# 程序设计与算法(一)

## C语言程序设计

郭 炜

微信公众号

微博：http://weibo.com/guoweiofpku

**学会程序和算法，走遍天下都不怕!**

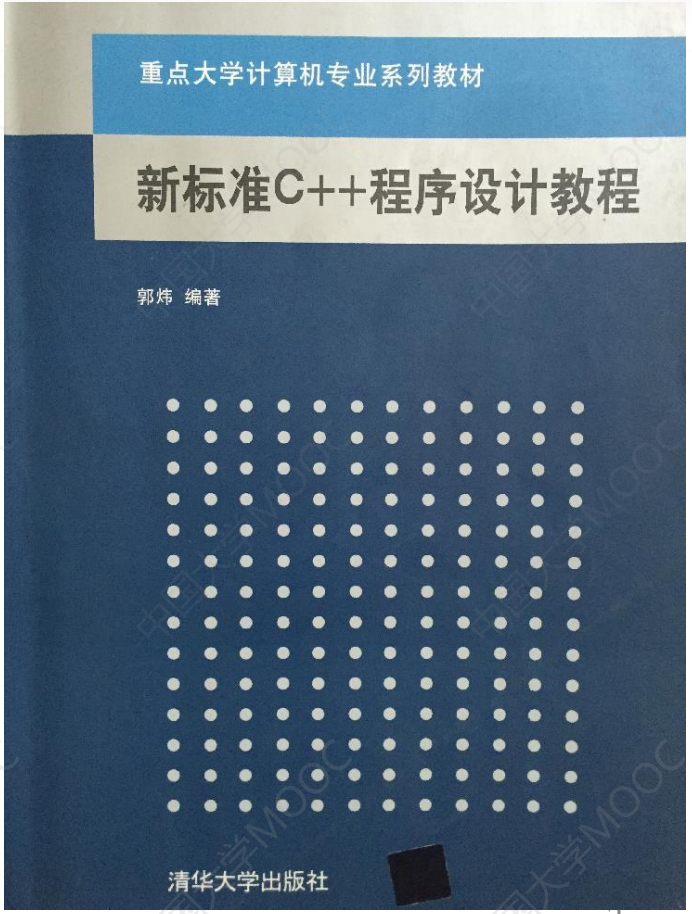
讲义照片均为郭炜拍摄

指定教材：

**《新标准C++程序设计教程》**

郭炜 编著

清华大学出版社



# 文件读写

中国大学MOOC

# 文件读写概述

九寨沟

中国大学MOOC

# 二进制文件和文本文件

- 文本文件

  内容是文字（哪国语言都行），用记事本打开能看到文字的文件。

- 二进制文件

  本质上所有文件都是0,1串，因此都是二进制文件。但是一般将内容不是文字，记事本打开看是乱码的文件，称为二进制文件。word文件记事本打开看是乱码，因此也是二进制文件。

中国大学MOOC

# 文件读写过程

- 文件读写相关函数在头文件 cstdio (或stdio.h) 中声明

  #include <cstdio>

- fopen函数打开文件, 返回 FILE * 指针，指向和文件相关的一个FILE变量 (FILE 是在cstdio中定义的struct)

- fscanf、fread,fgets,fgetc 等函数读文件，fprintf, fwrite,fputs,fputc 等函数写文件，都需要通过fopen返回的FILE *指针进行

- 文件读写结束后，一定要fclose关闭文件！！！

  否则可能导致写入文件的数据没被保存，或者无法打开其它文件

# 打开文件的函数

- FILE *  fopen(char * filename, char * mode)

- 以mode模式打开文件 filename，如果打开失败，返回NULL
  如果成功，返回一个 FILE * 指针，以后对文件的操作均通过该指针进行。

- FILE 是一个struct，记录被打开文件的各种信息。

- 打开失败的原因：文件不存在，文件名不合法，试图以写方式打开只读文件，没有权限……

中国大学MOOC

# 打开文件的模式

- fopen函数mode参数取值：

"r"： 读模式。打开文件用于读取信息。

"w"：写模式。创建打开文件用于写入信息。如果filename已经存在，则原文件立即丢失。

"a"： 添加模式。打开已有文件，在尾部添加新内容。如果文件不存在，则创建之。所有写入操作的效果都是往尾部添加内容。

"r+"：读写模式。打开文件，可读可写。如果文件本来存在，则原内容不会丢失；如果文件本来不存在，打开失败。

打开模式中加"b"即指定为二进制方式打开，如"rb","wb","ab","r+b"否则称为以文本方式打开。

# 二进制打开和文本打开

Linux,Unix下的换行符号：　‘\n’　(ASCII码：0x0a)
Windows 下的换行符号：　‘\r\n’　(ASCII码：　0x0d0a)
Mac OS下的换行符号：　‘\r’　(ASCII码：0x0d)

