13 文件

1.1 C 文件概述

所谓“文件”是指一组相关数据的有序集合。这个数据集有一个名称，叫做文件名。实际上在前面的各章中我们已经多次使用了文件，例如源程序文件、目标文件、可执行文件、库文件 (头文件)等。

文件通常是驻留在外部介质(如磁盘等)上的，在使用时才调入内存中来。从不同的角度可对文件作不同的分类。从用户的角度看，文件可分为普通文件和设备文件两种。

普通文件是指驻留在磁盘或其它外部介质上的一个有序数据集，可以是源文件、目标文件、可执行程序；也可以是一组待输入处理的原始数据，或者是一组输出的结果。对于源文件、目标文件、可执行程序可以称作程序文件，对输入输出数据可称作数据文件。

设备文件是指与主机相联的各种外部设备，如显示器、打印机、键盘等。在操作系统中,把外部设备也看作是一个文件来进行管理，把它们的输入、输出等同于对磁盘文件的读和写。

通常把显示器定义为标准输出文件,一般情况下在屏幕上显示有关信息就是向标准输出文件输出。如前面经常使用的 printf,putchar 函数就是这类输出。

键盘通常被指定标准的输入文件，从键盘上输入就意味着从标准输入文件上输入数据。scanf,getchar 函数就属于这类输入。

从文件编码的方式来看，文件可分为 ASCII 码文件和二进制码文件两种。ASCII 文件也称为文本文件，这种文件在磁盘中存放时每个字符对应一个字节，用于存放对应的 ASCII 码。例如，数 5678 的存储形式为：

ASCII 码：　　00110101　00110110　00110111　00111000

　　↓　　↓　　↓　　↓

十进制码：　　5　　6　　7　　8

共占用 4 个字节。

ASCII 码文件可在屏幕上按字符显示，例如源程序文件就是 ASCII 文件，用 DOS 命令 TYPE 可显示文件的内容。由于是按字符显示，因此能读懂文件内容。

二进制文件是按二进制的编码方式来存放文件的。

例如， 数 5678 的存储形式为：

00010110　00101110

只占二个字节。二进制文件虽然也可在屏幕上显示，但其内容无法读懂。C 系统在处理这些文件时，并不区分类型，都看成是字符流，按字节进行处理。

输入输出字符流的开始和结束只由程序控制而不受物理符号(如回车符)的控制。因此也把这种文件称作"流式文件"。

本章讨论流式文件的打开、关闭、读、写、 定位等各种操作。

1.1 文件指针

在 C 语言中用一个指针变量指向一个文件，这个指针称为文件指针。通过文件指针就可对它所指的文件进行各种操作。

定义说明文件指针的一般形式为：

FILE *指针变量标识符；

其中 FILE 应为大写，它实际上是由系统定义的一个结构，该结构中含有文件名、文件状态和文件当前位置等信息。在编写源程序时不必关心 FILE 结构的细节。

例如：

FILE *fp;

表示 fp 是指向 FILE 结构的指针变量，通过 fp 即可找存放某个文件信息的结构变量，然后按结构变量提供的信息找到该文件，实施对文件的操作。习惯上也笼统地把 fp 称为指向一个文件的指针。

1.1 文件的打开与关闭

文件在进行读写操作之前要先打开，使用完毕要关闭。所谓打开文件，实际上是建立文件的各种有关信息，并使文件指针指向该文件，以便进行其它操作。关闭文件则断开指针与文件之间的联系，也就禁止再对该文件进行操作。

在 C 语言中，文件操作都是由库函数来完成的。在本章内将介绍主要的文件操作函数。

1.1.1文件的打开(fopen 函数)

fopen 函数用来打开一个文件，其调用的一般形式为：

文件指针名=fopen(文件名,使用文件方式)；

其中，

“文件指针名”必须是被说明为 FILE 类型的指针变量；

“文件名”是被打开文件的文件名；
“使用文件方式”是指文件的类型和操作要求。
“文件名”是字符串常量或字符串数组。
例如：

```
FILE *fp;
fp=("file a","r");
```

其意义是在当前目录下打开文件 file a，只允许进行“读”操作，并使 fp 指向该文件。
又如：

```
FILE *fphzk
fphzk=("c:\\hzk16","rb")
```

