# 13.1C语言文件

## 1,两种文件存取方式(输入,输出方式)

顺序存取

直接存取

## 2,数据的两种存放形式

文本文件

二进制文件

## 13.2文件指针

定义文件类型指针变量的一般形式:

FILE \*指针变量名;

例如:

FILE \*fp1,\*fp2;

## 13.3打开文件

在使用文件之前,需打开文件.在C里使用fopen函数打开文件.格式为:

fopen(文件名,文件使用方式);

此函数返回一个指向FILE类型的指针.如:

FILE \*fp;

fp=fopen("file\_1","r");

如果调用成功,fp就指向file\_1,否则返回为NULL,所以为了保证文件的正确使用,要进行测试.采用如下语句:

If((fp=fopen("file\_1","r"))==NULL)

{

printf("Cannot open this file\n");

exit(0);

}

## 最常用的文件使用方式及其含义如下:

1,"r".为读而打开文本文件.(不存在则出错)

2,"rb".为读而打开二进制文件.

3,"w".为写而打开文本文件.(若不存在则新建,反之,则从文件起始位置写,原内容将被覆盖)

4,"wb".为写而打开二进制文件.

5,"a".为在文件后面添加数据而打开文本文件.(若不存在,则新建;反之,在原文件后追加)

6,"ab".为在文件后面添加数据而打开一个二进制文件.

最常用的文件使用方式及其含义如下:

7,"r+".为读和写而打开文本文件.(读时,从头开始;在写数据时,新数据只覆盖所占的空间,其后不变)

8,"rb+".为读和写而打开二进制文件.只是在随后的读写时,可以由位置函数设置读和写的起始位置.

9,"w+".首先建立一个新文件,进行写操作,随后可以从头开始读.(若文件存在,原内容将全部消失)

10,"wb+".功能与"w+"同.只是在随后的读写时,可以由位置函数设置读和写的起始位置.

最常用的文件使用方式及其含义如下:

11,"a+".功能与"a"相同;只是在文件尾部添加新的数据后,可以从头开始读.

12,"ab+".功能与"a+"相同;只是在文件尾部添加新数据之后,可以由位置函数设置开始读的起始位置.

## 13.4关闭文件

当文件的读写操作完成之后,使用fclose函数关闭文件.格式如下:

fclose(文件指针)

如:fclose(fp);

## 13.5调用getc(fgetc)和putc(fputc)函数进行输入和输出

1,调用putc(或fputc)函数输出一个字符

调用形式为:

putc(ch,fp);

功能是:将字符ch写到文件指针fp所指的文件中去.当输出成功,putc函数返回所输出的字符;否则,返回一个EOF值.EOF是在stdio.h库函数文件中定义的符号常量,其值等于-1.

13.5调用getc(fgetc)和putc(fputc)函数进行输入和输出

例如:把从键盘输入的文本按原样输出到名为file\_1.dat文件中,用字符@作为键盘输入结束标志.

#include

Void main()

{

FILE \*fpout;

char ch;

if(fpout=fpopen("file\_1","w")==NULL)

{

printf("Cannot open this file!\n");

exit(0);

}

ch=getchar();

while(ch!='@')

{ fputc(ch,fpout); ch=getchar(); }

fclose(fpout);

}

### 2.调用getc(或fgetc)函数输入一个字符

调用形式为:

ch=getc(pf);

功能是:从pf指定的文件中读如一个字符,并把它作为函数值返回.

例如:把一个已存在磁盘上的file\_1.dat文本文件中的内容,原样输出到终端屏幕上.

#include

void main(){

FILE \*fpin;

char ch;

if((fpin=fopen("file\_1.dat","r"))==NULL)

{ printf("Cann't open this file!\n");exit(0);}

ch=fgetc(fpin);

while (ch!=EOF)

{ putchar(ch); ch=fgetc(fpin);}

fclose(fpin);

}

## 13.6判断文件结束函数feof

EOF可以作为文本文件的结束 标志,但不能作为二进制文件的结束符.feof函数既可以判断二进制文件,又可以判断文本文件.