导致 Linux, Mac OS 文本文件在Windows 记事本中打开时不换行
openjudge是Linux系统，测试点数据即如此。

中国大学MOOC

# 二进制打开和文本打开

- Unix/Linux/Mac OS下打开文件，是否二进制方式无区别
- Windows下打开文件，如果模式不加"b"，即文本方式打开，则：

  读取文件时，所有的 '\r\n’ (0x0d0x0a)会被当做一个字符'\n'处理，即少读了一个字符'\r'。

  写入文件时，写入单独的'\n'(0x0a)时，系统自动在前面加一个'\r'，即多写了一个'\r'(0x0d)

  即要完全准确读写文件，应该用二进制方式打开。

中国大学MOOC

# 文件名的绝对路径和相对路径

- 绝对路径:

```
"c:\\tmp\\mydir\\some.txt"
```

- 相对路径:

```
"\\tmp\\mydir\\some.txt"
```

当前盘符的根目录下的tmp\dir\some.txt

```
"tmp\\mydir\\some.txt"
```

当前文件夹的tmp子文件夹里面的.....

```
"..\\tmp\\mydir\\some.txt"
```

当前文件夹的父文件夹下面的tmp子文件夹里面的.....

```
"..\\..\\tmp\\mydir\\some.txt"
```

当前文件夹的父文件夹的父文件夹下面的tmp子文件夹里面的.....

默认情况下，程序运行时的当前文件夹就是程序可执行文件所在的文件夹

# 文本文件读写

九寨沟

中国大学MOOC

# 文本文件读写

- int fscanf(FILE * fp, char * fmt, par1, par2 ....);

除第一个参数外，其余参数格式和用法同**scanf**。

和**scanf**的区别在于不是从键盘读入数据，而是从**fp**关联的文件中读入数据

- int fprintf (FILE * fp, char * fmt, par1, par2 ....);

除第一个参数外，其余参数格式和用法同 **printf**。

和**printf** 的区别在于不是将文字输出到屏幕，而是输出到**fp**关联的文件。

# 文本文件读写

- 写一个程序，将文件 in.txt 里面的整数排序后，输出到out.txt

例如，若in.txt 的内容为：

```
1 234 9 45 6 879
```

则执行本程序后，生成的out.txt的内容为：

```
1 6 9 45 234 879
```

中国大学MOOC

```
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
int a[1000];
int main()  {
        FILE * fpIn = fopen("in.txt","r");
        if ( fpIn == NULL)
                return 0; //文件打开失败
        FILE * fpOut = fopen("out.txt","w");
        int n = 0;
        while( fscanf(fpIn,"%d",a+n) != EOF)
                ++n;
        fclose(fpIn);
        sort(a,a+n);
        for (int i = 0;i < n; ++i)
                fprintf(fpOut,"%d ",a[i]);
        fclose(fpOut);
        return 0;
}
```

# 文本文件读写

- char * fgets ( char * str, int num, FILE * fp);

读取若干字符到str，碰到以下三种情况之一读取就结束：

1） 已经读入 （num-1） 个字符

2）碰到了换行符（换行符会被读入）

3） 文件全部读完

会自动往读入的最后一个字符后面加'\0'

读入成功则返回str，否则返回NULL

中国大学MOOC

# 文本文件读写

- 用fgets读取整个文本文件并输出

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
        char str[1000];
        FILE * fp = fopen("tmp.txt","r");
        while( fgets(str,990,fp))
                printf(str);
        fclose(fp);
        return 0;
}
// 对ANSI编码的文件能正确输出。对utf-8编码文件中的汉字，无法正确输出
```

# 文本文件读写

- int fputs (char * str, FILE * fp);

将字符串str输出到文件fp。若不成功，返回EOF(一般就是-1)

中国大学MOOC

# 二进制文件读写

张家界天门山

中国大学MOOC

# 文件的读写指针

- 每个打开的文件都有一个“读写指针”
- 读写指针标识文件读写操作的当前位置，该指针在哪里,读写操作就在哪里进行。
- 以"a"模式打开文件，读写指针指向文件末尾。其它情况读写指针指向文件开头

- "读写指针"和fopen返回的FILE * 指针不是一回事

# 文件的读写指针

- `int fseek ( FILE * fp, long long offset, int origin );`

将文件读写指针定位到距离origin位置offset字节处

origin取值：

SEEK_SET:　　文件开头（偏移量0处),此时offset为非负数
SEEK_END:　　文件末尾，此时offset为非正数
SEEK_CUR:　　当前文件读写指针位置，此时offset可正可负

# 二进制文件读写

- 用fread进行二进制读文件：

```
unsigned int fread ( void * ptr, unsigned int size,
                     unsigned int count, FILE * fp );
```

将文件读写指针指向的地方的 size*count 个字节内容，读入到内存地址ptr，然后将文件读写指针向后移动size*count字节。

返回值为： 成功读取的字节数/size

# 二进制文件读写

- 用fgetc 进行二进制读文件：

```
int fgetc ( FILE * fp );
```

读取文件fp中的下一个字节。如果文件已经读完，返回EOF(-1)。

注意必须用int存放函数返回值，不可用char存放返回值。否则函数返回 -1 (0xffffffff)时无法区分是读取了一个ASCII编码为0xff的字符，还是文件已经读完。

# 二进制文件读写

- 用fwrite二进制写文件：

