GACHA－二次元社区LOFTER-最美图片社交印象派-照片冲印、出书 签到赢iPhone 6 Plus >注册登录首页**日志**LOFTER相册音乐收藏博友[下载LOFTER客户端](http://www.lofter.com/app?act=qbbkrzydb_20150408_01)

C++ 通过以下几个类支持文件的输入输出：

ofstream: 写操作（输出）的文件类 (由ostream引申而来)

ifstream: 读操作（输入）的文件类(由istream引申而来)

fstream: 可同时读写操作的文件类 (由iostream引申而来)

打开文件(Open a file)

对这些类的一个对象所做的第一个操作通常就是将它和一个真正的文件联系起来，也就是说打开一个文件。被打开的文件在程序中由一个流对象(stream object)来表示 (这些类的一个实例) ，而对这个流对象所做的任何输入输出操作实际就是对该文件所做的操作。

要通过一个流对象打开一个文件，我们使用它的成员函数open()：void open (const char \* filename, openmode mode);

这里filename 是一个字符串，代表要打开的文件名，mode 是以下标志符的一个组合： ios::in 为输入(读)而打开文件

ios::out 为输出(写)而打开文件

ios::ate 初始位置：文件尾

ios::app 所有输出附加在文件末尾

ios::trunc 如果文件已存在则先删除该文件

ios::binary 二进制方式

这些标识符可以被组合使用，中间以”或”操作符(|)间隔。例如，如果我们想要以二进制方式打开文件"example.bin" 来写入一些数据，我们可以通过以下方式调用成员函数open（）来实现：ofstream file;

file.open ("example.bin", ios::out | ios::app | ios::binary);

ofstream, ifstream 和 fstream所有这些类的成员函数open 都包含了一个默认打开文件的方式，这三个类的默认方式各不相同： 类 参数的默认方式

ofstream ios::out | ios::trunc

ifstream ios::in

fstream ios::in | ios::out

只有当函数被调用时没有声明方式参数的情况下，默认值才会被采用。如果函数被调用时声明了任何参数，默认值将被完全改写，而不会与调用参数组合。

由于对类ofstream, ifstream 和 fstream 的对象所进行的第一个操作通常都是打开文件，这些类都有一个构造函数可以直接调用open 函数，并拥有同样的参数。这样，我们就可以通过以下方式进行与上面同样的定义对象和打开文件的操作：ofstream file ("example.bin", ios::out | ios::app | ios::binary);

两种打开文件的方式都是正确的。

你可以通过调用成员函数is\_open()来检查一个文件是否已经被顺利的打开了：bool is\_open();

它返回一个布尔(bool)值，为真（true）代表文件已经被顺利打开，假( false )则相反。

关闭文件(Closing a file)

当文件读写操作完成之后，我们必须将文件关闭以使文件重新变为可访问的。关闭文件需要调用成员函数close()，它负责将缓存中的数据排放出来并关闭文件。它的格式很简单：void close ();

这个函数一旦被调用，原先的流对象(stream object)就可以被用来打开其它的文件了，这个文件也就可以重新被其它的进程(process)所有访问了。

为防止流对象被销毁时还联系着打开的文件，析构函数(destructor)将会自动调用关闭函数close。

文本文件(Text mode files)

类ofstream, ifstream 和fstream 是分别从ostream, istream 和iostream 中引申而来的。这就是为什么 fstream 的对象可以使用其父类的成员来访问数据。

一般来说，我们将使用这些类与同控制台(console)交互同样的成员函数(cin 和 cout)来进行输入输出。如下面的例题所示，我们使用重载的插入操作符<<： // writing on a text file

#include <fiostream.h>

int main () {

ofstream examplefile ("example.txt");

if (examplefile.is\_open()) {

examplefile << "This is a line.\n";

examplefile << "This is another line.\n";

examplefile.close();

}

return 0;

}

file example.txt

This is a line.

This is another line.

从文件中读入数据也可以用与 cin的使用同样的方法： // reading a text file

#include <iostream.h>

#include <fstream.h>

#include <stdlib.h>

int main () {

char buffer[256];

ifstream examplefile ("example.txt");

if (! examplefile.is\_open())

{ cout << "Error opening file"; exit (1); }

while (! examplefile.eof() ) {

examplefile.getline (buffer,100);

cout << buffer << endl;

}

return 0;

}

This is a line.

This is another line.

上面的例子读入一个文本文件的内容，然后将它打印到屏幕上。注意我们使用了一个新的成员函数叫做eof ，它是ifstream 从类 ios 中继承过来的，当到达文件末尾时返回true 。

状态标志符的验证(Verification of state flags)

除了eof()以外，还有一些验证流的状态的成员函数（所有都返回bool型返回值）：

bad()

如果在读写过程中出错，返回 true 。例如：当我们要对一个不是打开为写状态的文件进行写入时，或者我们要写入的设备没有剩余空间的时候。

fail()

除了与bad() 同样的情况下会返回 true 以外，加上格式错误时也返回true ，例如当想要读入一个整数，而获得了一个字母的时候。

eof()

如果读文件到达文件末尾，返回true。

good()

这是最通用的：如果调用以上任何一个函数返回true 的话，此函数返回 false 。

要想重置以上成员函数所检查的状态标志，你可以使用成员函数clear()，没有参数。

获得和设置流指针(get and put stream pointers)

所有输入/输出流对象(i/o streams objects)都有至少一个流指针：

ifstream， 类似istream, 有一个被称为get pointer的指针，指向下一个将被读取的元素。

ofstream, 类似 ostream, 有一个指针 put pointer ，指向写入下一个元素的位置。

fstream, 类似 iostream, 同时继承了get 和 put

我们可以通过使用以下成员函数来读出或配置这些指向流中读写位置的流指针：

tellg() 和 tellp()

这两个成员函数不用传入参数，返回pos\_type 类型的值(根据ANSI-C++ 标准) ，就是一个整数，代表当前get 流指针的位置 (用tellg) 或 put 流指针的位置(用tellp).

