu trữ thành chuỗi các kí tự còn dạng nhị phân thì các số được lưu như trong bộ nhớ , nghĩa là dùng hai byte cho một số nguyên và 4 byte cho một số float .

3. Nhập xuất chuẩn : Chương trình dùng các hàm nhập xuất cấp 2 thường dễ hiểu hơn nên chúng ta sẽ nghiên cứu trước .

a. Nhập xuất kí tự : Để nhập kí tự vào tập tin ta dùng hàm putc() hay fputc().Để đọc kí tự từ tập tin ta dùng hàm getc() hay fgetc() . Chương trình ví dụ này là tạo lập các kí tự bằng cách gõ vào bàn phím mỗi lần một kí tự và ghi vào một tập tin trên đĩa . Chương trình dùng hàm fopen() để mở một tập tin , dùng hàm putc() để ghi lên tập tin , dùng kí tự enter để kết thúc chương trình .

Chương trình 3-1 :

```
#include <stdio.h>
#include <conio.h>
void main()
  {
    FILE *fp;
    char ch;
    printf("Nhap cac ki tu : ");
    fp=fopen("textfile","w");
    while ((ch=getche())!='\r')
      putc(ch,fp);
    fclose(fp);
  }
```

b. Mở một tập tin : Trước khi ghi một tập tin lên đĩa ta phải mở tập tin đó đã . Để mở tập tin , trước hết ta phải khai báo một con trỏ chỉ tới FILE . FILE là một structure chứa đựng các thông tin về cấu trúc của tập tin ví dụ như kích thước , vị trí của bộ đệm dữ liệu hiện hành . Cấu trúc FILE được khai báo trong stdio.h nên ta cần include tập tin này . Ngoài ra stdio.h còn xác định các tên và các biến khác được dùng trong chương trình hướng đến các tập tin . Do vậy trong chương trình ta có câu lệnh :

FILE *fp ;

Sau đó ta mở tập tin bằng lệnh :

fopen("textfile","w");

Khi viết như vậy sẽ làm cho hệ điều hành biết là mở một tập tin tên là textfile trong thư mục hiện hành để viết lên tập tin đó (nhờ "w") . Ta có thể cho tên đường dẫn đầy đủ nếu muốn mở tập tin ở thư mục bất kì . Hàm fopen() trả về một con trỏ chỉ đến cấu trúc FILE cho tập tin và con trỏ này được cất giữ trong biến fp . Chuỗi "w" được gọi là kiểu , nó có nghĩa là ghi lên tập tin . Các kiểu mở tập tin là :

- "r","rt" mở để đọc , tập tin phải có trên đĩa
- "w","wt" mở để ghi , nếu trên đĩa đã có tập tin thì nội dung bị ghi đè , nếu chưa có thì tập tin được tạo lập
- "a","at" mở để nối thêm, thông tin được ghi vào cuối tập tin cũ nếu đã có tập tin hay tạo mới tập tin
- "r+",''r+t" mở để vừa đọc và ghi , tập tin phải có trên đĩa
- "rb" mở một tập tin để đọc theo kiểu nhị phân . Tập tin phải có sẵn trên đĩa
- "r+b" mở một tập tin để đọc theo kiểu nhị phân . Tập tin phải có sẵn trên đĩa
- "w+","w+t" mở để vừa đọc và ghi , nội dung tập tin đã có trên đĩa sẽ bị ghi đè lên
- "wb" mở để ghi theo kiểu nhị phân , nếu trên đĩa đã có tập tin thì nội dung bị ghi đè , nếu chưa có thì tập tin được tạo lập
- "a+","a+t" mở để đọc và nối thêm , nếu tập tin chưa có thì nó sẽ được tạo ra
- "ab" mở để đọc và nối thêm theo kiểu nhị phân , nếu tập tin chưa có thì nó sẽ được tạo ra

c. Ghi lên tập tin : Khi tập tin đã được mở , ta có thể ghi lên tập tin từng kí tự một bằng cách dùng hàm :

putc(ch,fp)

Hàm putc() tương tự các hàm putch() và putchar() . Hàm putc() ghi lên tập tin có cấu trúc FILE được ấn định bởi biến fp nhận được khi mở tập tin . Tiến trình ghi được tiến hành cho đến khi nhấn enter .

d. Đóng tập tin : Khi không đọc ghi nữa ta cần đóng tập tin . Câu lệnh đóng tập tin là :

fclose(fp);

Ta báo cho hệ thống biết là cần đóng tập tin chỉ bởi fp .

e. Đọc tập tin : Nếu ta có thể ghi lên tập tin thì ta cũng có thể đọc từ tập tin . Ta có ví dụ sau :

