

# 程序设计基础

## Fundamental of Programming

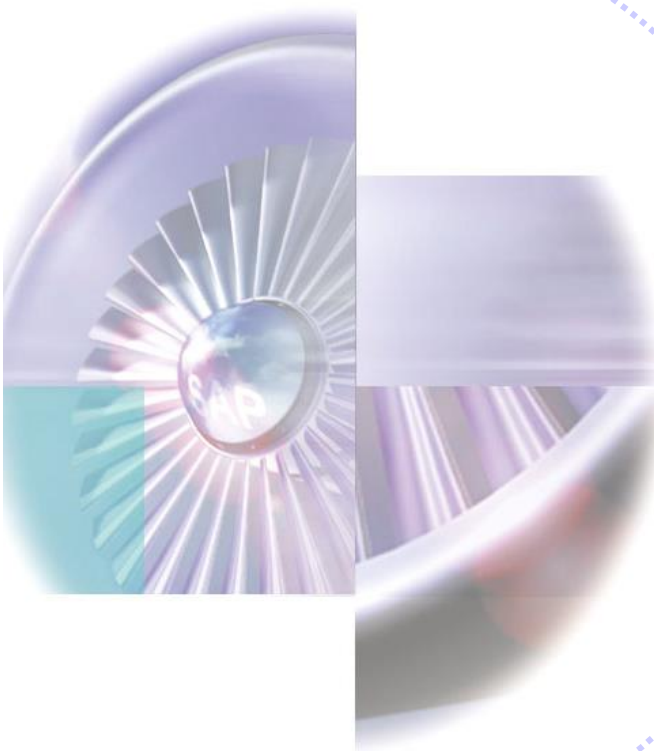
清华大学软件学院

刘玉身

[liuyushen@tsinghua.edu.cn](mailto:liuyushen@tsinghua.edu.cn)