例:编写程序,用于把一个文本文件(源)复制到另一个文件(目的)中,源文件名和目的文件名由命令行输入.命令形式如下:

可执行程序名 源文件名 目的文件名

#include

void filecopy(FILE\* ,FILE \*);

void main(int argc,char \*argv[]){

FILE \*fpin,\*fpout;

if(argc==3)

{ fpin=fopen(argv[1],"r");

fpout=fopen(argv[2],"w");

filecopy(fpin,fpout);

fclose(fpin);fclose(fpout);

}

else if(argc>3)

printf("The file names too many!!\n";

else

printf("There are no file names for input or output!!\n );

}

void filecopy(FILE \*fpin,FILE \*fpout)

{

char ch;

ch=getc(fpin);

while(!feof(fpin))

{putc(ch,fpout); ch=getc(fpin);}

}

## 13.7fscanf函数和fprintf函数

### 1,fscanf函数

fscanf只能从文本文件中按格式输入,和scanf函数相似,只不过输入的对象是磁盘上文本文件中的数据.调用形式为:

fscanf(文件指针,格式控制字符串,输入项表)

例如:fscanf(fp,"%d%d",&a,&b);

fscanf(stdin,"%d%d",&a,&b);

等价于scanf("%d%d",&a,&b);

### 3.fprintf函数

fprintf函数按格式将内存中的数据转换成对应的字符,并以ASCII代码形式输出到文本文件中.Fprintf函数和printf函数相似,只是将输出的内容按格式存放到磁盘的文本文件中.调用形式如下:

fprintf(文件指针,格式控制字符串,输出项表)

如:fprintf(fp,"%d %d",x,y);

以下语句 fprintf(stdout,"%d %d",x,y)

## 13.8fgets函数和fputs函数

### 1,fgets函数

fgets函数用来从文件中读入字符串.调用形式如下:

fgets(str,n,fp);

函数功能是:从fp所指文件中读入n-1个字符放入str为起始地址的空间内;如果在未读满n-1个字符时,则遇到换行符或一个EOF结束本次读操作,并已str作为函数值返回.

## 13.8fgets函数和fputs函数

### 2,fputs函数

fput函数把字符串输出到文件中.函数调用形式如下:

fputs(str,fp);

注意:为了便于读入,在输出字符串时,应当人为的加诸如"\n"这样的字符串.

#include <stdio.h>

#include <iostream>

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

{

char arr[10] ;

char \*ap = "hello!" ;

FILE \*fp ;

if ((fp = fopen("hello.txt", "wt+")) == NULL)

{

printf("error!") ;

exit(1) ;

}

fputs(ap, fp) ;

rewind(fp) ; //

fgets(arr, 10, fp) ;

printf("%s\n", arr) ;

fclose(fp) ;

return 0 ;

}

## 13.9fread函数和fwrite函数

例如有如下结构体:

struct st{

char num[8];

float mk[5];

}pers[30];

以下循环将把这30个元素中的数据输出到fp所指文件中.

for(i=0;i<30;i++)

fwrite(&pers[i],sizeof(struct st),1,fp);

## 13.9fread函数和fwrite函数

以下语句从fp所指的文件中再次将每个学生数据逐个读入到pers数组中.

i=0;

fread(&pers[i],sizeof(struct st),1,fp);

while(!feof(fp))

{ i++;

fread(&pers[i],sizeof(struct st),1,fp);

}

## 13.10文件定位函数

### 1,fseek函数

fseek函数用来移动文件位置指针到指定的位置上,接着的读或写操作将从此位置开始.函数的调用形式如下:

fseek(pf,offset,origin)

pf:文件指针

offset:以字节为单位的位移量,为长整形.

origin:是起始点,用来指定位移量是以哪个位置为基准的.

### 1,fseek函数

位移量的表示方法

标识符 数字 代表的起始点

SEEK\_SET 0 文件开始

SEEK\_END 2 文件末尾

SEEK\_CUR 1 文件当前位置

假设pf已指向一个二进制文件,则;

fseek(pf,30L,SEEK\_SET)

fseek(pf,-10L\*sizeof(int),SEEK\_END)

对于文本文件,位移量必须是0;如:

fseek(pf,0L,SEEK\_SET)

fseek(pf,0L,SEEK\_END)

### 2. ftell函数

ftell函数用以获得文件当前位置指针的位置,函数给出当前位置指针相对于文件开头的字节数.如;

long t;

t=ftell(pf);

当函数调用出错时,函数返回-1L.