其意义是打开 C 驱动器磁盘的根目录下的文件 hzk16，这是一个二进制文件，只允许按二进制方式进行读操作。两个反斜线“\\ ”中的第一个表示转义字符，第二个表示根目录。

使用文件的方式共有 12 种，下面给出了它们的符号和意义。

文件使用方式	意义
“rt”	只读打开一个文本文件，只允许读数据
“wt”	只写打开或建立一个文本文件，只允许写数据
“at”	追加打开一个文本文件，并在文件末尾写数据
“rb”	只读打开一个二进制文件，只允许读数据
“wb”	只写打开或建立一个二进制文件，只允许写数据
“ab”	追加打开一个二进制文件，并在文件末尾写数据
“rt+”	读写打开一个文本文件，允许读和写
“wt+”	读写打开或建立一个文本文件，允许读写
“at+”	读写打开一个文本文件，允许读，或在文件末追加数据
“rb+”	读写打开一个二进制文件，允许读和写
“wb+”	读写打开或建立一个二进制文件，允许读和写
“ab+”	读写打开一个二进制文件，允许读，或在文件末追加数据

对于文件使用方式有以下几点说明：

1) 文件使用方式由 r,w,a,t,b，+六个字符拼成，各字符的含义是：

```
r(read):        读
w(write):       写
a(append):      追加
t(text):        文本文件，可省略不写
b(banary):      二进制文件
+:              读和写
```

2) 凡用“r”打开一个文件时，该文件必须已经存在，且只能从该文件读出。
3) 用“w”打开的文件只能向该文件写入。若打开的文件不存在，则以指定的文件名建立该文件，若打开的文件已经存在，则将该文件删去，重建一个新文件。
4) 若要向一个已存在的文件追加新的信息，只能用“a”方式打开文件。但此时该文件必须是存在的，否则将会出错。
5) 在打开一个文件时，如果出错，fopen 将返回一个空指针值 NULL。在程序中可以用这一信息来判别是否完成打开文件的工作，并作相应的处理。因此常用以下程序段打开文件:
6)

```
if((fp=fopen("c:\\hzk16","rb")==NULL)
  {
   printf("\nerror on open c:\\hzk16 file!");
```

```
getch();
exit(1);
}
```

这段程序的意义是，如果返回的指针为空，表示不能打开 C 盘根目录下的 hzk16 文件，则给出提示信息"error on open c:\ hzk16 file!"，下一行 getch()的功能是从键盘输入一个字符，但不在屏幕上显示。在这里，该行的作用是等待，只有当用户从键盘敲任一键时，程序才继续执行，因此用户可利用这个等待时间阅读出错提示。敲键后执行 exit(1)退出程序。

7) 把一个文本文件读入内存时，要将 ASCII 码转换成二进制码，而把文件以文本方式写入磁盘时，也要把二进制码转换成 ASCII 码，因此文本文件的读写要花费较多的转换时间。对二进制文件的读写不存在这种转换。

8) 标准输入文件(键盘)，标准输出文件(显示器)，标准出错输出(出错信息)是由系统打开的，可直接使用。

1.1.1文件关闭函数（fclose 函数）

文件一旦使用完毕，应用关闭文件函数把文件关闭，以避免文件的数据丢失等错误。

fclose 函数调用的一般形式是：

```
fclose(文件指针);
```

例如：

```
fclose(fp);
```

正常完成关闭文件操作时，fclose 函数返回值为 0。如返回非零值则表示有错误发生。

1.1 文件的读写

对文件的读和写是最常用的文件操作。在 C 语言中提供了多种文件读写的函数：

- 字符读写函数 ：fgetc 和 fputc
- 字符串读写函数：fgets 和 fputs
- 数据块读写函数：freed 和 fwrite
- 格式化读写函数：fscanf 和 fprinf

下面分别予以介绍。使用以上函数都要求包含头文件 stdio.h。

1.1.1字符读写函数 fgetc 和 fputc

字符读写函数是以字符(字节)为单位的读写函数。每次可从文件读出或向文件写入一个字符。

1. **读字符函数 fgetc**

fgetc 函数的功能是从指定的文件中读一个字符，函数调用的形式为：

```
字符变量=fgetc(文件指针);
```

例如：

```
ch=fgetc(fp);
```

其意义是从打开的文件 fp 中读取一个字符并送入 ch 中。

对于 fgetc 函数的使用有以下几点说明：

1) 在 fgetc 函数调用中，读取的文件必须是以读或读写方式打开的。

2) 读取字符的结果也可以不向字符变量赋值，

例如：

fgetc(fp);

