操作系统

L18 文件系统接口

胡燕
大连理工大学 软件学院

文件概念

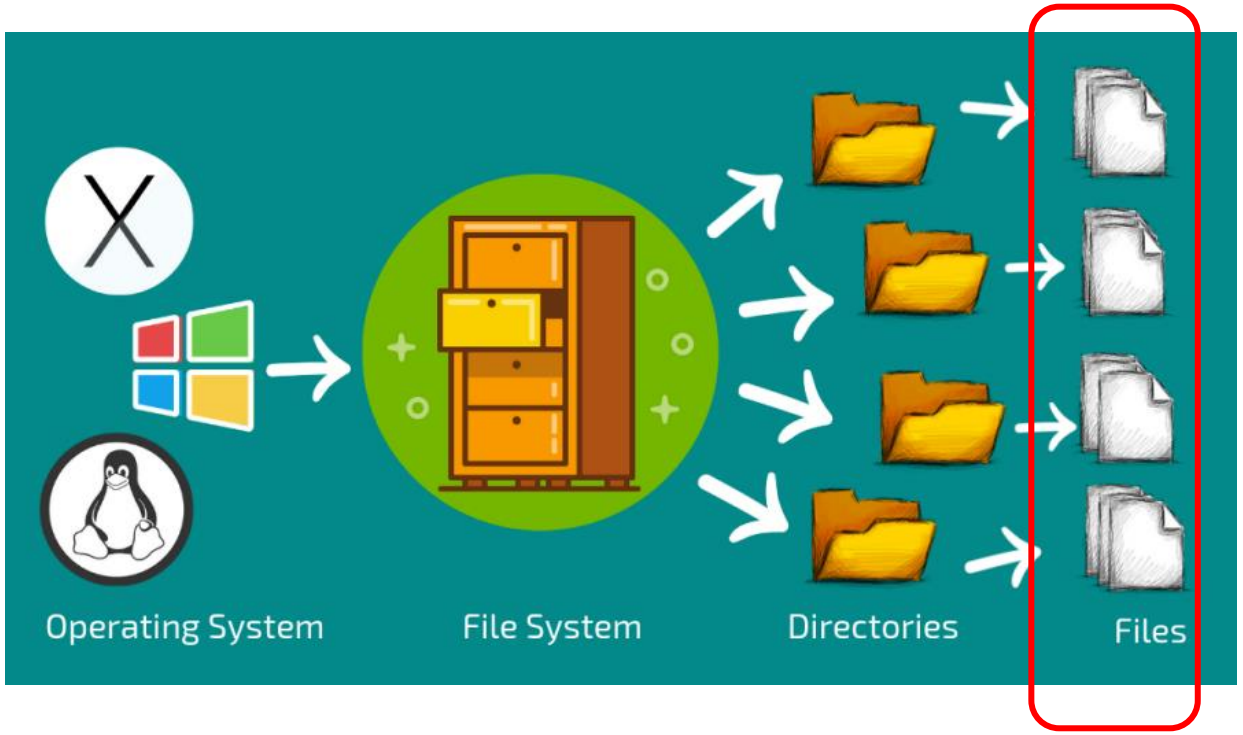
File Concepts

01

什么是文件

文件与文件系统

- 文件系统（File System）



文件是文件系统中的核心要素

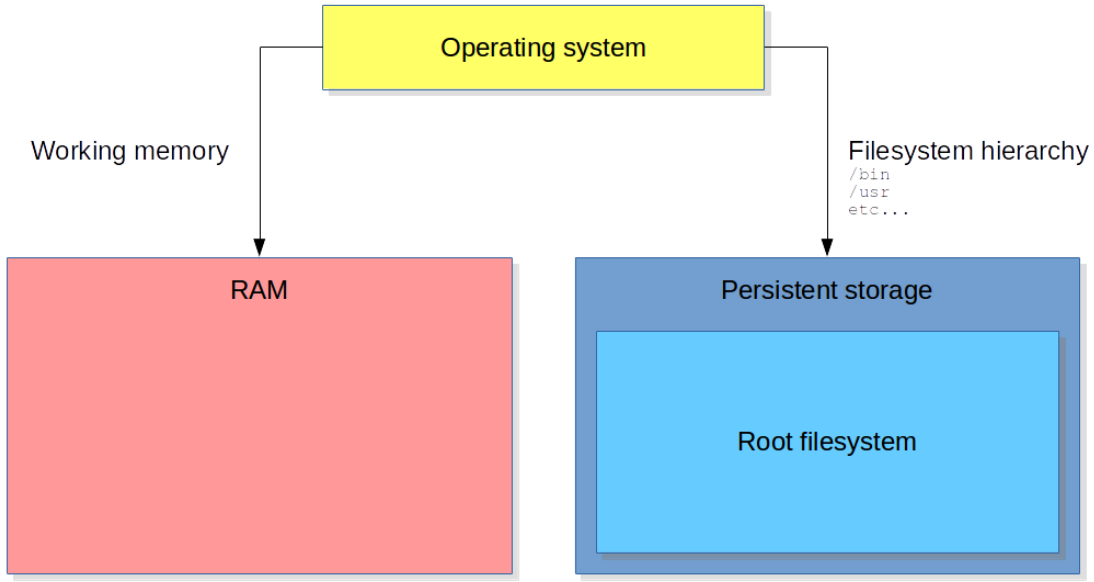
什么是文件

操作系统与文件系统之间是什么关系?

文件系统是操作系统的核心模块

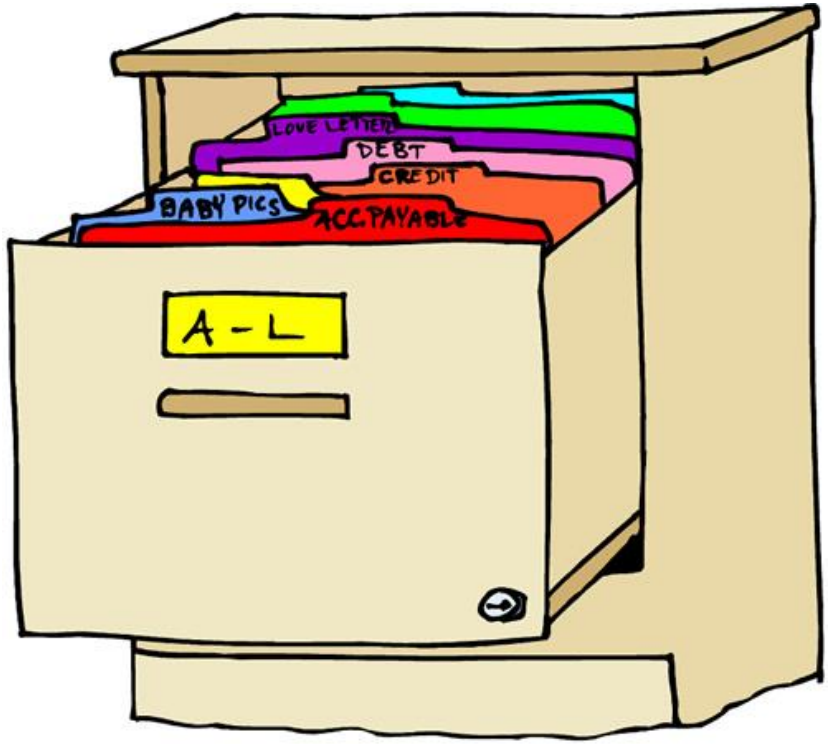
操作系统安装时，主要工作是：创建系统分区，并在系统分区建立根文件系统

根文件系统是一种文件系统，相对于普通文件系统，它的特殊之处在于：
根文件系统是内核启动时所mount的第一个文件系统，内核代码映像文件保存在根文件系统中，系统引导程序需要借助磁盘的启动扇区上的启动代码将内核加载。
并且，系统引导启动程序会在根文件系统挂载之后将从中把一些基本的初始化脚本和服务等加载到内存中去运行。



1-文件概念

什么是文件



文件系统中的文件概念源自现实生活中的文件资料

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

什么是文件

每个文件代表一段连续的逻辑数据

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，我们可以为其设定逻辑地址范围0-99

什么是文件

文件分类(根据内容性质):

- 数据文件

  .txt, .pdf, .doc

  ...

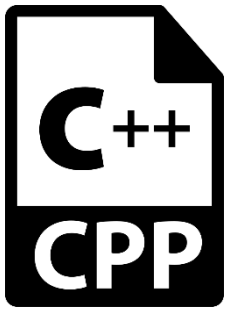
- 代码文件

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

  .obj, .o

  .lib, .dll, .a, .so

  .exe, .elf



什么是文件

文件分类(根据内容性质):

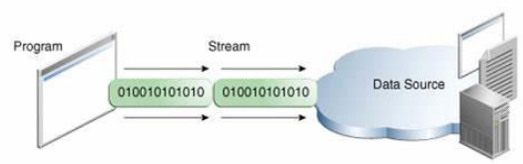
- UNIX文件类型

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.

