



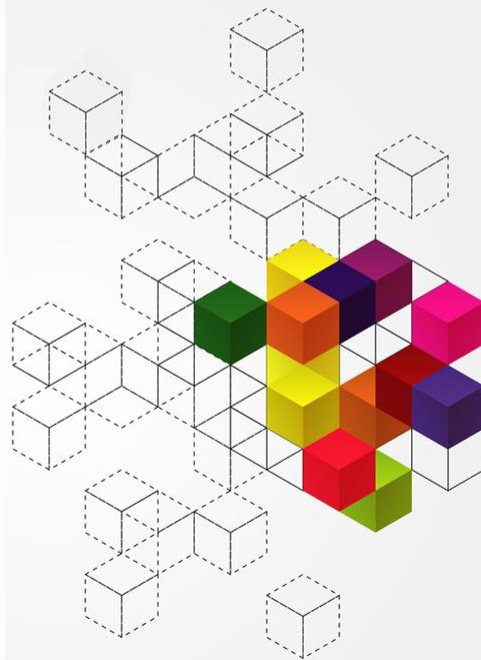
# 操作系统

Operating system

胡燕

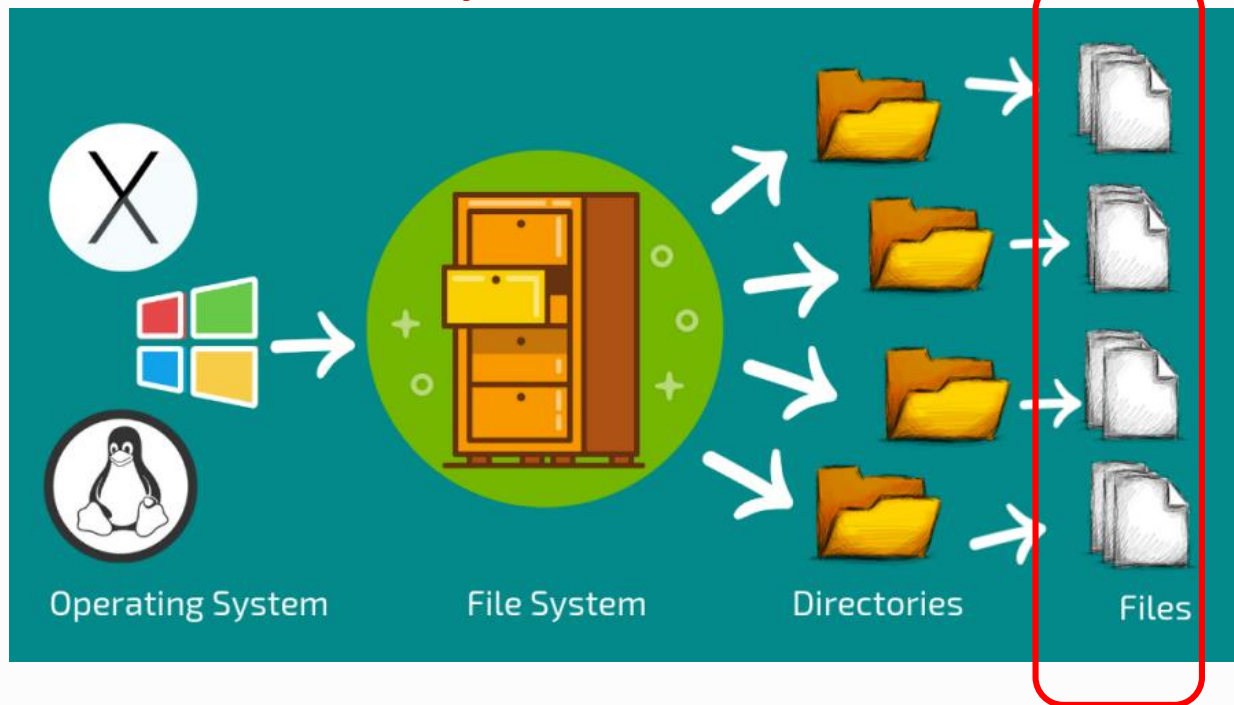
大连理工大学

- 一、 什么是文件
- 二、 文件分类
- 三、 文件逻辑结构
- 四、 文件属性

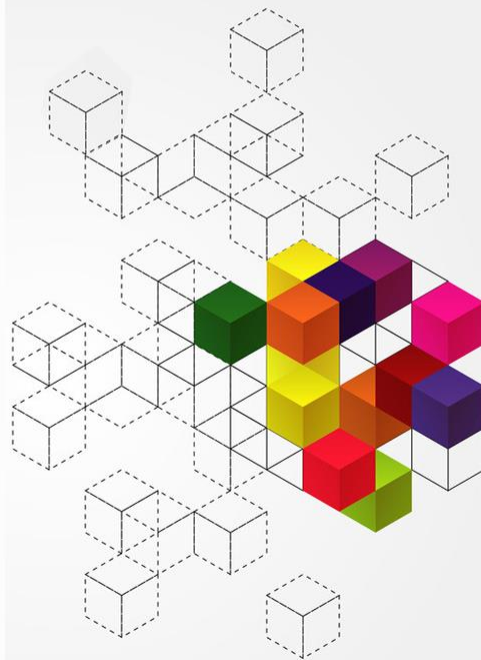


# 一、什么是文件

- 文件系统 (File System)



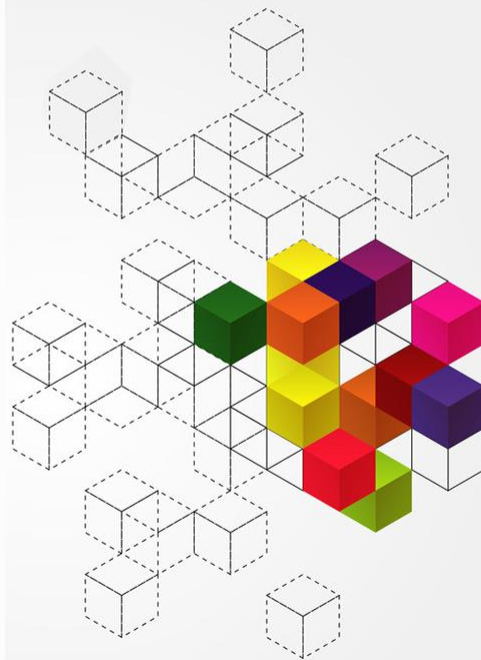
文件是文件系统中的核心要素



# 一、什么是文件

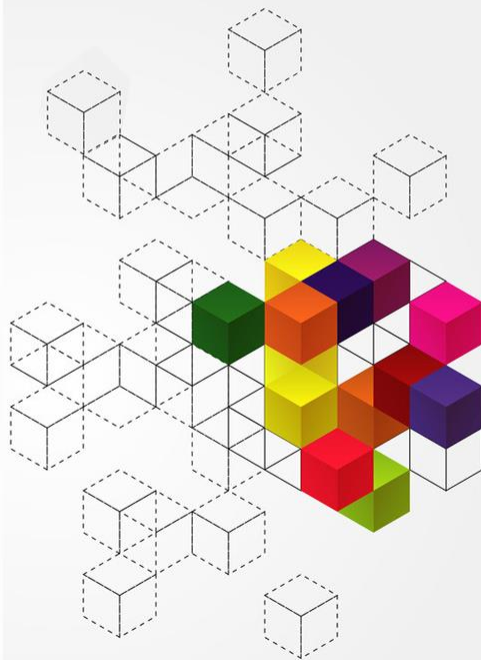


• 文件是什么？



# 一、什么是文件

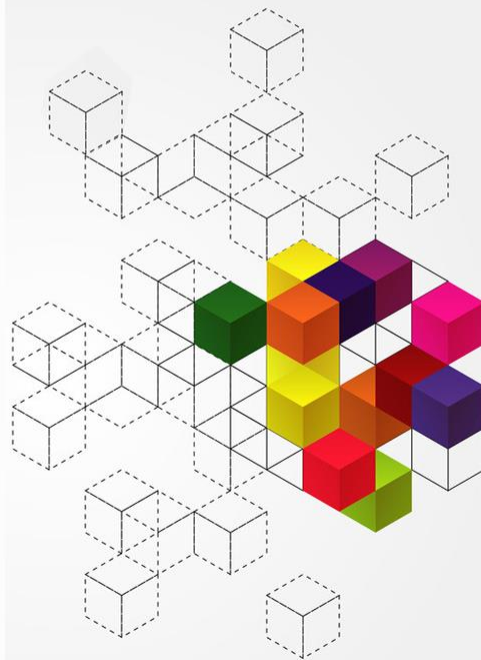
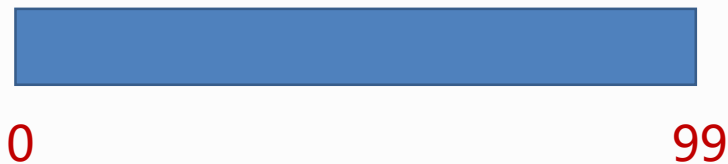
- 文件是操作系统在外部持久性存储设备上存储信息的基本单位
- 每个文件代表一段连续的逻辑数据



# 一、什么是文件

- 文件数据的逻辑地址
- 例如，一个文件中存放了长为100的字符串，其数据逻辑地址范围是0-99

File A



## 二、文件分类

### 按内容性质划分：

- 数据文件

.txt

.pdf

.doc

...

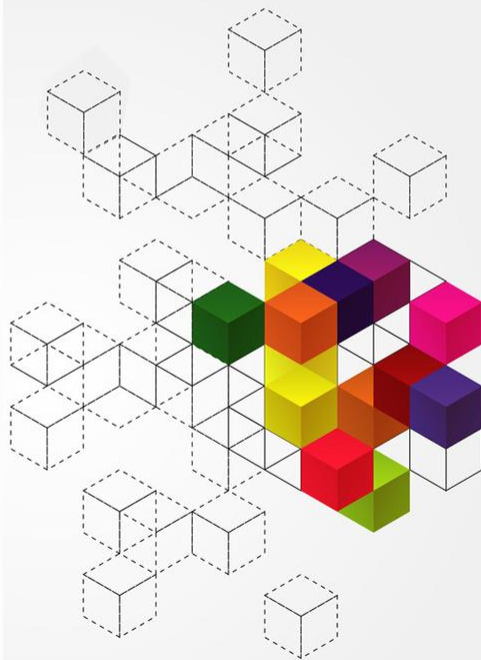
- 代码文件

.c, .cpp, .java, .py, ...

.obj, .o

.lib, .dll, .a, .so

.exe, .elf

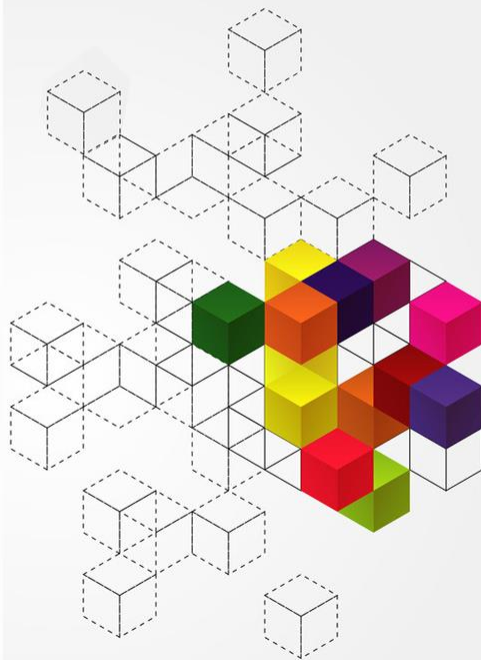




## 二、文件分类

### UNIX文件类型

文件类型	说明
1.普通文件	Ordinary or regular files
2.目录文件	a binary file used to track and locate other files and directories.
3.特殊文件（设备文件）	used for device I/O on UNIX and Linux systems
4.链接	a tool used for having multiple filenames that reference a single file on a physical disk
5.管道文件	tools that allow two or more system processes to communicate with each other
6.套接字	tools used for inter-process communication via network.

