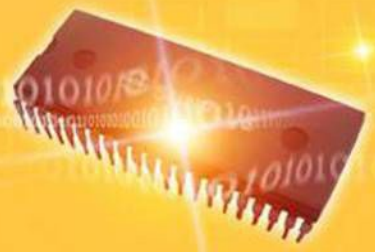
凌阳教育
www.sunplusedu.com
值得信赖的教育品牌

嵌入式系统工程师



Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

文件操作

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

大纲

- 文件的基本概念
- C语言对文件处理
- 文件的基本练习

大纲

- 文件的基本概念
- C语言对文件处理
- 文件的基本练习

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

文件的基本概念

- 凡是使用过计算机的人都不会对“文件”感到陌生

- 文件用来存放程序、文档、音频、视频数据、图片等
- 程序员，必须掌握编程实现创建、写入、读取文件等操作

文件的基本概念

- 文件的定义:
  - 磁盘文件: （我们通常认识的文件）

    指一组相关数据的有序集合,通常存储在外部介质(如磁盘)上，使用时才调入内存。
  - 设备文件:

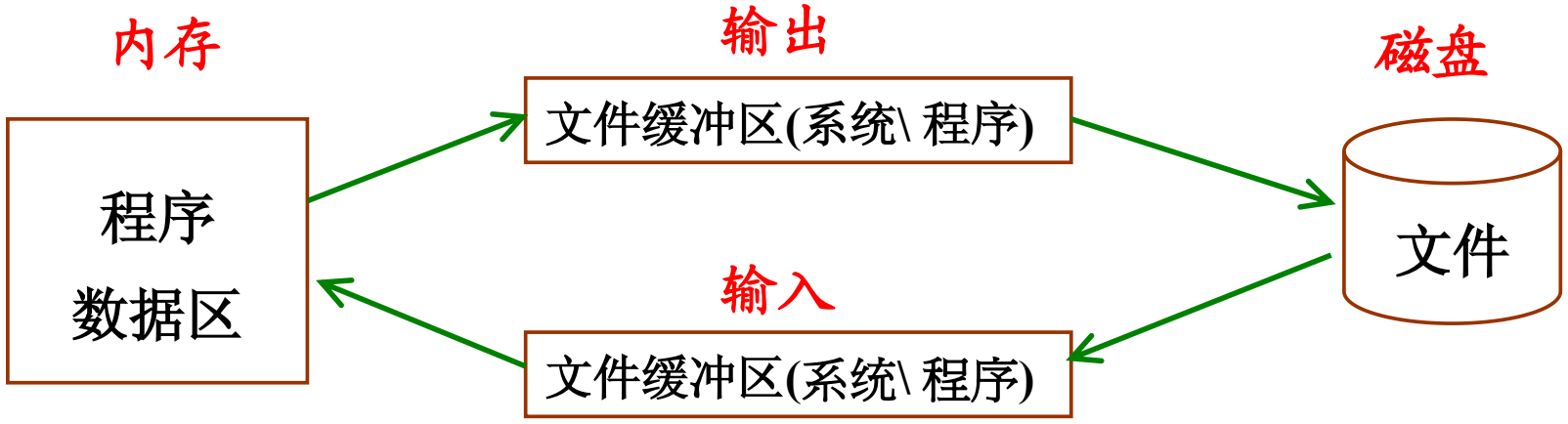
    在操作系统中把每一个与主机相连的输入、输出设备看作是一个文件，把它们的输入、输出等同于对磁盘文件的读和写。

    键盘：标准输入文件　　屏幕：标准输出文件

    其它设备：打印机、触摸屏、摄像头、音箱等

文件的基本概念

➢ 磁盘文件的存取



- 磁盘文件，一般保存在硬盘、光盘、U盘等掉电不丢失的磁盘设备中，在需要时调入内存
- 在内存中对文件进行编辑处理后，保存到磁盘中
- 程序与磁盘之间交互，不是立即完成，系统或程序可根据需要设置缓冲区，以提高存取效率