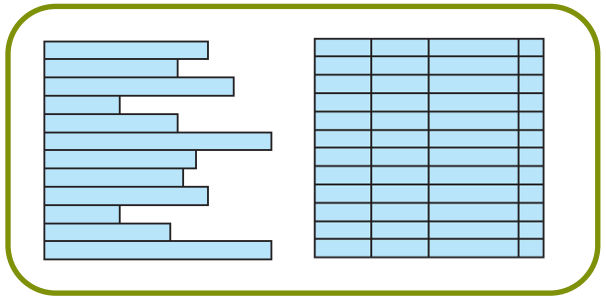
文件逻辑结构

- 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

文件数据在OS层面被看成字节流或字符流
- 无结构
- 逻辑地址以字节为单位编址
- 通过文件数据的逻辑地址访问具体文件内容

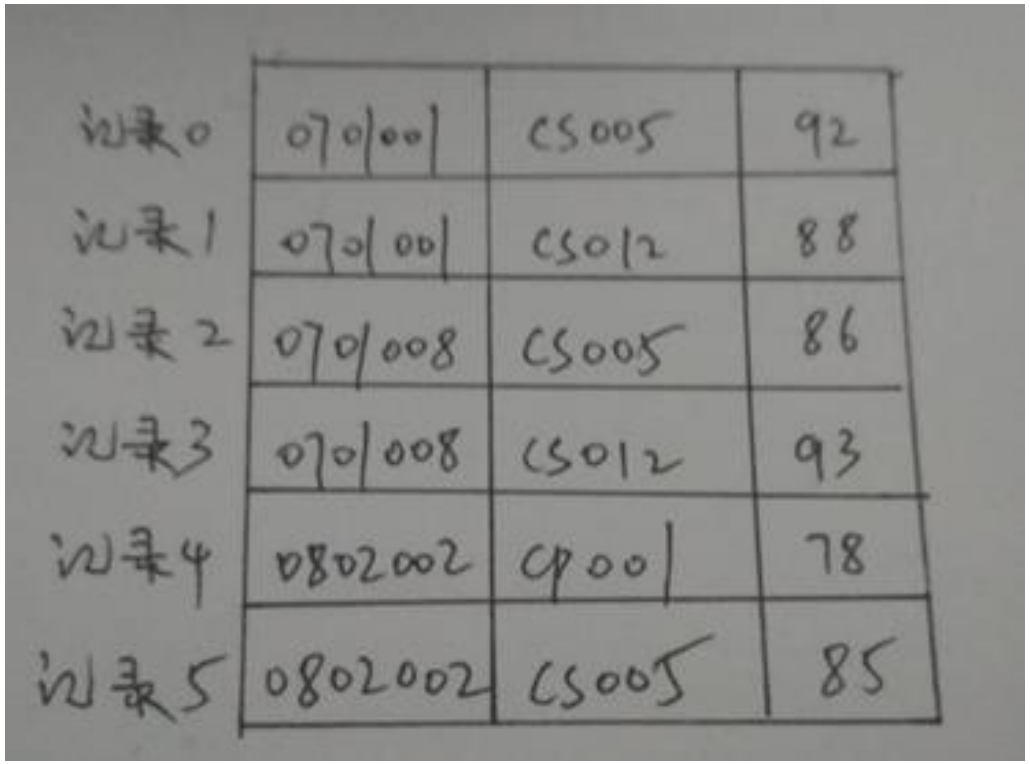
文件由固定格式的记录组成
- 记录按一定的顺序排列
- 可以根据记录的格式计算数据记录的地址（通常会为记录设定关键字，并以关键字建立记录索引）

有结构文件：定长记录

例如：

```
create table student (
    studentNo char(7),
    courseNo char(5),
    float score,
)
```

若一个字符占1B,float类型占4B，
那么student表中的一条记录占16B。



有结构文件：变长记录

在数据库系统中，以下几种情况会使用变长记录：
1.多种记录类型（即多个关系表）在一个文件中存储；
2.允许记录类型中包含一个或多个变长字段；
3.允许记录类型中包含重复字段，如数组等。

文件逻辑结构

按照记录式文件内的记录在逻辑上如何组织，可以分为

- 顺序逻辑结构
  文件中的记录一个接一个地排列
- 索引逻辑结构
  使用定长记录的顺序文件作为索引文件，对记录进行索引
- 索引顺序逻辑结构
  将记录分为一组一组，以组为单位建立索引；
  进行文件访问时，首先索引到记录组，再在组内顺序查找

示例：MySQL的数据库相关文件
.frm（表结构），.MYD(数据)，.MYI(索引)

文件逻辑结构概念小结

操作系统中对文件结构的支持以流式文件为主。
（文件的结构解析通常需要在应用层予以支持）

数据库管理系统（DBMS）中，对数据库表的存储多采用记录式结构。
（定长记录、变长记录、索引）

文件操作

File Operations

02

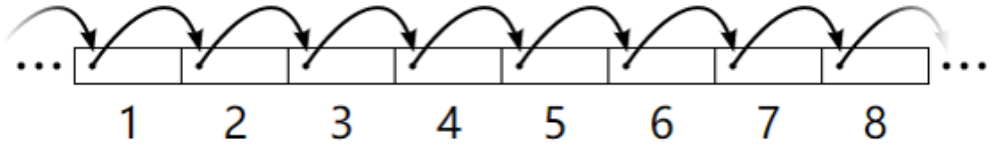
简介

- OS为文件对象提供的常规操作

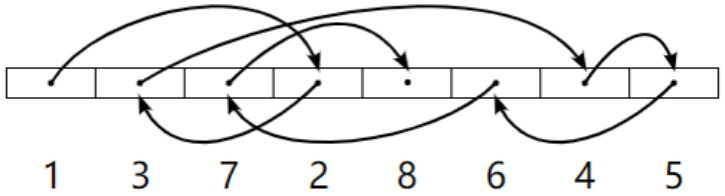
文件操作	说明
Open	打开文件（输入参数：文件名）
Close	关闭文件（输入参数：文件句柄）
Create	创建文件
Read	读文件内容
Write	写入文件
Seek	在文件内重新定位文件指针
Truncate	文件截短

文件的访问模式

1.顺序访问
Sequential Access



2.直接访问
Direct Access



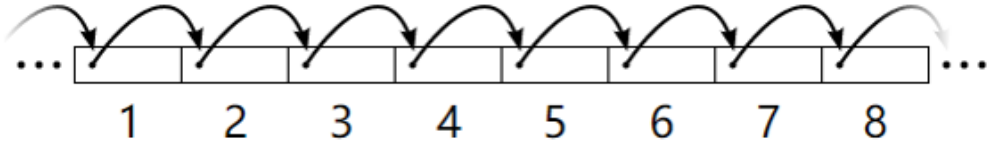
3.索引访问
Indexed Access

文件的访问模式

(1) 顺序访问模式

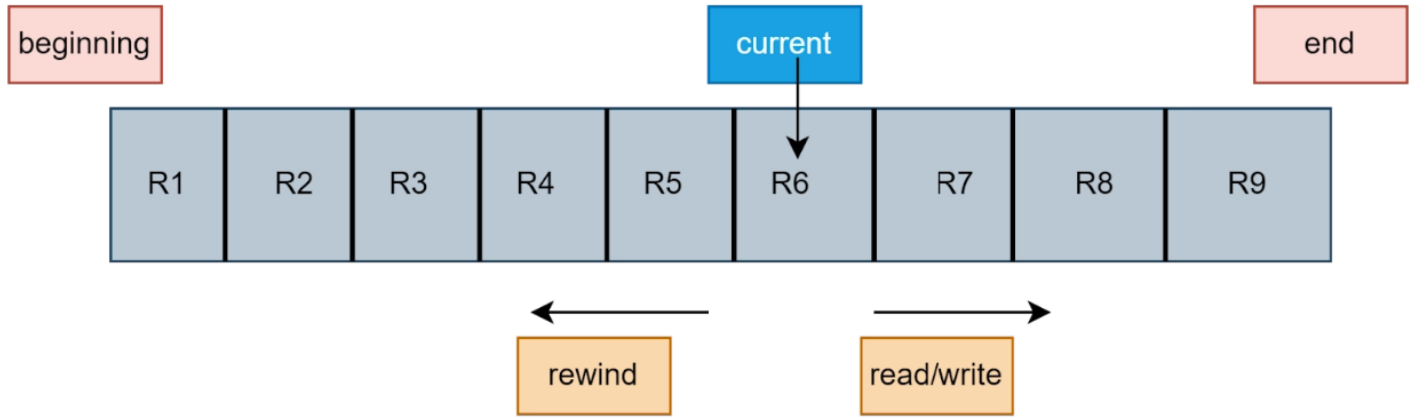
Simplest access method

Information processed in order，one record after the other
按顺序访问文件内容



典型代表：编译器（编译器处理代码文件时，采取顺序访问模式）

像磁带这样的顺序访问设备以及磁盘这样的随机访问设备，均能够支持文件的顺序访问模式

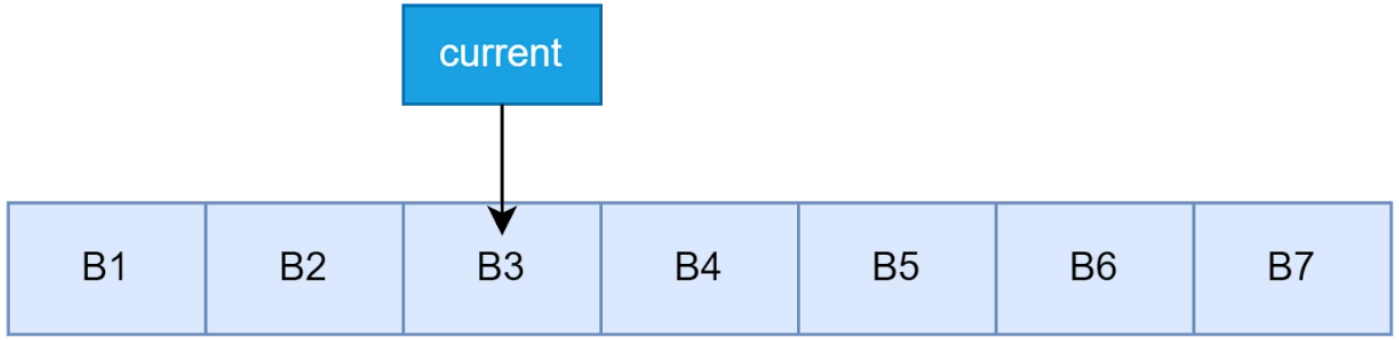


文件的访问模式

(2)随机访问模式

In random file access, a program may immediately jump to any location in the file.
支持按数据地址直接访问文件内数据的模式，亦称“直接访问模式” （Direct Access Mode）