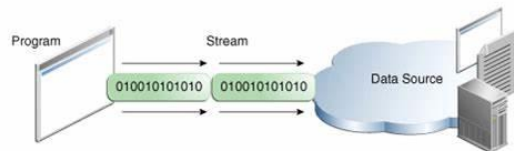




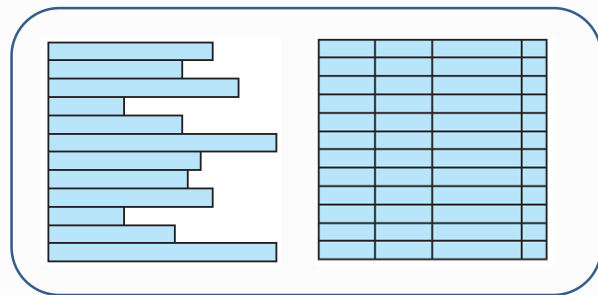
### 三、文件逻辑结构

- 文件的逻辑数据以什么方式组织？

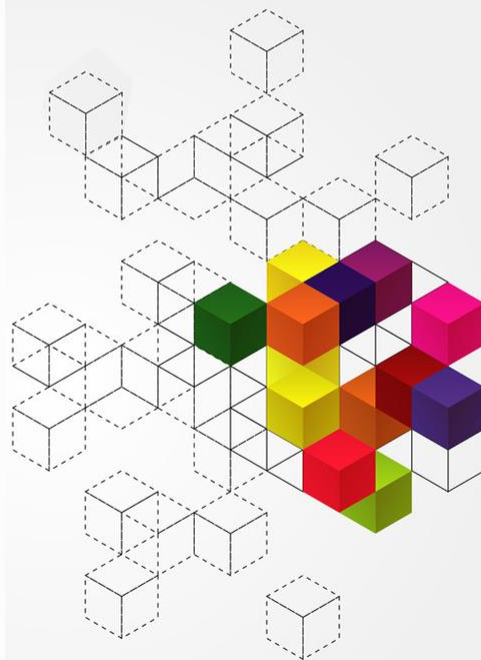
→ **文件逻辑结构**



**流式文件**



**记录式文件**



# 三、文件逻辑结构

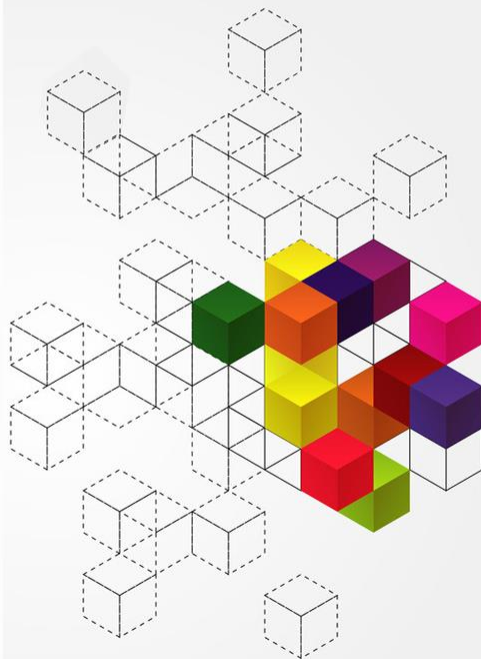
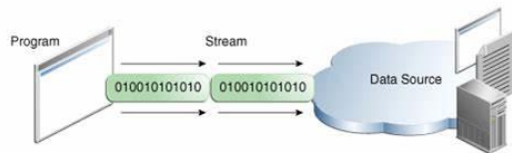
## -流式文件

- 文件0逻辑结构1:流式文件

无结构

查找文件中特定的逻辑内容，必须依靠  
对文件内容的全面搜索

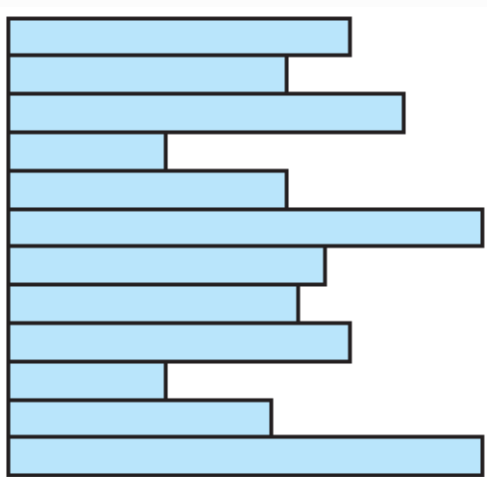
操作系统为了接口的统一性和简洁性，文件数据在系统调用层面通常都被看成字节流



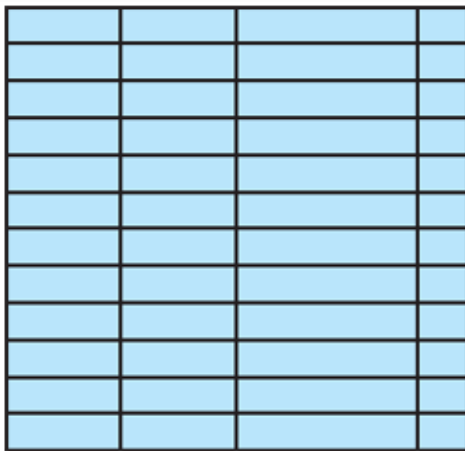
### 三、文件逻辑结构

-有结构文件

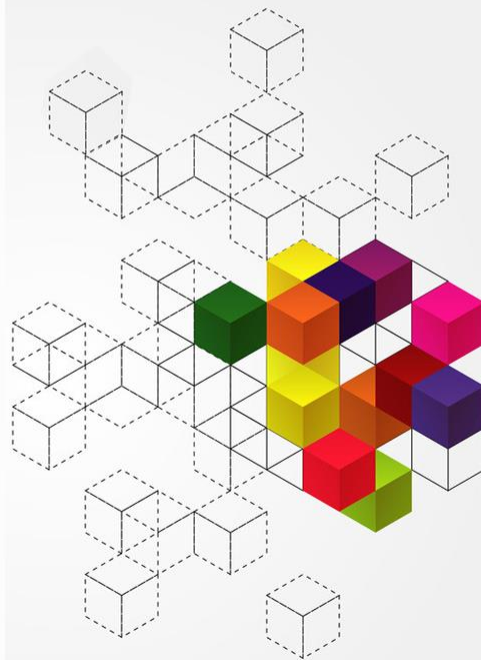
- 文件逻辑结构2:记录式文件



变长记录

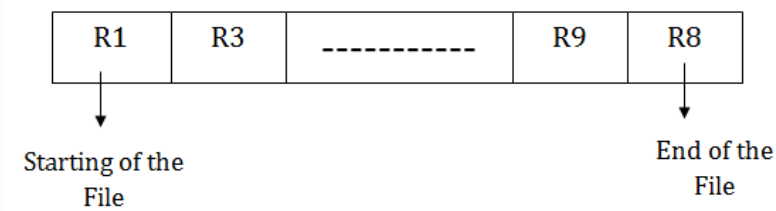


定长记录

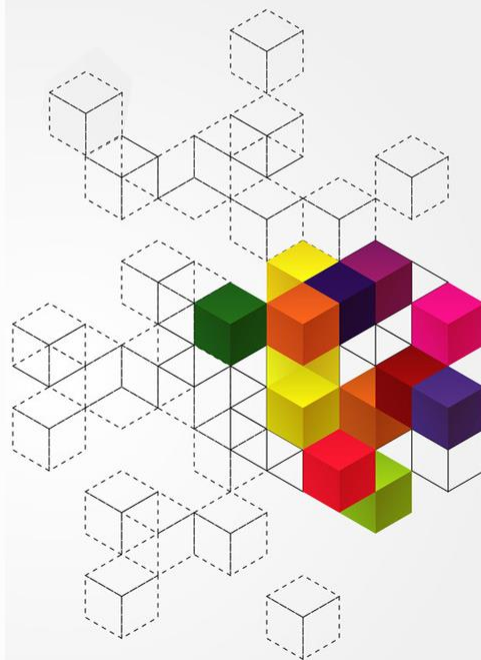
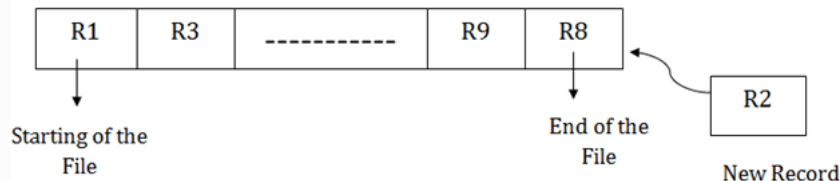


### 三、文件逻辑结构 -有结构文件

- 按照记录的不同组织形式，有结构文件可细分为：  
顺序文件、索引文件、索引顺序文件、哈希文件  
记录在文件中顺序存放



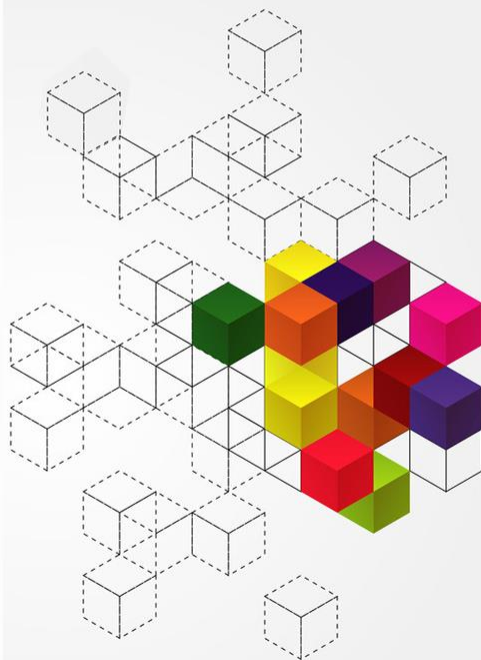
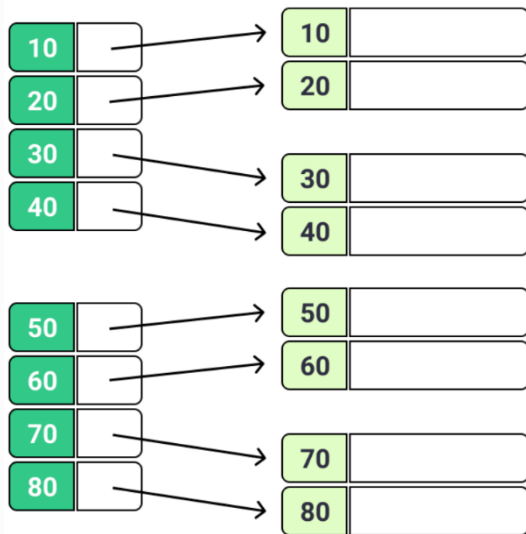
#### 新记录顺序插入



### 三、文件逻辑结构

-有结构文件

- 按照记录的不同组织形式，有结构文件可细分为：  
顺序文件、索引文件、索引顺序文件、哈希文件  
构建一个索引表，以记录的关键字排序



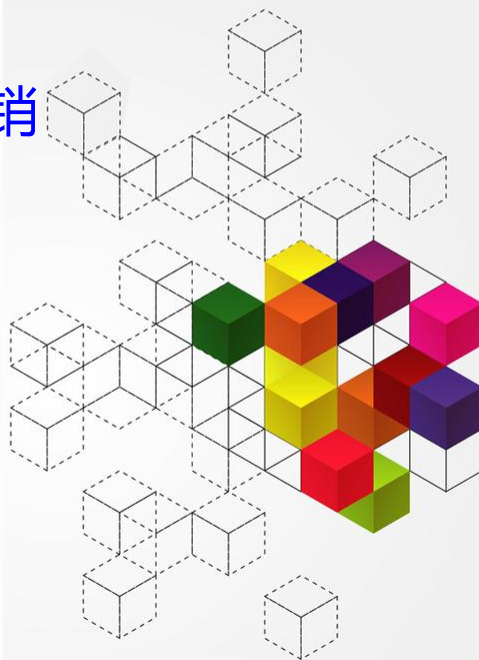
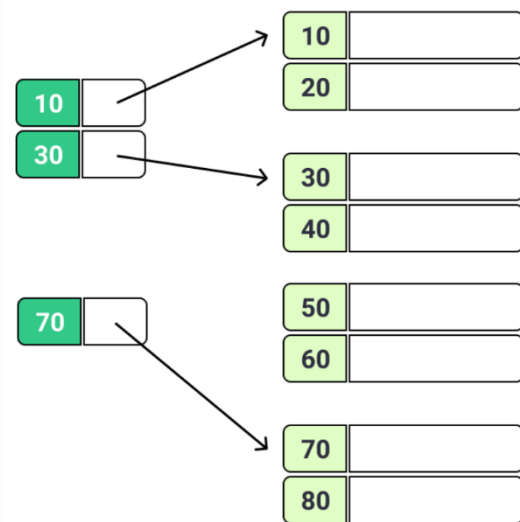
### 三、文件逻辑结构

-有结构文件

- 按照记录的不同组织形式，有结构文件可细分为：

顺序文件、索引文件、索引顺序文件、哈希文件

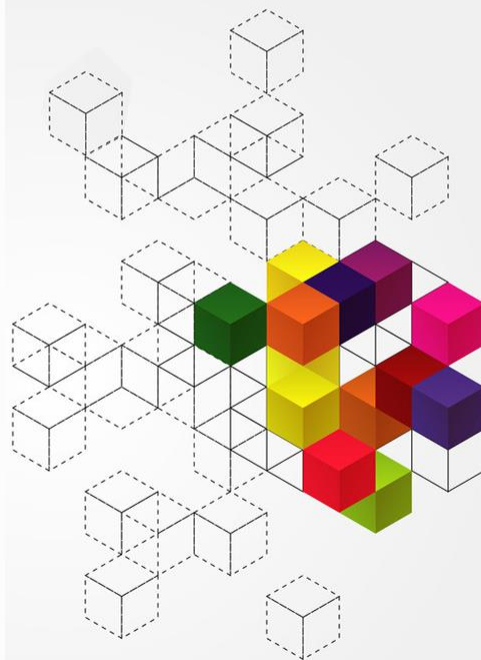
构建一个相对稀疏的索引表，避免过大的索引空间开销



## 四、文件属性

### 文件基本属性表

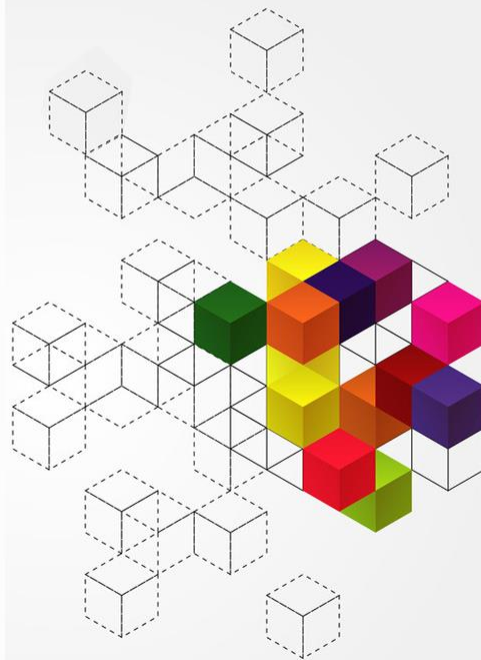
属性名称	说明
文件名	File Name
文件ID	File Identifier
文件类型	File Type
文件大小	File Size
文件保护信息	Protection Information
时间信息	Date of Creation,Last Modifer, etc.
文件所有者	File Owner