我们可以通过以下方式来测试一个文件的长度:

fseek(fp,0L,SEEK\_END);

t=ftell(fp);

### 3.rewind函数

调用形式为:

rewind(pf);

函数没有返回值.函数的功能是使文件的位置指针回到文件的开头.

## 13.10文件应用

在磁盘上的test.txt文件中放有10个不小于2的正整数,用函数调用方式编写程序.要求实现:

1,在被调函数prime中,判断和统计10个整数中的素数以及个数.

2,在主函数中将全部素数追加到磁盘文件test.txt的尾部,同时输出到屏幕上.

#include

#include

Int prime(int a[],int n)

{

int I,j,k=0,flag=0;

for(i=0;i { for(j=2;j if(a[i]%j==0)

{ flag=0; break;}

else flag=1;

if(flag)

{a[k]=a[i];k++;}

}

return k;

}

void main(){

int n,I,a[10];

FILE \*fp;

fp=fopen("test1-2.txt","r+");

for(n=0;n<10;n++)

fscanf(fp,"%d",&a[n]);

n=prime(a,n);

fseek(fp,o,2);

for(i=0;i {printf("%3d",a[i]);

fprintf(fp,"%3d",a[i]);

}

fclose(fp);

}

# 文件操作函数全

### clearerr（清除文件流的错误旗标）

相关函数 feof

表头文件 #include<stdio.h>

定义函数 void clearerr(FILE \* stream);

函数说明 clearerr（）清除参数stream指定的文件流所使用的错误旗标。

返回值

### fclose（关闭文件）

相关函数 close，fflush，fopen，setbuf

表头文件 #include<stdio.h>

定义函数 int fclose(FILE \* stream);

函数说明 fclose()用来关闭先前fopen()打开的文件。此动作会让缓冲区内的数据写入文件中，并释放系统所提供的文件资源。

返回值 若关文件动作成功则返回0，有错误发生时则返回EOF并把错误代码存到errno。

错误代码 EBADF表示参数stream非已打开的文件。

范例 请参考fopen（）。

### fdopen（将文件描述词转为文件指针）

相关函数 fopen，open，fclose

表头文件 #include<stdio.h>

定义函数 FILE \* fdopen(int fildes,const char \* mode);

函数说明 fdopen()会将参数fildes的文件描述词，转换为对应的文件指针后返回。参数mode字符串则代表着文件指针的流形态，此形态必须和原先文件描述词读写模式相同。关于mode字符串格式请参考fopen()。

返回值 转换成功时返回指向该流的文件指针。失败则返回NULL，并把错误代码存在errno中。

范例

#include<stdio.h>

main()

{

FILE \* fp =fdopen(0,”w+”);

fprintf(fp,”%s\n”,”hello!”);

fclose(fp);

}

执行 hello!

### feof（检查文件流是否读到了文件尾）

相关函数 fopen，fgetc，fgets，fread

表头文件 #include<stdio.h>

定义函数 int feof(FILE \* stream);

函数说明 feof()用来侦测是否读取到了文件尾，尾数stream为fopen（）所返回之文件指针。如果已到文件尾则返回非零值，其他情况返回0。

返回值 返回非零值代表已到达文件尾。

### fflush（更新缓冲区）

相关函数 write，fopen，fclose，setbuf

表头文件 #include<stdio.h>

定义函数 int fflush(FILE\* stream);

函数说明 fflush()会强迫将缓冲区内的数据写回参数stream指定的文件中。如果参数stream为NULL，fflush()会将所有打开的文件数据更新。

返回值 成功返回0，失败返回EOF，错误代码存于errno中。

错误代码 EBADF 参数stream指定的文件未被打开，或打开状态为只读。其它错误代码参考write（）。

### fgetc（由文件中读取一个字符）

相关函数 open，fread，fscanf，getc

表头文件 include<stdio.h>

定义函数 nt fgetc(FILE \* stream);