Items in a random access file can be accessed in any order



磁盘这样的随机访问设备，能够支持随机访问模式
像磁带这样的顺序访问设备，无法有效支持随机访问模式

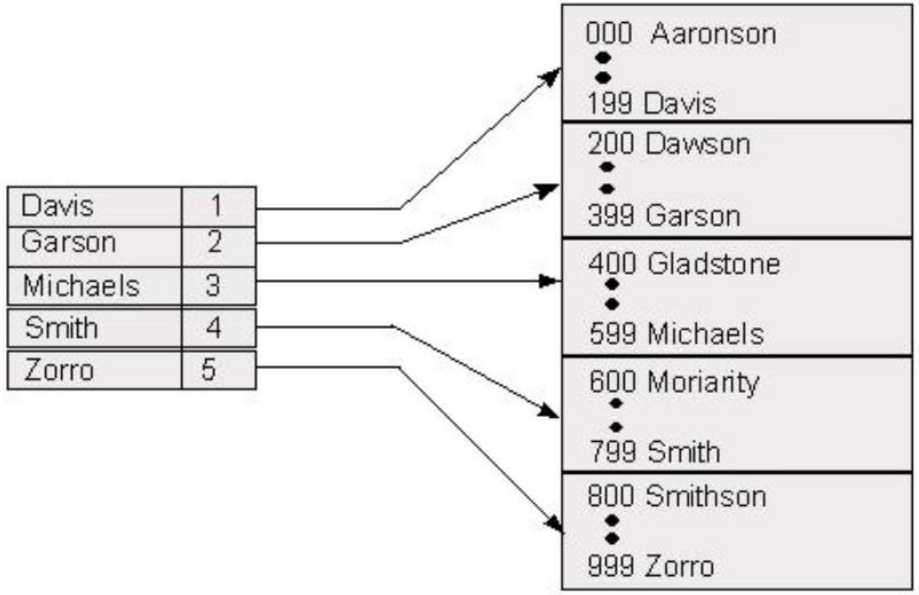
文件的访问模式

(3)索引访问模式

This method involves creating an index file that maps logical keys or data elements to their corresponding physical addresses within the file. Moreover, the system stores the index separately from the data file, enabling quick access to locate the desired data.

Provided in older commercial operating systems (e.g. IBM ISAM)

(Usually) handled separately by modern database systems



左侧是索引访问示意图
（逻辑结构：索引顺序文件）

文件的访问模式:示例

顺序访问：C程序示例

```
#include<stdio.h>

int main()  {
    FILE *fp;
    fp = fopen("scaler.txt","r");
    if(!fp) {
        printf("Error in opening file\n");
        return 0;
    }
    //Initially, the file pointer points to the starting of the file.
    printf("Position of the pointer : %ld\n",ftell(fp));

    char ch;
    while(fread(&ch,sizeof(ch),1,fp)==1) {
        //Here, we traverse the entire file and print its contents until we reach it's end.
        printf("%c",ch);
    }
    printf("Position of the pointer : %ld\n",ftell(fp));

    //Below, rewind() will bring it back to its original position.
    rewind(fp);
    printf("Position of the pointer : %ld\n",ftell(fp));

    fclose(fp);
    return 0;
}
```

文件的访问模式:示例

随机访问：C程序示例

```
#include<stdio.h>

int main()
{
    FILE *fp;
    fp = fopen("scaler.txt","r");
    if(!fp)
    {
        printf("Error: File cannot be opened\n");
        return 0;
    }
    //Move forward 6 bytes, thus we won't be seeing the first 6 bytes if we print till the end.
    fseek(fp, 6, 0);
    char ch;
    while(fread(&ch,sizeof(ch),1,fp)==1)
    {
        //Here, we traverse the entire file and print its contents until we reach its end.
        printf("%c",ch);
    }

    fclose(fp);
    return 0;
}
```

