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char fgetc(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。  
  
例：tmpnam(str1);  
  
  
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。   
  
　   
  
　　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<<hex<<123;要把3.1415926以5位精度输出：file1<<setpxecision(5)<<3.1415926。  
  
　　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