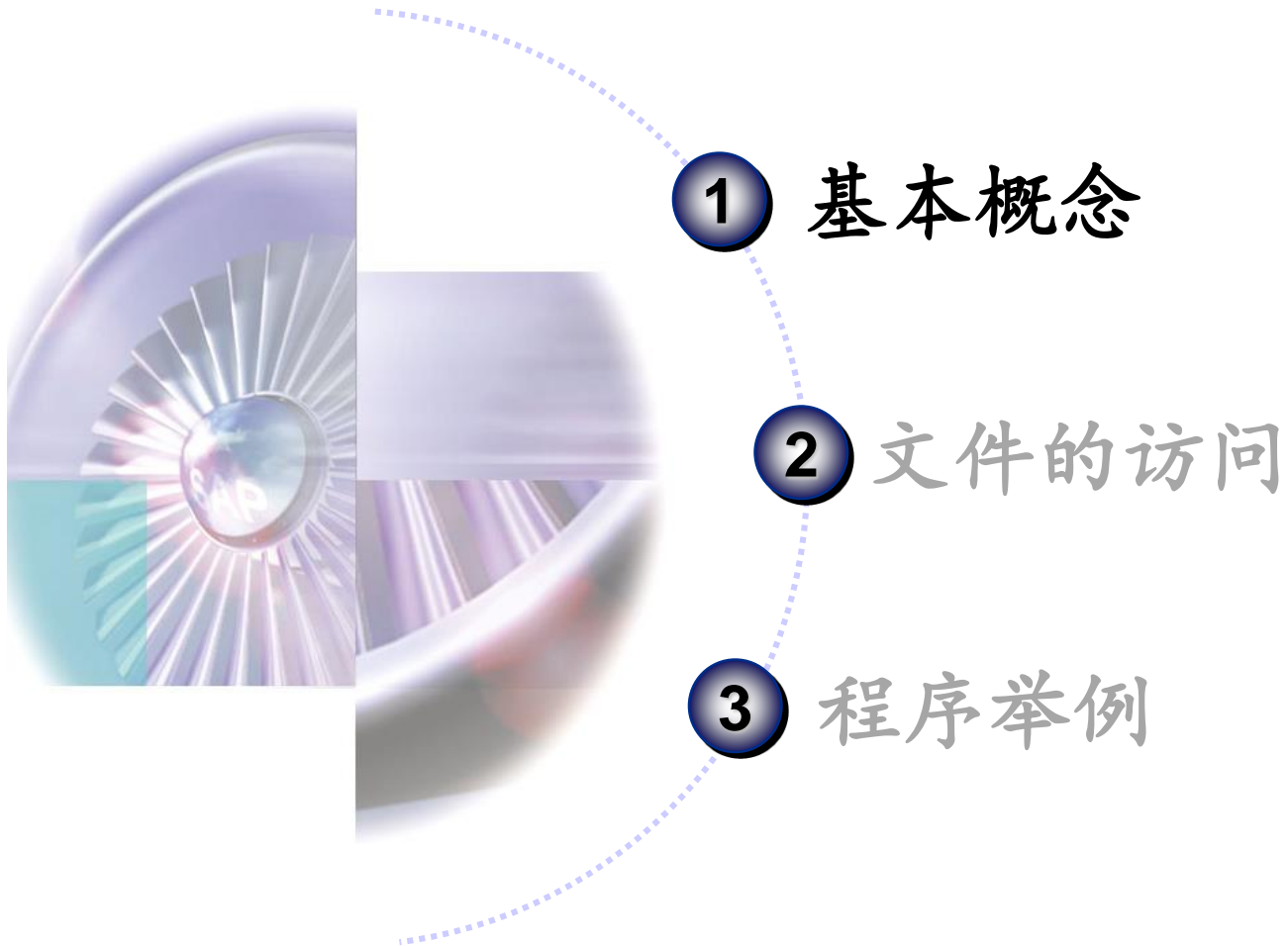
# Lecture 9: 文件

---

- 
- ① 基本概念
  - ② 文件的访问
  - ③ 程序举例

# Lecture 9: 文件

---



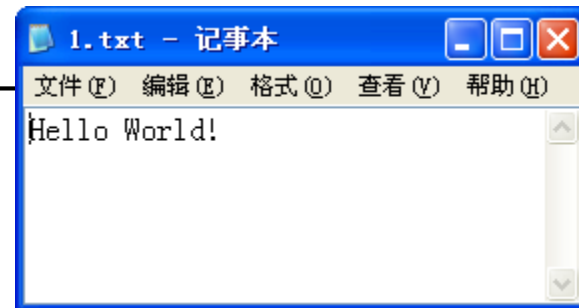
# 文件读写示例

---

## 问题描述：

将字符串 “Hello World!” 写入到磁盘文件 1.txt 中。

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    FILE *fp;                // 定义文件指针
    fp = fopen("1.txt", "w"); // 以写方式打开文件 1.txt
    if(fp == NULL)
    {
        printf("File open error!\n"); exit(0);
    }
    char *str = "Hello World!";
    fprintf(fp, "%s", str);   // 写文件
    if(fclose(fp))            // 关闭文件.返回非0无法正常关闭文件
    {
        printf("Can not close the file!\n"); exit(0);
    }
    return 0;
}
```



# 文件的概念

---

- 文件（File）：
  - 保存在外存储器上的一组数据的有序集合
- 特点：
  - 数据长久保存
  - 数据长度不定
  - 数据按顺序存取

# 文本文件和二进制文件

---

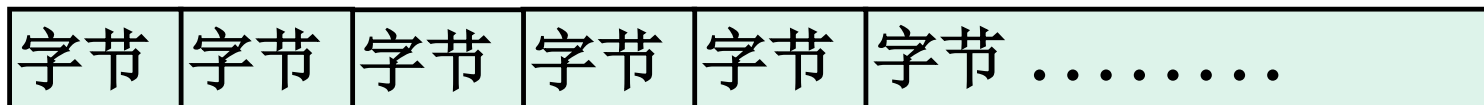
- C语言中的文件是数据流

- 文件的两种数据形式:

- ASCII码 (文本文件 text stream)

- 二进制码 (二进制文件 binary stream)

二进制文件是直接吧内存数据以二进制形式保存。



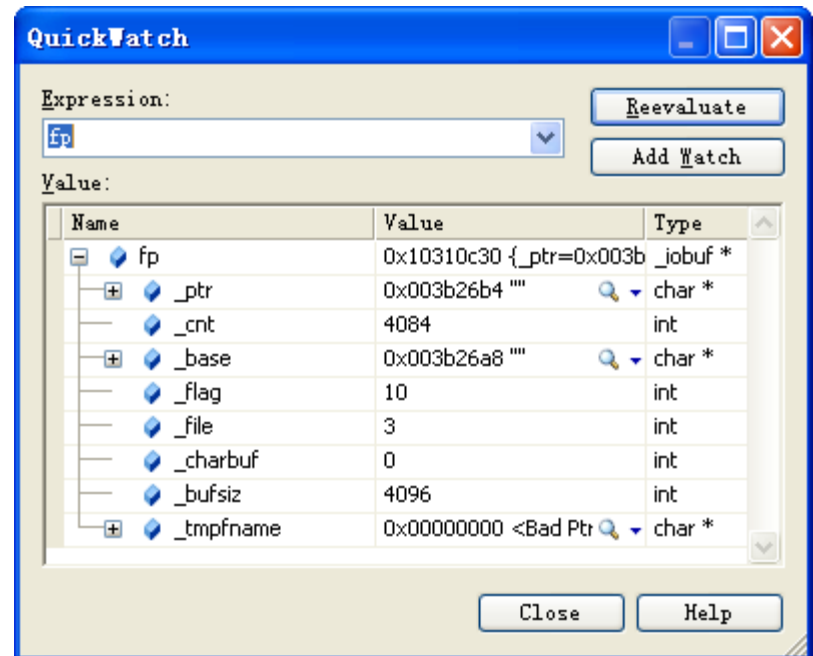
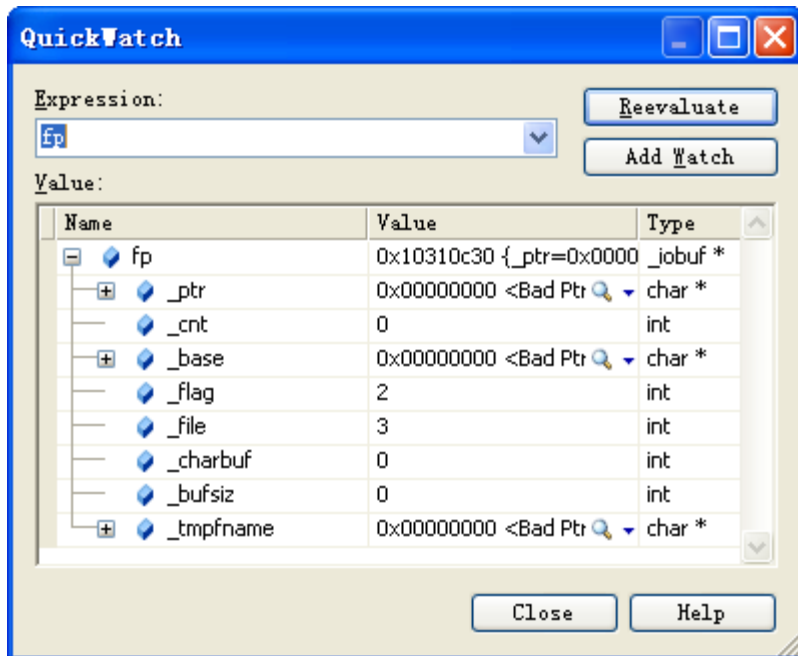
# 文件结构

---

- **FILE**: 结构类型
- 用 **typedef** 定义, **#include <stdio.h>**