# 本讲小结

- 什么是文件
- 文件分类
- 文件逻辑结构
- 文件属性



# 文件逻辑结构练习

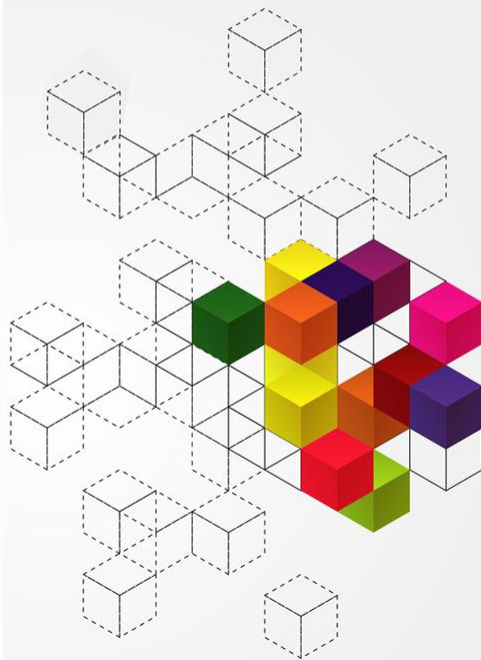
• 文件逻辑结构是（ ）的文件组织形式。

A. 在外部设备上

☒ B. 从用户角度看

C. 虚拟存储

D. 目录



# 文件逻辑结构练习

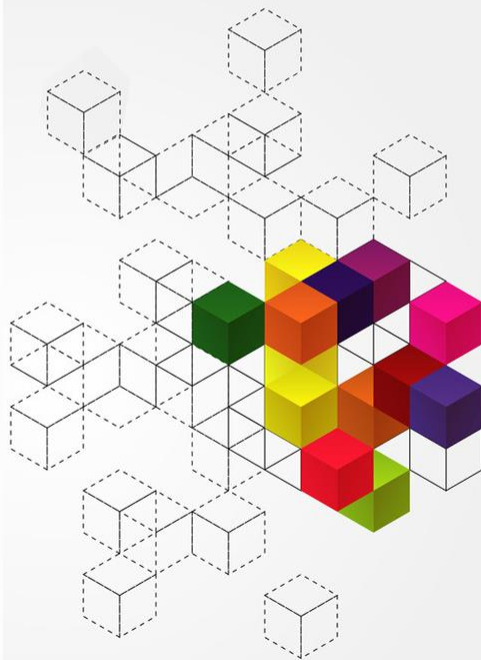
• 数据库文件的逻辑结构是（ ）。

A. 字符流式文件

B. 档案文件

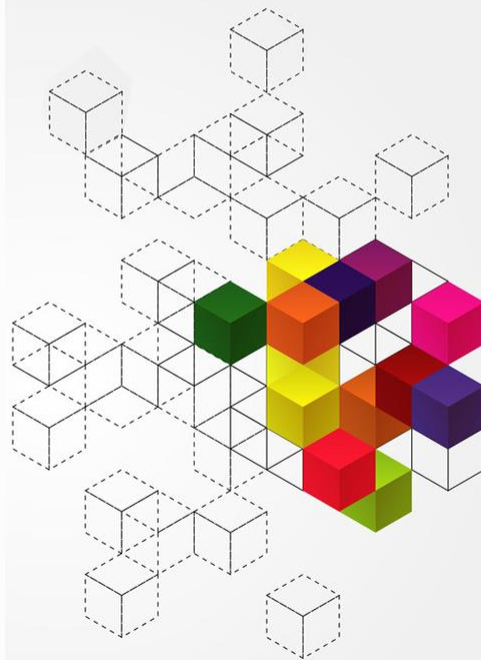
☒ C. 记录式文件

D. 只读文件



### 一、文件操作

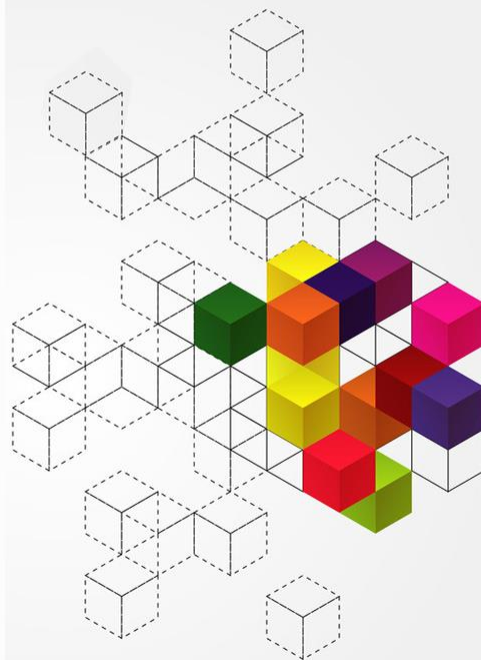
### 二、文件访问模式



# 一、文件操作

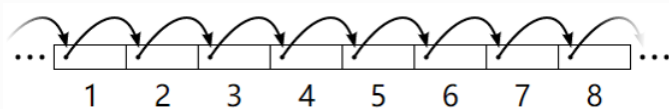
## OS为文件对象提供的常规操作接口

文件操作	说明
Open	打开文件（输入参数：文件名）
Close	关闭文件（输入参数：文件句柄）
Create	创建文件
Read	读文件内容
Write	写入文件
Seek	在文件内重新定位文件指针
Truncate	文件截短

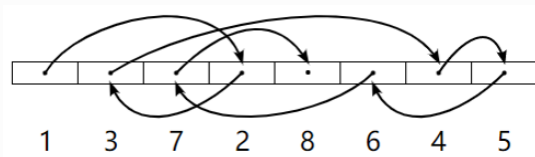


## 二、文件访问模式

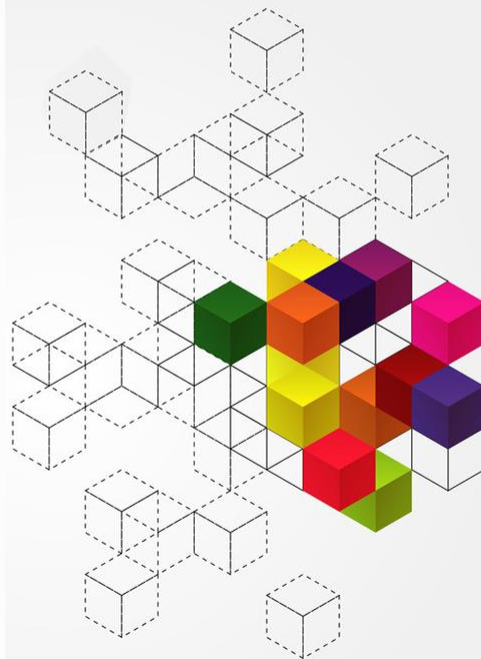
### 1. 顺序访问 Sequential Access



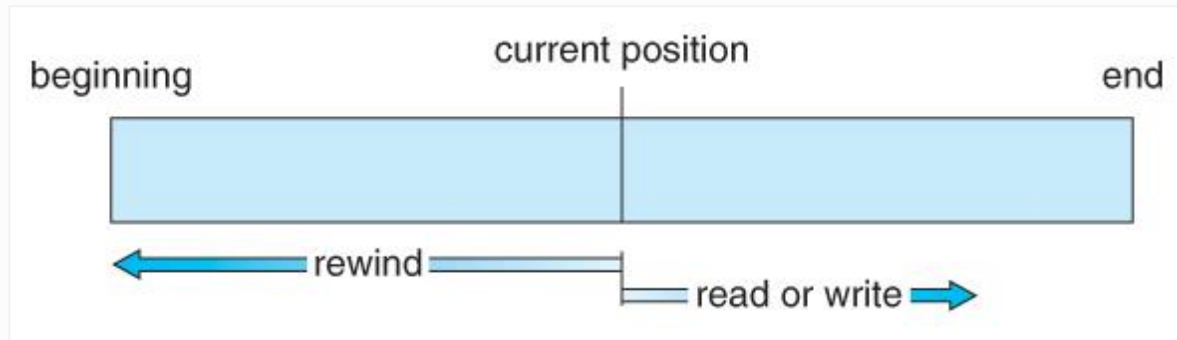
### 2. 直接访问 Direct Access



### 3. 索引访问 Indexed Access



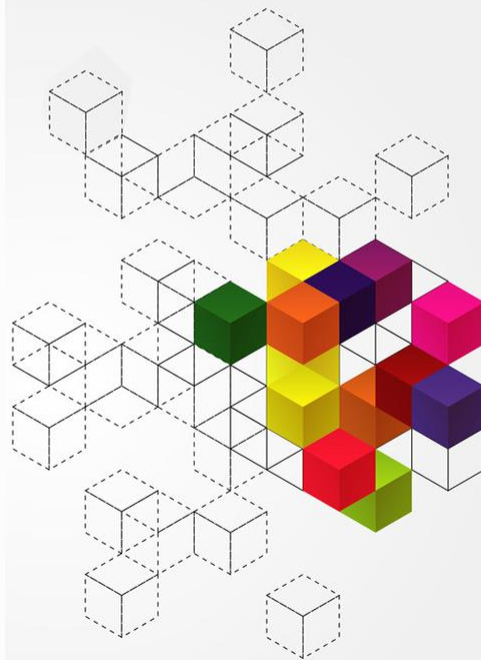
## 二、文件访问模式 - 顺序访问



从当前文件指针所指地址开始，沿逻辑地址增长的方向顺序访问文件内容

支持“倒带”操作 (rewind, reset)

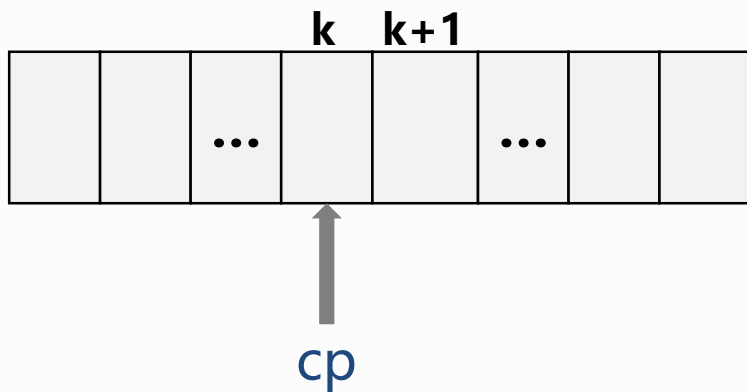
适用的典型存储介质: 磁带 (Tape)



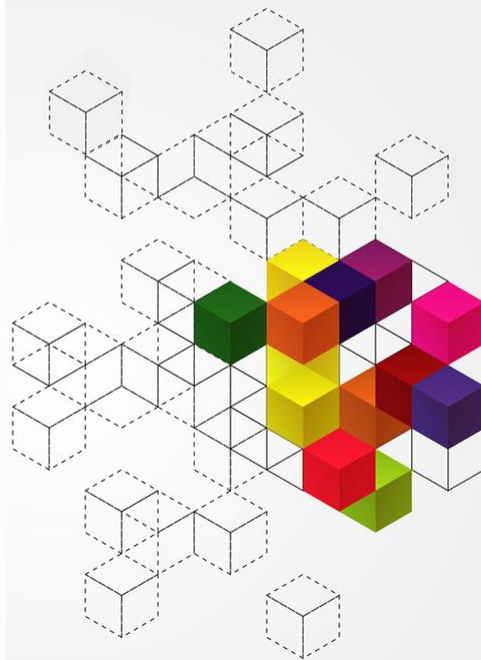


## 二、文件访问模式 - 顺序访问

File A

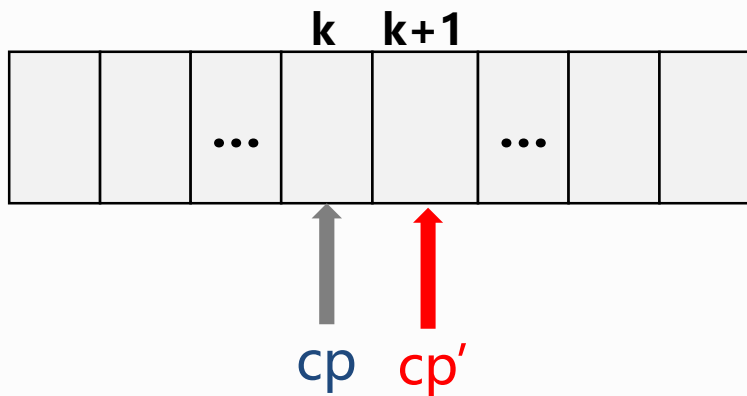


cp (current position)



## 二、文件访问模式 - 顺序访问

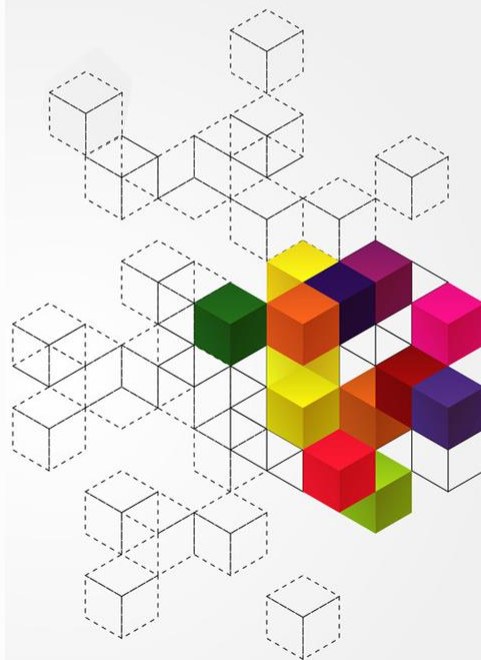
File A



$cp$  (current position)

