

本节内容

逻辑结构  
Vs  
物理结构

王道考研/CSKAOYAN.COM

1

傻傻分不清楚？

文件的逻辑结构

- 无结构文件
- 有结构文件
  - 顺序文件
    - 顺序存储
    - 链式存储
  - 索引文件
  - 索引顺序文件

文件的物理结构

- 连续分配
- 链接分配
- 索引分配

????

小朋友你是否有很多问号

王道考研/CSKAOYAN.COM

2

王道考研/CSKAOYAN.COM

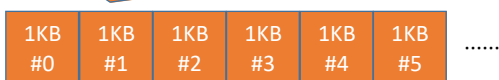
王道考研/CSKAOYAN.COM

2

## 物理结构（从操作系统视角看）

Hel lo wor ld! Hel lo wor ld! Hel lo wor ld! .....

操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！



被操作系统拆分为若干个块，逻辑块号相邻

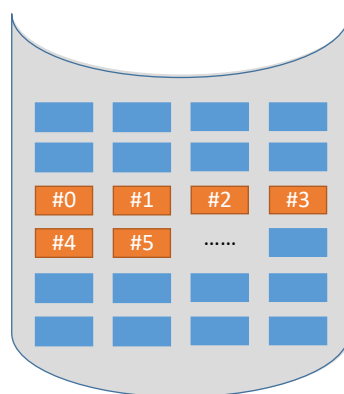
用户：

使用 C 语言库函数 `fseek`，将文件读写指针指向位置 `n`

使用 C 语言库函数 `fgetc`，从读写指针所指位置读出 1B 内容

指明逻辑地址

`fgetc` 底层使用了 `Read` 系统调用，操作系统将（逻辑块号，块内偏移量）转换为（物理块号，块内偏移量）



连续分配：逻辑上相邻的块物理上也相邻

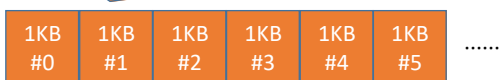
王道考研/CSKAOYAN.COM

5

## 物理结构（从操作系统视角看）

Hel lo wor ld! Hel lo wor ld! Hel lo wor ld! .....

操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！



被操作系统拆分为若干个块，逻辑块号相邻

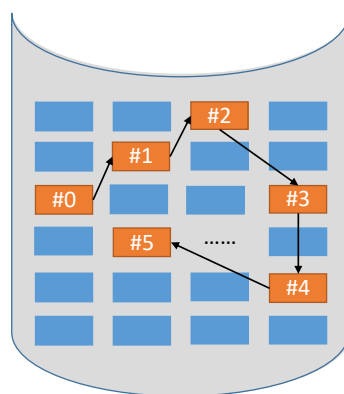
用户：

使用 C 语言库函数 `fseek`，将文件读写指针指向位置 `n`

使用 C 语言库函数 `fgetc`，从读写指针所指位置读出 1B 内容

指明逻辑地址

`fgetc` 底层使用了 `Read` 系统调用，操作系统将（逻辑块号，块内偏移量）转换为（物理块号，块内偏移量）



链接分配：逻辑上相邻的块在物理上用链接指针表示先后关系

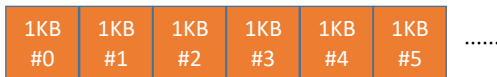
王道考研/CSKAOYAN.COM

6

## 物理结构（从操作系统视角看）

Hel lo wor ld! Hel lo wor ld! Hel lo wor ld! .....