```
typedef struct {  
    short          level;    /* 缓冲区使用量 */  
    unsigned       flags;    /* 文件状态标志 */  
    char           fd;       /* 文件描述符 */  
    short          bsize;    /* 缓冲区大小 */  
    unsigned char *buffer;   /* 文件缓冲区的首地址 */  
    unsigned char *curp;     /* 指向文件缓冲区的工作指针*/  
    unsigned char  hold;     /* 其他信息 */  
    unsigned       istemp;  
    short          token;  
} FILE;
```

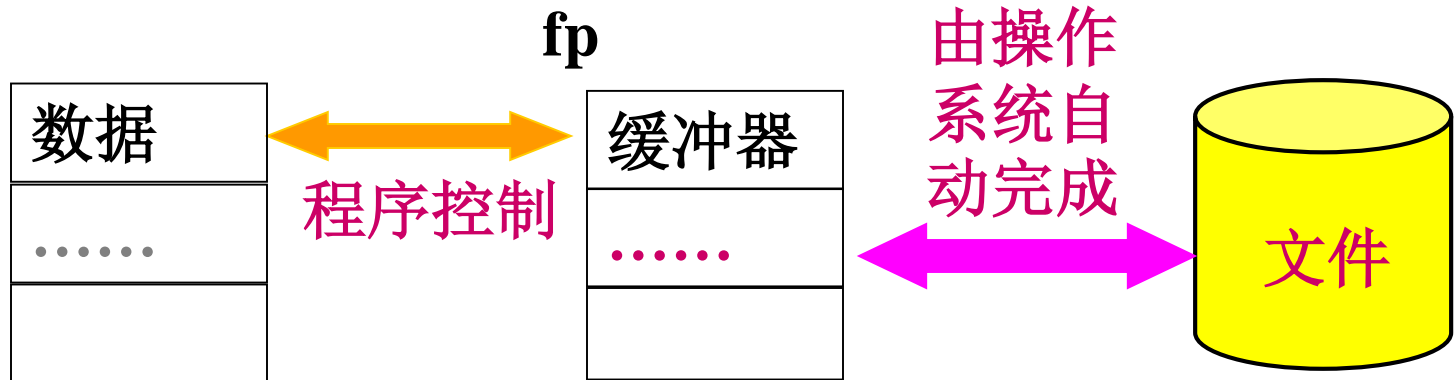




# 缓冲文件与文件类型指针

用文件指针指示文件缓冲区中具体读写的位置

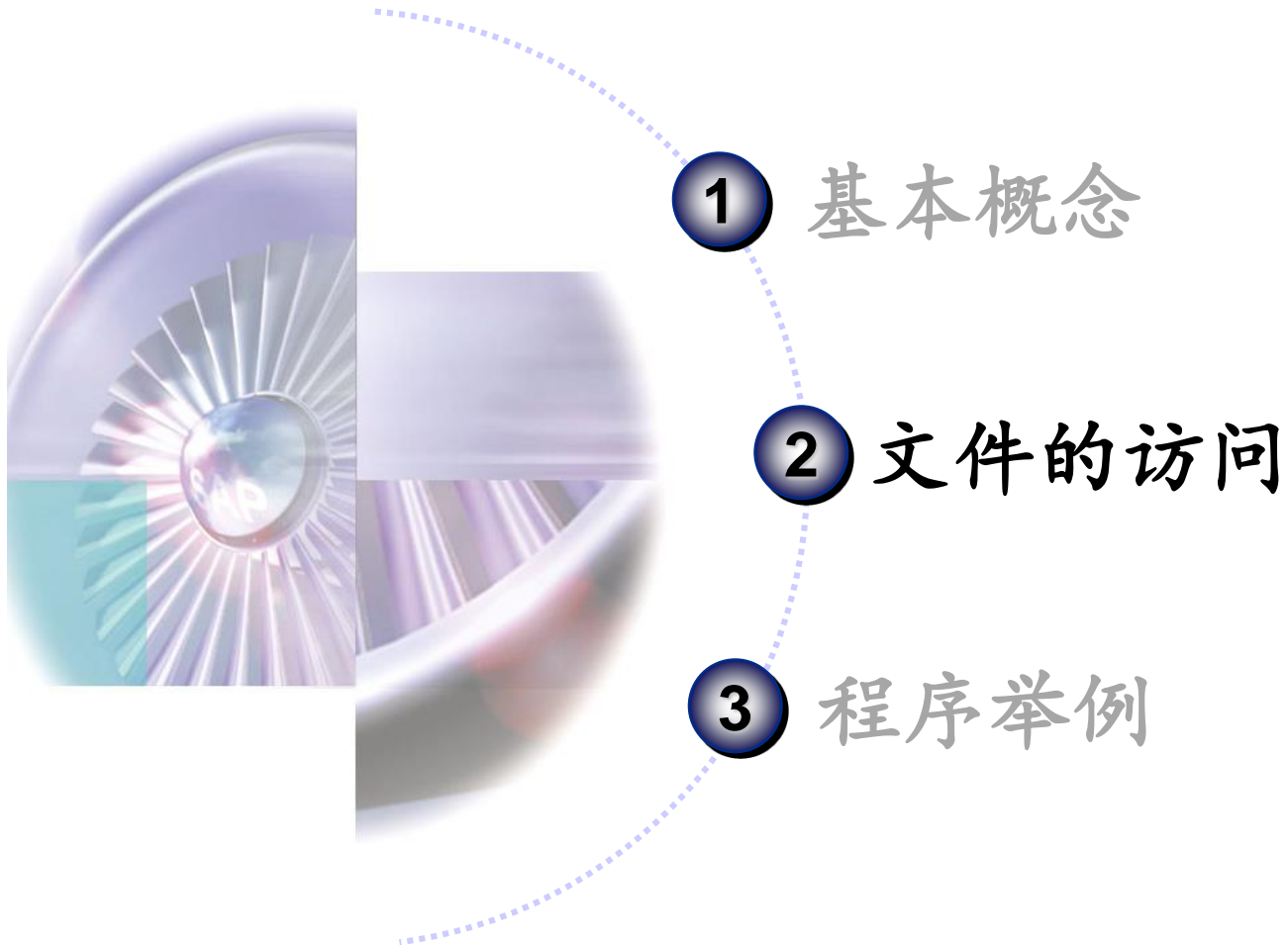
**FILE \*fp;**



同时使用多个文件时，每个文件都有缓冲区，用不同的文件指针分别指示。

# Lecture 9: 文件

---



# 读取学生成绩文件

---

## 问题描述：

已知一个数据文件 **f.txt** 中保存了5个学生的计算机等级考试成绩，包括学号、姓名和分数，文件内容如下，请将文件的内容读出并显示到屏幕中。

|               |            |           |
|---------------|------------|-----------|
| <b>301101</b> | <b>张文</b>  | <b>91</b> |
| <b>301102</b> | <b>陈慧</b>  | <b>85</b> |
| <b>301103</b> | <b>王卫东</b> | <b>76</b> |
| <b>301104</b> | <b>郑伟</b>  | <b>69</b> |
| <b>301105</b> | <b>郭温涛</b> | <b>55</b> |