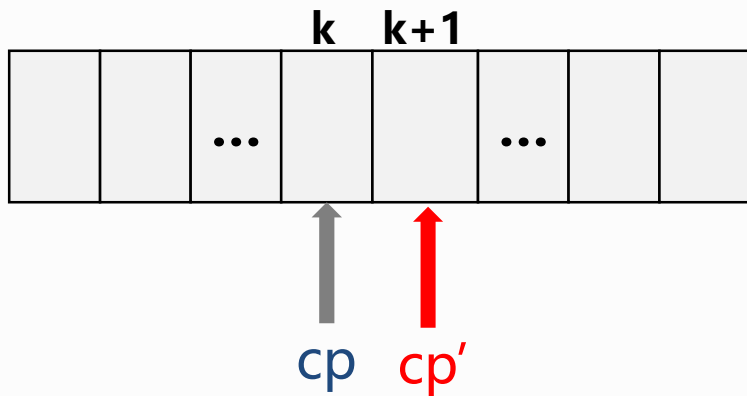
`read()`

读取 $cp$ 指向的元素(地址为 $k$ 的元素);  
 $cp = cp + 1;$



## 二、文件访问模式 - 顺序访问

File A

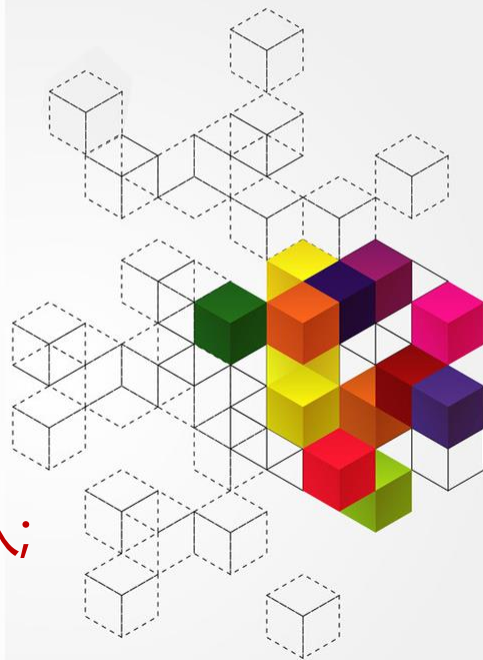


$cp$  (current position)

`read()`

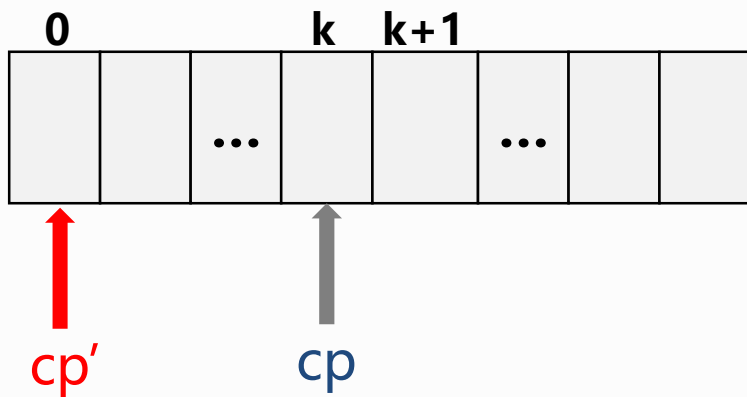
`write()`

对 $cp$ 指向的元素(地址为 $k$ 的元素)进行写入;  
 $cp = cp + 1;$



## 二、文件访问模式 - 顺序访问

File A



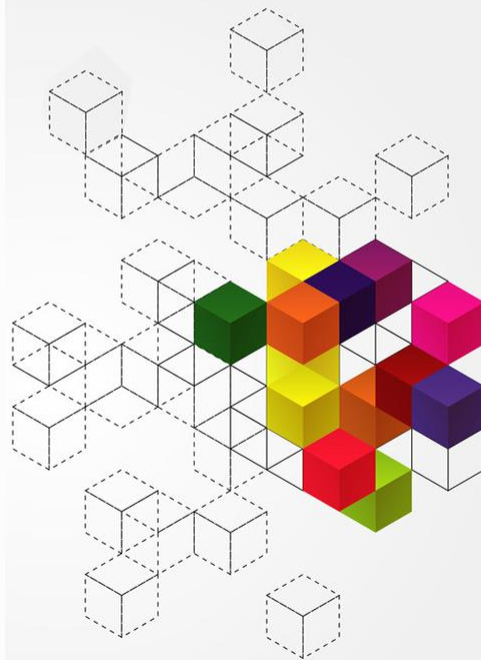
$cp$  (current position)

`read()`

`write()`

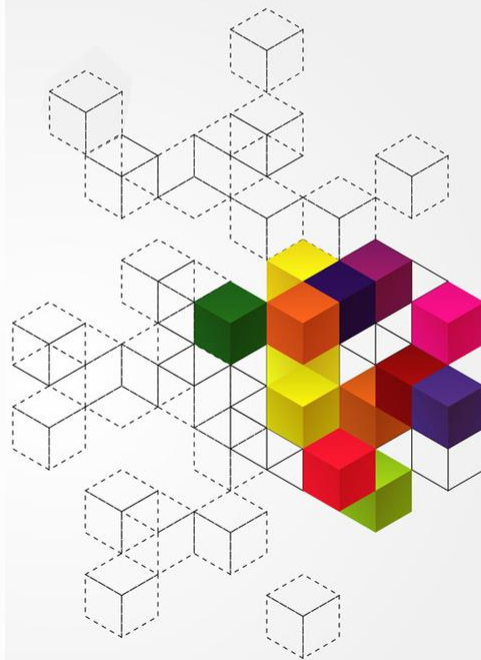
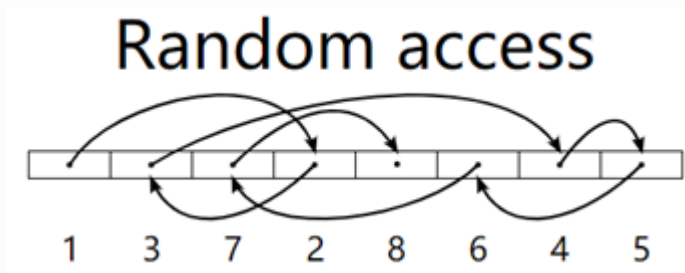
`reset()`

$cp = 0;$



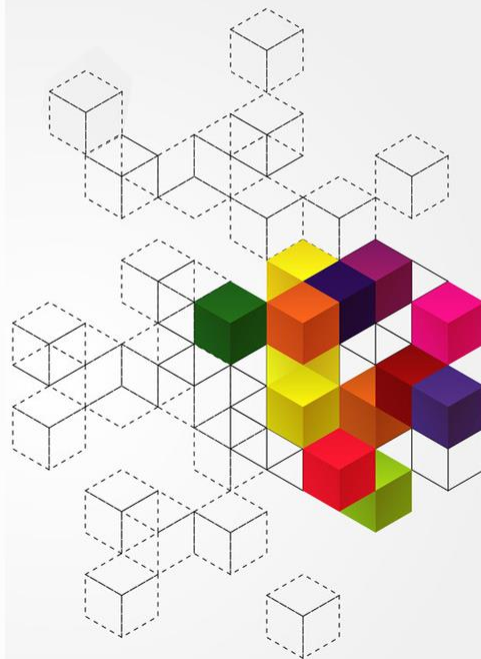
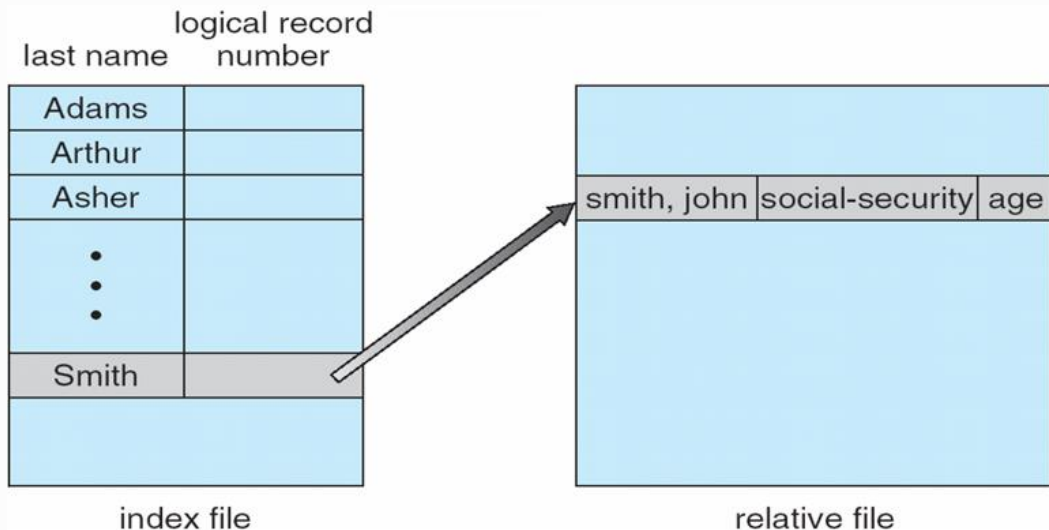
## 二、文件访问模式 - 随机访问

- 随机访问 (Random Access)
  - 又称直接访问 (Direct Access)
- 访问方式: 直接访问给定逻辑地址的文件内容
  - 基本操作:  $\text{read}(n)$ ,  $\text{write}(n)$ ,  $\text{seek}(n)$
  - 典型存储介质: 磁盘 (Magnetic Disk)



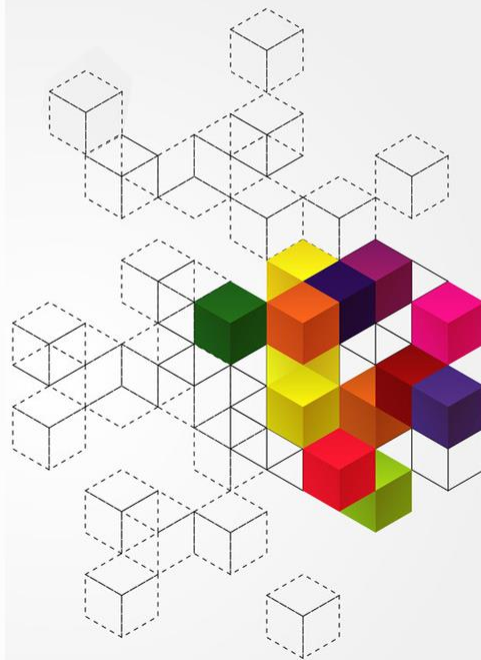
## 二、文件访问模式 -索引访问

- 基于记录关键字建立索引，以索引方式访问文件内容
  - 基本操作: read(key), write(key)
  - 典型应用: 数据库表 (DMBS Table Access)



# 本讲小结

- 文件操作
- 文件访问模式



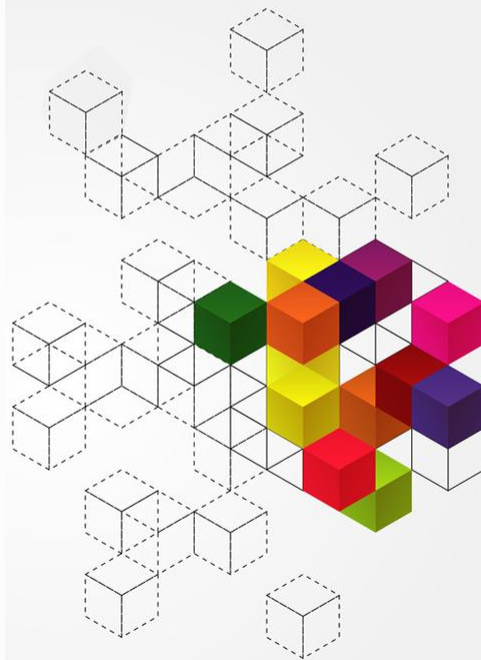


一、 磁盘分区结构

二、 目录结构

三、 目录操作

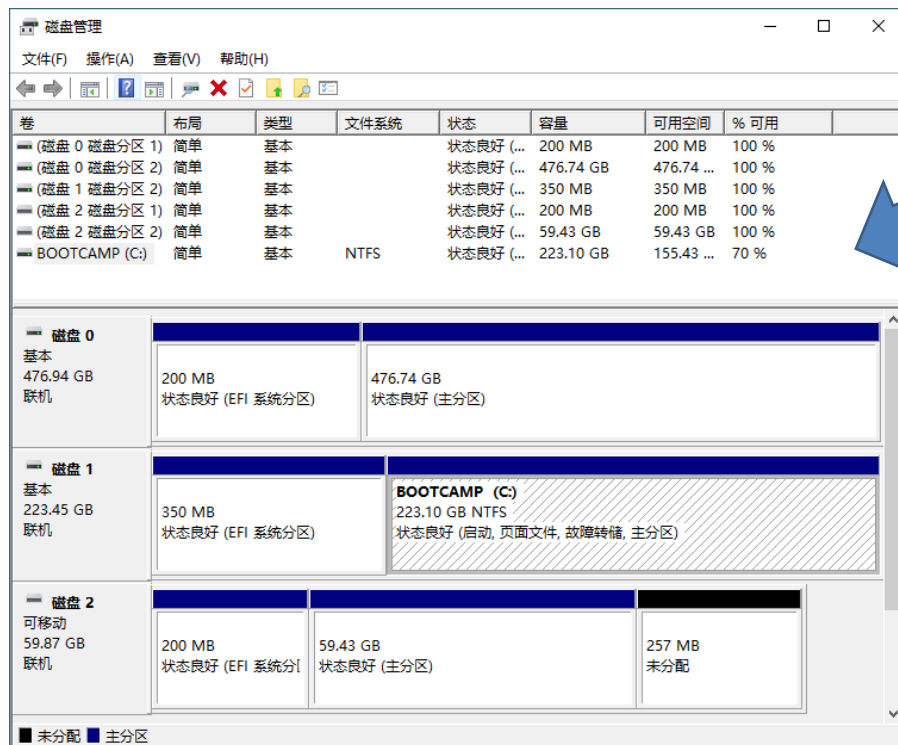
四、 目录设计原则



# 一、磁盘分区结构

## • 磁盘分区

- 每个分区使用前格式化为某一类型的文件系统



The screenshot shows the Windows 10 Disk Management console. At the top, a table lists all partitions across all disks. Below this, three detailed views are shown for Disk 0, Disk 1, and Disk 2. A blue arrow points from the 'Win10 分区示例' text to the Disk 0 section.