操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！

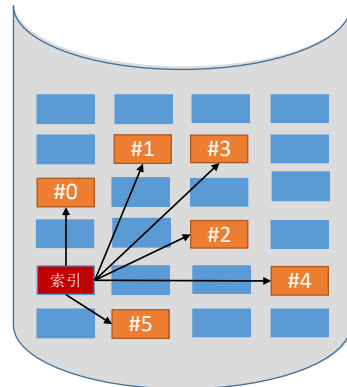


被操作系统拆分为若干个块，逻辑块号相邻

用户：  
使用 C 语言库函数 `fseek`，将文件读写指针指向位置 `n`  
使用 C 语言库函数 `fgetc`，从读写指针所指位置读出 1B 内容

指明逻辑地址

`fgetc` 底层使用了 `Read` 系统调用，操作系统将（逻辑块号，块内偏移量）转换为（物理块号，块内偏移量）



索引分配：操作系统为每个文件维护一张索引表，其中记录了逻辑块号→物理块号的映射关系

王道考研/CSKAOYAN.COM

7

## 例：C语言创建顺序文件

```
typedef struct {
    int number;        //学号
    char name[30];     //姓名
    char major[30];    //专业
} Student_info;

//以"写"方式打开文件
FILE *fp = fopen("students.info", "w");
if(fp == NULL) {
    printf("打开文件失败!");
    exit(0);
}

Student_info student[N]; //用数组保存N个学生信息
for(int i = 0; i < N; i++) { //生成 N 个学生信息
    student[i].number = i;
    student[i].name[0] = '?';
    student[i].major[0] = '?';
}

//将 N 个学生的信息写入文件
fwrite(student, sizeof(Student_info), N, fp);
fclose(fp);
```

用户视角：  
每个学生记录占 64B  
`sizeof(Student_info)`



```
//以"读"方式打开文件
FILE *fp = fopen("students.info", "r");
if(fp == NULL) {
    printf("打开文件失败!");
    exit(0);
}

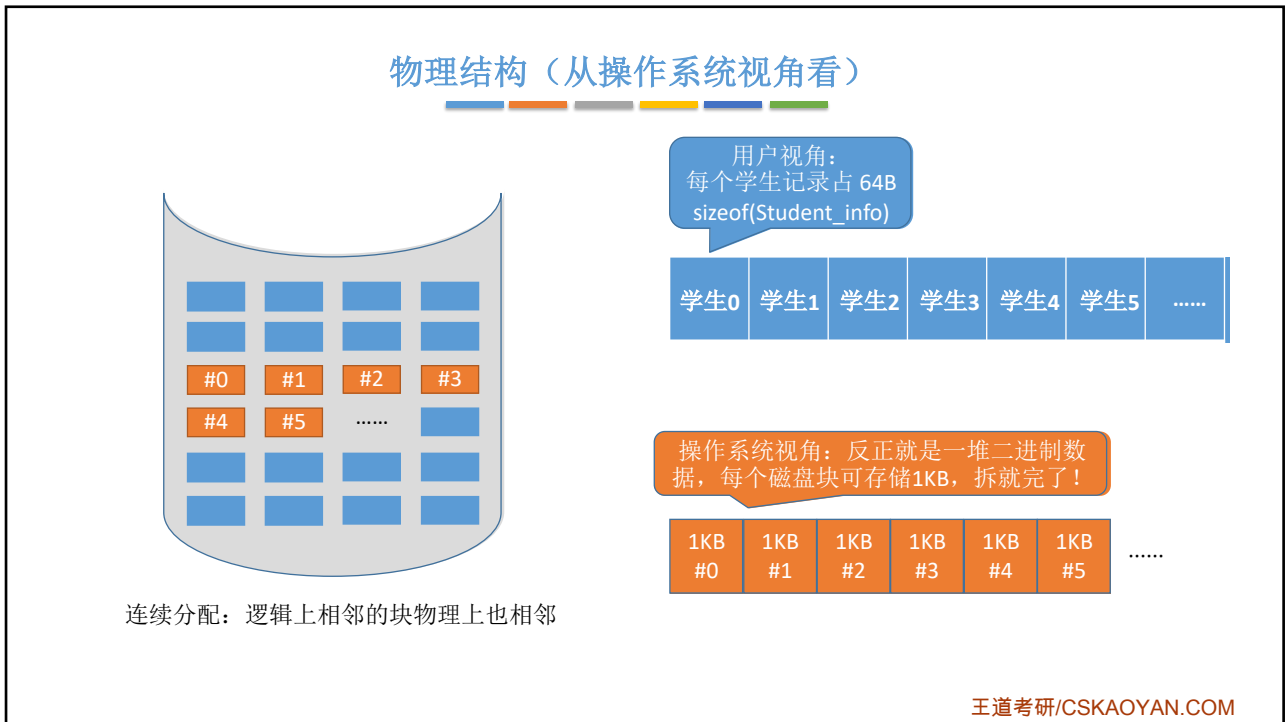
//文件读写指针指向编号为5的学生记录
fseek(fp, 5 * sizeof(Student_info), SEEK_SET);
Student_info stu;

//从文件读出1条记录，记录大小为 sizeof(Student_info)
fread(&stu, sizeof(Student_info), 1, fp);
printf("学生编号: %d\n", stu.number);
fclose(fp);
```

