**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。(数据加到文件尾，即使使用fseek函数改变了文件指针的位置)

复制代码代码如下:

r      Open text file for reading.  The stream is positioned at the beginning of the file.  
r+     Open for reading and writing.  The stream is positioned at the beginning of the file.  
w      Truncate file to zero length or create text file for writing.  The stream is positioned at the beginning of the file.  
w+     Open for reading and writing.  The file is created if it does not exist, otherwise it is truncated.  The  stream  is  posi‐  
       tioned at the beginning of the file.  
a      Open  for  appending  (writing at end of file).  The file is created if it does not exist.  The stream is positioned at the  
       end of the file.  
a+     Open for reading and appending (writing at end of file).  The file is created if it does not exist.  The initial file posi‐  
       tion for reading is at the beginning of the file, but output is always appended to the end of the file.

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**

复制代码代码如下:

#include<stdio.h>  
main()  
{  
FILE \* fp;  
fp=fopen(“noexist”,”a+”);  
if(fp= =NULL) return;  
fclose(fp);  
}

**1. fprintf**功能：传送格式化输出到一个文件中  
表头文件：#include<stdio.h>  
函数原型：int fprintf(FILE \*stream, char \*format[, argument,...]);  
FILE\* 一个FILE型的指针  
char\* 格式化输入函数，和printf里的格式一样  
返回值：成功时返回转换的字节数，失败时返回一个负数  
fp = fopen("/local/test.c","a+");  
fprintf(fp,"%s\n",str);  
  
**2. fscanf**功能：从一个流中执行格式化输入  
表头文件：#include<stdio.h>  
函数原型：int fscanf(FILE \*stream, char \*format[,argument...]);  
FILE\* 一个FILE型的指针  
char\* 格式化输出函数，和scanf里的格式一样  
返回值：成功时返回转换的字节数，失败时返回一个负数  
fp = fopen("/local/test.c","a+");  
fscanf(fp,"%s",str);  
  
**3. 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  
　  
**4.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（）。  
　  
**5.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**

复制代码代码如下:

#include<stdio.h>  
main()  
{  
FILE \* fp =fdopen(0,”w+”);  
fprintf(fp,”%s/n”,”hello!”);  
fclose(fp);  
}  
执行 hello!

**6.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。  
　  
**7.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（）。  
　  
**8.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**

复制代码代码如下:

#include<stdio.h>  
main()  
{  
FILE \*fp;  
int c;  
fp=fopen(“exist”,”r”);  
while((c=fgetc(fp))!=EOF)  
printf(“%c”,c);  
fclose(fp);  
}

**9.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 char \* 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**

复制代码代码如下:

#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\*/

**10.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**

复制代码代码如下:

#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

**12.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**

复制代码代码如下:

#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);  
}

**13.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**

**注意：**

size\_t fread(ptr, 256, 1, fp); 与

size\_t fread(ptr, 1, 256, fp); 的区别

它们都是读取256个，但是当总共只有128个字节时, 它们的返回值不同。

fread(ptr, 256, 1, fp) 返回0。 fread(ptr, 1, 256, fp)返回128.一般情况下建议用第二种。

**范例**

复制代码代码如下:

#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

**14.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**

复制代码代码如下:

#include<stdio.h>  
main()  
{  
FILE \* fp;  
fp=fopen(“/etc/passwd”,”r”);  
fp=freopen(“/etc/group”,”r”,fp);  
fclose(fp);  
}

**15.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**

复制代码代码如下:

#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

**16.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()。**　  
**17.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。(size为每次写的数据所占的大小，nmemb为写的次数)  
返回值 返回实际写入的nmemb次数。  
**范例**

复制代码代码如下:

#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（）。**

**18.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()。**　  
**19.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**

复制代码代码如下:

#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\*/

**20.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()**　  
**21.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**

复制代码代码如下:

#include<stdlib.h>  
main()  
{  
char template[ ]=”template-XXXXXX”;  
mktemp(template);  
printf(“template=%s/n”,template);  
}

**22.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（）。**　  
**23.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()。**　  
**24.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()  
  
25.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  
　  
**26.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  
  
**27.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  
  
**28.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  
  
**29.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。

复制代码代码如下:

#include <stdio.h>  
#include <stdlib.h>  
int main()  
{  
     FILE \*fp = NULL;  
     char\* str;  
     char re;   
     int num = 10;  
     str = (char\*)malloc(100);  
     //snprintf(str, 10,"test: %s", "0123456789012345678");  
    // printf("str=%s\n", str);  
     fp = fopen("/local/test.c","a+");  
     if (fp==NULL){  
        printf("Fail to open file\n");  
     }  
//     fseek(fp,-1,SEEK\_END);  
     num = ftell(fp);  
     printf("test file long:%d\n",num);  
     fscanf(fp,"%s",str);  
     printf("str = %s\n",str);  
     printf("test a: %s\n",str);  
     while ((re=getc(fp))!=EOF){//getc可以用作fgetc用  
        printf("%c",re);  
     }  
     //fread(str,10,10,fp);  
     fgets(str,100,fp);  
     printf("test a: %s\n",str);  
     sprintf(str,"xiewei test is:%s", "ABCDEFGHIGKMNI");  
     printf("str2=%s\n", str);  
   //  fprintf(fp,"%s\n",str);  
     fwrite(str,2,10,fp);  
     num = ftell(fp);  
     if(str!=NULL){  
        free(str);  
     }  
     fclose(fp);  
     return 0;  
}

用fopen打开文件时，以r+,w+,a+方式打开的文件都可以读和写，以r+方式打开文件，文件位置指针在文件开始处，所以写操作覆盖原来的内容；以w+方式打开文件时，如果文件存在，则原内容被删除，读操作将读不到内容，除非写入内容后读；以a+方式打开文件时，如果是读文件，从文件开始位置读，如果是写文件从文件结尾处写。