卷	布局	类型	文件系统	状态	容量	可用空间	% 可用
(磁盘 0 磁盘分区 1)	简单	基本		状态良好 (...)	200 MB	200 MB	100 %
(磁盘 0 磁盘分区 2)	简单	基本		状态良好 (...)	476.74 GB	476.74 ...	100 %
(磁盘 1 磁盘分区 2)	简单	基本		状态良好 (...)	350 MB	350 MB	100 %
(磁盘 2 磁盘分区 1)	简单	基本		状态良好 (...)	200 MB	200 MB	100 %
(磁盘 2 磁盘分区 2)	简单	基本		状态良好 (...)	59.43 GB	59.43 GB	100 %
BOOTCAMP (C:)	简单	基本	NTFS	状态良好 (...)	223.10 GB	155.43 ...	70 %

磁盘 0	布局	类型	文件系统	状态	容量	可用空间	% 可用
基本 476.94 GB 联机	200 MB 状态良好 (EFI 系统分区)	476.74 GB 状态良好 (主分区)					

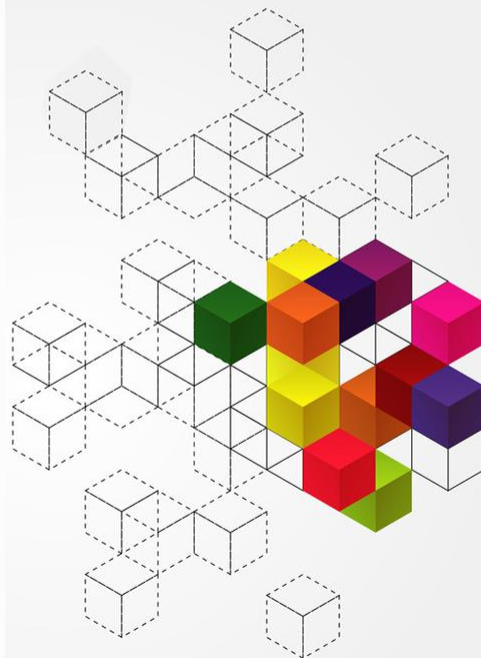
  

磁盘 1	布局	类型	文件系统	状态	容量	可用空间	% 可用
基本 223.45 GB 联机	350 MB 状态良好 (EFI 系统分区)	BOOTCAMP (C:) 223.10 GB NTFS 状态良好 (启动, 页面文件, 故障转储, 主分区)					

磁盘 2	布局	类型	文件系统	状态	容量	可用空间	% 可用
可移动 59.87 GB 联机	200 MB 状态良好 (EFI 系统分区)	59.43 GB 状态良好 (主分区)				257 MB 未分配	

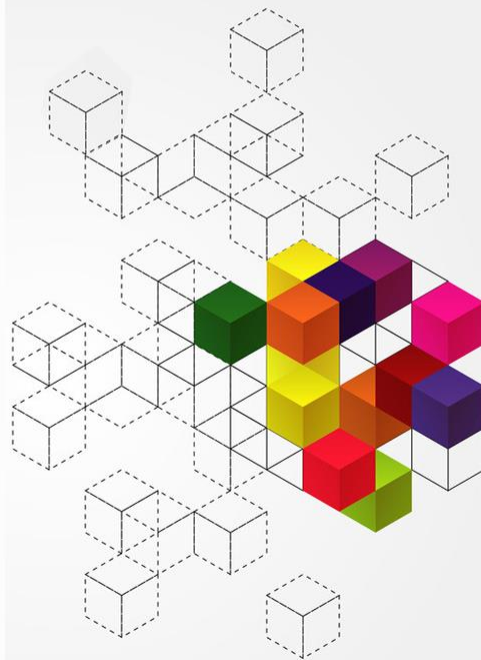
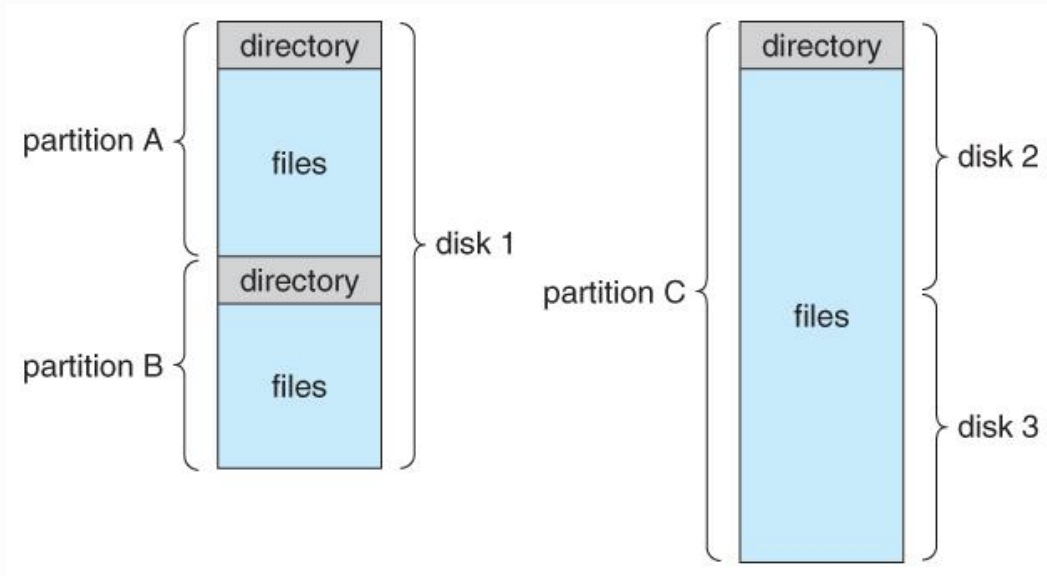
Win10  
分区示例



# 一、磁盘分区结构

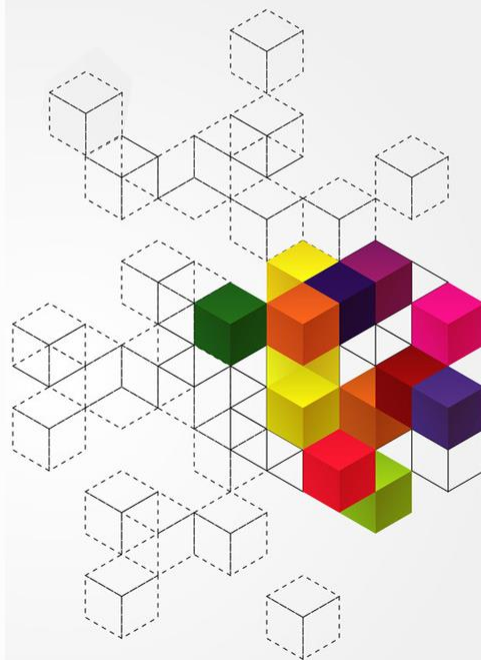
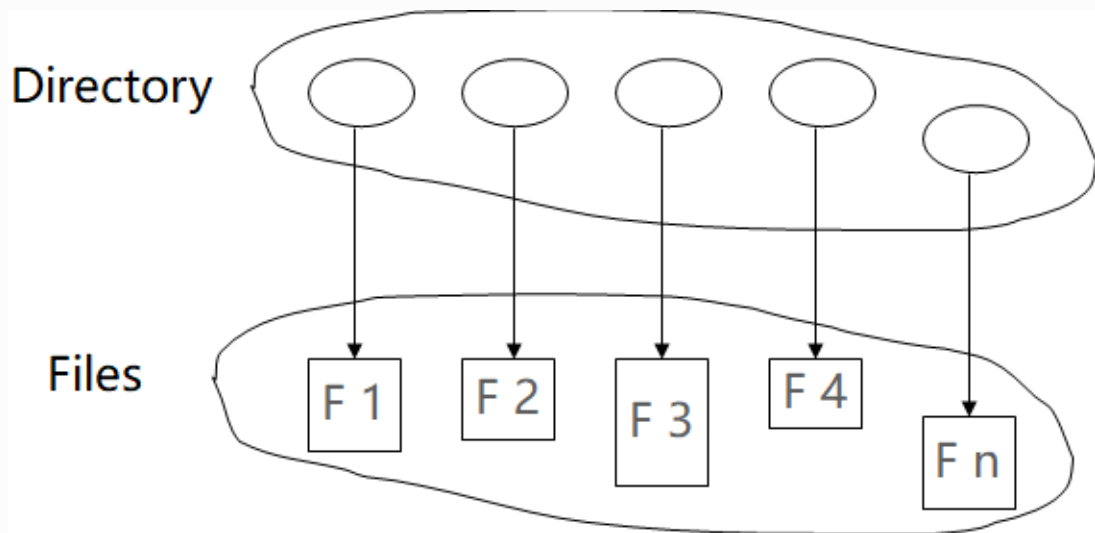
## • 磁盘分区模式

- 单个磁盘上多个分区
- 多个磁盘构成单个分区 (如RAID0)



## 二、目录结构

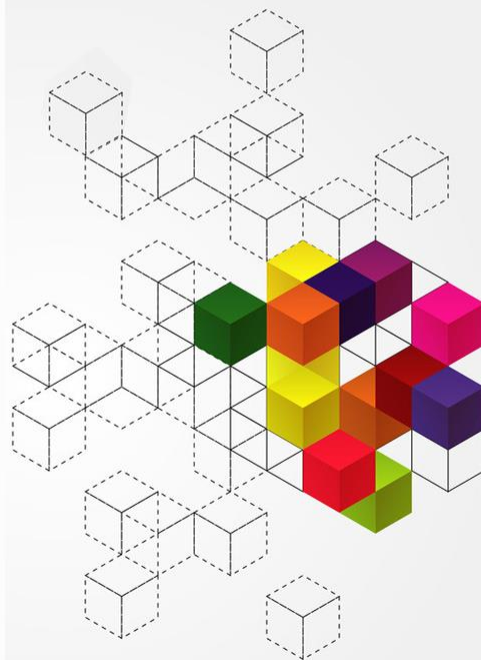
- 目录逻辑结构



## 二、目录结构

• 文件系统中，目录的主要作用是（ ）。

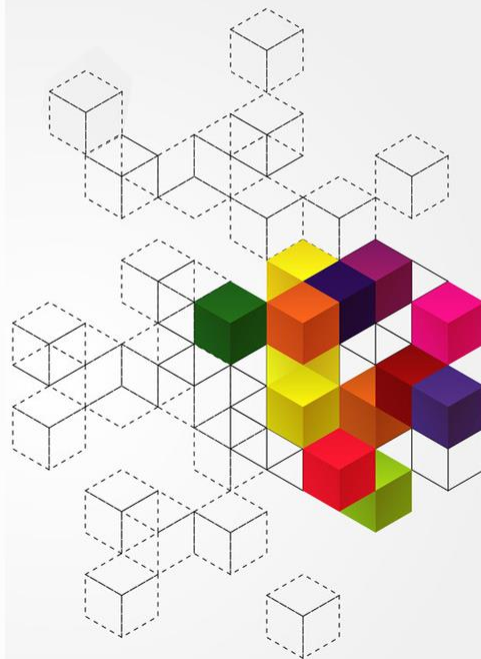
- A. 实现文件的按名存取
- B. 提高速度
- C. 节省空间
- D. 提高外存利用率



## 二、目录结构

- Linux下查看目录结构的命令：ls

```
mrx@mrx-Precision-3510: ~  
mrx@mrx-Precision-3510:~$ ls -ls  
总用量 60  
4 drwxrwxr-x 3 mrx mrx 4096 5月 12 04:43 Android  
4 drwxrwxr-x 3 mrx mrx 4096 5月 19 05:22 AndroidStudioProjects  
4 drwxrwxr-x 7 mrx mrx 4096 5月 20 18:47 code  
4 drwxr-xr-x 2 mrx mrx 4096 3月 12 10:21 Desktop  
4 drwxr-xr-x 7 mrx mrx 4096 5月 20 19:05 Documents  
4 drwxr-xr-x 5 mrx mrx 4096 5月 20 18:29 Downloads  
4 drwxr-xr-x 2 mrx mrx 4096 3月 11 18:22 Music  
4 drwxrwxr-x 4 mrx mrx 4096 5月 12 05:06 mywork  
4 drwxrwxr-x 3 mrx mrx 4096 3月 18 03:28 opt  
4 drwxr-xr-x 2 mrx mrx 4096 3月 11 18:22 Pictures  
4 drwxr-xr-x 2 mrx mrx 4096 3月 11 18:22 Public  
4 drwxr-xr-x 2 mrx mrx 4096 3月 12 10:18 Templates  
4 drwxrwxr-x 4 mrx mrx 4096 5月 14 17:53 tempwork  
4 drwxrwxr-x 7 mrx mrx 4096 3月 18 20:56 ubuntu_setup  
4 drwxr-xr-x 2 mrx mrx 4096 3月 11 18:22 Videos  
mrx@mrx-Precision-3510:~$
```

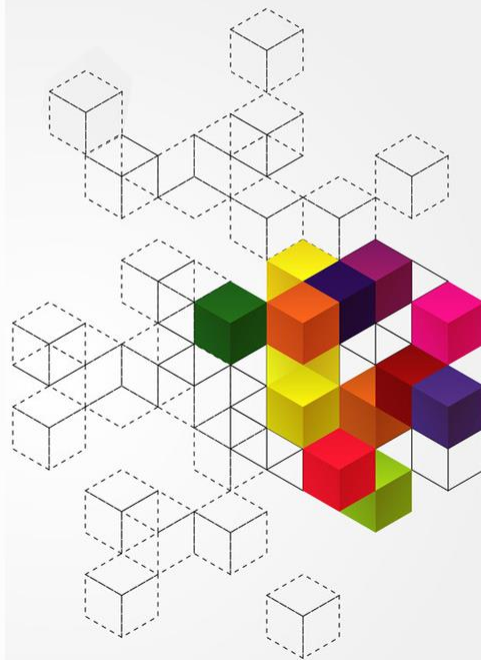


## 二、目录结构

- 目录操作示例：ls \*foo\*

```
mrx@mrx-Precision-3510: ~/foo
mrx@mrx-Precision-3510:~/foo$ ls -l *foo2*
总用量 0
mrx@mrx-Precision-3510:~/foo$ ls -d *foo2*
foo2
mrx@mrx-Precision-3510:~/foo$ ls -ls | grep *foo1*
4 drwxrwxr-x 2 mrx mrx 4096 5月 22 15:00 foo1
mrx@mrx-Precision-3510:~/foo$
```