函数说明 fgetc()从参数stream所指的文件中读取一个字符。若读到文件尾而无数据时便返回EOF。

返回值 getc()会返回读取到的字符，若返回EOF则表示到了文件尾。

范例

#include<stdio.h>

main()

{

FILE \*fp;

int c;

fp=fopen(“exist”,”r”);

while((c=fgetc(fp))!=EOF)

printf(“%c”,c);

fclose(fp);

}

### fgets（由文件中读取一字符串）

相关函数 open，fread，fscanf，getc

表头文件 include<stdio.h>

定义函数 har \* fgets(char \* s,int size,FILE \* stream);

函数说明 fgets()用来从参数stream所指的文件内读入字符并存到参数s所指的内存空间，直到出现换行字符、读到文件尾或是已读了size-1个字符为止，最后会加上NULL作为字符串结束。

返回值 gets()若成功则返回s指针，返回NULL则表示有错误发生。

范例

#include<stdio.h>

main()

{

char s[80];

fputs(fgets(s,80,stdin),stdout);

}

执行 this is a test /\*输入\*/

this is a test /\*输出\*/

### fileno（返回文件流所使用的文件描述词）

相关函数 open，fopen

表头文件 #include<stdio.h>

定义函数 int fileno(FILE \* stream);

函数说明 fileno()用来取得参数stream指定的文件流所使用的文件描述词。

返回值 返回文件描述词。

范例

#include<stdio.h>

main()

{

FILE \* fp;

int fd;

fp=fopen(“/etc/passwd”,”r”);

fd=fileno(fp);

printf(“fd=%d\n”,fd);

fclose(fp);

}

执行 fd=3

### fopen（打开文件）

相关函数 open，fclose

表头文件 #include<stdio.h>

定义函数 FILE \* fopen(const char \* path,const char \* mode);

函数说明 参数path字符串包含欲打开的文件路径及文件名，参数mode字符串则代表着流形态。

mode有下列几种形态字符串:

r 打开只读文件，该文件必须存在。

r+ 打开可读写的文件，该文件必须存在。

w 打开只写文件，若文件存在则文件长度清为0，即该文件内容会消失。若文件不存在则建立该文件。

w+ 打开可读写文件，若文件存在则文件长度清为零，即该文件内容会消失。若文件不存在则建立该文件。

a 以附加的方式打开只写文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾，即文件原先的内容会被保留。

a+ 以附加方式打开可读写的文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾后，即文件原先的内容会被保留。

上述的形态字符串都可以再加一个b字符，如rb、w+b或ab＋等组合，加入b字符用来告诉函数库打开的文件为二进制文件，而非纯文字文件。不过在POSIX系统，包含Linux都会忽略该字符。由fopen()所建立的新文件会具有S\_IRUSR|S\_IWUSR|S\_IRGRP|S\_IWGRP|S\_IROTH|S\_IWOTH(0666)权限，此文件权限也会参考umask值。

返回值 文件顺利打开后，指向该流的文件指针就会被返回。若果文件打开失败则返回NULL，并把错误代码存在errno中。

附加说明 一般而言，开文件后会作一些文件读取或写入的动作，若开文件失败，接下来的读写动作也无法顺利进行，所以在fopen()后请作错误判断及处理。

范例

#include<stdio.h>

main()

{

FILE \* fp;

fp=fopen(“noexist”,”a+”);

if(fp= =NULL) return;

fclose(fp);

}

### fputc（将一指定字符写入文件流中）

相关函数 fopen，fwrite，fscanf，putc

表头文件 #include<stdio.h>

定义函数 int fputc(int c,FILE \* stream);

函数说明 fputc 会将参数c转为unsigned char后写入参数stream指定的文件中。

返回值 fputc()会返回写入成功的字符，即参数c。若返回EOF则代表写入失败。

范例

#include<stdio.h>

main()