Chương trình 3-2 :

```
#include <stdio.h>
#include <conio.h>
main()
  {
    FILE *fp;
    int ch;
    clrscr();
    fp=fopen("textfile","r");
    while ((ch=getc(fp))!=EOF)
```

```
    printf("%c",ch);
  fclose(fp);
  getch();
 }
```

f. Kết thúc tập tin : Sự khác nhâu chủ yếu giữa chương trình đọc và ghi là chương trình đọc phải phân biệt được đâu là kí tự EOF . Nó không phải là một kí tự àm là một số nguyên do hệ điều hành gửi tới . Khi hết tập tin ta gặp mã kết thúc tập tin EOF (định nghĩa trong stdio.h bằng -1) và hàm foef() cho trị khác không . Người ta chọn -1 làm mã kết thúc vì nếu chưa gặp cuối tập tin thì sẽ đọc được một byte mà mã sẽ nằm trong khoảng 0-255 . Như vậy giá trị -1 không trùng với bất kì kí tự nào nào được đọc từ tập tin . Trong khi chương trình đang đọc và hiển thị các kí tự thì nó tìm kiếm mộ giá trị -1 hay EOF . Khi thấy giá trị này , chương trình sẽ kết thúc . Chúng ta dùng một biến nguyên cất giữ một kí tự đọc được , do đó ta có thể hiểu dấu EOF như là một trị nguyên có trị là -1 . Nếu dùng một biến kiểu char , chúg ta có thể dùng tất cả các kí tự từ 0..255 - đó là tổ hợp 8 bit . Do đó nếu dùng biến nguyên , ta bảo đảm rằng chỉ có một giá trị 16 bit là -1 , đó là dấu EOF .

g. Sự phiền phức khi mở tập tin : Hai chương trình ta trình bày trên có một lỗi tiểm ẩn . Nếu tập tin đã được chỉ định không mở được thì chương trình không chạy . Lỗi này có thể là do tập tin chưa có (khi đọc) hay đĩa không còn đủ chỗ(khi ghi). Do đó vấn đề là phải kiểm tra xem tập tin có mở được hay không , nếu tập tin không mở được thì hàm fopen() trả về trị 0(0 là NULL trong stdio.h) . Khi này C coi đây không phải là địa chỉ hợp lệ . Như vậy ta viết lại chương trình trên như sau

Chương trình 3-3 :

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
void main()
 {
  FILE *fp;
  int ch;
  clrscr();
  if ((fp=fopen("file","r"))==NULL)
    {
       printf("Khong mo duoc tap tin\n");
       getch();
       exit(1);
    }
  while ((ch=getc(fp))!=EOF)
    printf("%c",ch);
  fclose(fp);
 }
```

h. Đếm số kí tự : Khả năng đọc và ghi tập tin trên cơ sở các kí tự cho phép triển khai một số ứng dụng . Chúng ta xem xét chương trình đếm số kí tự sau :

Chương trình 3-4 :

```
#include <stdio.h>
#include <conio.h>
main(int argc,char *argv)
 {
  FILE *fp;
```

```
   char string[8];
   int count = 0;
   clrscr();
   if (argc!=2)
     {
        printf("Format c:\<ten chuong trinh>    <ten tap tin>");
        getch();
        exit(1);
     }
   if ((fp=fopen(argv[1],"r"))==NULL)
     {
        printf("Khong mo duoc tap tin\n");
        getch();
        exit(1);
     }
   while (getc(fp)!=EOF)
     count++;
   fclose(fp);
   printf("Tap tin %s co %d ki tu",argv[1],count);
   getch();
  }
```

i. Đếm số từ : Ta có thể sửa chương trình trên thành chương trình đếm số từ .

Chương trình 3-5 :

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
main(int argc,char *argv[])
  {
   FILE *fp;
   char ch,string[81];
   int count = 0;
   int white=1;
   clrscr();
   if (argc!=2)
     {
        printf("  Format c:\<Ten chuong trinh>  <tentaptin>\n");
        getch();
        exit(1);
     }
   if ((fp=fopen(argv[1],"r"))==NULL)
     {
        printf("Khong mo duoc tap tin\n");
        getch();
        exit(1);
     }
   while ((ch=getc(fp))!=EOF)
     switch(ch)
        {
         case ' ': /*nếu có dấu trống , dòng mới hay tab*/
```

```
            case '\t':
            case '\n': white++;
                       break;
            default:if(white)
                    {
                     white=0;
                     count++;
                    }
           }
   fclose(fp);
   printf("Tap tin %s co %d tu",argv[1],count);
   getch();
   return 0;
  }
```

k.Vào ra chuỗi : Đọc hay ghi chuỗi trên tập tin cũng tương tự như đọc hay ghi từng kí tự riêng lẻ . Ta xét một chương trình ghi chuỗi

Chương trình 3-6 :

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
void main()
  {
   FILE *fp;
   char string[8];
   clrscr();
   if ((fp=fopen("a.txt","w"))==NULL)
     {
        printf("Khong mo duoc tap tin\n");
        getch();
        exit(1);
     }
   while (strlen(gets(string))>0)
     {
        fputs(string,fp);
        fputs("\n",fp);
     }
   fclose(fp);
  }
```

Trong chương trình mỗi chuỗi kết thúc bằng cách gõ enter và kết thúc chương trình bằng cách gõ enter ở đầu dòng mới . Do fputs() không tự động thêm vào mã kết thúc để chuyển dòng mới nên ta phải thêm vào tập tin mã này . Chương trình đọc một chuỗi từ tập tin :

Chương trình 3-7 :

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
void main()
```

```
{
  FILE *fp;
  char string[81];
  clrscr();
  if ((fp=fopen("a.txt","r"))==NULL)
    {
       printf("Khong mo duoc tap tin\n");
       getch();
       exit(1);
    }
  while (fgets(string,81,fp)!=NULL)
    printf("%s",string);
  fclose(fp);
  getch();
}
```

