C 语言文件操作函数大全

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 * pathconst 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。

```
#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。