{

FILE \* fp;

char a[26]=”abcdefghijklmnopqrstuvwxyz”;

int i;

fp= fopen(“noexist”,”w”);

for(i=0;i<26;i++)

fputc(a,fp);

fclose(fp);

}

### fputs（将一指定的字符串写入文件内）

相关函数 fopen，fwrite，fscanf，fputc，putc

表头文件 #include<stdio.h>

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

函数说明 fputs()用来将参数s所指的字符串写入到参数stream所指的文件内。

返回值 若成功则返回写出的字符个数，返回EOF则表示有错误发生。

范例 请参考fgets（）。

### fread（从文件流读取数据）

相关函数 fopen，fwrite，fseek，fscanf

表头文件 #include<stdio.h>

定义函数 size\_t fread(void \* ptr,size\_t size,size\_t nmemb,FILE \* stream);

函数说明 fread()用来从文件流中读取数据。参数stream为已打开的文件指针，参数ptr指向欲存放读取进来的数据空间，读取的字符数以参数size\*nmemb来决定。Fread()会返回实际读取到的nmemb数目，如果此值比参数nmemb来得小，则代表可能读到了文件尾或有错误发生，这时必须用feof()或ferror()来决定发生什么情况。

返回值 返回实际读取到的nmemb数目。

附加说明

范例

#include<stdio.h>

#define nmemb 3

struct test

{

char name[20];

int size;

}s[nmemb];

int main(){

FILE \* stream;

int i;

stream = fopen(“/tmp/fwrite”,”r”);

fread(s,sizeof(struct test),nmemb,stream);

fclose(stream);

for(i=0;i<nmemb;i++)

printf(“name[%d]=%-20s:size[%d]=%d\n”,i,s.name,i,s.size);

}

执行

name[0]=Linux! size[0]=6

name[1]=FreeBSD! size[1]=8

name[2]=Windows2000 size[2]=11

### freopen（打开文件）

相关函数 fopen，fclose

表头文件 #include<stdio.h>

定义函数 FILE \* freopen(const char \* path,const char \* mode,FILE \* stream);

函数说明 参数path字符串包含欲打开的文件路径及文件名，参数mode请参考fopen()说明。参数stream为已打开的文件指针。Freopen()会将原stream所打开的文件流关闭，然后打开参数path的文件。

返回值 文件顺利打开后，指向该流的文件指针就会被返回。如果文件打开失败则返回NULL，并把错误代码存在errno中。

范例

#include<stdio.h>

main()

{

FILE \* fp;

fp=fopen(“/etc/passwd”,”r”);

fp=freopen(“/etc/group”,”r”,fp);

fclose(fp);

}

### fseek（移动文件流的读写位置）

相关函数 rewind，ftell，fgetpos，fsetpos，lseek

表头文件 #include<stdio.h>

定义函数 int fseek(FILE \* stream,long offset,int whence);

函数说明 fseek()用来移动文件流的读写位置。参数stream为已打开的文件指针，参数offset为根据参数whence来移动读写位置的位移数。

参数 whence为下列其中一种:

SEEK\_SET从距文件开头offset位移量为新的读写位置。SEEK\_CUR以目前的读写位置往后增加offset个位移量。

SEEK\_END将读写位置指向文件尾后再增加offset个位移量。

当whence值为SEEK\_CUR或SEEK\_END时，参数offset允许负值的出现。

下列是较特别的使用方式:

1) 欲将读写位置移动到文件开头时:fseek(FILE \*stream,0,SEEK\_SET);

2) 欲将读写位置移动到文件尾时:fseek(FILE \*stream,0,0SEEK\_END);

返回值 当调用成功时则返回0，若有错误则返回-1，errno会存放错误代码。

附加说明 fseek()不像lseek()会返回读写位置，因此必须使用ftell()来取得目前读写的位置。

范例

#include<stdio.h>

main()

{

FILE \* stream;

long offset;

fpos\_t pos;

stream=fopen(“/etc/passwd”,”r”);

fseek(stream,5,SEEK\_SET);

printf(“offset=%d\n”,ftell(stream));

rewind(stream);

fgetpos(stream,&pos);

printf(“offset=%d\n”,pos);

pos=10;

fsetpos(stream,&pos);

printf(“offset = %d\n”,ftell(stream));

fclose(stream);