Hàm fgets() nhận 3 đối số : địa chỉ nơi đặt chuỗi , chiều dài tối đa của chuỗi , và con trỏ chỉ tới tập tin .

l. Vấn đề sang dòng mới : Trong chương trình đếm kí tự ta thấy số kí tự đếm được bao giờ cũng nhỏ hơn số byte có trong tập tin này nhận được bằng lệnh dir của DOS . Khi ta ghi một tập tin văn bản vào đĩa , C tự động ghi vào đĩa cả hai mã CR và LF khi gặp mã sang dòng mới "\n" . Ngược lại khi đọc tập tin từ đĩa , các mã CR và LF được tổ hợp thành mã sang dòng mới . Chương trình sau minh hoa thêm về kĩ thuật vào ra chuỗi , nội dung tương tự lệnh type của DOS

Chương trình 3-8 :

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
main(int argc,char *argv[])
 {
  FILE *fp;
  char string[81];
  clrscr();
  if (argc!=2)
    {
       printf("Format c:\<ten chuong trinh>  <ten tap tin>");
       getch();
       exit(1);
    }
  if ((fp=fopen(argv[1],"r"))==NULL)
    {
       printf("Khong mo duoc tap tin\n");
       getch();
       exit(1);
    }
  while (fgets(string,81,fp)!=NULL)
    printf("%s",string);
  fclose(fp);
  getch();
  return 0;
```

```
}
```

m. Các tập tin chuẩn và máy in : Trên đây ta đã nói đến cách thức tiếp nhận một con trỏ tham chiếu dến một tập tin trên đĩa của hàm fopen() , C định nghĩa lại tê chuẩn của 5 tập tin chuẩn như sau :

Tên	Thiết bị
in	Thiết bị vào chuẩn (bàn phím)
out	Thiết bị ra chuẩn (màn hình)
err	Thiết bị lỗi chuẩn (màn hình)
aux	Thiết bị phụ trợ chuẩn(cổng nối tiếp)
prn	Thiết bị in chuẩn (máy in)

Ta có thể dùng các tên này để truy cập đến các thiết bị . Chương trình sau dùng hàm fgets(0 và fputs() để in nội dung một tập tin ra máy in

Chương trình 3-9 :

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
main(int argc,char *argv[])
 {
  FILE *fp1,*fp2;
  char string[81];
  clrscr();
  if (argc!=2)
    {
       printf("Format c:\<ten chuong trinh>  <ten tap tin>");
       getch();
       exit(1);
    }
  if ((fp1=fopen(argv[1],"r"))==NULL)
    {
       printf("Khong mo duoc tap tin\n");
       getch();
       exit(1);
    }
  if ((fp2=fopen("prn","w"))==NULL)
    {
       printf("Khong mo duoc may in\n");
       getch();
       exit(1);
    }
  while (fgets(string,81,fp1)!=NULL)
    fputs(string,fp2);
  fclose(fp1);
  fclose(fp2);
  getch();
  return 0;
 }
```

Trong chương trình trên máy in được coi là tập tin có tên là prn

n. Nhập xuất định dạng : Trước đây ta đã đề cập đến nhập xuất kí tự . Những số có định dạng cũng có thể ghi lên đĩa như các kí tự . Ta xét chương trình sau

Chương trình 3-10 :

```
#include <stdio.h>
#include <conio.h>
main()
  {
    FILE *p;
    int i,n;
    float x[4],y[4];
    clrscr();
    p=fopen("test.txt","w");
    printf("Cho so cap so can nhap n = ");
    scanf("%d",&n);
    fprintf(p,"%d\n",n);
    printf("Cho cac gia tri x va y\n");
    for (i=0;i<n;i++)
      {
          scanf("%f%f",&x[i],&y[i]);
          fprintf(p,"%f     %f\n",x[i],y[i]);
      }
    fclose(p);
  }
```

```
#include <stdio.h>
#include<conio.h>
#include <string.h>
void main()
  {
    FILE *fp;
    char name[40];
    int code;
    float height;
    int n,i;
    clrscr();
    fp=fopen("b.txt","w");
    printf("Cho so nguoi can nhap : ");
    scanf("%d",&n);
    for (i=0;i<n;i++)
      {
          printf("Nhap ten , ma so va chieu cao : ");
          scanf("%s%d%f",name,&code,&height);
          fprintf(fp,"%s  %d  %f",name,code,height);
      }
    fclose(fp);
  }
```

Chương trình 3-11 :