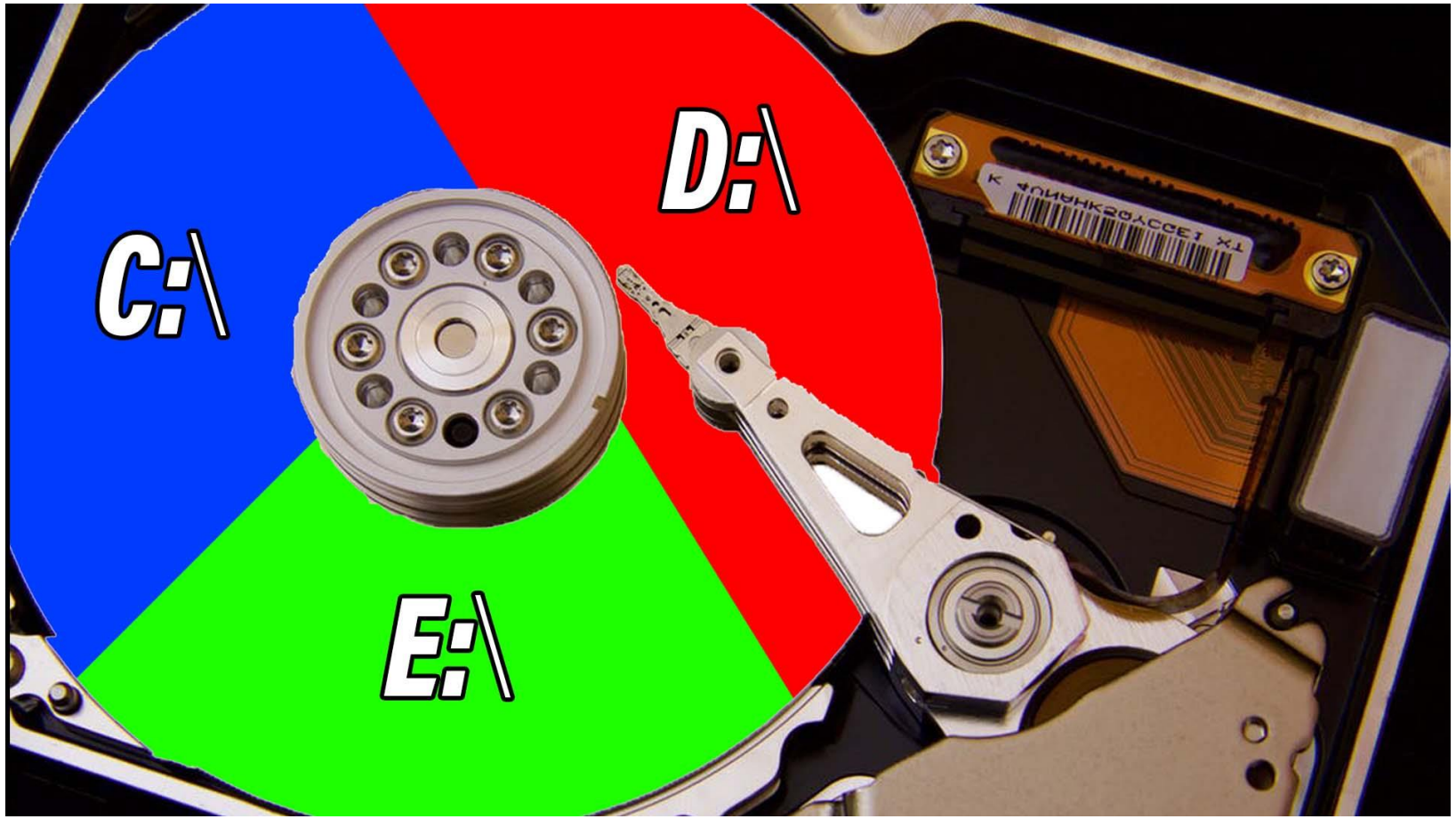
分区与目录结构

Partition and Directory Structure

03

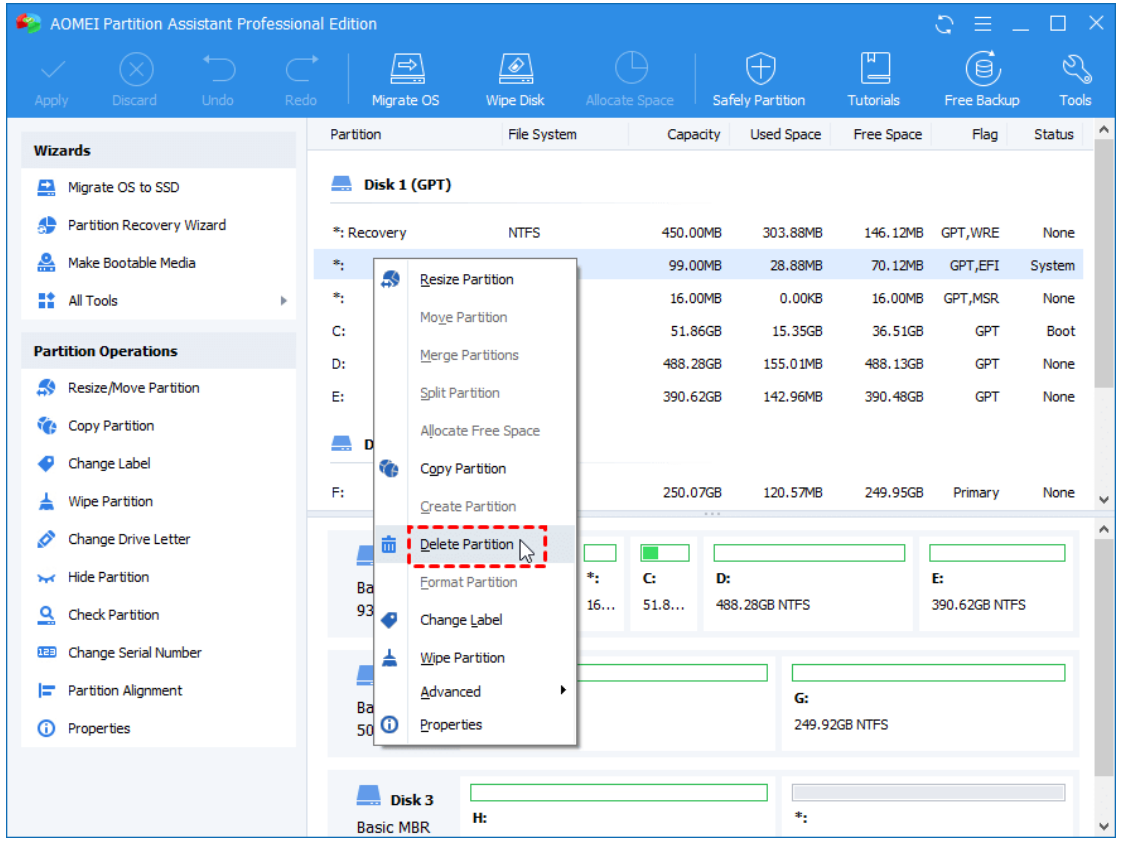
分区概念

- **磁盘分区**
  - 系统中的硬盘通常会被划分成多个分区



分区概念

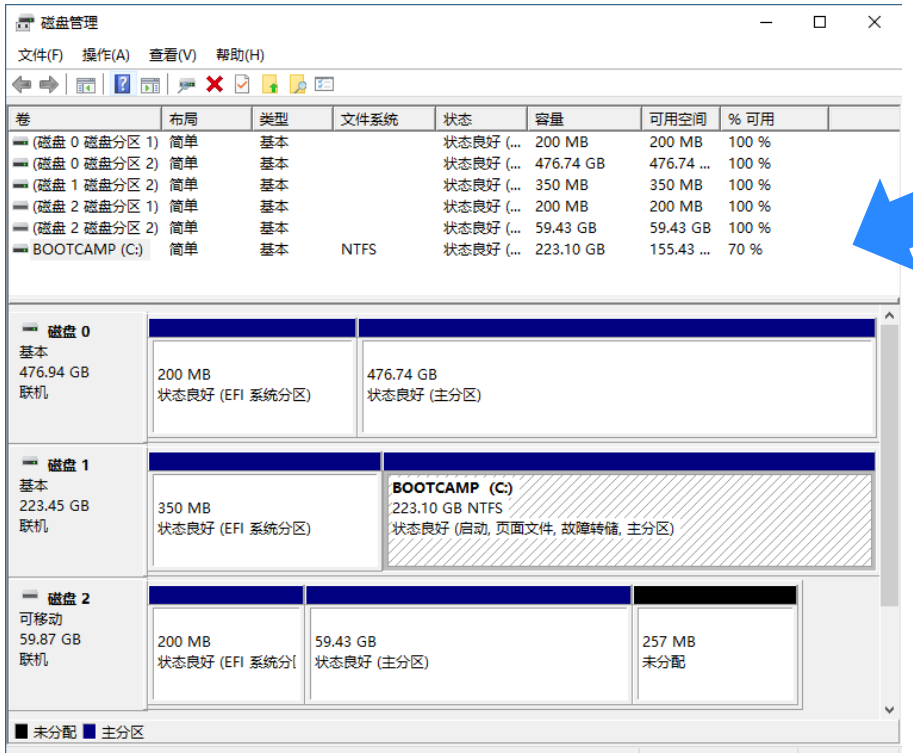
- **Partitions**
  - You can not just start writing files to a blank drive.
  - You must first create at least one container with a file system. We call this container a partition.