文件的基本概念

- 磁盘文件的分类
  - 一个文件通常是磁盘上一段命名的存储区
  - 计算机的存储在物理上是二进制的，所以物理上所有的磁盘文件本质上都是一样的：

    以字节为单位进行顺序存储
  - 从用户或者操作系统使用的角度（逻辑上）

    把文件分为：

    文本文件：基于字符编码的文件

    二进制文件：基于值编码的文件

文件的基本概念

- 文本文件:
  - 基于字符编码，常见编码有ASCII、UNICODE等
  - 一般可以使用文本编辑器直接打开

  例如：数5678的以ASCII存储形式为:
  ASCII码：00110101 00110110 00110111 00111000

  歌词文件(lrc):文本文件
- 二进制码文件:
  - 基于值编码,自己根据具体应用,指定某个值是什么意思
  - 把内存中的数据按其在内存中的存储形式原样输出到磁盘上
  - 一般需要自己判断或使用特定软件分析数据格式

  例如：数5678的存储形式为:
  二进制码：00010110 00101110
  音频文件(mp3):二进制文件

文件的基本概念

- 文本文件、二进制文件对比:
  - 译码:

    文本文件编码基于字符定长，译码容易些;

    二进制文件编码是变长的，译码难一些（不同的二进制文件格式，有不同的译码方式）。
  - 空间利用率:

    二进制文件用一个比特来代表一个意思(位操作);

    而文本文件任何一个意思至少是一个字符。
  - 可读性:

    文本文件用通用的记事本工具就几乎可以浏览所有文本文件

    二进制文件需要一个具体的文件解码器，比如读BMP文件，必须用读图软件。

大纲

- 文件的基本概念
- C语言对文件处理
  - 文件的打开与关闭
  - 文件的顺序读写
  - 文件的随机读写
  - 文件的出错检测
- 文件的基本练习

文件的打开与关闭

- C语言中不能直接操作文件

  采用**库函数**间接对文件进行操作

- C语言操作文件的基本流程为：
  - 在使用文件前要调用**打开函数**将文件打开

    打开文件会得到一个文件指针fp
  - 调用各种有关函数，利用fp对文件进行具体处理(**读或写**)
  - 在文件用完时，及时调用**关闭函数**来关闭文件

C语言中所有的文件操作都围绕文件指针完成

文件的打开与关闭

- 定义文件指针的一般形式为:

  FILE * 指针变量标识符;

  - FILE为大写，需要包含<stdio.h>
  - FILE是系统使用typedef定义出来的有关文件信息的一种结构体类型，结构中含有文件名、文件状态和文件当前位置等信息
  - 一般情况下，我们操作文件前必须定义一个文件指针**指向我们将要操作的文件**

- **实际编程中使用库函数操作文件，无需关心FILE结构体的细节**

文件的打开与关闭

➢ FILE在stdio.h文件中的文件类型声明:

```
typedef  struct
{   short level;                    //缓冲区“满”或“空”的程度
    unsigned flags;                 //文件状态标志
    char fd;                        //文件描述符
    unsigned charhold;              //如无缓冲区不读取字符
    short bsize;                    //缓冲区的大小
    unsigned char *buffer;          //数据缓冲区的位置
    unsigned ar*curp;               //指针，当前的指向
    unsigned istemp;                //临时文件，指示器
    shorttoken;                     //用于有效性检查
} FILE;
```

在缓冲文件系统中,每个被使用的文件都要在内存中开辟一块FILE类型的区域,存放与操作文件相关的信息

文件的打开与关闭

- c语言中有三个特殊的文件指针无需定义、打开可直接使用:

  stdin: 标准输入 默认为当前终端（键盘）

  我们使用的scanf、getchar函数默认从此终端获得数据

  stdout: 标准输出 默认为当前终端（屏幕）

  我们使用的printf、puts函数默认输出信息到此终端

  stderr: 标准出错 默认为当前终端（屏幕）

  当我们程序出错或者使用:perror函数时信息打印在此终端

文件的打开与关闭

- 任何文件使用之前必须打开，使用后必须关闭
- **打开文件**

```
FILE * fp = NULL;
fp = fopen（文件名，文件使用方式）;
```

①文件名:

要操作的文件的名字、可以包含路径信息

②文件使用方式:

"读"、"写"、"文本"或"二进制"等

③fp文件指针:

指向被打开的文件，失败返回空，成功返回相应指针

文件的打开与关闭

➢ 例:

```
FILE * fp_passwd = NULL;
fp_passwd = fopen( “passwd.txt” , “rt” );
if(fp_passwd==NULL)
    printf("file open error");
```