显示当前目录下文件名中包含foo的文件信息



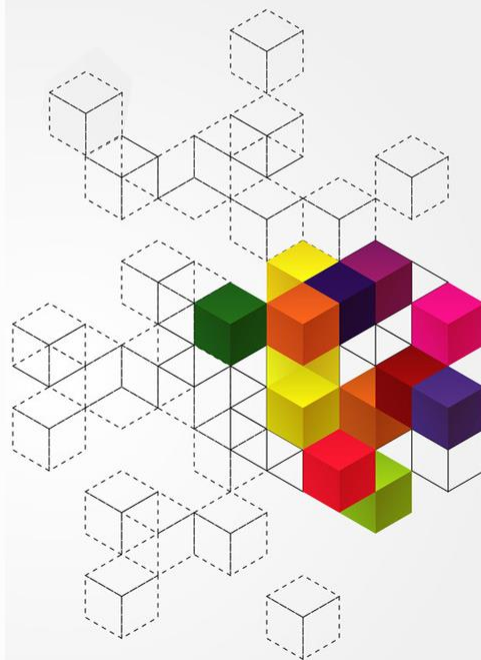


## 二、目录结构

- 目录操作示例: `ls -d */`

```
mrX@mrX-Precision-3510: ~/foo
mrX@mrX-Precision-3510:~/foo$ ls -d */
foo1/  foo2/  foo3/
mrX@mrX-Precision-3510:~/foo$
```

显示当前目录下所有的子目录

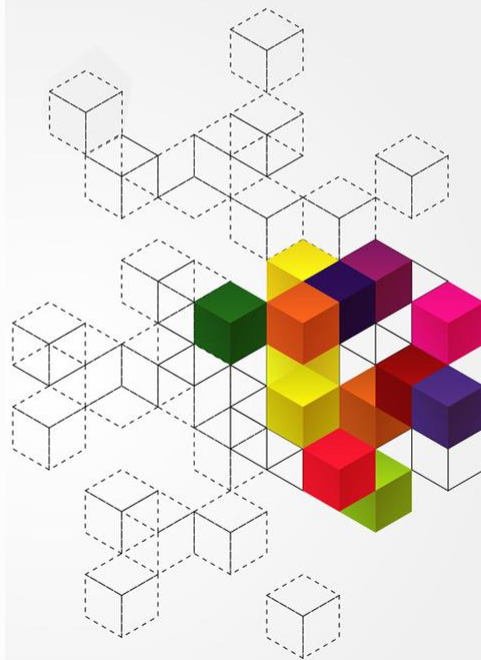




## 三、目录操作

### • 基本目录操作

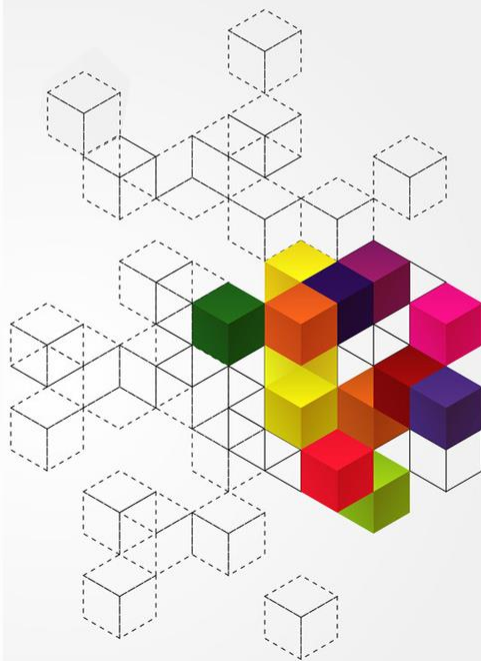
目录操作	说明
查找文件	按文件名在目录中查找对应目录项
创建文件	创建新文件时，要在所在目录中添加目录项
删除文件	删除文件时，需要同时删除目录中对应的目录项
显示目录内容	读取目录中所有目录项，并显示其属性
文件重命名	修改文件对应目录项
文件系统遍历	从根目录开始，处理每个目录的每个目录项



## 四、目录设计原则

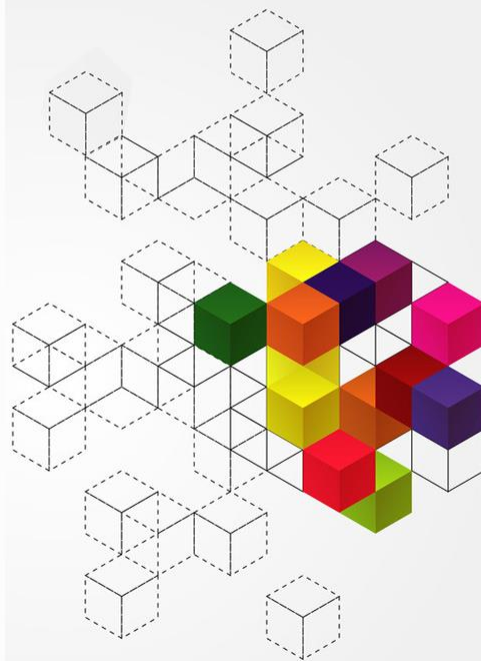
### 目录结构设计要注意3项指标：

- **目录内容的查找效率**。要能够在目录结构中迅速定位指定的文件
- **命名机制**。允许不同用户拥有同名文件；允许同一文件有多个不同的文件名。
- **文件分组功能**。要求能够按照用户按照文件的属性和类别将文件分成不同的分组。



# 本讲小结

- 磁盘分区结构
- 目录结构
- 目录操作
- 目录设计原则

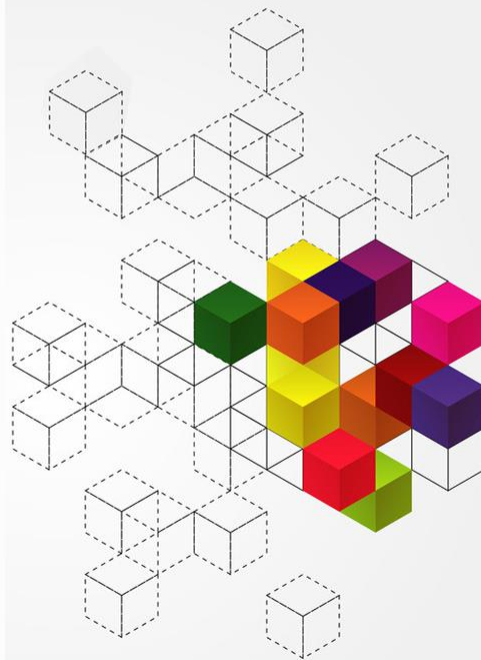


一、单级目录

二、二级目录

三、树状目录

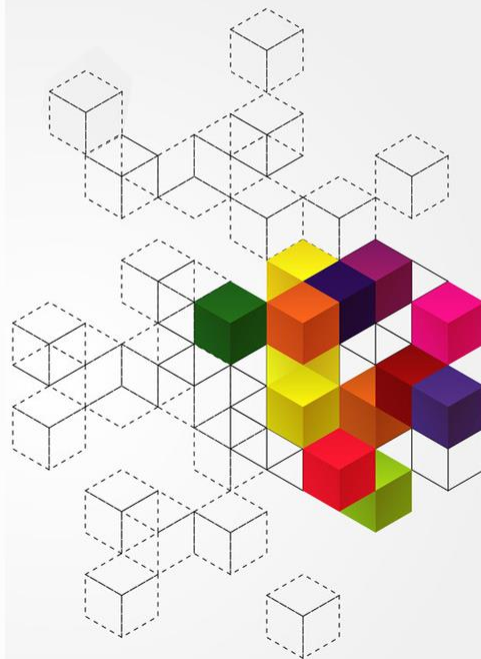
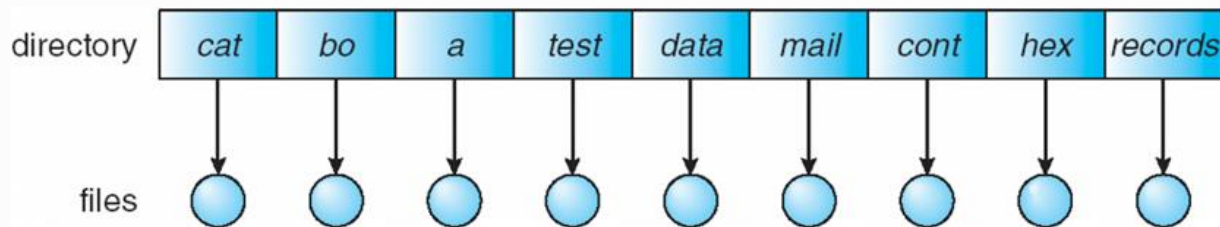
四、一般图结构目录



# 一、单级目录

- **Single Level Directory**

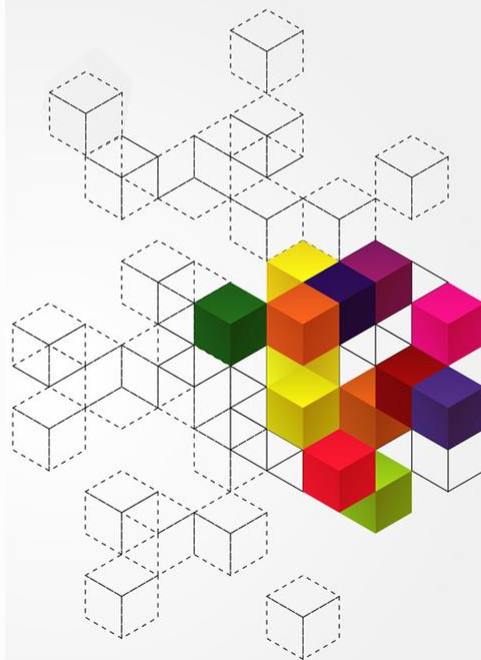
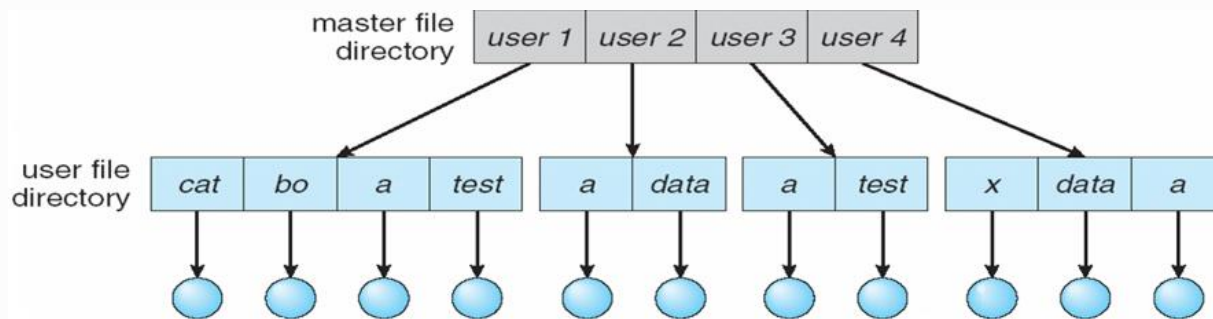
- 所有文件存放在单一目录内



## 二、二级目录

- **Two-Level Directory**

- 每个用户的文件放在单个目录下
- 一个主目录包含所有用户目录

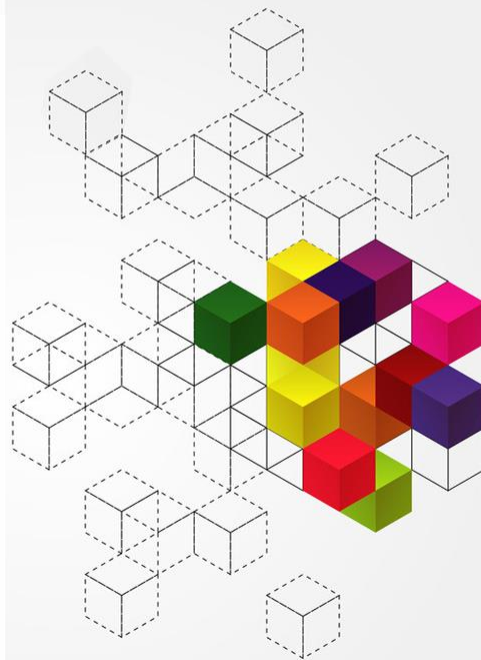
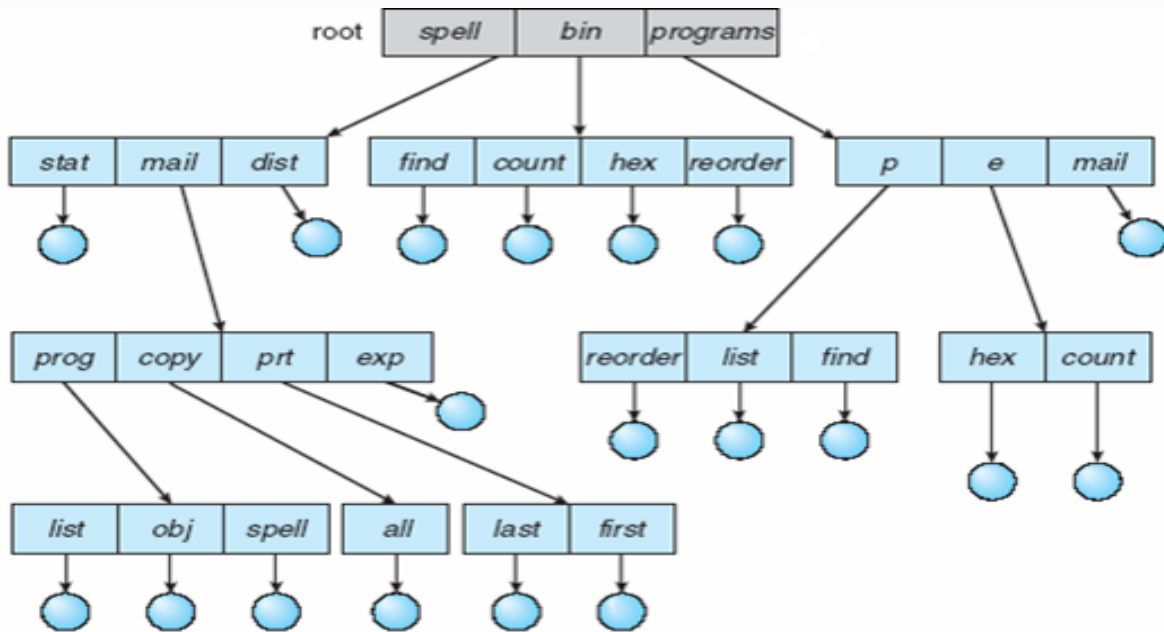