Windows系统分区（C:）　　NTFS

分区结构

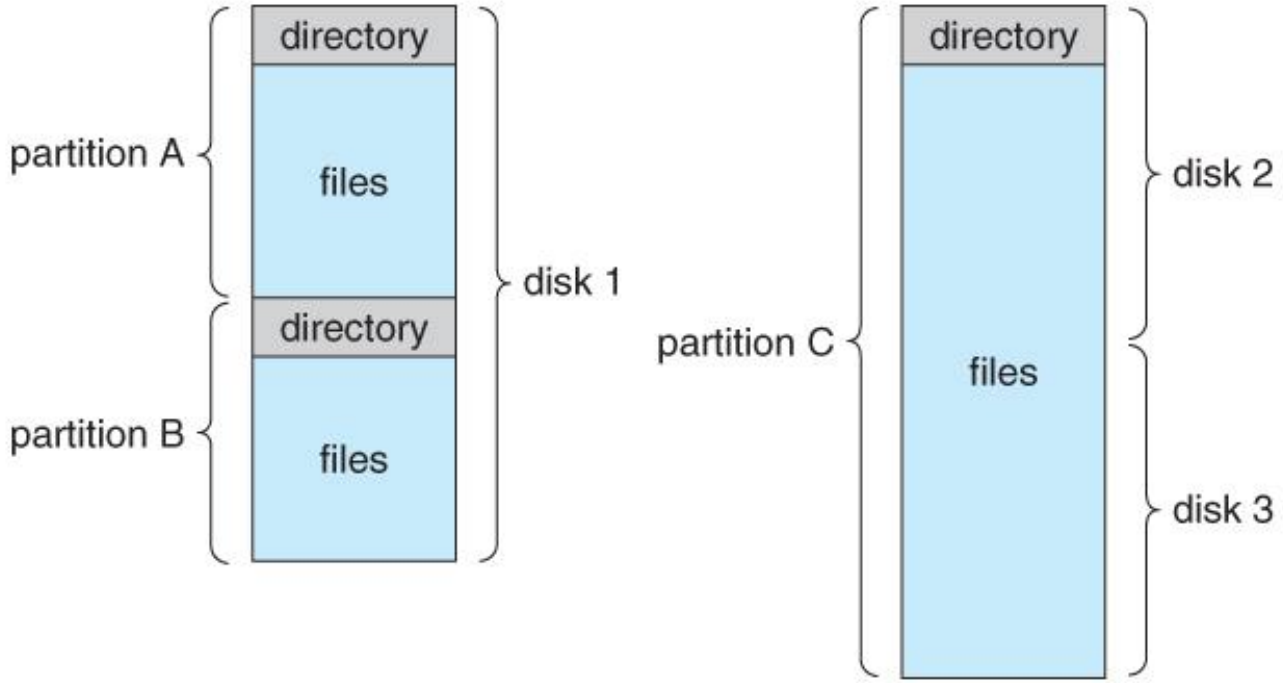
- **磁盘分区**
  - 每个分区使用前格式化为某一类型的文件系统



Win10
分区示例

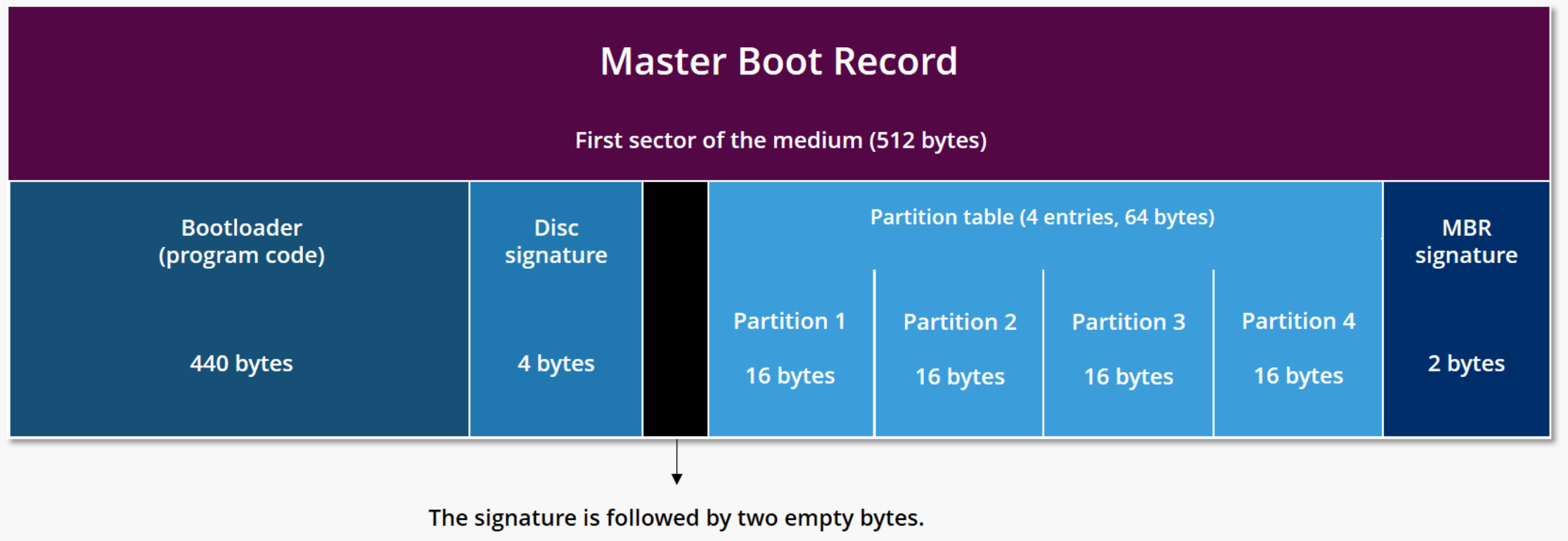
分区结构

- **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）



分区结构: MBR v.s. GPT

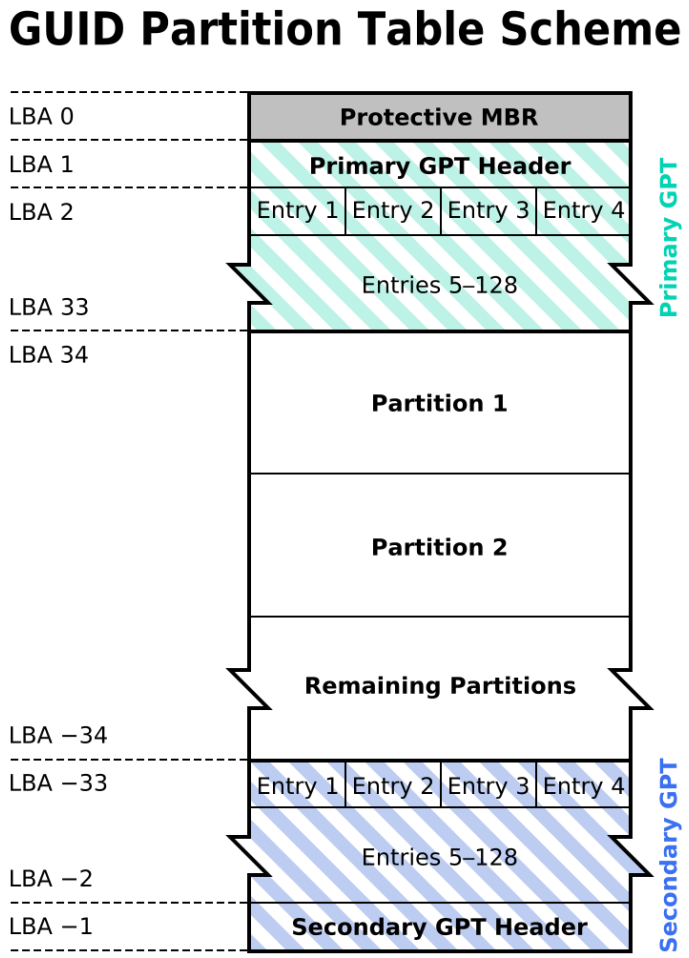
- **MBR （Master Boot Record）**



MBR only allows for four primary partitions per drive and doesn’ t support drives larger than 2 TB.

分区结构: MBR v.s. GPT

- **GPT**



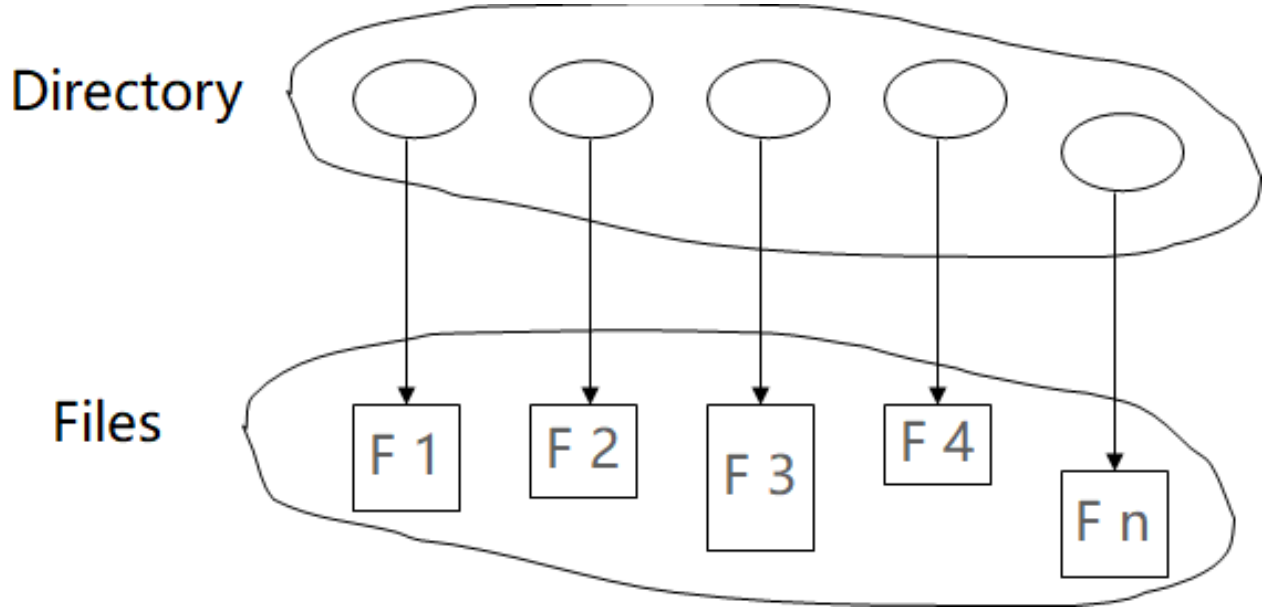
GPT allows you to create hundreds of partitions per drive and supports drives larger than one billion terabytes.