```

#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    FILE * fp;
    int num, score;
    char stname[20];
    if((fp = fopen("f.txt", "r")) == NULL)
    { //以只读方式打开文件f.txt
        printf("File open error!\n"); exit(0);
    }
    while(!feof(fp)) //检查文件是否结束:是,返回非0;否,返回0
    {
        fscanf(fp, "%d%s%d", &num, stname, &score);
        printf("%d\t%s\t%d\n", num, stname, score);
    };
    fclose(fp);
    return 0;
}

```

|        |     |    |
|--------|-----|----|
| 301101 | 张文  | 91 |
| 301102 | 陈慧  | 85 |
| 301103 | 王卫东 | 76 |
| 301104 | 郑伟  | 69 |
| 301105 | 郭温涛 | 55 |



| 文件(F)  | 编辑(E) | 格式(O) | 查看(V) | 帮助(H) |
|--------|-------|-------|-------|-------|
| 301101 | 张文    | 91    |       |       |
| 301102 | 陈慧    | 85    |       |       |
| 301103 | 王卫东   | 76    |       |       |
| 301104 | 郑伟    | 69    |       |       |
| 301105 | 郭温涛   | 55    |       |       |

# 打开文件

---

- 格式: **fp = fopen("文件名", "文件打开方式")**
  - 使文件指针与相应文件实体对应起来
  - 程序对文件指针进行操作, 即fp代表磁盘文件
- 说明: **fopen()** 的返回值
  - 执行成功, 则返回包含文件缓冲区等信息的FILE型地址, 赋给文件指针fp
  - 不成功, 则返回一个**NULL** (空值)
    - exit(0): 关闭所有打开的文件, 并终止程序的执行
    - 参数0表示程序正常结束; 非0参数表示不正常程序结束

# 文件打开方式