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

```
void main()
  {
    FILE *p;
    int i,n;
    float x[4],y[4];
    clrscr();
    p=fopen("test.txt","r");
    fscanf(p,"%d",&n);
    for (i=0;i<n;i++)
      {
          fscanf(p,"%f%f",&x[i],&y[i]);
          printf("\n%.3f%8.3f",x[i],y[i]);
      }
    fclose(p);
    getch();
  }
```

```
#include <stdio.h>
#include<conio.h>
#include <string.h>
void main()
  {
    FILE *fp;
    char name[2];
    int code,n,i;
    float height;
    clrscr();
    fp=fopen("b.txt","r");
    fscanf(fp,"%d",&n);
    for (i=0;i<n;i++)
      {
          fscanf(fp,"%s%d%f\n",name,&code,&height);
          printf("%s%3d%8.3f\n",name,code,height);
      }
    fclose(fp);
    getch();
  }
```

§3. KIỂU NHỊ PHÂN VÀ KIỂU VĂN BẢN

1. Mã sang dòng theo hai kiểu : Trong dạng văn bản , một kí tự chuyển dòng tương ứng với 2 mã CR và LF khi ghi vào tập tin trên đĩa . Ngược lại khi đọc , tổ hợp CR/LF trên đĩa tương ứng với kí tự sang dòng mới . Tuy nhiên nếu mở tập tin theo kiểu nhị phân thì 2 mã CR và LF là phân biệt nhau . Từ đó số kí tự mà chương trình đếm được khác với trường hợp mở tập tin bằng kiểu văn bản

Chương trình 3-12 : Chương trình đếm số kí tự bằng cách mở tập tin theo kiểu nhị phân

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

```
#include <conio.h>
void main(int argc,char *argv[])
  {
    FILE *fp;
    char string[81];
    int count=0;
    clrscr();
    if (argc!=2)
      {
          printf("Format c:\<ten chuong trinh>  <ten tap tin>");
          getch();
          exit(1);
      }
    if ((fp=fopen(argv[1],"rb"))==NULL)
      {
          printf("Khong mo duoc tap tin\n");
          getch();
          exit(1);
      }
    while (getc(fp)!=EOF)
      count++;
    fclose(fp);
    printf("Tap tin %s co %d ki tu",argv[1],count);
    getch();
  }
```

2. Mã kết thúc tập tin theo 2 kiểu : Sự khác biệt thứ hai khi mở tập tin theo kiểu nhị phân hay kiểu kí tự còn là ở chỗ nhìn nhận kí tự kết thúc tập tin . Nói chung các tập tin đều được quản lí theo kích thước của nó và khi đọc hết số byte đã chỉ ra trong kích thước tập tin thì dấu hiệu EOF sẽ được thông báo , dấu hiệu đó ứng với mã 1Ah(hay 26 ở hệ 10) . Khi đóng tập tin văn bản , mã 1A sẽ được tự động chèn vào cuối tập tin để làm dấu hiệu kết thúc tập tin (tương đương mã Ctrl-Z) . Do vậy nếu bằng cáh nào đó ta chèn mã 1A vào một vị trí giữa tập tin , thì khi mở tập tin theo kiểu văn bản và đọc đến mã này chương trình đọc sẽ ngừng hẳn vì chính lúc đó hàm đọc phát sinh giá trị -1 để báo cho chương trình là đã kết thúc tập tin . Nếu đã lưu số vào tập tin theo dạng nhị phân thì khi mở tập tin cần phải mở theo dạng nhị phân . Nếu không sẽ có một số nào đó là 1A và việc đọc tập tin theo kiểu văn bản sẽ kết thúc ngoài ý định . Tương tự , với tập tin mở theo kiểu nhị phân mã 10 không được nhìn nhận là mã sang dòng mới vì không được xem là tương ứng với tổ hợp CR/LF nữa.

3. Chương trình minh hoạ : Chúng ta xét một chương trình dùng kiểu nhị phân để khảo sát tập tin .

Chương trình 3-13 :

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define length          10
#define true   0
#define false  -1
void main(int agrc,char *argv[])
  {
```

```
  FILE *fp;
  int ch;
  int j,noteof;
  unsigned char string[length+1];
  clrscr();
  if (agrc!=2)
    {
       printf("Dang c:\<ten chuong trinh> <ten tap tin>");
       getch();
       exit(1);
    }
  if ((fp=fopen(argv[1],"rb"))==NULL)
    {
       printf("Khong mo duoc tap tin\n");
       getch();
       exit(1);
    }
  noteof=true;
  do
    {
       for (j=0;j<length;j++)
         {
          if ((ch=getc(fp))==EOF)
            noteof=false;
          printf("%3x",ch);
          if (ch>31)
            *(string+j)=ch;/* ki tu in duoc*/
          else
            *(string+j)='.';/* ki tu khong in duoc*/
         }
       *(string+j)='\0';
       printf("  %s\n",string);
    }
  while (noteof==true);
  fclose(fp);
  getch();
}
```

4. Các hàm fread và fwrite :

a. Ghi cấu trúc bằng fwrite : Ta xét một chương trình ghi cấu trúc lên đĩa . Trong chương trình ta dùng hàm fread() . Hàm này có 4 đối số : địa chỉ để ghi cấu trúc , kích thước của cấu trúc , số cấu trúc sẽ ghi và con trỏ chỉ tới tập tin .

Chương trình 3-14 :

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
void main()
 {
  char chso[10];
  FILE *fp;
```

```
  struct nguoi {
                  char ten[30];
                  int so;
                  float cao;
                 }nv;
  clrscr();
  if((fp=fopen("nhanvien.rec","wb"))==NULL)
    {
      printf("Khong mo duoc file\n");
      getch();
      exit(1);
    }
  do
    {
      printf("\nCho ten : ");
      gets(nv.ten);
      printf("Cho ma so : ");
      gets(chso);
      nv.so=atoi(chso);
      printf("Cho chieu cao :");
      gets(chso);
      nv.cao=atof(chso);
      fwrite(&nv,sizeof(nv),1,fp);
      printf("Tiep tuc khong y/n?");
    }
  while(getch()=='y');
  fclose(fp);
 }
```