但是读出的字符不能保存。

3) 在文件内部有一个位置指针。用来指向文件的当前读写字节。在文件打开时，该指针总是指向文件的第一个字节。使用 fgetc 函数后，该位置指针将向后移动一个字节。 因此可连续多次使用 fgetc 函数，读取多个字符。应注意文件指针和文件内部的位置指针不是一回事。文件指针是指向整个文件的，须在程序中定义说明，只要不重新赋值，文件指针的值是不变的。文件内部的位置指针用以指示文件内部的当前读写位置，每读写一次，该指针均向后移动，它不需在程序中定义说明，而是由系统自动设置的。

【例 13.1】读入文件 c1.doc，在屏幕上输出。

```
#include<stdio.h>
main()
{
  FILE *fp;
  char ch;
  if((fp=fopen("d:\\jrzh\\example\\c1.txt","rt"))==NULL)
    {
    printf("\nCannot open file strike any key exit!");
    getch();
    exit(1);
    }
  ch=fgetc(fp);
  while(ch!=EOF)
  {
    putchar(ch);
    ch=fgetc(fp);
  }
  fclose(fp);
}
```

本例程序的功能是从文件中逐个读取字符，在屏幕上显示。程序定义了文件指针 fp,以读文本文件方式打开文件"d:\\jrzh\\example\\ex1_1.c"，并使 fp 指向该文件。如打开文件出错，给出提示并退出程序。程序第 12 行先读出一个字符，然后进入循环，只要读出的字符不是文件结束标志(每个文件末有一结束标志 EOF)就把该字符显示在屏幕上，再读入下一字符。每读一次，文件内部的位置指针向后移动一个字符，文件结束时，该指针指向 EOF。执行本程序将显示整个文件。

2. 写字符函数 fputc

fputc 函数的功能是把一个字符写入指定的文件中，函数调用的形式为：

fputc(字符量，文件指针)；

其中，待写入的字符量可以是字符常量或变量，例如：

```
fputc('a',fp);
```

其意义是把字符 a 写入 fp 所指向的文件中。

对于 fputc 函数的使用也要说明几点：

1) 被写入的文件可以用写、读写、追加方式打开，用写或读写方式打开一个已存在的文件时将清除原有的文件内容，写入字符从文件首开始。如需保留原有文件内容，希望写入的字符以文件末开始存放，必须以追加方式打开文件。被写入的文件若不存在，则创建该文件。
2) 每写入一个字符，文件内部位置指针向后移动一个字节。
3) fputc 函数有一个返回值，如写入成功则返回写入的字符，否则返回一个 EOF。可用此来判断写入是否成功。

【例 13.2】从键盘输入一行字符，写入一个文件，再把该文件内容读出显示在屏幕上。

```
#include<stdio.h>
main()
{
  FILE *fp;
  char ch;
  if((fp=fopen("d:\\jrzh\\example\\string","wt+"))==NULL)
  {
    printf("Cannot open file strike any key exit!");
    getch();
    exit(1);
  }
  printf("input a string:\n");
  ch=getchar();
  while (ch!='\n')
  {
    fputc(ch,fp);
    ch=getchar();
  }
  rewind(fp);
  ch=fgetc(fp);
  while(ch!=EOF)
  {
    putchar(ch);
    ch=fgetc(fp);
  }
  printf("\n");
  fclose(fp);
}
```

程序中第 6 行以读写文本文件方式打开文件 string。程序第 13 行从键盘读入一个字符后进入循环，当读入字符不为回车符时，则把该字符写入文件之中，然后继续从键盘读入下

一字符。每输入一个字符，文件内部位置指针向后移动一个字节。写入完毕，该指针已指向文件末。如要把文件从头读出，须把指针移向文件头，程序第 19 行 rewind 函数用于把 fp 所指文件的内部位置指针移到文件头。第 20 至 25 行用于读出文件中的一行内容。