seekg() 和seekp()

这对函数分别用来改变流指针get 和put的位置。两个函数都被重载为两种不同的原型：

seekg ( pos\_type position );

seekp ( pos\_type position );

使用这个原型，流指针被改变为指向从文件开始计算的一个绝对位置。要求传入的参数类型与函数 tellg 和tellp 的返回值类型相同。

seekg ( off\_type offset, seekdir direction );

seekp ( off\_type offset, seekdir direction );

使用这个原型可以指定由参数direction决定的一个具体的指针开始计算的一个位移(offset)。它可以是： ios::beg 从流开始位置计算的位移

ios::cur 从流指针当前位置开始计算的位移

ios::end 从流末尾处开始计算的位移

流指针 get 和 put 的值对文本文件(text file)和二进制文件(binary file)的计算方法都是不同的，因为文本模式的文件中某些特殊字符可能被修改。由于这个原因，建议对以文本文件模式打开的文件总是使用seekg 和 seekp的第一种原型，而且不要对tellg 或 tellp 的返回值进行修改。对二进制文件，你可以任意使用这些函数，应该不会有任何意外的行为产生。

以下例子使用这些函数来获得一个二进制文件的大小： // obtaining file size

#include <iostream.h>

#include <fstream.h>

const char \* filename = "example.txt";

int main () {

long l,m;

ifstream file (filename, ios::in|ios::binary);

l = file.tellg();

file.seekg (0, ios::end);

m = file.tellg();

file.close();

cout << "size of " << filename;

cout << " is " << (m-l) << " bytes.\n";

return 0;

}

size of example.txt is 40 bytes.

二进制文件(Binary files)

在二进制文件中，使用<< 和>>，以及函数（如getline）来操作符输入和输出数据，没有什么实际意义，虽然它们是符合语法的。

文件流包括两个为顺序读写数据特殊设计的成员函数：write 和 read。第一个函数 (write) 是ostream 的一个成员函数，都是被ofstream所继承。而read 是istream 的一个成员函数，被ifstream 所继承。类 fstream 的对象同时拥有这两个函数。它们的原型是：

write ( char \* buffer, streamsize size );

read ( char \* buffer, streamsize size );

这里 buffer 是一块内存的地址，用来存储或读出数据。参数size 是一个整数值，表示要从缓存（buffer）中读出或写入的字符数。 // reading binary file

#include <iostream>

#include <fstream.h>

const char \* filename = "example.txt";

int main () {

char \* buffer;

long size;

ifstream file (filename, ios::in|ios::binary|ios::ate);

size = file.tellg();

file.seekg (0, ios::beg);

buffer = new char [size];

file.read (buffer, size);

file.close();

cout << "the complete file is in a buffer";

delete[] buffer;

return 0;

}

The complete file is in a buffer

阅读(57070)| 评论(3)

* [登录](https://passport.csdn.net/account/login?ref=toolbar)|[注册](http://passport.csdn.net/account/mobileregister?ref=toolbar&action=mobileRegister)

## [小孟的那些日子](http://blog.csdn.net/mak0000)

### 奋斗！

* [http://static.blog.csdn.net/images/ico_list.gif目录视图](http://blog.csdn.net/mak0000?viewmode=contents)
* [http://static.blog.csdn.net/images/ico_summary.gif摘要视图](http://blog.csdn.net/mak0000?viewmode=list)
* [http://static.blog.csdn.net/images/ico_rss.gif订阅](http://blog.csdn.net/mak0000/rss/list)

[学院APP首次下载，可得50C币！](http://edu.csdn.net/app)    [欢迎来帮助开源“进步”](http://bbs.csdn.net/topics/391876796)    [当讲师？爱学习？投票攒课吧](http://bss.csdn.net/m/topic/cuanke_vote)    [CSDN 2015博客之星评选结果公布](http://vote.blog.csdn.net/blogstar2015/result)

关闭

# [c++文件操作大全](http://blog.csdn.net/mak0000/article/details/3230199)

标签： [c++](http://www.csdn.net/tag/c%2b%2b)[file](http://www.csdn.net/tag/file)[stream](http://www.csdn.net/tag/stream)[ios](http://www.csdn.net/tag/ios)[buffer](http://www.csdn.net/tag/buffer)[fp](http://www.csdn.net/tag/fp)

2008-11-05 22:56 18863人阅读 [评论](http://blog.csdn.net/mak0000/article/details/3230199#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/mak0000/article/details/3230199#report)

版权声明：本文为博主原创文章，未经博主允许不得转载。

基于C的文件操作   
　　在ANSI C中，对文件的操作分为两种方式，即流式文件操作和I/O文件操作，下面就分别介绍之。

　　一、流式文件操作   
　　这种方式的文件操作有一个重要的结构FILE，FILE在stdio.h中定义如下：

　　typedef struct {   
　　int level; /\* fill/empty level of buffer \*/   
　　unsigned flags; /\* File status flags \*/   
　　char fd; /\* File descriptor \*/   
　　unsigned char hold; /\* Ungetc char if no buffer \*/   
　　int bsize; /\* Buffer size \*/   
　　unsigned char \_FAR \*buffer; /\* Data transfer buffer \*/   
　　unsigned char \_FAR \*curp; /\* Current active pointer \*/   
　　unsigned istemp; /\* Temporary file indicator \*/   
　　short token; /\* Used for validity checking \*/   
　　} FILE; /\* This is the FILE object \*/