b. Đọc cấu trúc bằng fread : Ta dùng hàm fread() để đọc cấu trúc ghi trên một tập tin . Các đối số của fread() cũng giống như fwrite() . Hàm fread() trả về số của những mục đã được đọc tới . Nếu tập tin đã kết thúc nó cho trị âm . Ta xét ví dụ sau :

Chương trình 3-15 :

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
void main()
 {
  FILE *fp;
  struct nguoi {
                  char ten[30];
                  int so;
                  float cao;
                 }nv;
  clrscr();
  if((fp=fopen("nhanvien.rec","rb"))==NULL)
    {
      printf("Khong mo duoc file\n");
      getch();
      exit(1);
    }
```

```
   do
     {
         printf("\nTen :%s\n",nv.ten);
         printf("Ma so :%03d\n",nv.so);
         printf("Chieu cao :%.2f\n",nv.cao);

     }
   while (fread(&nv,sizeof(nv),1,fp)==1);
   fclose(fp);
   getch();
  }
```

c. Ghi mảng bằng fwrite() : Hàm fwrite() cũng dùng ghi mảng lên đĩa . Ta xét ví dụ sau :

Chương trình 3-16 :

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int table[10]={1,2,3,4,5,6,7,8,9,10};
void main()
  {

   FILE *fp;
   clrscr();
   if((fp=fopen("table.rec","wb"))==NULL)
     {
         printf("Khong mo duoc file\n");
         getch();
         exit(1);
     }
   fwrite(table,sizeof(table),1,fp);
   fclose(fp);
  }
```

d. Đọc mảng bằng fread() : Sau khi ghi mảng lên đĩa ta có thể đọc các phần tử của mảng từ đĩa bằng hàm fread().

Chương trình 3-17 :

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

void main()
  {

   FILE *fp;
   int a[10];
   int i;
   clrscr();
   if((fp=fopen("table.rec","rb"))==NULL)
     {
```

```
        printf("Khong mo duoc file\n");
        getch();
        exit(1);
     }
   for (i=0;i<10;i++)
     {
        fread(a,sizeof(a),10,fp);
        printf("\%3d",a[i]);
     }
   fclose(fp);
   getch();
  }
```

e. Ví dụ về cơ sở dữ liệu : Ta xét chương trình quản lí nhân viên với các tập tin trên đĩa như sau :

Chương trình 3-18 :

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#define true 1
struct nguoi {
              char ten[30];
              int so;
              float cao;
             };
struct nguoi nv[10];
int n=0;
char numstr[10];

void main()
  {
   char ch;
   void newname(void);
   void listall(void);
   void wfile(void);
   void rfile(void);
   clrscr();
   while (true)
     {
        printf("\nGo 'e' de nhap nhan vien moi\n");
        printf("Go 'l'de liet ke nhan vien\n");
        printf("Go 'w' de ghi len dia\n");
        printf("Go 'r'de doc file tu dia\n");
        printf("Go 'q' de ket thuc chuong trinh\n\n");
        ch=getch();
        switch (ch)
          {
           case 'e':newname();
                    break;
           case 'l':listall();
```

```
                    break;
            case 'w':wfile();
                    break;
            case 'r':rfile();
                    break;
            case 'q': exit(1);
            default : printf("Nhap sai ki tu , chon lai!");
           }
      }
  }

void newname()
  {
    char numstr[81];
    printf("\nBan ghi so %d\nCho ten : ",n+1);
    gets(nv[n].ten);
    printf("Cho ma so co 3 chu so : ");
    gets(numstr);
    nv[n].so=atoi(numstr);
    printf("Cho chieu cao :");
    gets(numstr);
    nv[n++].cao=atof(numstr);
  }
void listall()
  {
    int j;
    if (n<1)
      printf("Danh sach rong\n");
    for (j=0;j<n;j++)
      {
          printf("\nBan ghi so %d\n",j+1);
          printf("Ten :%s\n",nv[j].ten);
          printf("Ma nhan vien : %3d\n",nv[j].so);
          printf("Chieu cao :%4.2f\n",nv[j].cao);
      }
  }

void wfile()
  {
    FILE *fp;
    if (n<1)
      {
          printf("Danh sach rong , khong ghi\n");
          getch();
          exit(1);
      }
    if ((fp=fopen("nv.rec","wb"))==NULL)
      {
          printf("Khong mo duoc file\n");
          getch();
```

```
        exit(1);
     }
   else
     {
        fwrite(nv,sizeof(nv[0]),n,fp);
        fclose(fp);
        printf("Da ghi %3d ban ghi len dia\n",n);
     }
  }

void rfile()
  {
   FILE *fp;
   if ((fp=fopen("nv.rec","rb"))==NULL)
     {
        printf("Khong mo duoc file\n");
        getch();
        exit(1);
     }
   else
     {
        while(fread(&nv[n],sizeof(nv[n]),1,fp)==1)
          {
           clrscr();
           printf("Ban ghi so %3d\n",n+1);
           printf("Ten nhan vien :%s\n",nv[n].ten);
           printf("Ma nhan vien :%3d\n",nv[n].so);
           printf("Chieu cao cua nhan vien :%.2f\n",nv[n].cao);
           getch();
           n++;
          }
        fclose(fp);
        printf("Xong ! Tong so ban ghi da doc %3d\n",n);
     }
  }
```