### 三、树状目录

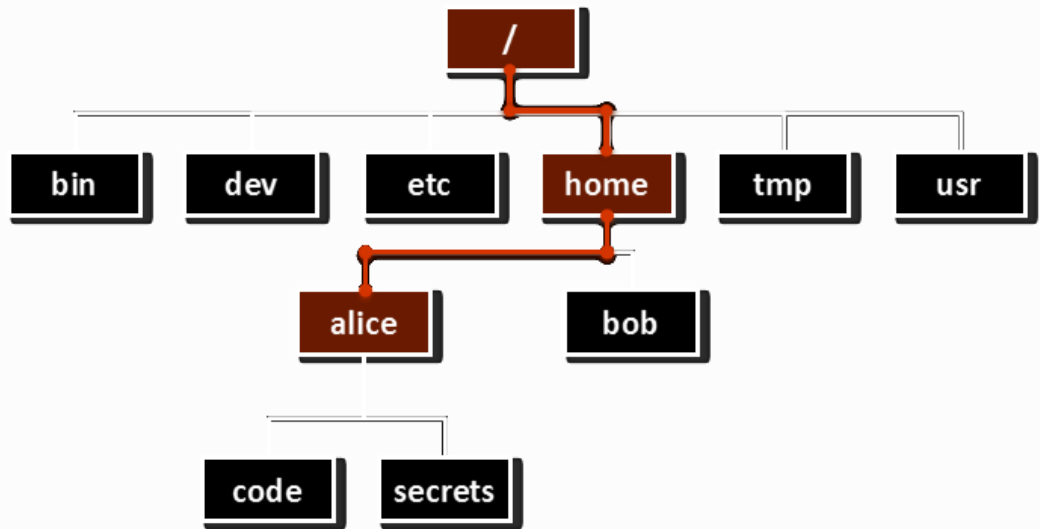
- **Tree-Structured Directory**

- 每个目录中都可包含多个子目录，最终形成树状目录



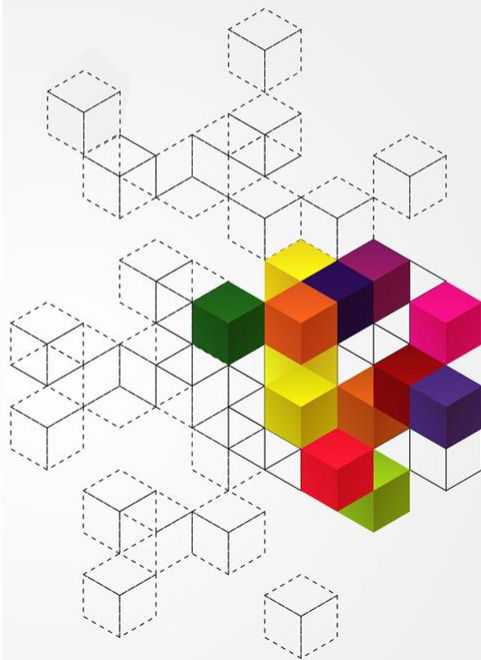
### 三、树状目录

- 绝对路径 (Absolute Path)



图中所示的alice是文件还是目录？其绝对路径是什么？

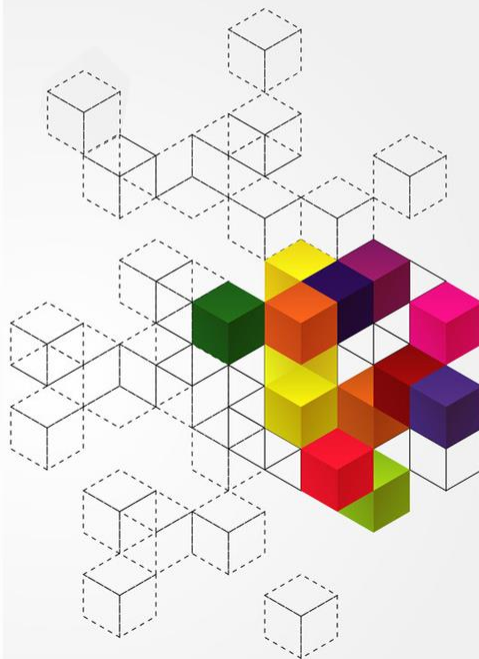
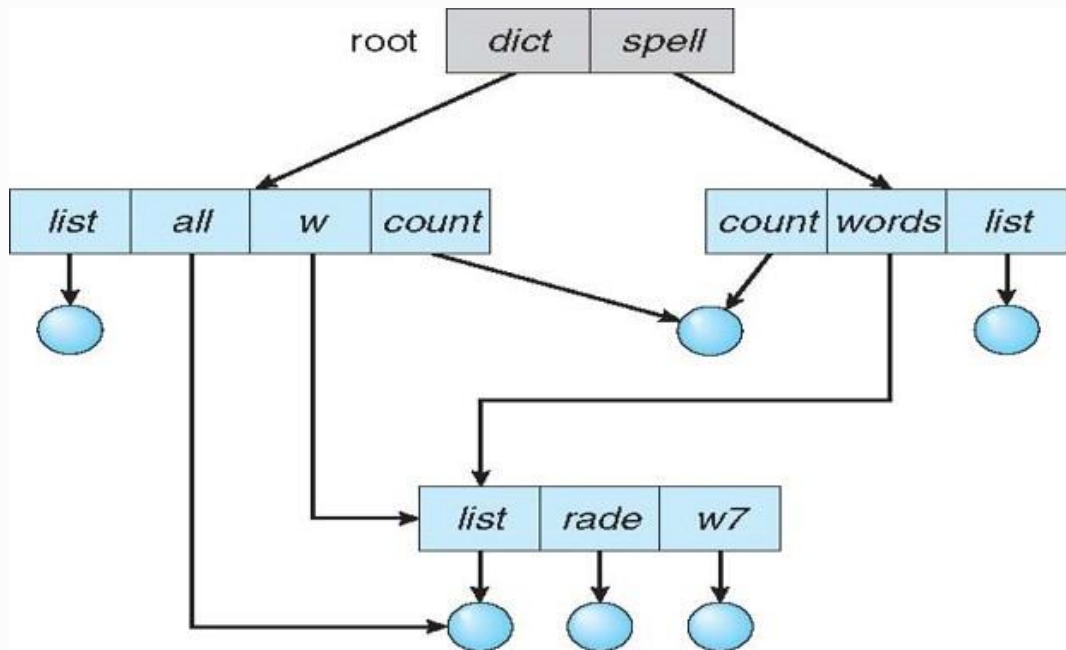
Alice是一个目录，其绝对路径是/home/alice



## 四、图结构目录

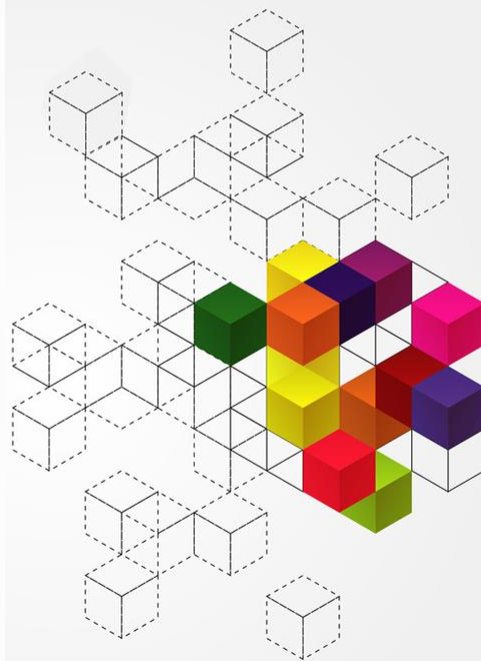
### • Acyclic-Graph Directory

- 不同目录中可能存在指向同一文件的目录项
- 但整体目录结构中不允许存在环路



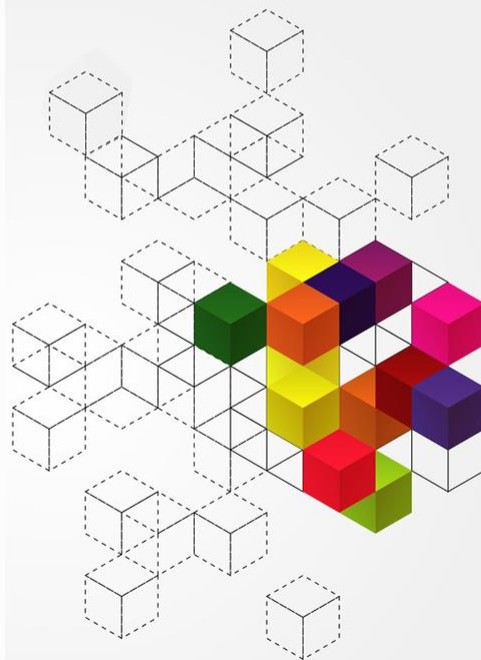
# 本讲小结

- 单级目录
- 二级目录
- 树状目录
- 图结构目录



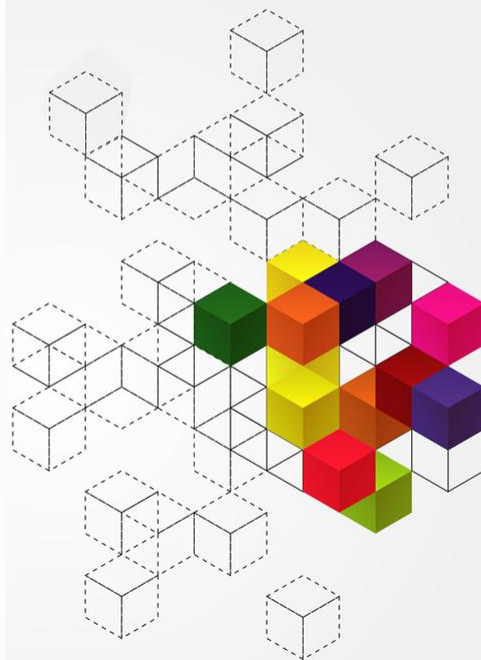
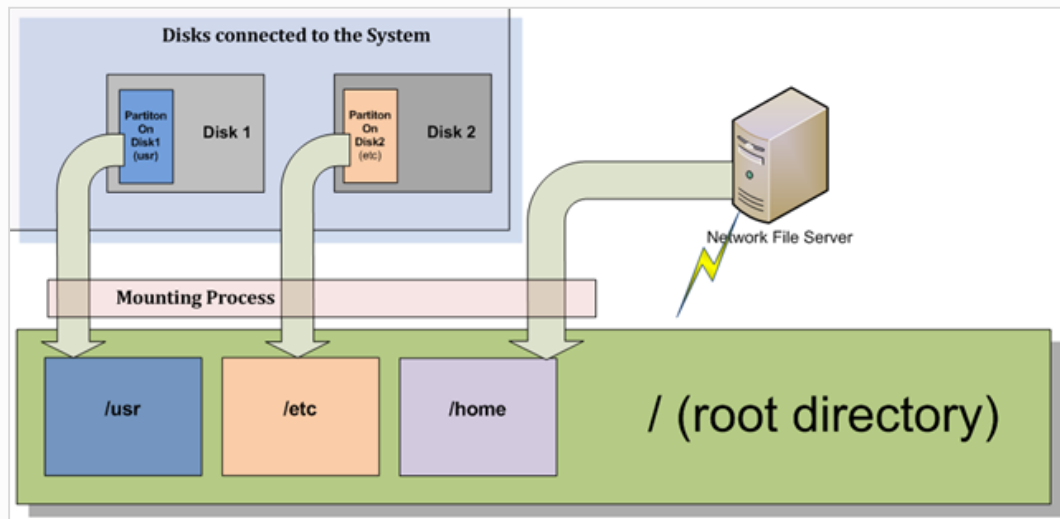
一、 文件系统加载