```
unsigned int fwrite ( void * ptr, unsigned int size,
                      unsigned int count, FILE * fp );
```

将内存地址ptr处的size*count个字节内容，写入到文件读写指针指向的地方，然后将文件读写指针向后移动size*count字节。

返回值为： 成功写入的字节数/size

# 二进制文件读写

- 用fputc二进制写文件：

```
int fputc ( int c, FILE * fp );
```

将 c 的低8位写入文件fp。

中国大学MOOC

# 二进制文件读写

- 在文件中写入和读取一个整数

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()  {
   FILE * fp = fopen("some.dat","wb");
   int x,y;
   scanf("%d",&x);
   fwrite(&x,sizeof(int),1,fp);
   fclose(fp);
   fp = fopen("some.dat","rb");
   fread(&y,sizeof(int),1,fp);
   fclose(fp);
   printf("%d",y);
   return 0;
}
```

# 二进制文件读写

- 从键盘输入几个学生的姓名和成绩，并以二进制文件形式保存

```
#include <iostream>
#include <cstdio>
using namespace std;
struct Student {
        char name[20];
        int score;
};
int main()  {
   Student s;
   FILE * fp = fopen("c:\\tmp\\students.dat","wb");
   while (scanf("%s%d",s.name,&s.score) != EOF)
        fwrite(&s, sizeof(s),1,fp);
   fclose(fp);
   return 0;
}
```

输入：

Tom 60
Jack 80
Jane 40
^Z+回车

则形成的 students.dat 为 72字节，用 记事本打开，呈现：

Tom ?? p R< Jack ?? p RP Jane ?? p R(

# 二进制文件读写

- 将 students.dat 文件的内容读出并显示

```
#include <iostream>
#include <cstdio>
using namespace std;
struct Student {
        char name[20];
        int score;
};
int main()  {
   Student s;
   FILE * fp = fopen("c:\\tmp\\students.dat","rb");
   while(fread(&s,sizeof(s),1,fp) == 1 )
        printf("%s %d\n",s.name,s.score);
   fclose(fp);
   return 0;
}
```

输出：

Tom 60

Jack 80

Jane 40

中国大学MOOC

# 二进制文件读写

## 一次读写一个数组

Student  students[20];

```
fread(students,sizeof(Student),20,fp);
fwrite(students,sizeof(Student),20,fp);
```

中国大学MOOC

# 二进制文件读写

用二进制文件存学生信息比用文本方式存的好处：

1） 可能节约空间
2） 便于快速读取、改写单个学生信息
3） 便于排序、查找学生信息

中国大学MOOC

# 二进制文件读写

- 将 students.dat 文件的Jane的名字改成Mike

```
#include <iostream>
#include <cstdio>
using namespace std;
struct Student {
        char name[20];
        int score;
};
```

# 二进制文件读写

- 将 students.dat 文件的Jane的名字改成Mike

```
int main()
{
    Student s;
    FILE * fp = fopen("c:\\tmp\\students.dat","r+b");
    if( !fp) {
        printf("error\n");     return 0;
    }
    fseek(fp, 2 * sizeof(s),SEEK_SET);    //定位读写指针到第三个记录
    fwrite("Mike",1,5,fp);
    fseek(fp,0,SEEK_SET);                 //定位读写指针到文件开头
    while( fread(&s,sizeof(s),1,fp) == 1)
        printf("%s %d\n",s.name,s.score);
    fclose(fp);
    return 0;
}
```

输出:
Tom 60
Jack 80
Mike 40

# 二进制文件读写

- 重要注意事项：

用"r+b"打开文件既读又写时，如果做了读操作，则做写操作之前一定要用fseek重新定位文件读写指针，哪怕此时指针位置恰好是正确的；同理如果做了写操作，则再做读操作之前也一定要用fseek重新定位文件读写指针。否则读写都不正确，可能造成文件数据混乱、丢失

# 文件拷贝程序mycopy 示例

```
/*用法示例:
  mycopy src.dat dest.dat
  即将 src.dat 拷贝到 dest.dat，如果 dest.dat 原来就有，则原来的文件会被覆
    盖   */
#include <iostream>
#include <cstdio>
using namespace std;
int main(int argc, char * argv[])
{
   if( argc != 3 ) {
       printf("File name missing!\n");
       return 0;
   }
```

# 文件拷贝程序mycopy 示例

```
FILE * fpIn = fopen(argv[1],"rb");//打开文件用于读
if( !fpIn ) {
    printf("Source file open error.\n");
    return 0;
}
FILE * fpOut = fopen(argv[2],"wb"); //打开文件用于写
if( !fpOut) {
    printf("New file open error.\n");
    fclose(fpIn); //打开的文件一定要关闭
    return 0;
}
```

# 文件拷贝程序mycopy 示例

```
    int c;   //必须是int
    while( (c = fgetc(fpIn)) != EOF) //每次读取一个字节
        fputc(c,fpOut);   //每次写入一个字节
    fclose(fpIn);
    fclose(fpOut);
    return 0;
}
```

**更好的做法2：蓝色三行替换成：**

```
    char buf[4096];
    int n;
    while( n = fread(buf,1,4096,fpIn))
        fwrite(buf,1,n,fpOut);
```