　　FILE这个结构包含了文件操作的基本属性，对文件的操作都要通过这个结构的指针来进行，此种文件操作常用的函数见下表 函数 功能   
　　fopen() 打开流   
　　fclose() 关闭流   
　　fputc() 写一个字符到流中   
　　fgetc() 从流中读一个字符   
　　fseek() 在流中定位到指定的字符   
　　fputs() 写字符串到流   
　　fgets() 从流中读一行或指定个字符   
　　fprintf() 按格式输出到流   
　　fscanf() 从流中按格式读取   
　　feof() 到达文件尾时返回真值   
　　ferror() 发生错误时返回其值   
　　rewind() 复位文件定位器到文件开始处   
　　remove() 删除文件   
　　fread() 从流中读指定个数的字符   
　　fwrite() 向流中写指定个数的字符   
　　tmpfile() 生成一个临时文件流   
　　tmpnam() 生成一个唯一的文件名

　　下面就介绍一下这些函数

　　1.fopen()   
　　fopen的原型是：FILE \*fopen(const char \*filename,const char \*mode)，fopen实现三个功能

　　为使用而打开一个流   
　　把一个文件和此流相连接   
　　给此流返回一个FILR指针   
　　参数filename指向要打开的文件名，mode表示打开状态的字符串，其取值如下表

　　字符串 含义   
　　"r" 以只读方式打开文件   
　　"w" 以只写方式打开文件   
　　"a" 以追加方式打开文件   
　　"r+" 以读/写方式打开文件，如无文件出错   
　　"w+" 以读/写方式打开文件，如无文件生成新文件

　　一个文件可以以文本模式或二进制模式打开，这两种的区别是：在文本模式中回车被当成一个字符'/n'，而二进制模式认为它是两个字符0x0D,0x0A；如果在文件中读到0x1B，文本模式会认为这是文件结束符，也就是二进制模型不会对文件进行处理，而文本方式会按一定的方式对数据作相应的转换。

　　系统默认的是以文本模式打开，可以修改全部变量\_fmode的值来修改这个设置，例如\_fmode=O\_TEXT；就设置默认打开方式为文本模式；而\_fmode=O\_BINARY；则设置默认打开方式是二进制模式。

　　我们也可以在模式字符串中指定打开的模式，如"rb"表示以二进制模式打开只读文件，"w+t"或"wt+"表示以文本模式打开读/写文件。

　　此函数返回一个FILE指针，所以申明一个FILE指针后不用初始化，而是用fopen()来返回一个指针并与一个特定的文件相连，如果成败，返回NULL。

　　例:

　　FILE \*fp;   
　　if(fp=fopen("123.456","wb"))   
　　puts("打开文件成功");   
　　else   
　　puts("打开文件成败");

2.fclose()   
　　fclose()的功能就是关闭用fopen()打开的文件，其原型是：int fclose(FILE \*fp);如果成功，返回0,失败返回EOF。

　　在程序结束时一定要记得关闭打开的文件，不然可能会造成数据丢失的情况，我以前就经常犯这样的毛病。

　　例：fclose(fp);

　　3.fputc()   
　　向流写一个字符，原型是int fputc(int c, FILE \*stream); 成功返回这个字符,失败返回EOF。

　　例：fputc('X',fp);

　　4.fgetc()   
　　从流中读一个字符，原型是int fputc(FILE \*stream); 成功返回这个字符,失败返回EOF。

　　例：char ch1=fgetc(fp);

　　5. fseek()   
　　此函数一般用于二进制模式打开的文件中，功能是定位到流中指定的位置，原型是int fseek(FILE \*stream, long offset, int whence);如果成功返回0，参数offset是移动的字符数，whence是移动的基准，取值是

　　符号常量 值 基准位置   
　　SEEK\_SET 0 文件开头   
　　SEEK\_CUR 1 当前读写的位置   
　　SEEK\_END 2 文件尾部

　　例：fseek(fp,1234L,SEEK\_CUR);//把读写位置从当前位置向后移动1234字节(L后缀表示长整数)

　　fseek(fp,0L,2);//把读写位置移动到文件尾

　　6.fputs()   
　　写一个字符串到流中，原型int fputs(const char \*s, FILE \*stream);

　　例：fputs("I Love You",fp);

　　7.fgets()   
　　从流中读一行或指定个字符，原型是char \*fgets(char \*s, int n, FILE \*stream); 从流中读取n-1个字符，除非读完一行，参数s是来接收字符串，如果成功则返回s的指针，否则返回NULL。

　　例：如果一个文件的当前位置的文本如下

　　Love ,I Have

　　But ........

　　如果用

　　fgets(str1,4,file1);

　　则执行后str1="Lov"，读取了4-1=3个字符，而如果用

　　fgets(str1,23,file1);

　　则执行str="Love ,I Have"，读取了一行(不包括行尾的'/n')。

8.fprintf()   
　　按格式输入到流，其原型是int fprintf(FILE \*stream, const char \*format[, argument, ...]);其用法和printf()相同，不过不是写到控制台，而是写到流罢了

　　例：fprintf(fp,"%2d%s",4,"Hahaha");

　　9.fscanf()   
　　从流中按格式读取，其原型是int fscanf(FILE \*stream, const char \*format[, address, ...]);其用法和scanf()相同，不过不是从控制台读取，而是从流读取罢了。

　　例：fscanf(fp,"%d%d" ,&x,&y);

　　10.feof()   
　　检测是否已到文件尾，是返回真，否则返回0，其原型是int feof(FILE \*stream);

　　例：if(feof(fp))printf("已到文件尾");

　　11.ferror()   
　　原型是int ferror(FILE \*stream);返回流最近的错误代码，可用clearerr()来清除它，clearerr()的原型是void clearerr(FILE \*stream);

　　例：printf("%d",ferror(fp));

　　12.rewind()   
　　把当前的读写位置回到文件开始，原型是void rewind(FILE \*stream);其实本函数相当于fseek(fp,0L,SEEK\_SET);

　　例：rewind(fp);

　　12.remove()   
　　删除文件，原型是int remove(const char \*filename); 参数就是要删除的文件名，成功返回0。

　　例：remove("c://io.sys");