§4. CÁC FILE NGẪU NHIÊN

Các tập tin đề cập trước đây là các tập tin tuần tự , nghĩa là tập tin mà khi đọc hay ghi đề theo chế độ tuần tự từ đầu đến cuối tập tin . Đối với tập tin tuần tự ta không thể đọc hay ghi một cách trực tiếp tại một vị trí bất kì trên tập tin . Tập tin ngẫu nhiên cho phép ta truy cập ngẫu nhiên vào những vị trí cần thiết trên tập tin . Các hàm dùng khi truy cập tập tin ngẫu nhiên là :

rewind() : di chuyển con trỏ tập tin về đầu tập tin

Cú pháp : void rewind(FILE *fp);

fseek() : di chuyển con trỏ tập tin về vị trí mong muốn

Cú pháp : int fseek(FILE *fp , long sb , int xp)

fp - con trỏ tập tin

sb - số byte cần di chuyển

xp - vị trí xuất phát mà việc dịch chuyển đươợ bắt đầu từ đó . xp có thể có các giá trị sau :

xp=SEEK_SET hay 0 : xuất páht từ đầu tập tin

xp=SEEK_CUR hay 1 : xuất phát từ vị trí con trỏ hiện tại

xp=SEEK_END hay 2 : xuất páht từ cuối tập tin

ftell() : cho biết vị trí hiện tại của con trỏ tập tin

Ta xét chương trình ví dụ sau :

Chương trình 3-19 :

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
void main()
  {
    struct nguoi {
                   char ten[30];
                   int so;
                   float cao;
                  }nv;
    int recno;
    FILE *fp;
    long int offset;
    clrscr();
    if ((fp=fopen("nhanvien.rec","r"))==NULL)
      {
         printf("Khong mo duoc file\n");
         getch();
         exit(1);
      }
    printf("Ban muon doc ban ghi thu may : ");
    scanf("%d",&recno);
    recno--;
    offset=recno*sizeof(nv);
    if (fseek(fp,offset,0)!=0)
      {
         printf("Khong di chuyen duoc con tro file toi do\n");
         getch();
         exit(1);
      }
    fread(&nv,sizeof(nv),1,fp);
    printf("Ten :%s\n",nv.ten);
    printf("Ma nhan vien : %3d\n",nv.so);
    printf("Chieu cao :%4.2f\n",nv.cao);
    getch();
  }
```

§5. LỖI VÀO RA

Nói chung , khi mở tập tin thành công ta có thể ghi lên nó . Tuy nhiên , nhiều trường hợp không mở được tập tin nhưng ta không biết lỗi do đâu . Để xác định llỗi ta dùng hàm

ferror() . Hàm này có đối số là con trỏ tập tin . Hàm sẽ có giá trị không nếu không có lỗi gì . Ngược lại hàm cho giá trị khác không . Ta cũng có thể dùng hàm perror() để chỉ nội dung lỗi .

Chương trình 3-20 :

```
#include <stdio.h>
#include<conio.h>
#include <string.h>
#include <stdlib.h>
void main()
  {
    FILE *fp;
    char name[40],numstr[10];
    int code;
    float height;
    int n,i;
    clrscr();
    fp=fopen("a:\newfile.txt","w");
    printf("Cho so nguoi can nhap : ");
    gets(numstr);
    n=atoi(numstr);
    for (i=0;i<n;i++)
      {
          printf("Nhap ten : ");
          gets(name);
          printf("Nhap ma so : ");
          gets(numstr);
          code=atoi(numstr);
          printf("Nhap chieu cao : ");
          gets(numstr);
          height=atof(numstr);
          fprintf(fp,"%s  %d  %f",name,code,height);
          if (ferror(fp))
            {
              perror("Loi ghi file ");
              getch();
              exit(1);
            }
      }
    fclose(fp);
  }
```

Sau lỗi do ta ghi , trình biên dịch sẽ thông báo lỗi cụ thể trong câu " Loi ghi file : no such file ỏ directory"

§6. VÀO RA Ở MỨC HỆ THỐNG

1.Các tập tin tiêu đề và biến chuẩn : Trong cách vào ra ở mức hệ thống , ta phải khởi tạo bộ đệm rồi đặt dữ liệu vào đó trước ghi hay đọc . Vào ra ở mức hệ thống có lợi ở chỗ lượng mã ít hơn vào ra chuẩn và tốc độ sẽ nhanh hơn . Để dùng các hàm cấp 1 ta phải cần các tập tin tiêu đề sau :

io.h - chứa các prototype của các hàm cấp 1
fcntl.h - chứa các định nghĩa quyền truy cập
sys/stat.h - chá các định nghĩa thuộc tính
dó.h - chứa các thuộc tính theo DOS

2. Tóm tắt các hàm : creat - tạo tập tin mới
_creat - tạo tập tin mới theo kiểu nhị phân
open - mở tập tin
_open - mở tập tin đã tồn tại
close và _close - đóng tập tin
chmod - thay đổi thuộc tính của tập tin
_chmode - thay đổi thuộc tính của tập tin theo kiểu DOS
perror - thông báo lỗi (stdlib.h)
write - ghi một dãy các byte
read - đọc một dãy các byte
lseek - dùng di chuyển con trỏ vị trí

3. Đọc tập tin theo cách vào ra hệ thống : Ta có chương trình đọc tập tin từ đĩa và hiển thị lên màn hình theo cách vào ra hệ thống .

Chương trình 3-21 :