说明:

以**只读方式**打开**当前目录**下一个叫passwd.txt的文件

返回一个文件指针赋给fp_passwd

判断打开是否成功，如果失败，返回NULL（重要）

文件的打开与关闭

- 第一个参数的几种形式
- **相对路径:**

fp_passwd = fopen(“passwd.txt” , “r”);//同例子

打开当前目录passdw文件: 源文件(源程序)所在目录

fp_passwd = fopen(“./test/passwd.txt” , “r”);

打开当前目录(test)下passwd.txt文件

fp_passwd = fopen(“../passwd.txt” , “r”);

打开当前目录上一级目录（相对当前目录）passwd.txt文件

- **绝对路径:**

fp_passwd = fopen(“c://test//passwd.txt” , “r”);

打开C盘test目录下一个叫passwd.txt文件

文件的打开与关闭

- 第二个参数的几种形式（打开文件的方式）

  读写权限：r　w　a　+

- r: 以只读方式打开文件

  文件**不存在**返回NULL;

  文件**存在**返回文件指针，进行后续的读操作

- w: 以只写方式打开文件

  文件**不存在**，以指定文件名创建此文件;

  若文件**存在**，**清空文件内容**，进行写操作;

  如果文件打不开（比如文件只读），返回NULL

- a: 以追加方式打开文件

  文件**不存在**，以指定文件名创建此文件(同w)

  若文件**存在**，从文件的**结尾处**进行写操作

- +: 同时以读写打开指定文件

文件的打开与关闭

- 第二个参数的几种形式（打开文件的方式）

  打开方式：b　t(可以省略)

- 指**打开方式**，与文件的**存储方式**无关，与**操作系统**有关
- **Windows平台下**
  - 以“文本”方式打开文件

    当读取文件的时候，系统会将所有的"\r\n"转换成"\n"

    当写入文件的时候，系统会将"\n"转换成"\r\n"写入
  - 以"二进制"方式打开文件，则读\写都不会进行这样的转换
- **在Unix/Linux平台下**

  “文本”与“二进制”模式没有区别。

  \r\n作为两个字符原样输入输出

文件的打开与关闭

模　式	功　能
r或rb	以只读方式打开一个文本文件（不创建文件）
w或wb	以写方式打开文件（使文件长度截断为0字节，创建一个文件）
a或ab	以添加方式打开文件，即在末尾添加内容，当文件不存在时，创建文件用于写
r+或rb+	以可读、可写的方式打开文件(不创建新文件)
w+或wb+	以可读、可写的方式打开文件 （使文件长度为0字节，创建一个文件）
a+或ab+	以添加方式打开文件，打开文件并在末尾更改文件 （如果文件不存在，则创建文件）

文件的打开与关闭

关闭文件

- 调用的一般形式是:

  fclose (文件指针);

  文件指针: 指向要关闭的文件

- 返回值:

  ①关闭文件成功，返回值为0.

  ②关闭文件失败，返回值非零.

- 例如:

```
FILE * fp = NULL;
fp = fopen( “passwd.txt” , “r” );
fclose(fp);
```

文件的打开与关闭

练习：

1. 以只读、文本方式打开当前路径下一个叫test.txt文件
2. 以只写、二进制方式打开同一个文件
3. 以追加(a)、文本的方式打开同一个文件
4. 同时以读/写、二进制方式打开同一个文件，要求若文件不存在，提示出错
5. 同时以读/些、文本方式打开同一个文件，要求若文件不存在，创建此文件
6. 若打开成功则关闭相应的文件

大纲

- 文件的基本概念
- C语言对文件处理
  - 文件的打开与关闭
  - 文件的顺序读写
  - 文件的随机读写
  - 文件的出错检测
- 文件的基本练习

文件的顺序读写

- **对文件操作最常用的是：“读”和“写”**
- C语言提供了多种对文件读写的函数：

字节读写函数： fgetc和fputc

字符串读写函数： fgets和fputs

数据块读写函数： fread和fwrite

格式化读写函数： fscanf和fprintf

以上函数可完成对文件内容的顺序读写

文件的顺序读写

- 字节的读写

ch = fgetc(fp); //读一个字节

说明:从指定文件读一个字节赋给ch(以“读”或“读写”方式打开)