【例 13.3】把命令行参数中的前一个文件名标识的文件，复制到后一个文件名标识的文件中，如命令行中只有一个文件名则把该文件写到标准输出文件(显示器)中。

```
#include<stdio.h>
main(int argc,char *argv[])
{
 FILE *fp1,*fp2;
 char ch;
 if(argc==1)
 {
   printf("have not enter file name strike any key exit");
   getch();
   exit(0);
 }
  if((fp1=fopen(argv[1],"rt"))==NULL)
  {
    printf("Cannot open %s\n",argv[1]);
    getch();
    exit(1);
  }
  if(argc==2) fp2=stdout;
  else if((fp2=fopen(argv[2],"wt+"))==NULL)
  {
    printf("Cannot open %s\n",argv[1]);
    getch();
    exit(1);
  }
  while((ch=fgetc(fp1))!=EOF)
    fputc(ch,fp2);
  fclose(fp1);
  fclose(fp2);
}
```

本程序为带参的 main 函数。程序中定义了两个文件指针 fp1 和 fp2，分别指向命令行参数中给出的文件。如命令行参数中没有给出文件名，则给出提示信息。程序第 18 行表示如果只给出一个文件名，则使 fp2 指向标准输出文件(即显示器)。程序第 25 行至 28 行用循环语句逐个读出文件 1 中的字符再送到文件 2 中。再次运行时，给出了一个文件名，故输出给标准输出文件 stdout，即在显示器上显示文件内容。第三次运行，给出了二个文件名，因此把 string 中的内容读出，写入到 OK 之中。可用 DOS 命令 type 显示 OK 的内容。

1.1.1字符串读写函数 fgets 和 fputs

1. **读字符串函数 fgets**

函数的功能是从指定的文件中读一个字符串到字符数组中，函数调用的形式为：

fgets(字符数组名,n,文件指针);

其中的 n 是一个正整数。表示从文件中读出的字符串不超过 n-1 个字符。在读入的最后一个字符后加上串结束标志'\0'。

例如：

fgets(str,n,fp);

的意义是从 fp 所指的文件中读出 n-1 个字符送入字符数组 str 中。

【例 13.4】从 string 文件中读入一个含 10 个字符的字符串。

```
#include<stdio.h>
main()
{
  FILE *fp;
  char str[11];
  if((fp=fopen("d:\\jrzh\\example\\string","rt"))==NULL)
  {
    printf("\nCannot open file strike any key exit!");
    getch();
    exit(1);
  }
  fgets(str,11,fp);
  printf("\n%s\n",str);
  fclose(fp);
}
```

本例定义了一个字符数组 str 共 11 个字节，在以读文本文件方式打开文件 string 后，从中读出 10 个字符送入 str 数组，在数组最后一个单元内将加上'\0'，然后在屏幕上显示输出 str 数组。输出的十个字符正是例 13.1 程序的前十个字符。

对 fgets 函数有两点说明：

1) 在读出 n-1 个字符之前，如遇到了换行符或 EOF，则读出结束。

2) fgets 函数也有返回值，其返回值是字符数组的首地址。

2. **写字符串函数 fputs**

fputs 函数的功能是向指定的文件写入一个字符串，其调用形式为：

fputs(字符串,文件指针);

其中字符串可以是字符串常量，也可以是字符数组名，或指针变量，例如：

fputs("abcd",fp);

其意义是把字符串“abcd"写入 fp 所指的文件之中。

【例 13.5】在例 13.2 中建立的文件 string 中追加一个字符串。

```
#include<stdio.h>
main()
```

```
{
  FILE *fp;
  char ch,st[20];
  if((fp=fopen("string","at+"))==NULL)
  {
    printf("Cannot open file strike any key exit!");
    getch();
    exit(1);
}
  printf("input a string:\n");
  scanf("%s",st);
  fputs(st,fp);
  rewind(fp);
  ch=fgetc(fp);
  while(ch!=EOF)
  {
    putchar(ch);
    ch=fgetc(fp);
  }
  printf("\n");
  fclose(fp);
}
```

本例要求在 string 文件末加写字符串，因此，在程序第 6 行以追加读写文本文件的方式打开文件 string。然后输入字符串，并用 fputs 函数把该串写入文件 string。在程序 15 行用 rewind 函数把文件内部位置指针移到文件首。再进入循环逐个显示当前文件中的全部内容。