```
fp = fopen('f.txt', 'r')
```

- 文件打开方式参数表

| 文 本 文 件 (ASCII) |          | 二 进 制 文 件(Binary) |          |
|-----------------|----------|-------------------|----------|
| 使用方式            | 含 义      | 使用方式              | 含 义      |
| “ r ”           | 打开只读文件   | “ rb ”            | 打开只读文件   |
| “ w ”           | 建立只写新文件  | “ wb ”            | 建立只写新文件  |
| “ a ”           | 打开添加写文件  | “ ab ”            | 打开添加写文件  |
| “ r+ ”          | 打开读/写文件  | “ rb+ ”           | 打开读/写文件  |
| “ w +”          | 建立读/写新文件 | “ wb+ ”           | 建立读/写新文件 |
| “ a +”          | 打开读/写文件  | “ ab+ ”           | 打开读/写文件  |

# 文件读写与打开方式

if (读文件)

指定的文件必须存在，否则出错；

if (写文件) // 指定的文件可以存在，也可以不存在

if (以 "w" 方式写)

if (该文件已经存在)

原文件将被删去重新建立；

else

按指定的名字新建一个文件；

else if (以 "a" 方式写)

if (该文件已经存在)

写入的数据将被添加到指定文件原有数据的后面，不会删去原来的内容；

else

按指定的名字新建一个文件（与“w”相同）；

if (文件同时读和写)

使用 "r+"、"w+" 或 "a+" 打开文件



# 关闭文件

---

- 格式: **fclose(文件指针)**
  - 把缓冲区中的数据写入磁盘扇区, 确保写文件的正常完成
  - 释放文件缓冲区单元和FILE结构体, 使文件指针与具体文件脱钩。
- 说明: **fclose()** 的返回值
  - 返回0: 正常关闭文件
  - 返回非0: 无法正常关闭文件

# 打开多个文件

---

- C语言允许同时打开多个文件
  - 不同的文件对应不同的文件指针
  - 不允许同一个文件在关闭前再次打开

# 文件读写函数

---

- 字符读写函数: `fgetc()`, `fputc()`
- 字符串读写函数: `fgets()`, `fputs()`
- 格式化读写函数: `fscanf()`, `fprintf()`
- 二进制读写函数: `fread()`, `fwrite()`
- 其他相关函数:
  - 检测文件结尾函数 `feof()`
  - 检测文件读写出错函数 `ferror()`
  - 清除末尾标志和出错标志函数 `clearerr()`
  - 文件定位的函数 `fseek()`

# 数据块读写fread()和fwrite()

---

- **fread(buffer, size, count, fp);**

从二进制文件中读入一个数据块到变量

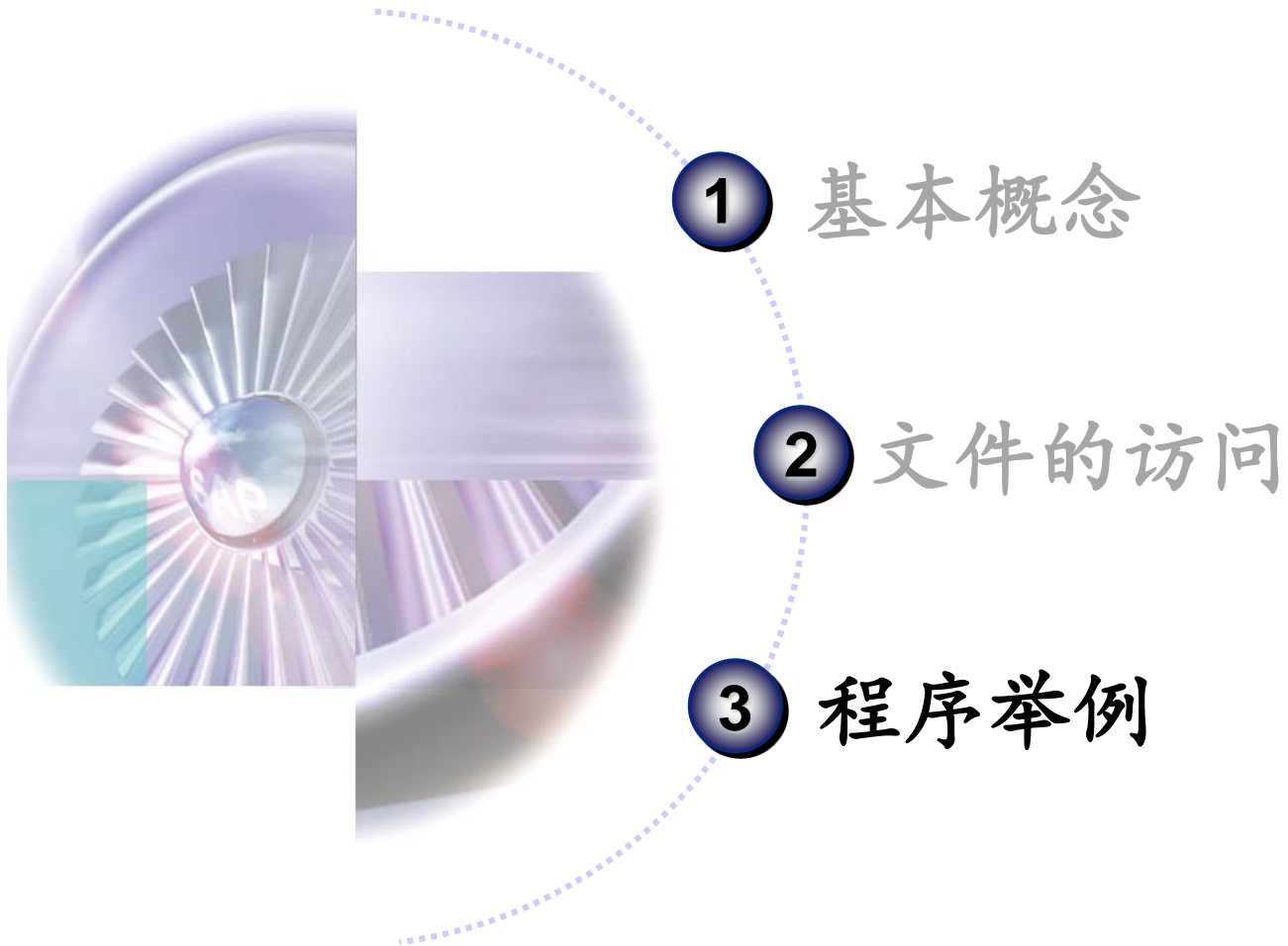
- **fwrite(buffer, size, count, fp);**

向二进制文件中写入一个数据块

- **buffer:** 指针，表示存放数据的首地址;
- **size:** 数据块的字节数
- **count:** 要读写的数据块块数
- **fp:** 文件指针

# Lecture 9: 文件

---



# 二进制文件读写



## 问题描述:

利用**结构体**，**读取文本**数据文件 1.txt 中保存的学生信息，每个学生的信息包括：学号、姓名和分数。

并把该文件**保存**为 2.a 的**二进制**文件，再从 2.a 中**读取**数据并**显示到**屏幕上。

---

## 问题分析：

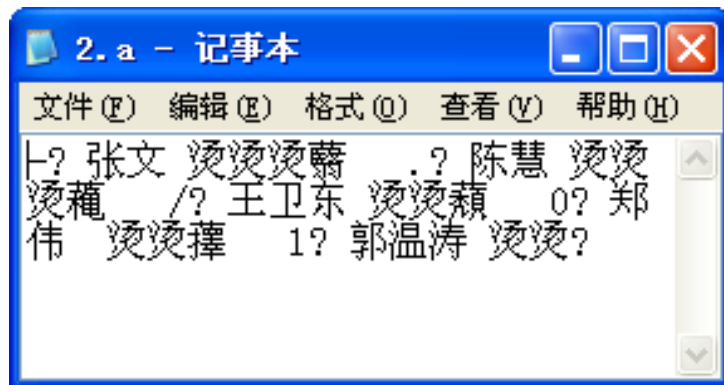
- 1、如何存储/读取为二进制文件？
- 2、如何判断文件读取结束？
- 3、`fEOF()` 是否有问题？