```
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <io.h>
#define BUFFSIZE  512
char buff[BUFFSIZE];
void main(int argc,char *argv[])
  {
   int inhandle,bytes,i;
   clrscr();
   if (argc!=2)
     {
        printf("Dang <ten chuong trinh>   <ten tap tin>");
        getch();
        exit(1);
     }
   if ((inhandle=open(argv[1],O_RDONLY|O_BINARY))<0)
     {
        printf("Khong mo duoc file\n");
        getch();
        exit(1);
     }
   while ((bytes=read(inhandle,buff,BUFFSIZE))>0)
     for (i=0;i<bytes;i++)
        putch(buff[i]);
   close(inhandle);
  }
```

4. Khởi tạo bộ đệm : Trong chương trình ta phải định nghĩa bộ đệm bằng phát biểu

```
#define BUFFSIZE  512
char buff[BUFFSIZE]
```

Nhờ đó ta đọc được dữ liệu từ đĩa vào bộ đệm buff . Với DOS , kích thước bộ đệm nên chọn là bội số của 512.

5. Mở một tập tin : Cũng giống như vào ra bằng hàm cấp 2 , ta phải mở tập tin trước khi đọc hay ghi bằng phát biểu :

```
inhandle=open(argv[1],O_RDONLY | O_BINARY);
```

Biểu thức này thiết lập sự liên lạc giữa tập tin và hệ điều hành . Trong biểu thức ta cần một hằng lag oflag là dấu hiệu cho biết mức độ dùng tập tin .

oflag	ý nghĩa
O_APPEND	Đặt con trỏ ở cuối tập tin
O_CREAT	Tạo tập tin mới để ghi(không có hiệu quả nếu tập tin đã có)
O_RDONLY	Mở một tập tin để chỉ đọc
O_RDWR	Mở một tập tin để chỉ đọc hay ghi
O_TRUNC	Mở và cắt bỏ bớt tập tin
O_WRONLY	Mở tập tin để ghi
O_BINARY	Mở tập tin kiểu nhị phân
O_TEXT	Mở tập tin kiểu văn bản

6. Danh số của tập tin : Trong vào ra chuẩn , con trỏ tập tin sẽ nhận được ngay khi gọi hàm fopen() còn trong nhập xuất bằng hàm cấp 1 ta nhậ được giá trị nguyên gọi là danh số của tập tin . Đây là số gán cho một tập tin cụ thể để tham chiếu đến tập tin này . Nếu hàm open() cho ta giá trị -1 nghĩa là danh số không đúng và phát sinh lỗi .

7. Đọc tập tin vào bộ đệm : Để đọc tập tin vào bộ đệm ta dùng lệnh :

```
byte = read(inhandle , buff , BUFSIZE);
```

Hàm này có 3 đối số : danh số của tập tin , địa chỉ của bộ đệm và số byte cực đại cần đọc . Giá trị của hàm read() chỉ ra số byte đã đọc được .

8. Đóng tập tin : Để đóng tập tin ta dùng lệnh

```
close(inhandle);
```

9. Thông báo lỗi : Khi hàm open() cho giá trị -1 , nghĩa là có lỗi . Dạng lỗi sẽ được đọc bằng perror() . Ta có chương trình ví dụ

Chương trình 3-22 :

```
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <io.h>
#define BUFFSIZE  512
char buff[BUFFSIZE];
void main(int argc,char *argv[])
  {
    int inhandle,bytes,i;
    clrscr();
    if (argc!=2)
      {
          printf("Dang <ten chuong trinh>   <ten tap tin>");
          getch();
          exit(1);
      }
    if ((inhandle=open(argv[1],O_RDONLY|O_BINARY))<0)
```