　　13.fread()   
　　从流中读指定个数的字符，原型是size\_t fread(void \*ptr, size\_t size, size\_t n, FILE \*stream);参数ptr是保存读取的数据，void\*的指针可用任何类型的指针来替换，如char\*、int \*等等来替换；size是每块的字节数；n是读取的块数，如果成功，返回实际读取的块数(不是字节数)，本函数一般用于二进制模式打开的文件中。

　　例：

　　char x[4230];   
　　FILE \*file1=fopen("c://msdos.sys","r");   
　　fread(x,200,12 ,file1);//共读取200\*12=2400个字节

　　14.fwrite()   
　　与fread对应，向流中写指定的数据，原型是size\_t fwrite(const void \*ptr, size\_t size, size\_t n, FILE \*stream);参数ptr是要写入的数据指针，void\*的指针可用任何类型的指针来替换，如char\*、int \*等等来替换；size是每块的字节数；n是要写的块数，如果成功，返回实际写入的块数(不是字节数)，本函数一般用于二进制模式打开的文件中。

　　例：

　　char x[]="I Love You";   
　　fwire(x, 6,12,fp);//写入6\*12=72字节

　　将把"I Love"写到流fp中12次，共72字节

　　15.tmpfile()   
　　其原型是FILE \*tmpfile(void); 生成一个临时文件，以"w+b"的模式打开，并返回这个临时流的指针，如果失败返回NULL。在程序结束时，这个文件会被自动删除。

　　例：FILE \*fp=tmpfile();

　　16.tmpnam();   
　　其原型为char \*tmpnam(char \*s); 生成一个唯一的文件名，其实tmpfile()就调用了此函数，参数s用来保存得到的文件名，并返回这个指针，如果失败，返回NULL。

borland c++ bulder的文件操作总结２

二、直接I/O文件操作   
　　这是C提供的另一种文件操作，它是通过直接存/取文件来完成对文件的处理，而上篇所说流式文件操作是通过缓冲区来进行；流式文件操作是围绕一个FILE指针来进行，而此类文件操作是围绕一个文件的"句柄"来进行，什么是句柄呢？它是一个整数，是系统用来标识一个文件(在WINDOWS中，句柄的概念扩展到所有设备资源的标识)的唯一的记号。此类文件操作常用的函数如下表，这些函数及其所用的一些符号在io.h和fcntl.h中定义，在使用时要加入相应的头文件。

　　函数 说明   
　　open() 打开一个文件并返回它的句柄   
　　close() 关闭一个句柄   
　　lseek() 定位到文件的指定位置   
　　read() 块读文件   
　　write() 块写文件   
　　eof() 测试文件是否结束   
　　filelength() 取得文件长度   
　　rename() 重命名文件   
　　chsize() 改变文件长度

　　下面就对这些函数一一说明：

　　1.open()   
　　打开一个文件并返回它的句柄，如果失败，将返回一个小于0的值，原型是int open(const char \*path, int access [, unsigned mode]); 参数path是要打开的文件名，access是打开的模式，mode是可选项。表示文件的属性，主要用于UNIX系统中，在DOS/WINDOWS这个参数没有意义。其中文件的打开模式如下表。

　　符号 含义 符号 含义 符号 含义   
　　O\_RDONLY 只读方式 O\_WRONLY 只写方式 O\_RDWR 读/写方式   
　　O\_NDELAY 用于UNIX系统 O\_APPEND 追加方式 O\_CREAT 如果文件不存在就创建   
　　O\_TRUNC 把文件长度截为0 O\_EXCL 和O\_CREAT连用，如果文件存在返回错误 O\_BINARY 二进制方式   
　　O\_TEXT 文本方式

　　对于多个要求，可以用"|"运算符来连接，如O\_APPEND|O\_TEXT表示以文本模式和追加方式打开文件。

　　例：int handle=open("c://msdos.sys",O\_BINARY|O\_CREAT|O\_WRITE)

　　2.close()   
　　关闭一个句柄，原型是int close(int handle);如果成功返回0

　　例：close(handle)

　　3.lseek()   
　　定位到指定的位置，原型是：long lseek(int handle, long offset, int fromwhere);参数offset是移动的量，fromwhere是移动的基准位置，取值和前面讲的fseek()一样，SEEK\_SET：文件首部；SEEK\_CUR：文件当前位置；SEEK\_END：文件尾。此函数返回执行后文件新的存取位置。

　　例：

　　lseek(handle,-1234L,SEEK\_CUR);//把存取位置从当前位置向前移动1234个字节。   
　　x=lseek(hnd1,0L,SEEK\_END);//把存取位置移动到文件尾，x=文件尾的位置即文件长度

4.read()   
　　从文件读取一块，原型是int read(int handle, void \*buf, unsigned len);参数buf保存读出的数据，len是读取的字节。函数返回实际读出的字节。

　　例：char x[200];read(hnd1,x,200);

　　5.write()   
　　写一块数据到文件中，原型是int write(int handle, void \*buf, unsigned len);参数的含义同read()，返回实际写入的字节。

　　例：char x[]="I Love You";write(handle,x,strlen(x));

　　7.eof()   
　　类似feof()，测试文件是否结束，是返回1，否则返回0;原型是：int eof(int handle);

　　例：while(!eof(handle1)){……};

　　8.filelength()   
　　返回文件长度，原型是long filelength(int handle);相当于lseek(handle,0L,SEEK\_END)

　　例：long x=filelength(handle);

　　9.rename()   
　　重命名文件，原型是int rename(const char \*oldname, const char \*newname); 参数oldname是旧文件名，newname是新文件名。成功返回0

　　例：rename("c://config.sys","c://config.w40");

　　10.chsize();   
　　改变文件长度，原型是int chsize(int handle, long size);参数size表示文件新的长度，成功返回0，否则返回-1，如果指定的长度小于文件长度，则文件被截短；如果指定的长度大于文件长度，则在文件后面补'/0'。

　　例：chsize(handle,0x12345);