文本文件: 读到文件结尾返回EOF

二进制文件: 读到文件结尾,使用feof(后面会讲)判断结尾

fputc(ch, fp); //写一个字符

说明:把一个ch变量中的值(1个字节)写到指定的文件

如果输出成功,则返回输出的字节;

如果输出失败,则返回一个EOF。

EOF是在stdio.h文件中定义的符号常量,值为-1

文件的顺序读写

例子： 01.fgetc.c

从一个指定文件(文本文件)中读取所有信息打印在屏幕上

```
1  #include <stdio.h>
2  int main(void)
3  {
4      FILE *fp;
5      char ch;
6      fp=fopen("test.txt","r+");
7      if(fp==NULL)
8      {
9          printf("Cannot open the file\n");
10         return 0;
11     }
12     while(  (ch = fgetc(fp))!=EOF  )
13     {
14         printf("%c",ch);
15     }
16     fclose(fp);
17     return 0;
18 }
```

文件的顺序读写

- 练习
- 从一个文件(文本文件)中读取所有信息，写入另一个文件中
- 参考：r+ w+ fgetc fputc EOF

文件的顺序读写

➢ 字符串的读写

fgets(str,n,fp); //读一个字符串

说明：从fp指向的文件读入n-1个字符

在读入n-1个字符之前遇到换行符或EOF，读入提前结束

在最后加一个’\0’，str为存放数据的首地址

fputs(“china”,fp); //写一个字符串

说明：向指定的文件写一个字符串

第一个参数可以是字符串常量、字符数组名或字符指针

字符串末尾的’\0’不会写到文件中

文件的顺序读写

例子：从一个文件中读取一个字符串，输出到另一个文件中

02.fgets_fputs.c

```c
#include <stdio.h>
int main(void)
{
    FILE *fp_read,*fp_write;
    char string1[100];
    if((fp_read=fopen("src.txt","r+"))==NULL)
    {
        printf("Cannot open the file\n");
        return 0;
    }
    if((fp_write=fopen("dest.txt","w+"))==NULL)
    {
        printf("Cannot open the file\n");
        return 0;
    }
    fgets(string1, 10, fp_read);
    printf("%s\n",string1);
    fputs(string1,fp_write);
    fclose(fp_read);
    fclose(fp_write);
    return 0;
}
```

```c
fgets(string,100,stdin);
fputs(string,stdout);
```

从标准输入读入一个字符串
往标准输出写入一个字符串

文件的顺序读写

- **数据块的读写**

```
fread(buffer,size,count,fp);
fwrite(buffer,size,count,fp);
```

说明:

参数:

buffer: 指向存储数据空间的首地址的指针

size: 一次读写的数据块大小

count: 要读写的数据块个数

fp: 指向要进行写操作的文件指针

返回值:

实际读写的数据块数（不是总数据大小，重要）

文件的顺序读写

03.fread_fwrite.c

例

从键盘输入一个结构体数组数据，输出到文件，再读入并显示

```c
#include <stdio.h>
struct stu
{
    char name[10];
    int num;
    int age;
}boya[10],boyb[2];
int main()
{
    FILE *fp;
    int i;
    if((fp=fopen("test.txt","wb+"))==NULL)
    {
        printf("Cannot open file!");
        return 0;
    }
    printf("input data\n");
    printf("name、num、age、addr: \n");
    for(i=0;i<2;i++)
        scanf("%s %d %d",boya[i].name,&boya[i].num,&boya[i].age);
    fwrite(boya,sizeof(struct stu),2,fp);      //将学生信息写入文件中
    rewind(fp); //文件指针经过写操作已经到了最后，需要复位
    fread(boyb,sizeof(struct stu),2,fp);     //将文件中的数据读入到内存中
    for(i=0;i<2;i++)
        printf("%s %d %d\n",boyb[i].name,boyb[i].num,boyb[i].age);
    fclose(fp);
    return 0;
}
```

文件的顺序读写

➢ 格式化的读写

函数调用:

fprintf（文件指针，格式字符串，输出表列）;

fscanf （文件指针，格式字符串，输入表列）;

函数功能:

从磁盘文件中读入或输出字符

文件的顺序读写

➢ 例：从键盘读入一组数据，写入文件再读出

04.fprintf_fscanf.c

```
#include <stdio.h>
int main(void)
{
    FILE *fp;
    char ch1='a',ch2;
    int num1=50,num2;
    char string1[20]="hello",string2[20];
    float score1=85.5,score2;

    if((fp=fopen("test.txt","wb+"))==NULL)
    {
        printf("Cannot open the file\n");
        return 0;
    }
    fprintf(fp,"%c %d %s %f\n",ch1,num1,string1,score1);
    rewind(fp);
    fscanf(fp,"%c %d %s %f\n",&ch2,&num2,&string2,&score2);
    printf("%c %d %s %f\n",ch2,num2,string2,score2);
    fclose(fp);
    return 0;
}
```

文件的顺序读写

- 注意:
- 用fprintf和fscanf函数对磁盘文件读写
  使用方便，容易理解，但:
  在输入时要将ASCII码转换为二进制形式
  在输出时将二进制形式转换成字符，花费时间较多

- 在内存与磁盘频繁交换数据的情况下，最好不用fprintf和fscanf函数，而用fread和fwrite函数。

大纲

- 文件的基本概念
- C语言对文件处理
  - 文件的打开与关闭
  - 文件的顺序读写
  - 文件的随机读写
  - 文件的出错检测
- 文件的基本练习

文件的随机读写

- 前面介绍的对文件的读写方式都是顺序读写，即读写文件只能从头开始，顺序读写各个数据;
- 但在实际问题中常要求只读写文件中某一指定的部分

  例如：读取文件第200--30个字节

- 为了解决这个问题可以移动文件内部的位置指针到需要读写的位置，再进行读写，这种读写称为随机读写
- 实现随机读写的关键是要按要求移动位置指针，这称为文件的定位.
- 完成文件定位的函数有:

  rewind、ftell、fseek函数

文件定位与随机读写

- **rewind函数**

  void rewind(文件指针);

  函数功能:

  把文件内部的位置指针移到文件首

  调用形式:

  rewind(文件指针);

- 例如:

```
fwrite(pa,sizeof(struct stu),2,fp );
rewind(fp);
fread( pb,sizeof(struct stu),2,fp);
```

前面我们已经多次用到

文件定位与随机读写

ftell函数

定义函数:

long ftell(文件指针);

函数功能:

取得文件流目前的读写位置.

返回值:

返回当前位置(距离文件起始的字节数)，出错时返回-1.

例如:

```
int length;
length = ftell(fp);
```

文件定位与随机读写

- **fseek函数（一般用于二进制文件）**

定义函数:

int fseek(文件类型指针，位移量，起始点);

函数功能:

移动文件流的读写位置.

- 说明：起始位置

文件开头	SEEK_SET	0
文件当前位置	SEEK_CUR	1
文件末尾	SEEK_END	2

位移量:以起始点为基点，向前、后移动的字节数.

文件定位与随机读写

➢ fseek函数应用举例

- fseek(fp, 100, SEEK_SET);
  将位置指针从文件头 向前移100个字节处
- fseek(fp, 50, SEEK_CUR);
  将位置指针从当前位置 向前移动50个字节处
- fseek(fp, -50, SEEK_CUR);
  将位置指针从当前位置 向回移动50个字节处
- fseek(fp, -50, SEEK_END);
  将位置指针从文件尾 退回50个字节处.

文件定位与随机读写

05.fseek.c

例

往文件中写入两个结构体，读取第二个结构体

```
#include <stdio.h>
struct stu
{
    char name[10];
    int num;
    int age;
}boya[2],boyb;
main()
{
    FILE *fp;
    int i;
    if((fp=fopen("test.txt","wb+"))==NULL)
    {
        printf("Cannot open file!");
        return 0;
    }
    printf("name、num、age\n");
    for(i=0;i<2;i++)
        scanf("%s %d %d",boya[i].name,&boya[i].num,&boya[i].age);
    fwrite(boya,sizeof(struct stu),2,fp);
    fseek(fp,sizeof(struct stu),SEEK_SET);
    fread(&boyb,1,sizeof(struct stu),fp);
    printf("%s %d %d\n",boyb.name,boyb.num,boyb.age);
    fclose(fp);
    return 0;
}
```