```
    {
        perror("Khong mo duoc file\n");
        getch();
        exit(1);
    }
  while ((bytes=read(inhandle,buff,BUFFSIZE))>0)
    for (i=0;i<bytes;i++)
        putch(buff[i]);
  close(inhandle);
 }
```

10. Thao tác trên bộ đệm : Việc đưa tập tin vào bộ đệm có lợi là cho phép truy cập trên bộ đệm thay vì trên tập tin . Làm như vậy nhanh hơn truy cập trên đĩa .
Chương trình sau cho phép tìm một từ trong một tập tin văn bản .
Chương trình 3-23 :

```
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <string.h>
#include <io.h>
#define BUFFSIZE  1024
char buff[BUFFSIZE];

void main(int argc,char *argv[])
 {
  int inhandle,bytes;
  void search(char *,int);
  clrscr();
  if (argc!=3)
    {
        printf("Dang <ten chuong trinh> <ten tap tin> <tu can tim>");
        getch();
        exit(1);
    }
  if ((inhandle=open(argv[1],O_TEXT))<0)
    {
        printf("Khong mo duoc file %s\n",argv[1]);
        getch();
        exit(1);
    }
  while ((bytes=read(inhandle,buff,BUFFSIZE))>0)
    search(argv[2],bytes);
  close(inhandle);
  printf("Khong tim thay");
  getch();
 }

void search(char *cau,int buflen)
 {
```

```
  char *p,*ptr;
  ptr=buff;
  while ((ptr=memchr(ptr,cau[0],buflen))!=NULL)
    if (memcmp(ptr,cau,strlen(cau))==0)
        {
          printf("Tu xuat hien lan dau trong cau tai vi tri %d:\n",ptr-buff+1);
          for (p=ptr;p<ptr+strlen(cau);p++)
            putch(*p);
          exit(1);
        }
    else
        ptr++;
 }
```

11. Hàm dùng bộ đệm : Hàm search() là chương trình con minh hoạ cách dùng bộ đệm . Ta có một hàm memchr() dạng :

```
ptr = memchr(ptr , cau[0] , buflen);
```

Hàm này dùng để tìm vị trí của kí tự cau[0] trong chuỗi chỉ bởi ptr và độ f\dài của phần cần tìm trong bộ đệm là buflen . Chương trình sẽ truyền argv[2] cho cau . Hàm này cho giá trị NULL khi không tìm thấy kí tự cần tìm . Ngược lại nó sẽ cho địa chỉ của kí tự đã tìm thấy trong bộ đệm . Việc so sánh các chuỗi cau và chuỗi ptr được tiến hành nhờ hàm memcmp trong câu lệnh :

```
if ((memcmp(ptr,cau,strlen(cau))==0)
```

Hàm này cũng có 3 đối số là : chuỗi thu nhất ptr , chuỗi thu hai cau và đo dai can so sanh strlen(cau)

12. Ghi lên tập tin : Ghi thông tin lên tập tin phức tạp hơn đọc từ tập tin . Ta có chương trình ví dụ sau dùng để chép từ một tập tin này sang tập tin khác.

Chương trình 3-24 :

```
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <io.h>
#include <sys\stat.h>
#define BUFFSIZE  4096
char buff[BUFFSIZE];

void main(int argc,char *argv[])
 {
  int inhandle,outhandle,bytes;

  clrscr();
  if (argc!=3)
    {
      printf("Dang <ten chuong trinh> <ten tap tin 1> <ten tap tin 2>");
      getch();
      exit(1);
    }
  if ((inhandle=open(argv[1],O_RDWR|O_BINARY))<0)
    {
```

```
        printf("Khong mo duoc file %s\n",argv[1]);
        getch();
        exit(1);
     }
   if ((outhandle=open(argv[2],O_CREAT|O_WRONLY|O_BINARY,S_IWRITE))<0)
     {
        printf("Khong mo duoc file %s\n",argv[2]);
        getch();
        exit(1);
     }
   while ((bytes=read(inhandle,buff,BUFFSIZE))>0)
     write(outhandle,buff,bytes);
   close(inhandle);
   close(outhandle);
   printf("Da chep xong");
   getch();
 }
```

Trong ví dụ trên ta mở một lúc 2 tập tin với danh số là inhamdle và outhandle Biểu thức mở tập tin nguồn không có gì đặc biệt còn biểu thức mở tập tin đích có dạng :

outhandle = open(argv[2] ,O_CREAT | O_WRONLY | O_BINARY , S_IWRITE)

với O_CREAT để tạo tập tin trên đĩa

O_WRONLY để chỉ ghi lên tập tin

O_BINARY để mở tập tin theo kiểu nhị phân

Khi mở tập tin với O_CREAT , đối thứ 3 của open() là một trong 3 trị :

S_IWRITE : chỉ cho phép ghi lên tập tin

S_IREAD : chỉ cho phép đọc từ tập tin

S_IWRITE | S_IREAD : cho phép đọc và ghi lên tập tin

Để dùng các trị này phải khai báo #include <sys\stat.h> sau khai báo #include<fcntl.h> . Hàm write() có đối tương tự như read() . Trong vòng lặp while hàm read() báo số byte đọc được qua biến bytes và hàm write() sẽ biết số bytes cần ghi vào tập tin đích . Trong chương trình ta dùng bộ đệm với kích thước khá lớn để chương trình chạy nhanh .