1.1.1数据块读写函数 fread 和 fwtrite

C 语言还提供了用于整块数据的读写函数。可用来读写一组数据，如一个数组元素，一个结构变量的值等。

读数据块函数调用的一般形式为：

```
fread(buffer,size,count,fp);
```

写数据块函数调用的一般形式为：

```
fwrite(buffer,size,count,fp);
```

其中：

buffer　是一个指针，在 fread 函数中，它表示存放输入数据的首地址。在 fwrite 函数中，它表示存放输出数据的首地址。

size　表示数据块的字节数。

count　表示要读写的数据块块数。

fp　　表示文件指针。

例如：

fread(fa,4,5,fp);

其意义是从 fp 所指的文件中，每次读 4 个字节(一个实数)送入实数组 fa 中，连续读 5 次，即读 5 个实数到 fa 中。

【例 13.6】从键盘输入两个学生数据，写入一个文件中，再读出这两个学生的数据显示在屏幕上。

```
#include<stdio.h>
struct stu
{
  char name[10];
  int num;
  int age;
  char addr[15];
}boya[2],boyb[2],*pp,*qq;
main()
{
  FILE *fp;
  char ch;
  int i;
  pp=boya;
  qq=boyb;
  if((fp=fopen("d:\\jrzh\\example\\stu_list","wb+"))==NULL)
  {
    printf("Cannot open file strike any key exit!");
    getch();
    exit(1);
  }
  printf("\ninput data\n");
  for(i=0;i<2;i++,pp++)
  scanf("%s%d%d%s",pp->name,&pp->num,&pp->age,pp->addr);
  pp=boya;
  fwrite(pp,sizeof(struct stu),2,fp);
  rewind(fp);
  fread(qq,sizeof(struct stu),2,fp);
  printf("\n\nname\tnumber      age      addr\n");
  for(i=0;i<2;i++,qq++)
  printf("%s\t%5d%7d     %s\n",qq->name,qq->num,qq->age,qq->addr);
  fclose(fp);
}
```

本例程序定义了一个结构 stu,说明了两个结构数组 boya 和 boyb 以及两个结构指针变量 pp 和 qq。pp 指向 boya,qq 指向 boyb。程序第 16 行以读写方式打开二进制文件"stu_list",输入二个学生数据之后，写入该文件中，然后把文件内部位置指针移到文件首，读出两块学生数据后，在屏幕上显示。

1.1.1格式化读写函数 fscanf 和 fprintf

fscanf 函数，fprintf 函数与前面使用的 scanf 和 printf 函数的功能相似，都是格式化读写函数。两者的区别在于 fscanf 函数和 fprintf 函数的读写对象不是键盘和显示器，而是磁盘文件。

这两个函数的调用格式为：

fscanf(文件指针,格式字符串,输入表列);

fprintf(文件指针,格式字符串,输出表列);

例如：

```
fscanf(fp,"%d%s",&i,s);
fprintf(fp,"%d%c",j,ch);
```

用 fscanf 和 fprintf 函数也可以完成例 10.6 的问题。修改后的程序如例 10.7 所示。

【例 13.7】用 fscanf 和 fprintf 函数成例 10.6 的问题。

```
#include<stdio.h>
struct stu
{
  char name[10];
  int num;
  int age;
  char addr[15];
}boya[2],boyb[2],*pp,*qq;
main()
{
  FILE *fp;
  char ch;
  int i;
  pp=boya;
  qq=boyb;
  if((fp=fopen("stu_list","wb+"))==NULL)
  {
    printf("Cannot open file strike any key exit!");
    getch();
    exit(1);
  }
  printf("\ninput data\n");
  for(i=0;i<2;i++,pp++)
    scanf("%s%d%d%s",pp->name,&pp->num,&pp->age,pp->addr);
  pp=boya;
  for(i=0;i<2;i++,pp++)
    fprintf(fp,"%s %d %d %s\n",pp->name,pp->num,pp->age,pp->
            addr);
  rewind(fp);
  for(i=0;i<2;i++,qq++)
```

```
    fscanf(fp,"%s %d %d %s\n",qq->name,&qq->num,&qq->age,qq->addr);
  printf("\n\nname\tnumber       age       addr\n");
  qq=boyb;
  for(i=0;i<2;i++,qq++)
    printf("%s\t%5d  %7d       %s\n",qq->name,qq->num,  qq->age,
           qq->addr);
  fclose(fp);
}
```