　　--------------------------------------------------------------------------------

　　如果熟悉汇编可能会发现这种方式和汇编语言的DOS功能调用句柄式文件操作很像，比如open()就像DOS服务的3CH号功能调用，其实这种操作还有两种类型的函数就是直接用DOS功能来完成的，如\_open()，\_dos\_open()等等。有兴趣可自已查询BCB的帮助。

　　同流式文件操作相同，这种也提供了Unicode字符操作的函数，如\_wopen()等等，用于9X/NT下的宽字符编程，有兴趣可自已查询BCB的帮助。

　　另外，此种操作还有lock(),unlock(),locking()等用于多用户操作的函数，但在BCB中用得并不多，我就不介绍了，但如果要用C来写CGI，这些就必要的常识了，如果你有这方面的要求，那就得自已好好看帮助了。

　　到这里，我就把基于C的文件操作介绍完了，下一篇将介绍基于C++的文件操作，敬请期待。

在C++中，有一个stream这个类，所有的I/O都以这个"流"类为基础的，包括我们要认识的文件I/O，stream这个类有两个重要的运算符：

　　1、插入器(<<)   
　　向流输出数据。比如说系统有一个默认的标准输出流(cout)，一般情况下就是指的显示器，所以，cout<<"Write Stdout"<<'/n';就表示把字符串"Write Stdout"和换行字符('/n')输出到标准输出流。

　　2、析取器(>>)   
　　从流中输入数据。比如说系统有一个默认的标准输入流(cin)，一般情况下就是指的键盘，所以，cin>>x;就表示从标准输入流中读取一个指定类型(即变量x的类型)的数据。

　　在C++中，对文件的操作是通过stream的子类fstream(file stream)来实现的，所以，要用这种方式操作文件，就必须加入头文件fstream.h。下面就把此类的文件操作过程一一道来。

　　一、打开文件   
　　在fstream类中，有一个成员函数open()，就是用来打开文件的，其原型是：

　　void open(const char\* filename,int mode,int access);

　　参数：

　　filename：　　要打开的文件名   
　　mode：　　　　要打开文件的方式   
　　access：　　　打开文件的属性   
　　打开文件的方式在类ios(是所有流式I/O类的基类)中定义，常用的值如下：

　　ios::app：　　　以追加的方式打开文件   
　　ios::ate：　　　文件打开后定位到文件尾，ios:app就包含有此属性   
　　ios::binary： 　以二进制方式打开文件，缺省的方式是文本方式。两种方式的区别见前文   
　　ios::in：　　　 文件以输入方式打开   
　　ios::out：　　　文件以输出方式打开   
　　ios::nocreate： 不建立文件，所以文件不存在时打开失败　   
　　ios::noreplace：不覆盖文件，所以打开文件时如果文件存在失败   
　　ios::trunc：　　如果文件存在，把文件长度设为0   
　　可以用"或"把以上属性连接起来，如ios::out|ios::binary

　　打开文件的属性取值是：

　　0：普通文件，打开访问   
　　1：只读文件   
　　2：隐含文件   
　　4：系统文件   
　　可以用"或"或者"+"把以上属性连接起来 ，如3或1|2就是以只读和隐含属性打开文件。

　　例如：以二进制输入方式打开文件c:/config.sys

　　fstream file1;   
　　file1.open("c://config.sys",ios::binary|ios::in,0);

　　如果open函数只有文件名一个参数，则是以读/写普通文件打开，即：

　　file1.open("c://config.sys");<=>file1.open("c://config.sys",ios::in|ios::out,0);

　　另外，fstream还有和open()一样的构造函数，对于上例，在定义的时侯就可以打开文件了：

　　fstream file1("c://config.sys");

　　特别提出的是，fstream有两个子类：ifstream(input file stream)和ofstream(outpu file stream)，ifstream默认以输入方式打开文件，而ofstream默认以输出方式打开文件。

　　ifstream file2("c://pdos.def");//以输入方式打开文件   
　　ofstream file3("c://x.123");//以输出方式打开文件

　　所以，在实际应用中，根据需要的不同，选择不同的类来定义：如果想以输入方式打开，就用ifstream来定义；如果想以输出方式打开，就用ofstream来定义；如果想以输入/输出方式来打开，就用fstream来定义。

二、关闭文件   
　　打开的文件使用完成后一定要关闭，fstream提供了成员函数close()来完成此操作，如：file1.close();就把file1相连的文件关闭。

　　三、读写文件   
　　读写文件分为文本文件和二进制文件的读取，对于文本文件的读取比较简单，用插入器和析取器就可以了；而对于二进制的读取就要复杂些，下要就详细的介绍这两种方式

　　1、文本文件的读写   
　　文本文件的读写很简单：用插入器(<<)向文件输出；用析取器(>>)从文件输入。假设file1是以输入方式打开，file2以输出打开。示例如下：

　　file2<<"I Love You";//向文件写入字符串"I Love You"   
　　int i;   
　　file1>>i;//从文件输入一个整数值。

　　这种方式还有一种简单的格式化能力，比如可以指定输出为16进制等等，具体的格式有以下一些

　　操纵符 功能 输入/输出   
　　dec 格式化为十进制数值数据 输入和输出   
　　endl 输出一个换行符并刷新此流 输出   
　　ends 输出一个空字符 输出   
　　hex 格式化为十六进制数值数据 输入和输出   
　　oct 格式化为八进制数值数据 输入和输出   
　　setpxecision(int p) 设置浮点数的精度位数 输出

　　比如要把123当作十六进制输出：file1<  
　　2、二进制文件的读写   
　　①put()   
　　put()函数向流写入一个字符，其原型是ofstream &put(char ch)，使用也比较简单，如file1.put('c');就是向流写一个字符'c'。

　　②get()   
　　get()函数比较灵活，有3种常用的重载形式：

　　一种就是和put()对应的形式：ifstream &get(char &ch);功能是从流中读取一个字符，结果保存在引用ch中，如果到文件尾，返回空字符。如file2.get(x);表示从文件中读取一个字符，并把读取的字符保存在x中。