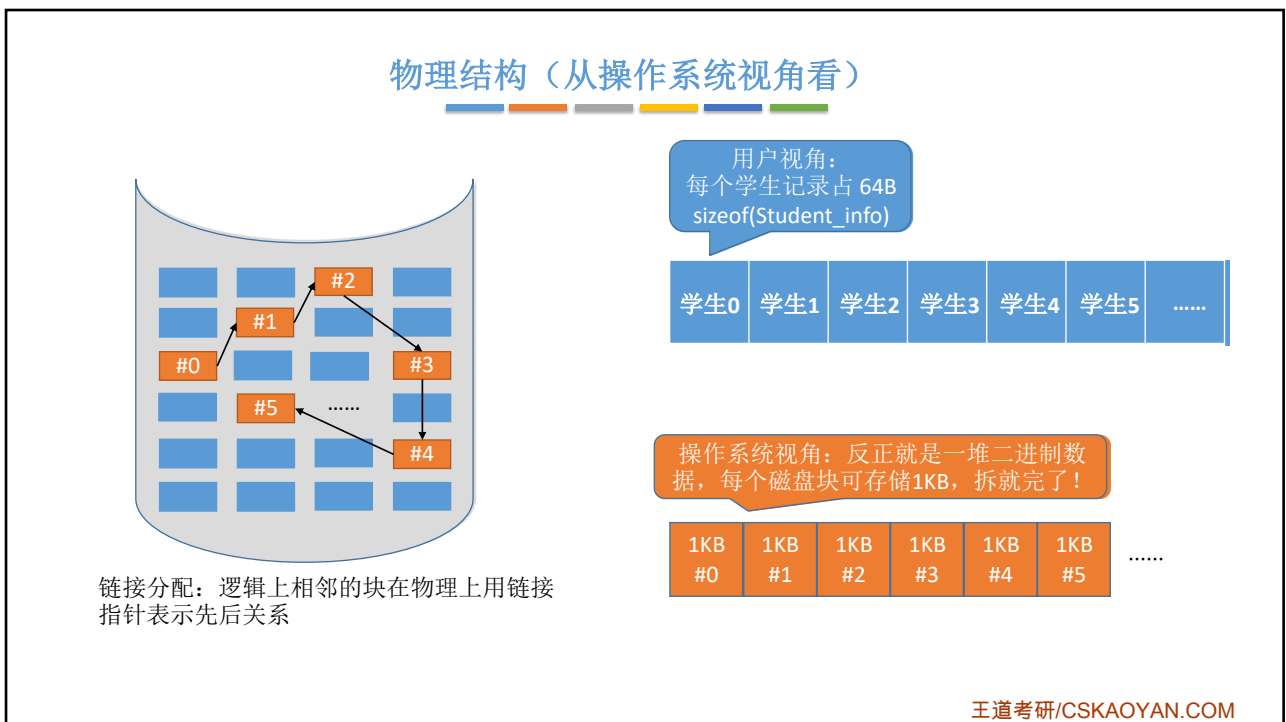
用户用逻辑地址访问文件

王道考研/CSKAOYAN.COM

8

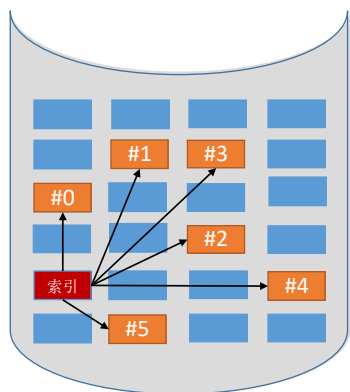


9



10

## 物理结构（从操作系统视角看）



索引分配：操作系统为每个文件维护一张索引表，其中记录了逻辑块号→物理块号的映射关系

用户视角：  
每个学生记录占 64B  
sizeof(Student\_info)

学生0 学生1 学生2 学生3 学生4 学生5 .....

操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！

1KB #0 1KB #1 1KB #2 1KB #3 1KB #4 1KB #5 .....

王道考研/CSKAOYAN.COM

11

## 懵逼点：顺序文件采用顺序存储/链式存储

顺序文件：各个记录可以顺序存储或链式存储。

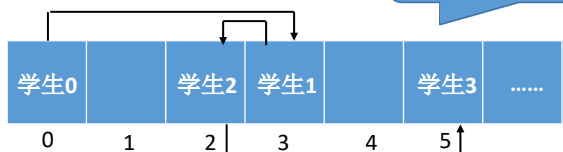
顺序存储，各条记录相邻这存放

