# 流

## 简介

## 文件

## IO

## 流

### 打开流

### 关闭流

## 操作

### 文件拷贝

### 读写

#### fgets

#### fread

#### fwrite

#### 文件结束函数

#### 错误指示函数

### 缓冲

### 定位

在Linux中通过man fseek获取定位函数：

#### fseek

原型：int fseek(FILE \*stream, long offset, int whence);

参数：offset表示偏移量，whence表示具体移动到某个标志性位置（可以是文件头，文件尾或者当前位置）。

作用：针对某个标志性位置做偏移

举例：

#include <stdio.h>

#include <stdlib.h> //man free, man malloc

#include <string.h>

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

{

//一次性完成数据的传递

//计算出源文件的大小

//准备一个合适的内存，大小（源文件本身大小）

//一次性读取源文件

//一次性写入源文件

//关闭

int ret = -1;

//拷贝 源文件（source.c）到目标文件（dst.c)

FILE \*fp\_s = NULL;//源文件流

fp\_s = fopen("source.c","r");//只读 r

if ( NULL == fp\_s )

{

perror("fopen source.c");

return -1;

}

FILE \*fp\_d = NULL;//目标文件流

fp\_d = fopen("dst.c","w");//写w

if ( NULL == fp\_d )

{

perror("fopen dst.c");

return -1;

}

//准备内存

char \*buff = NULL;

//计算出源文件的大小

ret = fseek(fp\_s,0,SEEK\_END);//定位到文件尾部

if ( -1 == ret )

{

perror("fseek");

return -1;

}

long file\_size;

file\_size = ftell(fp\_s);//计算出大小

if ( -1 == file\_size )

{

perror("ftell");

return -1;

}

//文件位置－－－文件头

ret = fseek(fp\_s,0,SEEK\_SET);

if ( -1 == ret )

{

perror("fseek");

return -1;

}

//准备一个合适的内存，大小（源文件本身大小）

buff = malloc(file\_size);//堆

if( NULL == buff )

{

perror("malloc buff");

return -1;

}

memset(buff,'\0',file\_size);//内存每次完成后清0

//读

//从源文件默认光标（文件位置处）读取一个长度信息，存放到用户内存里面

ret = fread(buff,1,file\_size,fp\_s);

if ( 20 > ret )

{

if ( feof(fp\_s))//ferror

{

//文件尾

ret = fwrite(buff,1,ret,fp\_d);

if ( 0 == ret )

{

//bug...

perror("fwrite");

return -1;

}

}else

{

//出错

perror("fread");

return -1;

}

}

//写

//

ret = fwrite(buff,1,ret,fp\_d);

if ( 0 == ret )

{

//bug...

perror("fwrite");

return -1;

}

free(buff);

buff = NULL;

fclose(fp\_s);

fp\_s = NULL;

fclose(fp\_d);

fp\_d = NULL;

}

#### ftell

原型：long ftell(FILE \*stream);

作用：测试当前光标位置距离文件头的位置（该距离使用long类型标识，单位是按照字节计算）。

可以利用fseek定位到文件头，然后ftell计算偏移量，这样就可以计算出文件的大小。

#### rewind

#### fgetpos

#### fsetpos