　　另一种重载形式的原型是： int get();这种形式是从流中返回一个字符，如果到达文件尾，返回EOF，如x=file2.get();和上例功能是一样的。

　　还有一种形式的原型是：ifstream &get(char \*buf,int num,char delim='/n')；这种形式把字符读入由 buf 指向的数组，直到读入了 num 个字符或遇到了由 delim 指定的字符，如果没使用 delim 这个参数，将使用缺省值换行符'/n'。例如：

　　file2.get(str1,127,'A');//从文件中读取字符到字符串str1，当遇到字符'A'或读取了127个字符时终止。

　　③读写数据块   
　　要读写二进制数据块，使用成员函数read()和write()成员函数，它们原型如下：

　　read(unsigned char \*buf,int num);   
　　write(const unsigned char \*buf,int num);

　　read()从文件中读取 num 个字符到 buf 指向的缓存中，如果在还未读入 num 个字符时就到了文件尾，可以用成员函数 int gcount();来取得实际读取的字符数；而 write() 从buf 指向的缓存写 num 个字符到文件中，值得注意的是缓存的类型是 unsigned char \*，有时可能需要类型转换。

　　例：

　　unsigned char str1[]="I Love You";   
　　int n[5];   
　　ifstream in("xxx.xxx");   
　　ofstream out("yyy.yyy");   
　　out.write(str1,strlen(str1));//把字符串str1全部写到yyy.yyy中   
　　in.read((unsigned char\*)n,sizeof(n));//从xxx.xxx中读取指定个整数，注意类型转换   
　　in.close();out.close();

　　四、检测EOF   
　　成员函数eof()用来检测是否到达文件尾，如果到达文件尾返回非0值，否则返回0。原型是int eof();

　　例：　　if(in.eof())ShowMessage("已经到达文件尾！");

五、文件定位   
　　和C的文件操作方式不同的是，C++ I/O系统管理两个与一个文件相联系的指针。一个是读指针，它说明输入操作在文件中的位置；另一个是写指针，它下次写操作的位置。每次执行输入或输出时，相应的指针自动变化。所以，C++的文件定位分为读位置和写位置的定位，对应的成员函数是 seekg()和 seekp()，seekg()是设置读位置，seekp是设置写位置。它们最通用的形式如下：

　　istream &seekg(streamoff offset,seek\_dir origin);   
　　ostream &seekp(streamoff offset,seek\_dir origin);

　　streamoff定义于 iostream.h 中，定义有偏移量 offset 所能取得的最大值，seek\_dir 表示移动的基准位置，是一个有以下值的枚举：

　　ios::beg：　　文件开头   
　　ios::cur：　　文件当前位置   
　　ios::end：　　文件结尾   
　　这两个函数一般用于二进制文件，因为文本文件会因为系统对字符的解释而可能与预想的值不同。

　　例：

　　file1.seekg(1234,ios::cur);//把文件的读指针从当前位置向后移1234个字节   
　　file2.seekp(1234,ios::beg);//把文件的写指针从文件开头向后移1234个字节

在BCB中也提供了文件操作的函数，这些函数的功能和前面所介绍的大致相同，但这类函数和BCB关系紧密，能使用BCB中的AnsiString等数据类型，在BCB中用这种方式的文件操作是最方便的，下面我就把这种文件操作详细介绍。

　　在BCB提供的这组文件操作函数中，可分为三种类型，就是：1、文件名函数，2、文件管理函数；3、文件I/O函数。

　　1、文件名函数   
　　文件名函数可以对文件的名称、所在子目录、驱动器和扩展名等进行操作。下表列出这些函数及其功能。

　　函数 说明   
　　ExpandFileName() 返回文件的全路径(含驱动器、路径)   
　　ExtractFileExt() 从文件名中抽取扩展名   
　　ExtractFileName() 从文件名中抽取不含路径的文件名   
　　ExtractFilePath() 从文件名中抽取路径名   
　　ExtractFileDir() 从文件名中抽取目录名   
　　ExtractFileDrive() 从文件名中抽取驱动器名   
　　ChangeFileExt() 改变文件的扩展名   
　　ExpandUNCFileName() 返回含有网络驱动器的文件全路径   
　　ExtractRelativePath() 从文件名中抽取相对路径信息   
　　ExtractShortPathName() 把文件名转化为DOS的8·3格式   
　　MatchesMask() 检查文件是否与指定的文件名格式匹配

　　下面就把这些函数作一一介绍：

　　⑴ExpandFileName()   
　　原型：extern PACKAGE AnsiString \_\_fastcall ExpandFileName(const AnsiString FileName);

　　功能：返回文件的全路径(含驱动器、路径)

　　参数：FileName:要处理的文件名

　　例：ShowMessage(ExpandFileName(Application->ExeName));//显示你的程序文件名，如C:/MyBCB/Sample1.EXE

　　⑵ExtractFileExt()   
　　原型：extern PACKAGE AnsiString \_\_fastcall ExtractFileExt(const AnsiString FileName);

　　功能：从文件名中抽取扩展名

　　参数：FileName:要处理的文件名（全路径）

　　例：ShowMessage(ExtractFileExt(Application->ExeName));//显示".exe"

　　⑶ExtractFileName()   
　　原型：extern PACKAGE AnsiString \_\_fastcall ExtractFileName(const AnsiString FileName);

　　功能：从文件名中抽取不含路径的文件名

　　参数：FileName:要处理的文件名

　　例：ShowMessage(ExtractFileExt("c://Winnt//SOL.EXE"));//显示"SOL.EXE"

　　⑷ExtractFilePath()   
　　原型：extern PACKAGE AnsiString \_\_fastcall ExtractFilePath(const AnsiString FileName);

　　功能：从文件名中抽取路径名

　　参数：FileName:要处理的文件名

　　例：ShowMessage(ExtractFilePath("Winnt//SOL.EXE"));//显示"Winnt/"