}

执行 offset = 5

offset =0

offset=10

### ftell（取得文件流的读取位置）

相关函数 fseek，rewind，fgetpos，fsetpos

表头文件 #include<stdio.h>

定义函数 long ftell(FILE \* stream);

函数说明 ftell()用来取得文件流目前的读写位置。参数stream为已打开的文件指针。

返回值 当调用成功时则返回目前的读写位置，若有错误则返回-1，errno会存放错误代码。

错误代码 EBADF 参数stream无效或可移动读写位置的文件流。

范例 参考fseek()。

### fwrite（将数据写至文件流）

相关函数 fopen，fread，fseek，fscanf

表头文件 #include<stdio.h>

定义函数 size\_t fwrite(const void \* ptr,size\_t size,size\_t nmemb,FILE \* stream);

函数说明 fwrite()用来将数据写入文件流中。参数stream为已打开的文件指针，参数ptr指向欲写入的数据地址，总共写入的字符数以参数size\*nmemb来决定。Fwrite()会返回实际写入的nmemb数目。

返回值 返回实际写入的nmemb数目。

范例

#include<stdio.h>

#define set\_s (x,y) {strcoy(s[x].name,y);s[x].size=strlen(y);}

#define nmemb 3

struct test

{

char name[20];

int size;

}s[nmemb];

main()

{

FILE \* stream;

set\_s(0,”Linux!”);

set\_s(1,”FreeBSD!”);

set\_s(2,”Windows2000.”);

stream=fopen(“/tmp/fwrite”,”w”);

fwrite(s,sizeof(struct test),nmemb,stream);

fclose(stream);

}

执行 参考fread（）。

### getc（由文件中读取一个字符）

相关函数 read，fopen，fread，fgetc

表头文件 #include<stdio.h>

定义函数 int getc(FILE \* stream);

函数说明 getc()用来从参数stream所指的文件中读取一个字符。若读到文件尾而无数据时便返回EOF。虽然getc()与fgetc()作用相同，但getc()为宏定义，非真正的函数调用。

返回值 getc()会返回读取到的字符，若返回EOF则表示到了文件尾。

范例 参考fgetc()。

### getchar（由标准输入设备内读进一字符）

相关函数 fopen，fread，fscanf，getc

表头文件 #include<stdio.h>

定义函数 int getchar(void);

函数说明 getchar()用来从标准输入设备中读取一个字符。然后将该字符从unsigned char转换成int后返回。

返回值 getchar()会返回读取到的字符，若返回EOF则表示有错误发生。

附加说明 getchar()非真正函数，而是getc(stdin)宏定义。

范例

#include<stdio.h>

main()

{

FILE \* fp;

int c,i;

for(i=0li<5;i++)

{

c=getchar();

putchar(c);

}

}

执行 1234 /\*输入\*/

1234 /\*输出\*/

### gets（由标准输入设备内读进一字符串）

相关函数 fopen，fread，fscanf，fgets

表头文件 #include<stdio.h>

定义函数 char \* gets(char \*s);

函数说明 gets()用来从标准设备读入字符并存到参数s所指的内存空间，直到出现换行字符或读到文件尾为止，最后加上NULL作为字符串结束。

返回值 gets()若成功则返回s指针，返回NULL则表示有错误发生。

附加说明 由于gets()无法知道字符串s的大小，必须遇到换行字符或文件尾才会结束输入，因此容易造成缓冲溢出的安全性问题。建议使用fgets()取代。

范例 参考fgets()

### mktemp（产生唯一的临时文件名）

相关函数 tmpfile

表头文件 #include<stdlib.h>

定义函数 char \* mktemp(char \* template);

函数说明 mktemp()用来产生唯一的临时文件名。参数template所指的文件名称字符串中最后六个字符必须是XXXXXX。产生后的文件名会借字符串指针返回。

返回值 文件顺利打开后，指向该流的文件指针就会被返回。如果文件打开失败则返回NULL，并把错误代码存在errno中。

附加说明 参数template所指的文件名称字符串必须声明为数组，如:

char template[ ]=”template-XXXXXX”；

不可用char \* template=”template-XXXXXX”；

范例

#include<stdlib.h>

main()