学生0 学生1 学生2 学生3 学生4 学生5 .....

支持随机访问：指可以直接确定第*i*条记录的逻辑地址

```
typedef struct {
    int number;           //学号
    char name[30];        //姓名
    char major[30];       //专业
} Student_info;
```

链式存储，各条记录离散着存放，用指针表示先后关系



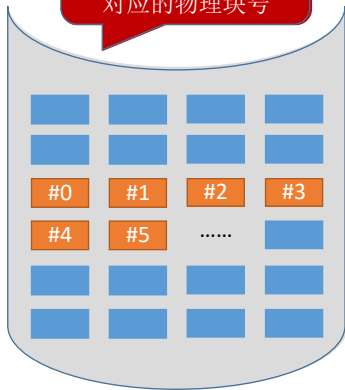
```
typedef struct {
    int number;           //学号
    char name[30];        //姓名
    char major[30];       //专业
    int next;             //下一个学生记录的存放位置
} Student_info;
```

王道考研/CSKAOYAN.COM

12

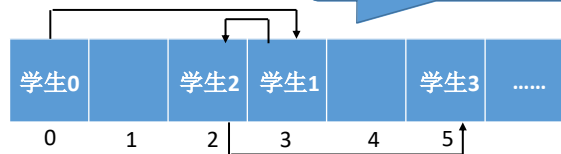
## 链式存储的顺序文件采用连续分配...