　　⑸ExtractFileDir()   
　　原型：extern PACKAGE AnsiString \_\_fastcall ExtractFileDir(const AnsiString FileName);

　　功能：从文件名中抽取目录名(和上个函数不同，不包括最后的"/")

　　参数：FileName:要处理的文件名

　　例：ShowMessage(ExtractFileDir("Winnt//SOL.EXE"));//显示"Winnt",注意和上个函数的区别

　　⑹ExtractFileDrive()   
　　原型：extern PACKAGE AnsiString \_\_fastcall ExtractFileDrive(const AnsiString FileName);

　　功能：从文件名中抽取驱动器名

　　参数：FileName:要处理的文件名

　　例：ShowMessage(ExtractFileDrive("c://Winnt//SOL.EXE"));//显示"c:"

　　⑺ChangeFileExt()   
　　原型：extern PACKAGE System::AnsiString \_\_fastcall ChangeFileExt(const System::AnsiString FileName, const System::AnsiString Extension);

　　功能：更改文件名的扩展名，不是对真正的文件进行改名，只是对文件名这个字符串进行处理

　　参数：FileName:要改名的文件名，Extension：新的扩展名

　　例：ShowMessage(ChangeFileExt("c://Winnt//SOL.EXE",".OOO"));//显示"c:/winnt/SOL.OOO"

　　⑻ExpandUNCFileName()   
　　原型：extern PACKAGE AnsiString \_\_fastcall ExpandUNCFileName(const AnsiString FileName);

　　功能：返回含有网络驱动器的文件全路径，格式为：//机器名/共享名/文件名

　　参数：FileName:要处理的文件名

　　例：ShowMessage(ExpandUNCFileName("F://Winnt//SOL.EXE"));/\*如果F:是映射的网络驱动器[//NT40/WINNT](file:///\\nt40\WINNT)，则显示"[//NT40/WINNT/SOL.EXE"\*/](file:///\\nt40\WINNT\SOL.EXE%22*\)

　　⑼ExtractRelativePath()   
　　原型：extern PACKAGE AnsiString \_\_fastcall ExtractRelativePath(const AnsiString BaseName, const AnsiString DestName);

　　功能：从文件名中抽取相对路径信息，如"../sss/ss.asd"这种形式

　　参数：BaseName：基准文件名；DestName：目标文件名

　　例：ShowMessage(ExtractRelativePath("D://Source//c//1.123","D://Source//Asm//dz.asm"));/\*显示"../asm/dz.asm"\*/

　　(10) ExtractShortPathName()   
　　原型：extern PACKAGE AnsiString \_\_fastcall ExtractShortPathName(const AnsiString FileName);

　　功能：把文件名转换为DOS的8、3格式

　　参数：FileName:要处理的文件名

　　例：ShowMessage(ExtractShortPathName("E://Program Files//Dual Wheel Mouse//4dmain.exe"));/\*显示"E:/Progra~1/dualwh~1/4dmain.exe"\*/

　　⑾MatchesMask()   
　　原型：extern PACKAGE bool \_\_fastcall MatchesMask(const AnsiString Filename, const AnsiString Mask);

　　功能：检查文件是否与指定的文件名格式匹配

　　参数：FileName:要处理的文件名；Mask:文件名格式，支持通配符

　　例：ShowMessage(MatchesMask("Lxf.exe","\*.?x?));//显示"true"

2、文件管理函数   
　　这类函数包括设置和读取驱动器、子目录和文件的有关的各种操作，下表列出这类操作常用的函数及其功能。

　　函数 功能   
　　CreateDir() 创建新的子目录   
　　DeleteFile() 删除文件   
　　DirectoryExists() 判断目录是否存在   
　　DiskFree() 获取磁盘剩余空间   
　　DiskSize() 获取磁盘容量   
　　FileExists() 判断文件是否存在   
　　FileGetAttr() 获取文件属性   
　　FileGetDate() 获取文件日期   
　　GetCurrentDir() 获取当前目录   
　　RemoveDir() 删除目录   
　　SetCurrentDir() 设置当前目录

　　下面就把这些函数作一一介绍：

　　⑴CreateDir()   
　　原型：extern PACKAGE bool \_\_fastcall CreateDir(const System::AnsiString Dir);

　　功能：建立子目录，如果成功返回true，否则返回false

　　参数：Dir:要建立的子目录的名字

　　例：Create("ASM");//在当前目录下建立一个名为ASM的子目录

　　⑵DeleteFile()   
　　原型：extern PACKAGE bool \_\_fastcall DeleteFile(const System::AnsiString FileName);

　　功能：删除文件，如果成功返回true，否则返回false

　　参数：FileName:要删除的文件名

　　例：if(OpenDialog1->Execute())DeleteFile(OpenDialog1->FileName);

　　⑶DirectoryExists()   
　　原型：extern PACKAGE bool \_\_fastcall DirectoryExists(const System:: AnsiString Name);

　　功能：检测目录是否存在，如果存在返回true，否则返回false

　　参数：Name:要检测的目录名

　　例：if(!DirectoryExists("ASM"))CreateDir("ASM");//如果ASM这个目录不存在则创建之

　　⑷DiskFree()   
　　原型：extern PACKAGE \_\_int64 \_\_fastcall DiskFree(Byte Drive);

　　功能：检测磁盘剩余空间，返回值以字节为单位，如果指定的磁盘无效，返回-1

　　参数：Drive：磁盘的代号，0表示当前盘, 1=A,2=B,3=C 以此类推

　　例：ShowMessage(DiskFree(0));//显示当前盘的剩余空间

　　⑸DiskSize()   
　　原型：extern PACKAGE \_\_int64 \_\_fastcall DiskSize(Byte Drive);

　　功能：检测磁盘容量，返回值以字节为单位，如果指定的磁盘无效，返回-1

　　参数：Drive：磁盘的代号，0表示当前盘, 1=A,2=B,3=C 以此类推

　　例：ShowMessage(DiskFree(0));//显示当前盘的容量

　　⑹FileExists()   
　　原型：extern PACKAGE bool \_\_fastcall FileExists(const AnsiString FileName);