```
#include <stdio.h>
#include <stdlib.h>
typedef struct student
{
    int ID; char name[10]; int score;
} STUDENT;
int main()
{
    FILE *fp1, *fp2; STUDENT s; //定义两个文件指针+一个结构体变量
    if((fp1 = fopen("1.txt", "r")) == NULL)        exit(0);
    if((fp2 = fopen("2.a", "wb")) == NULL)          exit(0);
    while(!feof(fp1)) {
        fscanf(fp1, "%d %s %d", &s.ID, s.name, &s.score);
        printf("%d\t%s\t%d\n", s.ID, s.name, s.score);
        fwrite(&s, sizeof(STUDENT), 1, fp2); //写数据块
    };
    fclose(fp1); fclose(fp2); //先关闭文件fp2, 再打开fp2
    if((fp2 = fopen("2.a", "rb")) == NULL)          exit(0);
    while(!feof(fp2)) {
        fread(&s, sizeof(STUDENT), 1, fp2); //读二进制数据块
        printf("%d\t%s\t%d\n", s.ID, s.name, s.score);
    };
    fclose(fp2);
}
```



|        |     |    |
|--------|-----|----|
| 301101 | 张文  | 91 |
| 301102 | 陈慧  | 85 |
| 301103 | 王卫东 | 76 |
| 301104 | 郑伟  | 69 |
| 301105 | 郭温涛 | 55 |

|        |     |    |
|--------|-----|----|
| 301101 | 张文  | 91 |
| 301102 | 陈慧  | 85 |
| 301103 | 王卫东 | 76 |
| 301104 | 郑伟  | 69 |
| 301105 | 郭温涛 | 55 |
| 301105 | 郭温涛 | 55 |



二进制文件

什么问题导致多读了一遍？