分区上可以承载的文件系统

- 众多文件系统
  - FAT文件系统
  - NTFS文件系统
  - EXT文件系统
  - …

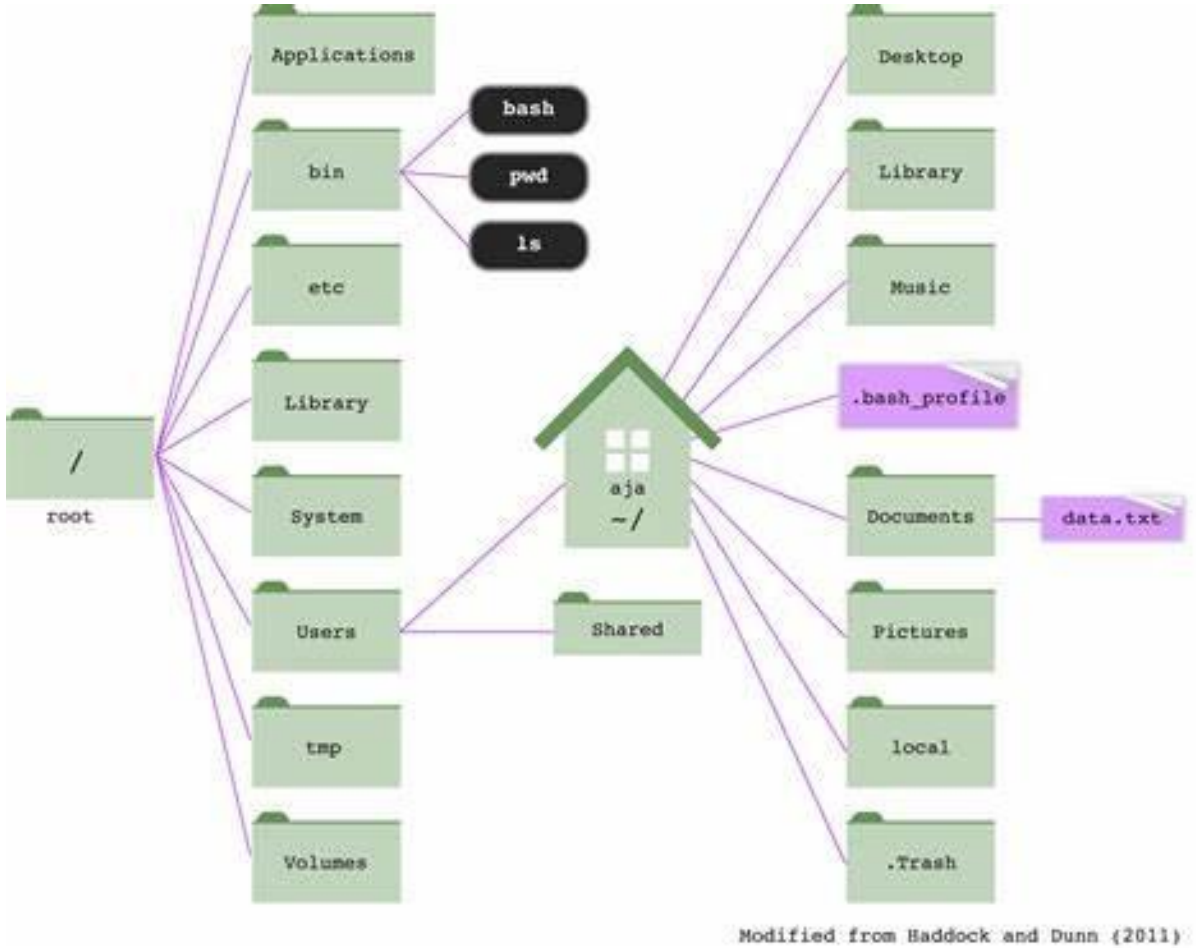
目录结构

- **文件系统**
  - **从逻辑上，文件系统由一组目录和文件组成**



目录结构

- **按名访问**
  - **在文件系统中，每个目录和文件都有名字**
  - **用户通过目录名和文件名访问目录或文件**



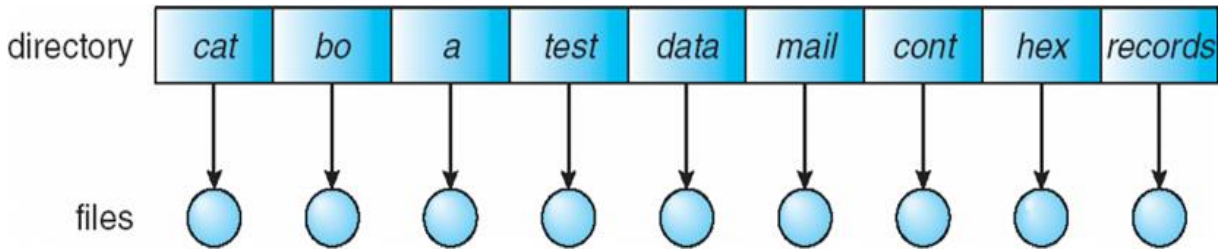
典型目录结构

Typical Directory Structures

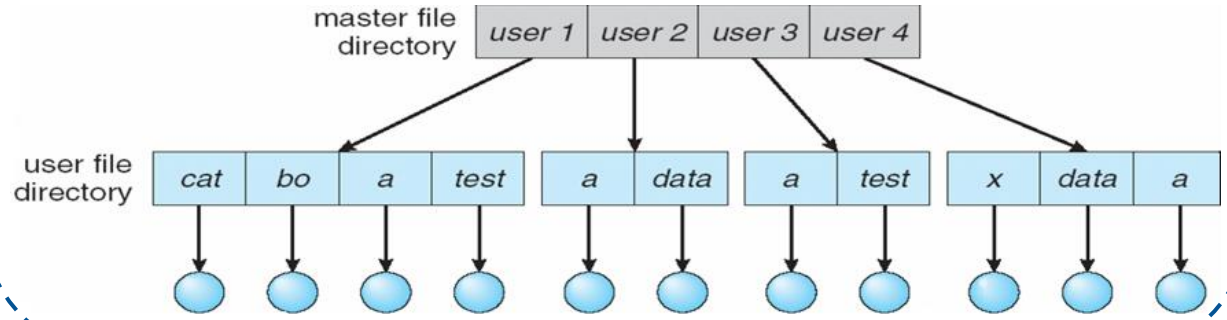
04

目录结构

- **Single Level Directory**
  - **所有文件存放在单一目录内**

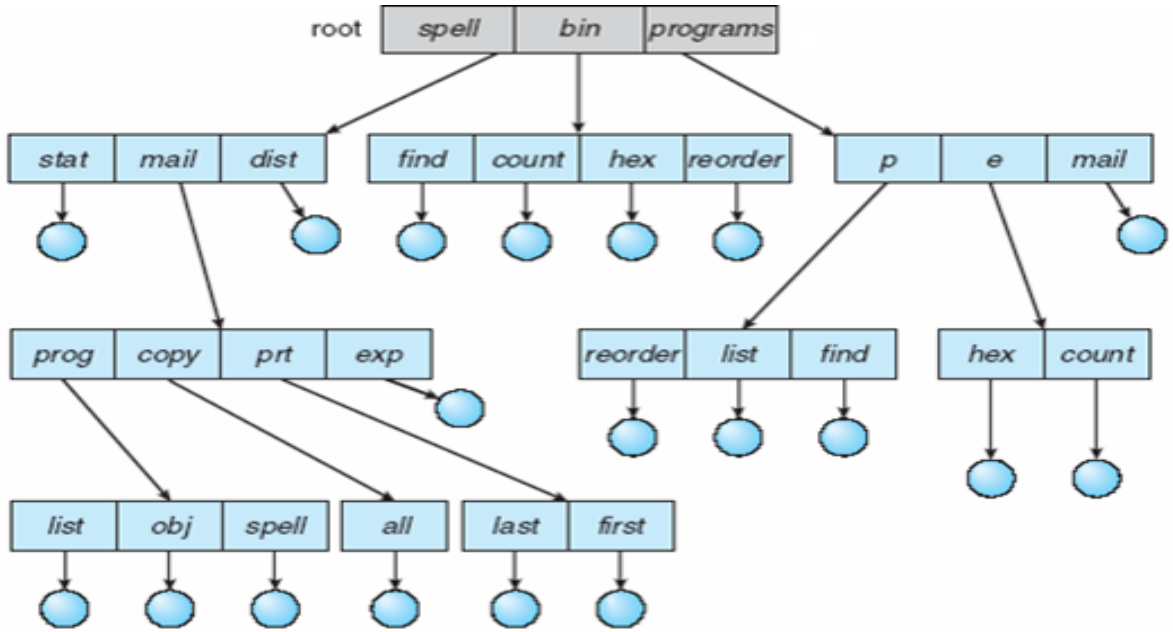


- **Two-Level Directory**
  - 每个用户的文件放在单个目录下
  - 一个主目录包含所有用户目录

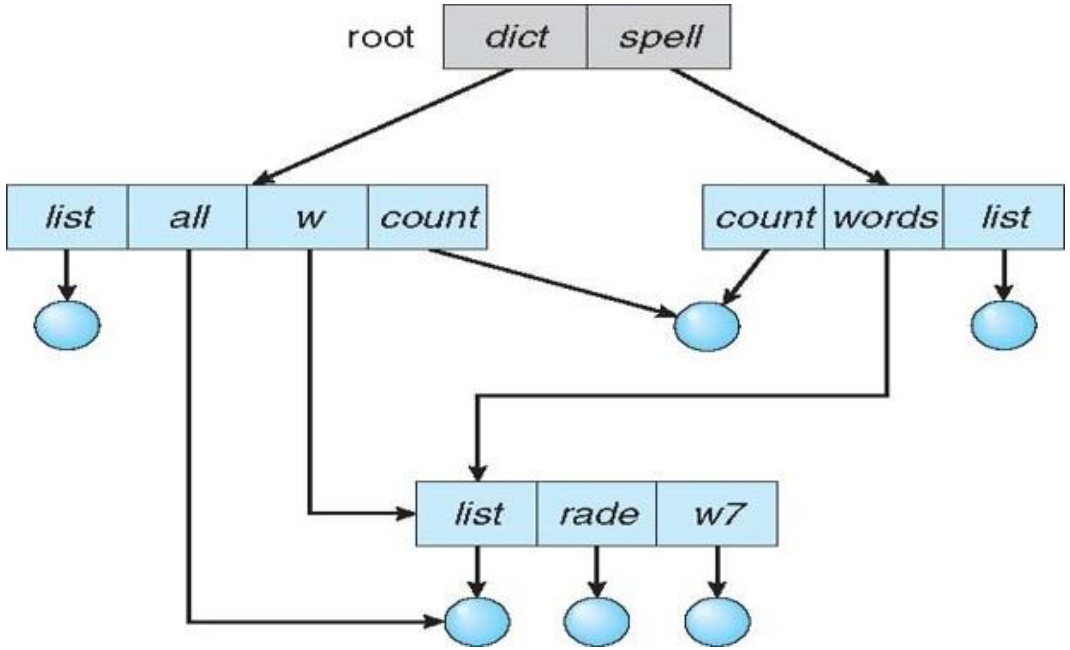


目录结构

- **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

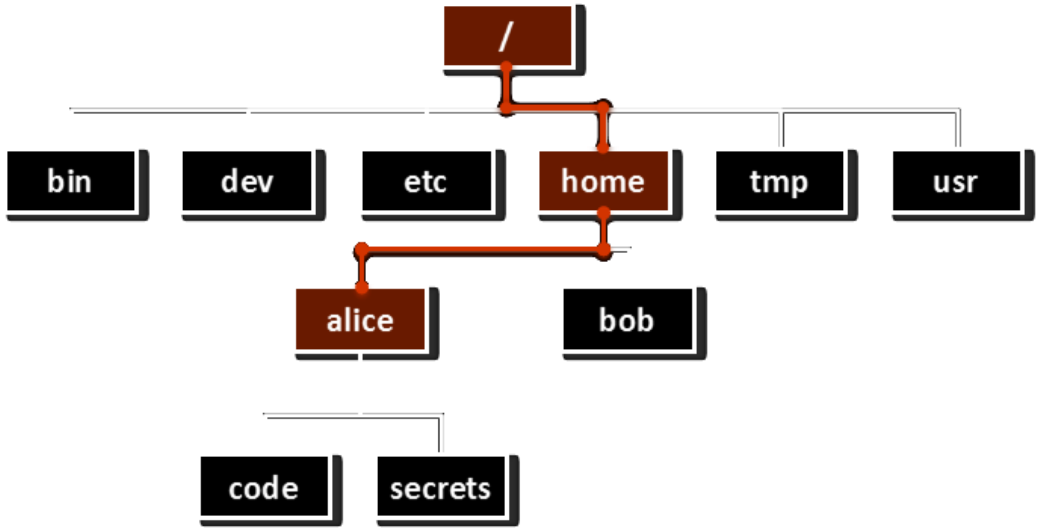


- **Acyclic-Graph Directory**
  - 不同目录中可能存在指向同一文件的目录项
  - 但整体目录结构中不允许存在环路



目录结构

- **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

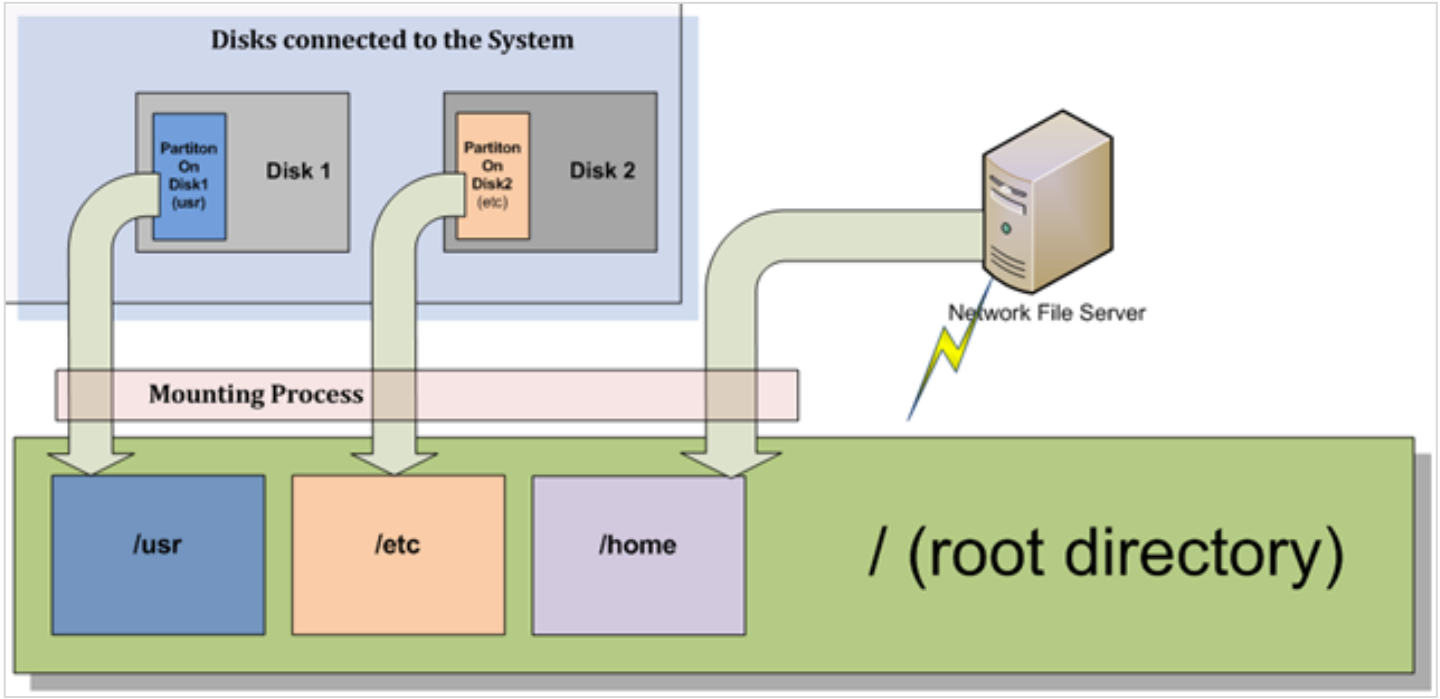