与例 10.6 相比，本程序中 fscanf 和 fprintf 函数每次只能读写一个结构数组元素，因此采用了循环语句来读写全部数组元素。还要注意指针变量 pp,qq 由于循环改变了它们的值，因此在程序的 25 和 32 行分别对它们重新赋予了数组的首地址。

1.1 文件的随机读写

前面介绍的对文件的读写方式都是顺序读写，即读写文件只能从头开始，顺序读写各个数据。 但在实际问题中常要求只读写文件中某一指定的部分。为了解决这个问题可移动文件内部的位置指针到需要读写的位置，再进行读写，这种读写称为随机读写。

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。

1.1.1文件定位

移动文件内部位置指针的函数主要有两个，即 rewind 函数和 fseek 函数。

rewind 函数前面已多次使用过，其调用形式为：

```
rewind(文件指针);
```

它的功能是把文件内部的位置指针移到文件首。

下面主要介绍 fseek 函数。

fseek 函数用来移动文件内部位置指针，其调用形式为：

```
fseek(文件指针,位移量,起始点);
```

其中：

“文件指针”指向被移动的文件。

“位移量”表示移动的字节数，要求位移量是 long 型数据，以便在文件长度大于 64KB 时不会出错。当用常量表示位移量时，要求加后缀“L”。

“起始点”表示从何处开始计算位移量，规定的起始点有三种：文件首，当前位置和文件尾。其表示方法如下表。

起始点	表示符号	数字表示
文件首	SEEK_SET	0
当前位置	SEEK_CUR	1
文件末尾	SEEK_END	2

例如：

```
fseek(fp,100L,0);
```

其意义是把位置指针移到离文件首 100 个字节处。

还要说明的是 fseek 函数一般用于二进制文件。在文本文件中由于要进行转换，故往往计算的位置会出现错误。

1.1.1文件的随机读写

在移动位置指针之后，即可用前面介绍的任一种读写函数进行读写。由于一般是读写一个数据据块，因此常用 fread 和 fwrite 函数。

下面用例题来说明文件的随机读写。

【例 13.8】在学生文件 stu_list 中读出第二个学生的数据。

```
#include<stdio.h>
struct stu
{
  char name[10];
  int num;
  int age;
  char addr[15];
}boy,*qq;
main()
{
  FILE *fp;
  char ch;
  int i=1;
  qq=&boy;
  if((fp=fopen("stu_list","rb"))==NULL)
  {
    printf("Cannot open file strike any key exit!");
    getch();
    exit(1);
  }
  rewind(fp);
  fseek(fp,i*sizeof(struct stu),0);
  fread(qq,sizeof(struct stu),1,fp);
  printf("\n\nname\tnumber      age      addr\n");
  printf("%s\t%5d  %7d      %s\n",qq->name,qq->num,qq->age,
         qq->addr);
}
```

文件 stu_list 已由例 13.6 的程序建立，本程序用随机读出的方法读出第二个学生的数据。程序中定义 boy 为 stu 类型变量，qq 为指向 boy 的指针。以读二进制文件方式打开文件，程序第 22 行移动文件位置指针。其中的 i 值为 1，表示从文件头开始，移动一个 stu 类型的长度，然后再读出的数据即为第二个学生的数据。

1.1 文件检测函数

C 语言中常用的文件检测函数有以下几个。

1.1.1文件结束检测函数 feof 函数

调用格式：

feof(文件指针);

功能：判断文件是否处于文件结束位置，如文件结束，则返回值为 1，否则为 0。

1.1.1读写文件出错检测函数

ferror 函数调用格式：

ferror(文件指针);

功能：检查文件在用各种输入输出函数进行读写时是否出错。如 ferror 返回值为 0 表示未出错，否则表示有错。

1.1.1文件出错标志和文件结束标志置 0 函数

clearerr 函数调用格式：

clearerr(文件指针);