支持随机访问：指可以直接找到逻辑块号对应的物理块号

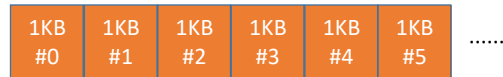


连续分配：逻辑上相邻的块物理上也相邻

链式存储，各条记录离散着存放，用指针表示先后关系



操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！

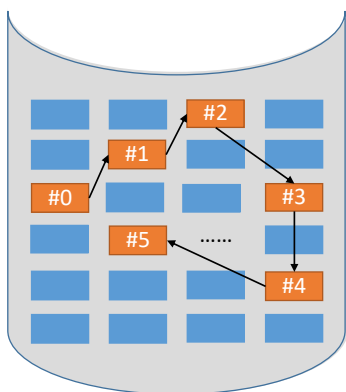


王道考研/CSKAOYAN.COM

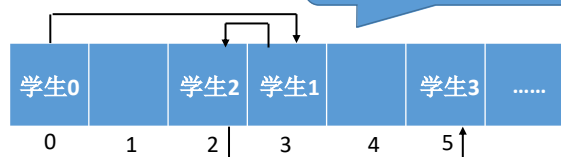
13

## 链式存储的顺序文件采用链接分配...

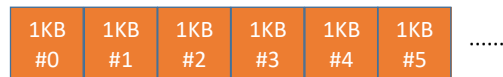
链式存储，各条记录离散着存放，用指针表示先后关系



文件内部各条记录链式存储：由创建文件的用户自己设计的  
文件整体用链接分配：由操作系统决定



操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！



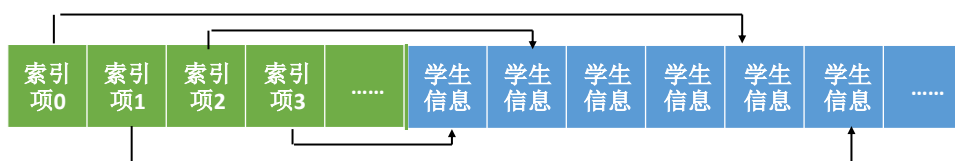
王道考研/CSKAOYAN.COM

14

## 逻辑结构：索引文件

```
typedef struct {
    int number;    //学号
    int addr;      //学生记录的逻辑地址
} IndexTable;
```

```
typedef struct {
    char name[30]; //姓名
    char major[30]; //专业
    //还可添加其他各种各样的学生信息
} Student_info;
```

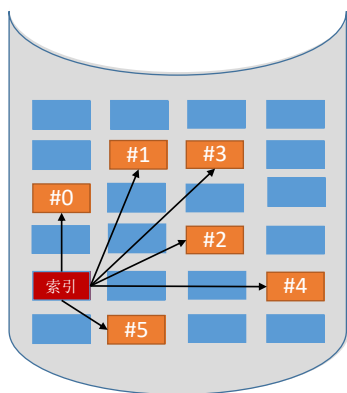
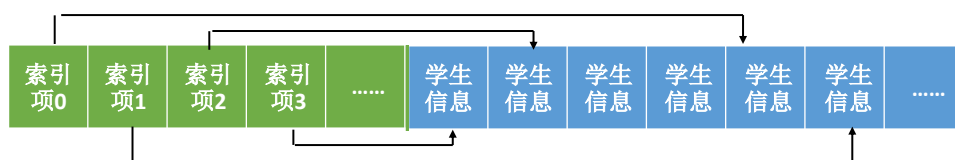


索引文件：从用户视角来看，整个文件依然是连续存放的。如：前1MB存放索引项，后续部分存放记录。

王道考研/CSKAOYAN.COM

15

## 索引文件采用索引分配...



操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！

1KB	1KB	1KB	1KB	1KB	1KB	.....
#0	#1	#2	#3	#4	#5	

索引文件的索引表：用户自己建立的，映射：关键字→记录存放的逻辑地址

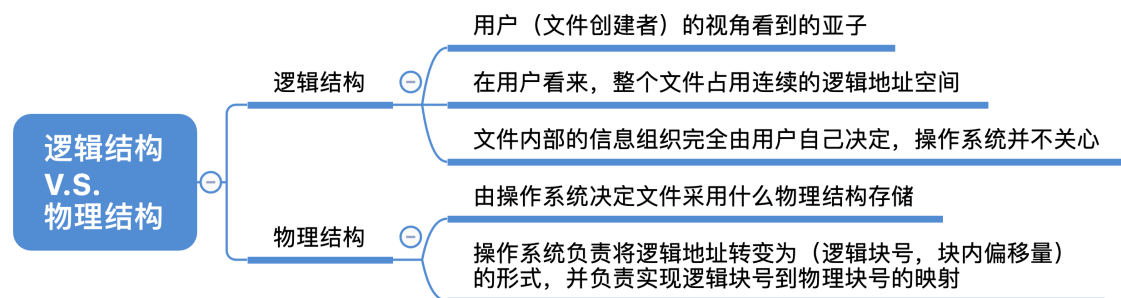
索引分配的索引表：操作系统建立的，映射：逻辑块号→物理块号

王道考研/CSKAOYAN.COM

16



## 慢下来消化一下8



王道考研/CSKAOYAN.COM