FS加载与保护

File System mounting and protection

05

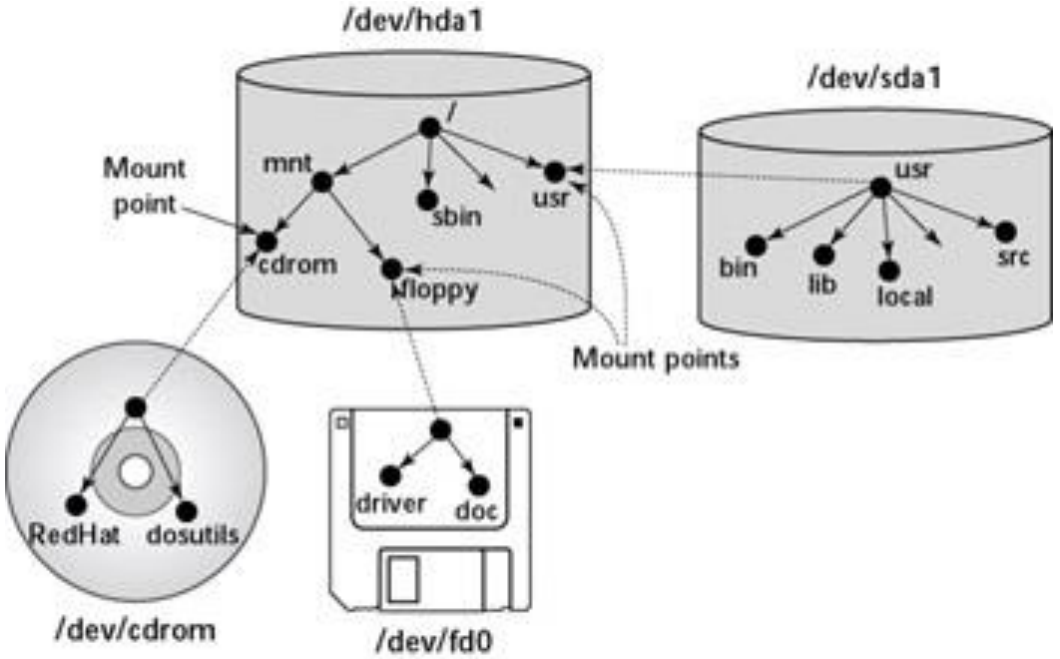
文件系统加载

- 文件系统加载是文件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



文件系统加载

- 文件系统加载操作
  - 加载点：mount point
  - /mnt/cdrom
  - /mnt/floppy
  - …



文件保护

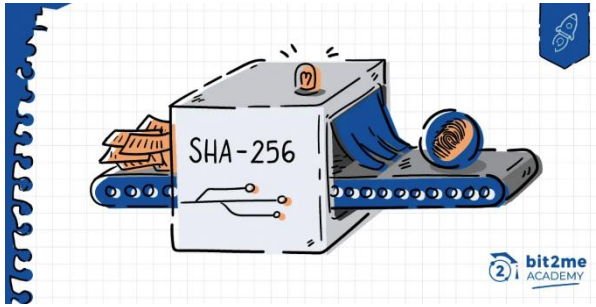
- 文件是信息的重要载体
  - 在信息互联的年代，要求计算机用户了解文件保护机制，增强数据安全意识，避免信息泄漏
  - 操作系统通常为用户提供多种文件保护的机制，保证系统和用户隐私安全

- 文件保护的目的
  - 避免文件被用户或程序错误或恶意使用
  - 确保共享资源的使用符合系统资源保护策略
  - 保证错误的使用对系统伤害最小化
  - 注意：保护机制的作用是为系统提供资源保护机制，有效使用这些策略对文件资源实施保护，还要依赖系统管理员的规范操作

文件保护

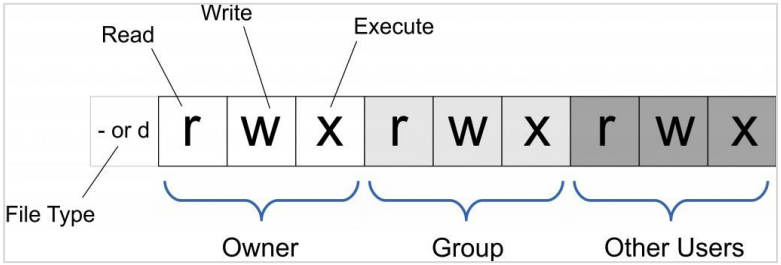
文件完整性验证

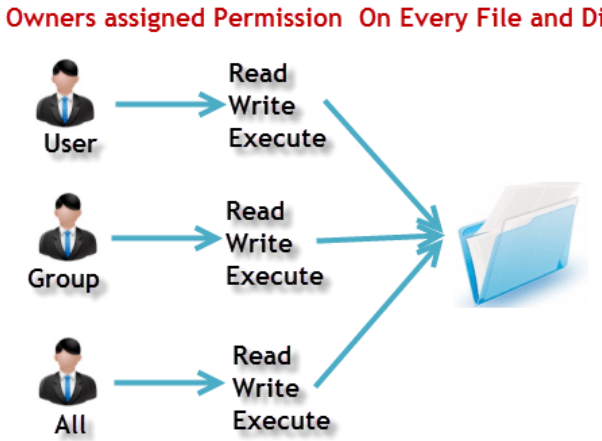
- 使用哈希（Hashing）机制
- 典型的方法有：MD5、SHA-256



Linux文件访问权限控制

- 文件操作权限类型：r， w， x
- 3类文件属主：user, group, all





Summary

小结：

- 文件概念
- 文件操作
- 分区与目录结构
- 典型目录结构
- 文件系统加载与保护

Open操作

File open/close

还能回忆起用标准C库中的fopen()和fclose()打开和关闭文件吗?