```
while(!feof(fp2))  
{  
}
```

- 
- 问题：用`feof()`做判断，导致在读文件时，`fread()`多读取了一次
    - `feof(fp)`用来测试`fp`所指向的文件当前状态是否为“文件结束”。如果文件结束，则返回1，否则返回0。适合于二进制文件和文本文件。
  - C FAQ-12.3的解释是“在C语言中，只有输入例程试图读并失败以后才能得到文件结束符。... `fgets()`在遇到文件结束符的时候返回`NULL`。实际上，在任何情况下，都完全没必要使用`feof()`。”

```

#include <stdio.h>
#include <stdlib.h>
typedef struct student
{
    int ID; char name[10]; int score;
} STUDENT;
int main()
{
    FILE *fp1, *fp2; STUDENT s;
    if((fp1 = fopen("1.txt", "r")) == NULL)        exit(0);
    if((fp2 = fopen("2.a", "wb")) == NULL)          exit(0);
    while(!feof(fp1)) {
        fscanf(fp1, "%d %s %d", &s.ID, s.name, &s.score);
        printf("%d\t%s\t%d\n", s.ID, s.name, s.score);
        fwrite(&s, sizeof(STUDENT), 1, fp2);
    };
    fclose(fp1); fclose(fp2);
    if((fp2 = fopen("2.a", "rb")) == NULL)          exit(0);
    while(fread(&s, sizeof(STUDENT), 1, fp2) != NULL) {
        printf("%d\t%s\t%d\n", s.ID, s.name, s.score);
    };
    fclose(fp2);
}

```

|        |     |    |
|--------|-----|----|
| 301101 | 张文  | 91 |
| 301102 | 陈慧  | 85 |
| 301103 | 王卫东 | 76 |
| 301104 | 郑伟  | 69 |
| 301105 | 郭温涛 | 55 |

## 思考题:台阶问题

---

### 问题描述:

楼梯有 $N$  ( $N < 25$ ) 级台阶, 上楼时一步可以走一级台阶, 也可以走二级或三级台阶。请编写一个递归程序, 计算共有多少种不同的走法?

---

思路：

1、**简化**问题，探索**规律**。先从**个别**再到**一般**，要善于对多个因素作分解，化难为易。

2. 定义函数

**f(int n)** —— 楼梯台阶数

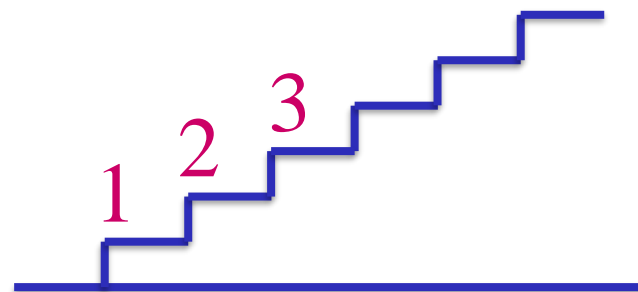
怎样编写这种程序？思路还是先从最简单的情况分析起，走一走看，慢慢理出思路。

1、只有一级台阶时

$$f(1) = 1;$$

2、只有二级台阶时

$$f(2) = 2;$$



3、只有三级台阶时

$$f(3) = 4; \quad // \text{ 4种走法: } 1-1-1, 1-2, 2-1, 3$$

4、有  $n$  级台阶时 ( $n > 3$ )

$$f(n) = ?$$

---

## 有 $n$ 级台阶时 ( $n > 3$ )

- ① 最后一步走一个台阶，前  $n-1$  个台阶有  $f(n-1)$  种走法。
- ② 最后一步走两个台阶，前  $n-2$  个台阶有  $f(n-2)$  种走法。
- ③ 最后一步走三个台阶，前  $n-3$  个台阶有  $f(n-3)$  种走法。
- ④ 所以共有  $f(n) = f(n-1) + f(n-2) + f(n-3)$  种走法。