　　功能：检测文件是否存在，如果存在返回true，否则返回false

　　参数：FileName:要检测的文件名

　　例：if(FileExists("AAA.ASM"))DeleteFile("AAA.ASM");

　　⑺FileGetAttr()   
　　原型：extern PACKAGE int \_\_fastcall FileGetAttr(const AnsiString FileName);

　　功能：取得文件属性，如果出错返回-1

　　返回值如下表，如果返回$00000006表示是一个具有隐含和系统属性的文件(4+2)

　　常量 值 含义   
　　faReadOnly $00000001 只读文件   
　　faHidden $00000002 隐含文件   
　　faSysFile $00000004 系统文件   
　　faVolumeID $00000008 卷标   
　　faDirectory $00000010 目录   
　　faArchive $00000020 归档文件

　　例：if(FileGetAttr("LLL.TXT")&0x2)ShowMessage("这是一个有隐含属性的文件");

　　与此对应的有FileSetAttr() ，请自已查阅帮助系统

　　⑻FileGetDate()   
　　原型：extern PACKAGE int \_\_fastcall FileGetDate(int Handle);

　　功能：返回文件的建立时间到1970-1-1日0时的秒数

　　参数：Handle:用FileOpen()打开的文件句柄。

　　例：

　　int i=FileOpen("C://autoexec.bat",fmOpenRead);   
　　ShowMessage(FileGetDate(i));   
　　FileClose(i);

　　与此对应的有FileSetDate()，请自已查阅帮助系统

　　⑼GetCurrentDir()   
　　原型：extern PACKAGE AnsiString \_\_fastcall GetCurrentDir();

　　功能：取得当前的目录名

　　例：ShowMessage(GetCurrentDir());

　　⑽RemoveDir()   
　　原型：extern PACKAGE bool \_\_fastcall RemoveDir(const AnsiString Dir);

　　功能：删除目录，如果成功返回true，否则返回false

　　参数：Dir:要删除的目录名

　　例：if(DiectoryExists("ASM"))RemoveDir("ASM");

　　⑾SetCurrentDir()   
　　原型：extern PACKAGE bool \_\_fastcall SetCurrentDir(const AnsiString Dir);

　　功能：设置当前目录，如果成功返回true，否则返回false

　　参数：Dir:要切换到的目录名

　　例：SetCurrentDir("C://WINDOWS");

3、文件I/O函数   
　　这类函数完成对文件的读写相关的操作，这种类型的操作和C的基于I/O文件操作类似，下表列出这类操作常用的函数及其功能。

　　FileOpen() 打开文件   
　　FileClose() 关闭文件   
　　FileRead() 读文件   
　　FileSeek() 文件定位   
　　FileWrite() 写文件   
　　FileCreate() 创建文件

　　下面就对这些函数作详细介绍。

　　⑴FileOpen()   
　　原型：extern PACKAGE int \_\_fastcall FileOpen(const AnsiString FileName, int Mode);

　　功能：打开文件，如果成功返回其句柄，否则返回-1

　　参数：FileName:要打开的文件名；Mode：打开的方式，取值如下表，可用"或"("|")运算符连接。

　　常量 值 说明   
　　-------------------------------------------------------------   
　　fmOpenRead 0 以只读属性打开   
　　fmOpenWrite 1 以只写属性打开   
　　fmOpenReadWrite 2 以读/写属性打开   
　　fmShareCompat 0 兼容FCB方式(汇编中有相应的DOS功能调用，感兴趣自已查阅相关资料)   
　　fmShareExclusive 16 共享方式：以独占方式打开，在关闭以前，别人不能访问   
　　fmShareDenyWrite 32 共享方式：拒绝写访问   
　　fmShareDenyRead 48 共享方式：拒绝读访问   
　　fmShareDenyNone 64 共享方式：无限制，允许读写   
　　例：int i=FileOpen("C://WINDOWS//Win.ini",fmOpenReadWrite|fmShareExclusive);

　　⑵FileClose()   
　　原型：extern PACKAGE void \_\_fastcall FileClose(int Handle);

　　功能：关闭打开的句柄。

　　参数：Handle:要关闭的句柄

　　例：FileClose(i);

　　⑶FileRead()   
　　原型：extern PACKAGE int \_\_fastcall FileRead(int Handle, void \*Buffer, int Count);

　　功能：读文件，返回实际读取的字节数，句柄必须首先由FileOpen或FileCreate创建。

　　参数：Handle:要读取的句柄；Buffer:存放读取的数据的缓冲区；Count:想读取的字节数

　　例：char str[400];FileRead(hnd1,str,400);

　　⑷FileSeek()   
　　原型：extern PACKAGE int \_\_fastcall FileSeek(int Handle, int Offset, int Origin);

　　功能：移动文件读取指针，成功返回文件指针的位置，失败返回-1

　　参数：Handle:相关联的句柄；Offset:移动的量；Orgin:移动的基准，0=文件头，1=当前位置，2=文件尾。

　　例：ShowMessage(FileSeek(hnd1,0,2));//取得文件的长度

　　⑸FileWrite()   
　　原型：extern PACKAGE int \_\_fastcall FileWrite(int Handle, const void \*Buffer, int Count);

　　功能：写文件，返回实际写入的字节数，句柄必须首先由FileOpen或FileCreate创建。

　　参数：Handle:要写入的句柄；Buffer:存放写入数据的缓冲区；Count:想写入的字节数

　　例：char str[]="I Love You";FileWrite(hnd1,str,strlen(str));

　　⑹FileCreate()   
　　原型：extern PACKAGE int \_\_fastcall FileCreate(const AnsiString FileName);

　　功能：创建文件。成功返回其句柄，否则返回-1

　　参数：FileName:要创建的文件名

　　例：if(!FileExists("KC.C"))hnd1=FileCreate("KC.C");