文件定位与随机读写

- 练习：

  将一个未知大小的文件(文本文件)全部读入内存，并显示在屏幕上

参考：fseek ftell rewind fread malloc

大纲

- 文件的基本概念
- C语言对文件处理
  - 文件的打开与关闭
  - 文件的顺序读写
  - 文件的随机读写
  - 文件的出错检测
- 文件的基本练习

文件检测函数

- 文件结束检测函数feof

调用格式：feof(文件指针);

功能：　判断文件是否处于文件结束位置

常配合fgetc、fgets、fread等读函数判断是否到文件结束

返回值：　文件未结束返回0，文件**已结束**返回非0

文件检测函数

06.feof.c

```c
#include <stdio.h>
int main(void)
{
    FILE *fp;
    char ch;
    fp=fopen("test.txt","r+");
    if(fp==NULL)
    {
        printf("Cannot open the file\n");
        return 0;
    }
    while(1)
    {
        ch = fgetc(fp);
        if(feof(fp)!=0)
            break;
        printf("%c",ch);
    }
    fclose(fp);
    return 0;
}
```


Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

文件检测函数

- 读写文件出错检测函数ferror

调用格式：ferror(文件指针);

功能:

检查文件在用各种输入输出函数进行读写时是否出错

比如：以只读方式打开一个文件，调用写函数就会发生错误

只要出现错误标志，就一直保留，直到对同一文件调用clearerr函数或rewind函数，或任何其他一个输入输出函数。

返回值:

为0表示未出错，否则表示有错

文件检测函数

07.ferror.c

```c
#include <stdio.h>
int main(void)
{
    FILE *fp;
    char ch;
    fp = fopen("DUMMY.FIL", "w");

    ch=getc(fp);
    if (ferror(fp))
    {
        printf("Error reading from DUMMY.FIL\n");
        clearerr(fp);
    }
    fclose(fp);
    return 0;
}
```

文件检测函数

- 文件出错标志和文件结束标志置0函数clearerr

调用格式：clearerr(文件指针);

功能：

本函数用于清除出错标志和文件结束标志，使它们为0值

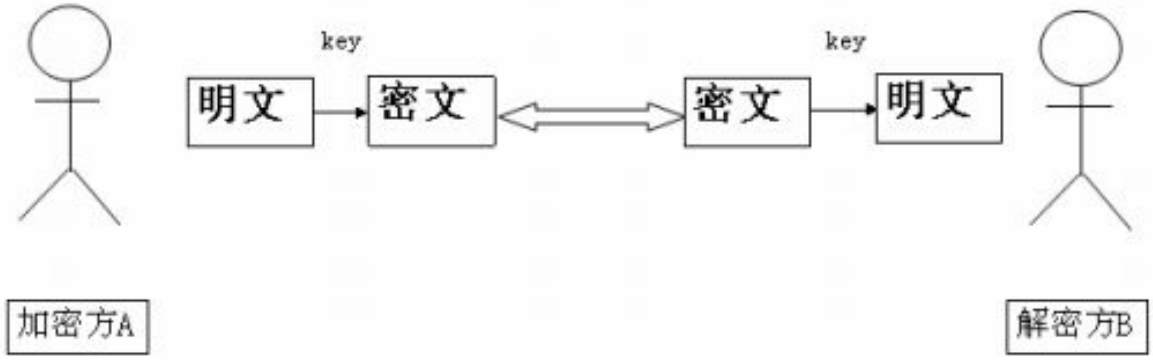
无返回值.

大纲

- 文件的基本概念
- C语言对文件处理
  - 文件的打开与关闭
  - 文件的顺序读写
  - 文件的随机读写
  - 文件的出错检测
- 文件的基本练习

文件检测函数

- 对称加密体制是传统而经典的加密体制策略。
- 对称加密体制即加密方A和解密方B共享一个密钥key
  - 加密方A使用该密钥key对要保密的文件进行加密操作，从而生成密文
  - 解密方B 同样使用该密钥key 对加密方生成的加密文件实施解密操作，从而生成明文。



对称密码体制示意

凌阳教育
www.sunplusedu.com

值得信赖的教育品牌

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com