功能：本函数用于清除出错标志和文件结束标志，使它们为 0 值。

1.1 C 库文件

C 系统提供了丰富的系统文件，称为库文件，C 的库文件分为两类，一类是扩展名为".h"的文件，称为头文件，在前面的包含命令中我们已多次使用过。在".h"文件中包含了常量定义、 类型定义、宏定义、函数原型以及各种编译选择设置等信息。另一类是函数库，包括了各种函数的目标代码，供用户在程序中调用。 通常在程序中调用一个库函数时，要在调用之前包含该函数原型所在的".h" 文件。

下面给出 Turbo C 的全部".h"文件。

Turbo C 头文件

- ALLOC.H　　说明内存管理函数(分配、释放等)。
- ASSERT.H　　定义 assert 调试宏。
- BIOS.H　　说明调用 IBM—PC ROM BIOS 子程序的各个函数。
- CONIO.H　　说明调用 DOS 控制台 I/O 子程序的各个函数。
- CTYPE.H　　包含有关字符分类及转换的名类信息(如 isalpha 和 toascii 等)。
- DIR.H　　包含有关目录和路径的结构、宏定义和函数。
- DOS.H　　定义和说明 MSDOS 和 8086 调用的一些常量和函数。
- ERRON.H　　定义错误代码的助记符。
- FCNTL.H　　定义在与 open 库子程序连接时的符号常量。

- FLOAT.H　　包含有关浮点运算的一些参数和函数。
- GRAPHICS.H　说明有关图形功能的各个函数，图形错误代码的常量定义，正对不同驱动程序的各种颜色值，及函数用到的一些特殊结构。
- IO.H　　包含低级 I/O 子程序的结构和说明。
- LIMIT.H　　包含各环境参数、编译时间限制、数的范围等信息。
- MATH.H　　说明数学运算函数，还定了 HUGE VAL 宏， 说明了 matherr 和 matherr 子程序用到的特殊结构。
- MEM.H　　说明一些内存操作函数(其中大多数也在 STRING.H 中说明)。
- PROCESS.H　说明进程管理的各个函数，spawn…和 EXEC …函数的结构说明。
- SETJMP.H　定义 longjmp 和 setjmp 函数用到的 jmp buf 类型,说明这两个函数。
- SHARE.H　　定义文件共享函数的参数。
- SIGNAL.H　　定义 SIG[ZZ(Z) [ZZ)]IGN 和 SIG[ZZ(Z) [ZZ)]DFL 常量，说明 rajse 和 signal 两个函数。
- STDARG.H　　定义读函数参数表的宏。(如 vprintf,vscarf 函数)。
- STDDEF.H　　定义一些公共数据类型和宏。
- STDIO.H　　定义 Kernighan 和 Ritchie 在 Unix System V 中定义的标准和扩展的类型和宏。还定义标准 I/O 预定义流：stdin,stdout 和 stderr，说明 I/O 流子程序。
- STDLIB.H　　说明一些常用的子程序：转换子程序、搜索/ 排序子程序等。
- STRING.H　　说明一些串操作和内存操作函数。
- SYS\STAT.H　定义在打开和创建文件时用到的一些符号常量。
- SYS\TYPES.H　说明 ftime 函数和 timeb 结构。
- SYS\TIME.H　定义时间的类型 time[ZZ(Z) [ZZ)]t。
- TIME.H　　定义时间转换子程序 asctime、localtime 和 gmtime 的结构，ctime、difftime、 gmtime、 localtime 和 stime 用到的类型，并提供这些函数的原型。
- VALUE.H　　定义一些重要常量，包括依赖于机器硬件的和为与 Unix System V 相兼容而说明的一些常量，包括浮点和双精度值的范围。

1.1 本章小结

1. C 系统把文件当作一个“流"，按字节进行处理。
2. C 文件按编码方式分为二进制文件和 ASCII 文件。
3. C 语言中，用文件指针标识文件，当一个文件被 打开时，可取得该文件指针。
4. 文件在读写之前必须打开，读写结束必须关闭。
5. 文件可按只读、只写、读写、追加四种操作方式打开，同时还必须指定文件的类型是二进制文件还是文本文件。
6. 文件可按字节，字符串，数据块为单位读写，文件也可按指定的格式进行读写。
7. 文件内部的位置指针可指示当前的读写位置，移动该指针可以对文件实现随机读写。