{

char template[ ]=”template-XXXXXX”;

mktemp(template);

printf(“template=%s\n”,template);

}

### putc（将一指定字符写入文件中）

相关函数 fopen，fwrite，fscanf，fputc

表头文件 #include<stdio.h>

定义函数 int putc(int c,FILE \* stream);

函数说明 putc()会将参数c转为unsigned char后写入参数stream指定的文件中。虽然putc()与fputc()作用相同，但putc()为宏定义，非真正的函数调用。

返回值 putc()会返回写入成功的字符，即参数c。若返回EOF则代表写入失败。

范例 参考fputc（）。

### putchar（将指定的字符写到标准输出设备）

相关函数 fopen，fwrite，fscanf，fputc

表头文件 #include<stdio.h>

定义函数 int putchar (int c);

函数说明 putchar()用来将参数c字符写到标准输出设备。

返回值 putchar()会返回输出成功的字符，即参数c。若返回EOF则代表输出失败。

附加说明 putchar()非真正函数，而是putc(c，stdout)宏定义。

范例 参考getchar()。

### rewind（重设文件流的读写位置为文件开头）

相关函数 fseek，ftell，fgetpos，fsetpos

表头文件 #include<stdio.h>

定义函数 void rewind(FILE \* stream);

函数说明 rewind()用来把文件流的读写位置移至文件开头。参数stream为已打开的文件指针。此函数相当于调用fseek(stream,0,SEEK\_SET)。

返回值

范例 参考fseek()

### setbuf（设置文件流的缓冲区）

相关函数 setbuffer，setlinebuf，setvbuf

表头文件 #include<stdio.h>

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

函数说明 在打开文件流后，读取内容之前，调用setbuf()可以用来设置文件流的缓冲区。参数stream为指定的文件流，参数buf指向自定的缓冲区起始地址。如果参数buf为NULL指针，则为无缓冲IO。Setbuf()相当于调用:setvbuf(stream,buf,buf?\_IOFBF:\_IONBF,BUFSIZ)

返回值

### setbuffer（设置文件流的缓冲区）

相关函数 setlinebuf，setbuf，setvbuf

表头文件 #include<stdio.h>

定义函数 void setbuffer(FILE \* stream,char \* buf,size\_t size);

函数说明 在打开文件流后，读取内容之前，调用setbuffer()可用来设置文件流的缓冲区。参数stream为指定的文件流，参数buf指向自定的缓冲区起始地址，参数size为缓冲区大小。

返回值

### setlinebuf（设置文件流为线性缓冲区）

相关函数 setbuffer，setbuf，setvbuf

表头文件 #include<stdio.h>

定义函数 void setlinebuf(FILE \* stream);

函数说明 setlinebuf()用来设置文件流以换行为依据的无缓冲IO。相当于调用:setvbuf(stream,(char \* )NULL,\_IOLBF,0);请参考setvbuf()。

返回值

### setvbuf（设置文件流的缓冲区）

相关函数 setbuffer，setlinebuf，setbuf

表头文件 #include<stdio.h>

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

函数说明 在打开文件流后，读取内容之前，调用setvbuf()可以用来设置文件流的缓冲区。参数stream为指定的文件流，参数buf指向自定的缓冲区起始地址，参数size为缓冲区大小，参数mode有下列几种

\_IONBF 无缓冲IO

\_IOLBF 以换行为依据的无缓冲IO

\_IOFBF 完全无缓冲IO。如果参数buf为NULL指针，则为无缓冲IO。

返回值

### ungetc（将指定字符写回文件流中）

相关函数 fputc，getchar，getc

表头文件 #include<stdio.h>

定义函数 int ungetc(int c,FILE \* stream);

函数说明 ungetc()将参数c字符写回参数stream所指定的文件流。这个写回的字符会由下一个读取文件流的函数取得。

返回值 成功则返回c字符，若有错误则返回EOF