```

#include <stdio.h>
#include <stdlib.h>
int Step(int n);
int main()
{
    int n, res;
    scanf("%d", &n);
    res = Step(n);
    printf("%d", res);
}
int Step(int n)
{
    int res;
    if (n == 1)        return 1;
    else if(n == 2) return 2;
    else if(n == 3) return 4;
    else if(n > 3)
        return (Step(n-1) + Step(n-2) + Step(n-3));
    return 0;
}

```

楼梯有 1级台阶, 有1多种走法  
 楼梯有 2级台阶, 有2多种走法  
 楼梯有 3级台阶, 有4多种走法  
 楼梯有 4级台阶, 有7多种走法  
 楼梯有 5级台阶, 有13多种走法  
 楼梯有 6级台阶, 有24多种走法  
 楼梯有 7级台阶, 有44多种走法  
 楼梯有 8级台阶, 有81多种走法  
 楼梯有 9级台阶, 有149多种走法  
 楼梯有10级台阶, 有274多种走法  
 .....



## 此类问题求解的基本方法

---

- 首先，确认：能否容易的得到简单情况的解？
- 然后，假设：规模为 $N-1$ 的情况已经得到解决。
- 最后，重点分析：当规模扩大到 $N$ 时，如何枚举出所有的情况，并且要确保对于每一种子情况都能用已经得到的数据解决。

# 递推的经典问题

---

斐波那契级数：



一对新生小兔，一个月后长成中兔，从第三个月开始长成大兔并每个月生一对小兔。按此规律，一年后共有多少对兔子？

---

---

| 第n个月       | 大兔对数     | 中兔对数     | 小兔对数     | 总对数       |
|------------|----------|----------|----------|-----------|
| <b>1</b>   | <b>0</b> | <b>0</b> | <b>1</b> | <b>1</b>  |
| <b>2</b>   | <b>0</b> | <b>1</b> | <b>0</b> | <b>1</b>  |
| <b>3</b>   | <b>1</b> | <b>0</b> | <b>1</b> | <b>2</b>  |
| <b>4</b>   | <b>1</b> | <b>1</b> | <b>1</b> | <b>3</b>  |
| <b>5</b>   | <b>2</b> | <b>1</b> | <b>2</b> | <b>5</b>  |
| <b>6</b>   | <b>3</b> | <b>2</b> | <b>3</b> | <b>8</b>  |
| <b>7</b>   | <b>5</b> | <b>3</b> | <b>5</b> | <b>13</b> |
| <b>...</b> |          |          |          |           |

# Fibonacci 数列

---

$$f_n = \begin{cases} 1 & n = 1 \\ 1 & n = 2 \\ f_{n-1} + f_{n-2} & n \geq 3 \end{cases}$$

即：1、1、2、3、5、8、13、21、34 ...

关键：确定问题的递归特性

关键：分析出递归公式

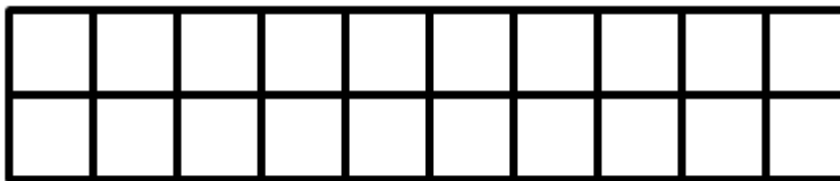
关键：分析出递归边界

## 思考题：铺地砖问题

讨论

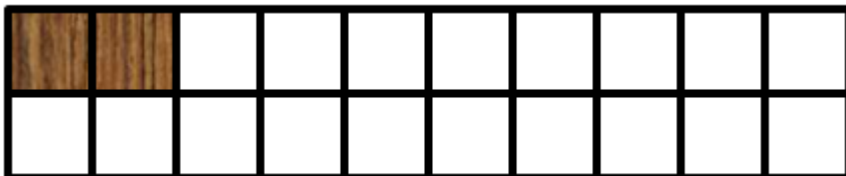
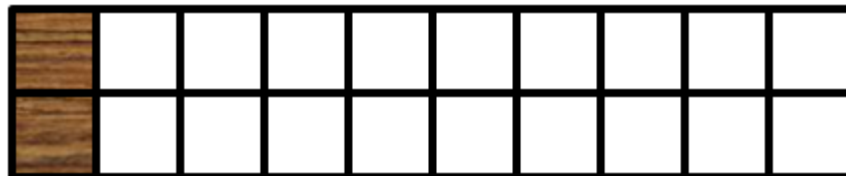
### 问题描述：

有一个长度为 $n$ ，宽度为2的地面，有若干块长为2，宽为1的地砖，请问用此地砖铺完这个地面共有多少种方法？



## 不同的题目，相似的建模方法！

1. 假设铺完长度为 $n$ 的地面有 $f(n)$ 种方法；
2. 如果第一块地砖**竖**起来铺，还剩下长度为 $n-1$ 的地面，有 $f(n-1)$ 种方法；
3. 如果第一块地转**横**着铺，那么还剩下长度为 $n-2$ 的地面，有 $f(n-2)$ 种铺法；
4. 因此  $f(n) = f(n-1) + f(n-2)$ ;



递归边界:

$$f(1) = 1;$$

$$f(2) = 2;$$