二、 文件保护



# 一、文件系统加载

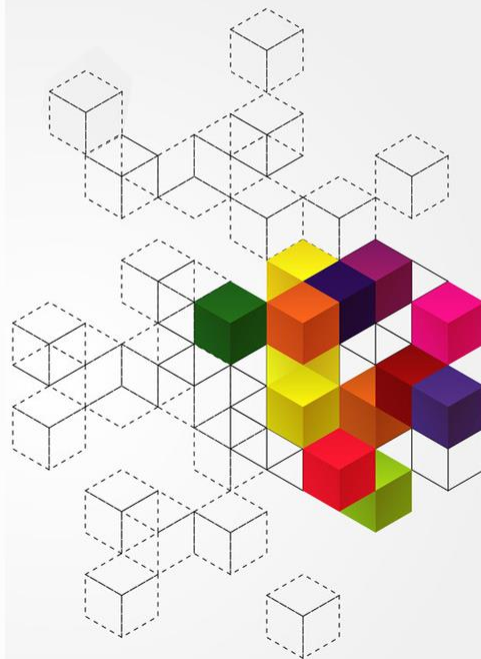
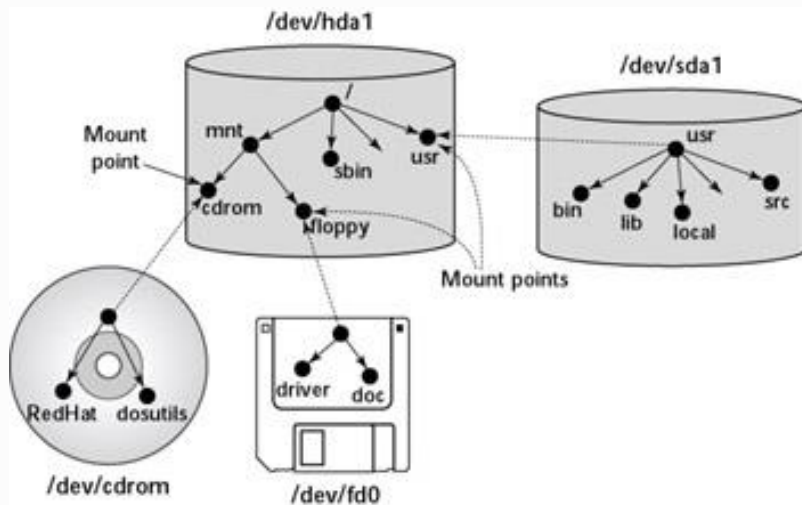
- 文件系统加载是操作系统初始化的关键步骤
- 文件系统加载操作，将本地磁盘或远程服务器的磁盘上的文件系统加载到根文件系统的特定加载点



# 一、文件系统加载

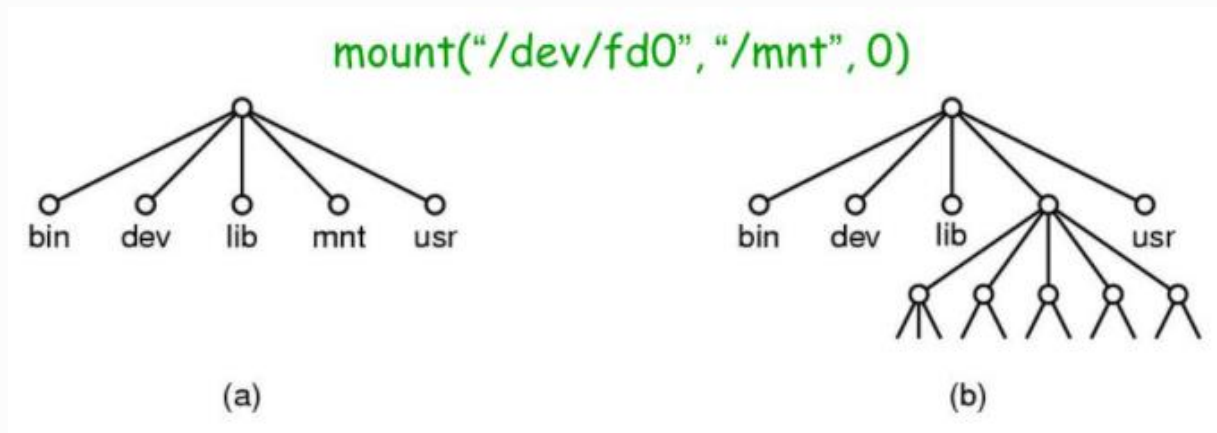
## • 文件系统加载操作

- 加载点: mount point
- /mnt/cdrom
- /mnt/floppy
- ...

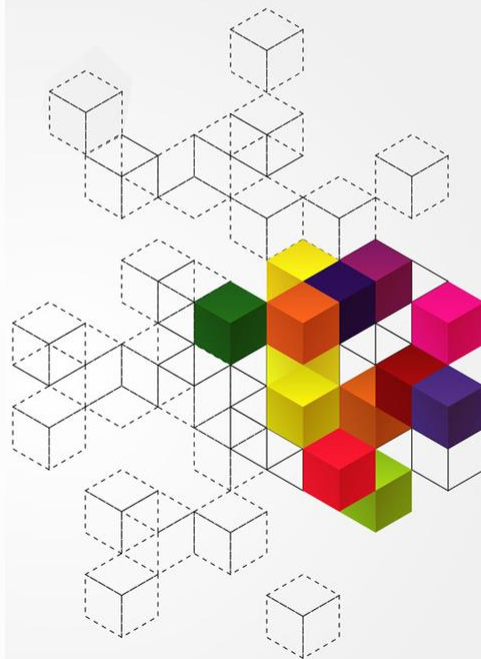


# 一、文件系统加载

- 文件系统加载操作能够将两个文件系统合二为一



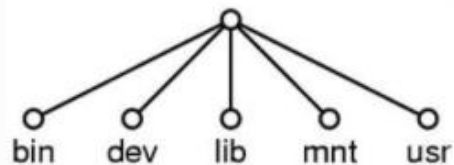
- 例如需要将设备/dev/fd0到根分区中的/mnt目录，可以通过调用mount系统调用来完成



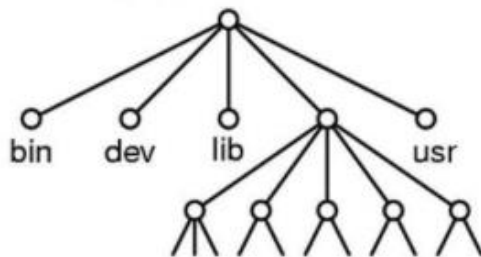


# 一、文件系统加载

## • Umount系统调用



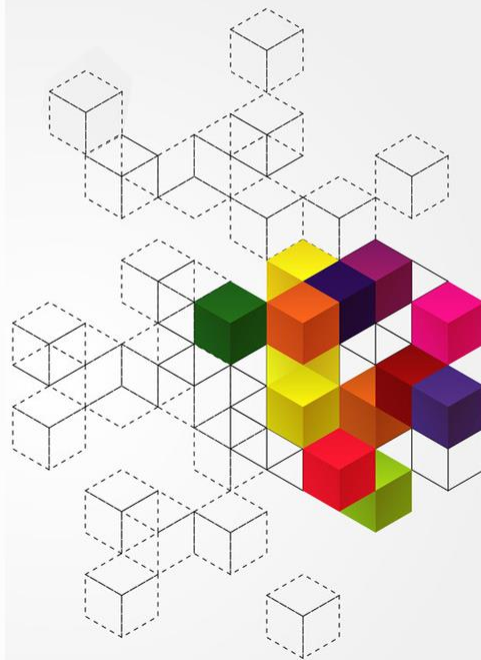
(a)



(b)



• `Umount( "/mnt" );`

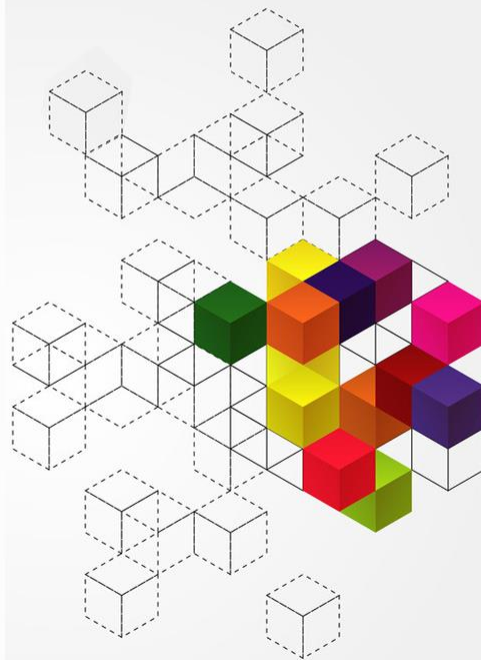


# 一、文件系统加载

## • Linux下的mount操作

```
dave@howtogeek:~/Downloads$ sudo mount -t iso9660 -o loop TinyCore-current.iso /mnt
[sudo] password for dave:
mount: /mnt: WARNING: device write-protected, mounted read-only.
dave@howtogeek:~/Downloads$
```

- 此示例：在Linux命令行下，通过mount命令将一个iso映像文件以iso9660文件系统格式，加载到加载点/mnt
  - mount是涉及到设备访问的关键命令，要求root权限(需要在命令前加上sudo)

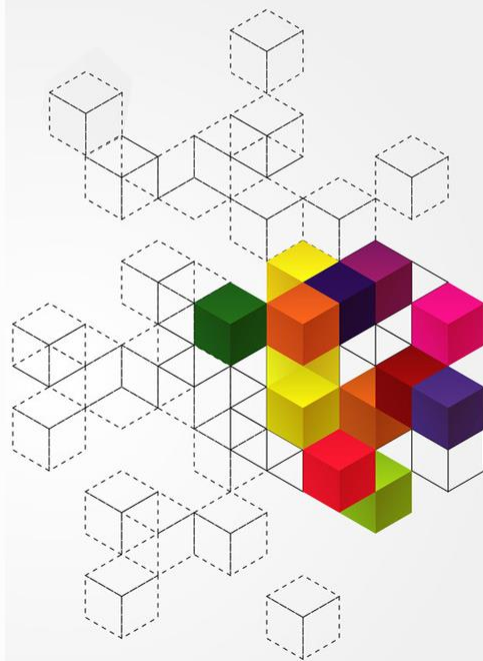


## 二、文件保护

### • 文件保护的目

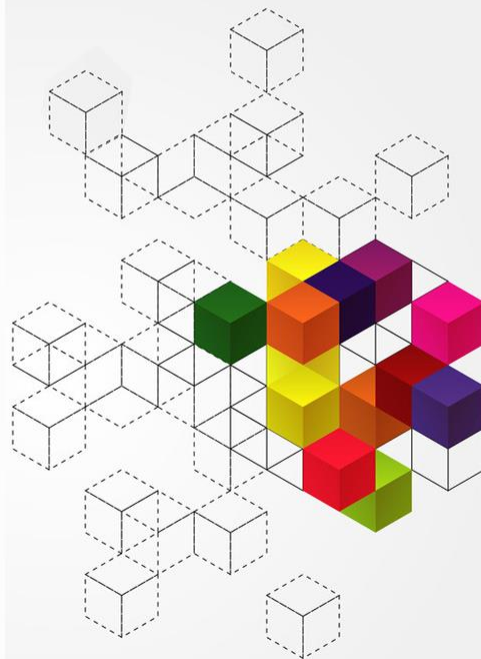
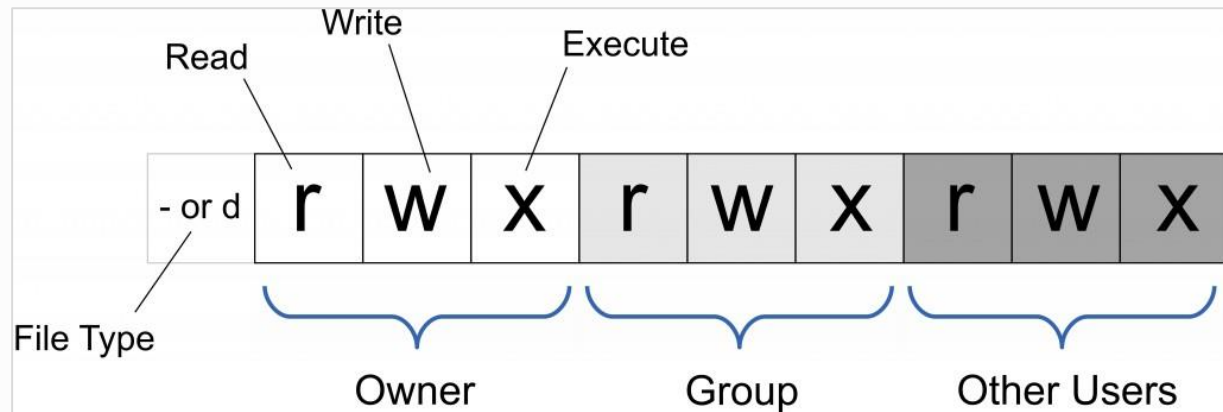
- 避免文件被用户/程序错误或恶意地使用
- 保证共享文件资源的使用符合系统资源保护策略
- 最小化文件的错误使用行为对系统所造成的伤害

操作系统提供文件资源**保护机制**  
系统管理员利用这些机制**实施安全策略**



## 二、文件保护

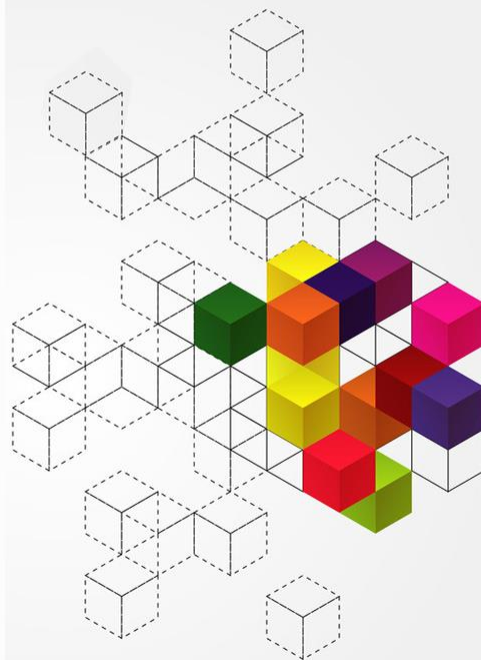
- Linux中的文件访问控制信息



## 二、文件保护

- Linux中的文件保护手段1：9-bit permission model

-rw-rw-r--	1	pbg	staff	31200	Sep 3 08:30	intro.ps
drwx-----	5	pbg	staff	512	Jul 8 09:33	private/
drwxrwxr-x	2	pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2	jwg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1	pbg	staff	9423	Feb 24 2017	program.c
-rwxr-xr-x	1	pbg	staff	20471	Feb 24 2017	program
drwx--x--x	4	tag	faculty	512	Jul 31 10:31	lib/
drwx-----	3	pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3	pbg	staff	512	Jul 8 09:35	test/



# 本讲小结

- 文件系统加载
- 文件保护