主观题 10分

设置

请手写代码，用标准C中的文件操作接口，实现打开一个文件，向文件中写入 Hello World, greeting from <your name, student NO.>. 将书写的代码截屏提交。

作答

Open操作

File open/close

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    // pointer demo to FILE
    FILE* demo;

    // Creates a file "demo_file"
    // with file access as write-plus mode
    demo = fopen("demo_file.txt", "w+");
    // adds content to the file
    fprintf(demo, "%s %s %s", "Welcome", "to", “OS class");
    // closes the file pointed by demo
    fclose(demo);

    return 0;
}
```

```
root@a88ea803f069:~/app/oslab/misc/c-file-io# make
gcc -o fopen_demo_01 fopen_demo_01.c
root@a88ea803f069:~/app/oslab/misc/c-file-io# ./fopen_demo_01
root@a88ea803f069:~/app/oslab/misc/c-file-io# cat demo_file.txt
Welcome to OS class
root@a88ea803f069:~/app/oslab/misc/c-file-io#
```

fseek操作

C文件的直接访问支持

```
#include <stdio.h>

int main()
{
    FILE* fp;
    fp = fopen("demo_file.txt", "r");

    // Moving pointer to the very beginning
    fseek(fp, 0, SEEK_SET);
    // Printing position of pointer
    printf("%ld\n", ftell(fp));
    char c;
    fread(&c, 1, 1, fp);
    printf("char at current pos: %c\n", c);

    return 0;
}
```

```
root@a88ea803f069:~/app/oslab/misc/c-file-io# make
gcc -o fseek_demo_02 fseek_demo_02.c
root@a88ea803f069:~/app/oslab/misc/c-file-io# ./fseek_demo_02
0
char at current pos: W
root@a88ea803f069:~/app/oslab/misc/c-file-io#
```

fseek操作

C文件的直接访问支持

```
root@a88ea803f069:~/app/oslab/misc/c-file-io# make
touch fseek_demo_03.c
gcc -o fseek_demo_03 fseek_demo_03.c
root@a88ea803f069:~/app/oslab/misc/c-file-io# cat demo_file.txt
Welcome to OS class
root@a88ea803f069:~/app/oslab/misc/c-file-io# ./fseek_demo_03
3
char at current pos: c
root@a88ea803f069:~/app/oslab/misc/c-file-io#
```

```c
#include <stdio.h>

int main()
{
    FILE* fp;
    fp = fopen("demo_file.txt", "r");

    // Moving pointer to the very beginning
    fseek(fp, 3, SEEK_SET);
    // Printing position of pointer
    printf("%ld\n", ftell(fp));
    char c;
    fread(&c, 1, 1, fp);
    printf("char at current pos: %c\n", c);

    return 0;
}
```

fseek操作

C文件的直接访问支持

```c
#include <stdio.h>

int main()
{
    FILE* fp;
    fp = fopen("demo_file.txt", "r");

    // Moving pointer to the very beginning
    fseek(fp, 0, SEEK_END);
    // Printing position of pointer
    printf("%ld\n", ftell(fp));
    char c;
    fread(&c, 1, 1, fp);
    printf("char at current pos: %c\n", c);

    return 0;
}
```

```
root@a88ea803f069:~/app/oslab/misc/c-file-io# make
touch fseek_demo_04.c
gcc -o fseek_demo_04 fseek_demo_04.c
root@a88ea803f069:~/app/oslab/misc/c-file-io# ./fseek_demo_04
20
char at current pos:
root@a88ea803f069:~/app/oslab/misc/c-file-io#
```

fseek操作

C文件的直接访问支持

```
root@a88ea803f069:~/app/oslab/misc/c-file-io# cat demo_file.txt
Welcome to OS class
```

```c
#include <stdio.h>

int main()
{
    FILE* fp;
    fp = fopen("demo_file.txt", "r");

    // Moving pointer to the very beginning
    fseek(fp, -2, SEEK_END);
    // Printing position of pointer
    printf("%ld\n", ftell(fp));
    char c;
    fread(&c, 1, 1, fp);
    printf("char at current pos: %c\n", c);

    return 0;
}
```

本代码的输出应该是?

fseek操作

C文件的直接访问支持

```
#include <stdio.h>

int main()
{
    FILE* fp;
    fp = fopen("demo_file.txt", "r");

    // Moving pointer to the very beginning
    fseek(fp, -2, SEEK_END);
    // Printing position of pointer
    printf("%ld\n", ftell(fp));
    char c;
    fread(&c, 1, 1, fp);
    printf("char at current pos: %c\n", c);

    return 0;
}
```

```
root@a88ea803f069:~/app/oslab/misc/c-file-io# make
touch fseek_demo_05.c
gcc -o fseek_demo_05 fseek_demo_05.c
root@a88ea803f069:~/app/oslab/misc/c-file-io# ./fseek_demo_05
18
char at current pos: s
root@a88ea803f069:~/app/oslab/misc/c-file-io#
```

rewind操作

rewind示例

```c
#include <stdio.h>

int main()
{
    FILE* fp;
    fp = fopen("demo_file.txt", "r");

    // Moving pointer to the very beginning
    fseek(fp, -2, SEEK_END);
    // Printing position of pointer
    printf("%ld\n", ftell(fp));

    rewind(fp);
    printf("%ld\n", ftell(fp));

    return 0;
}
```

```
root@a88ea803f069:~/app/oslab/misc/c-file-io# make
touch rewind_demo_01.c
gcc -o rewind_demo_01 rewind_demo_01.c
root@a88ea803f069:~/app/oslab/misc/c-file-io# ./rewind_demo_01
18
0
root@a88ea803f069:~/app/oslab/misc/c-file-io#
```

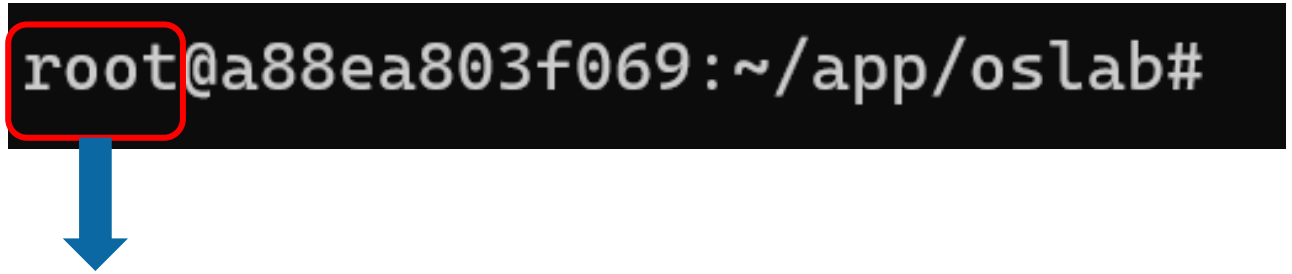
命令提示符

Linux中shell的命令提示符

```
root@a88ea803f069:~/app/oslab#
```

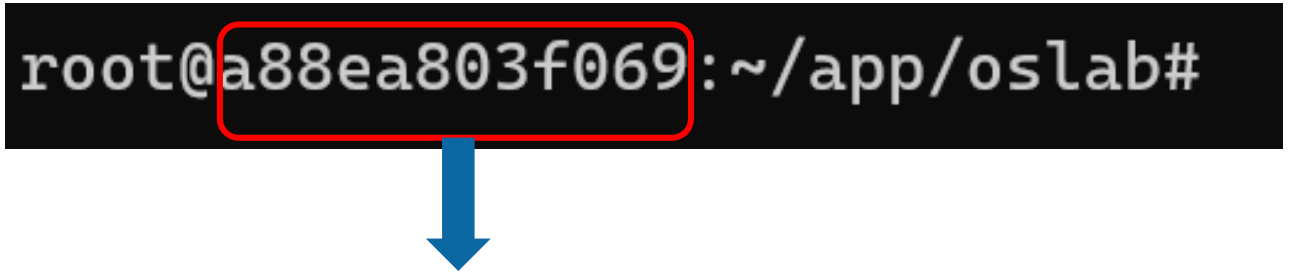
命令提示符

Linux中shell的命令提示符



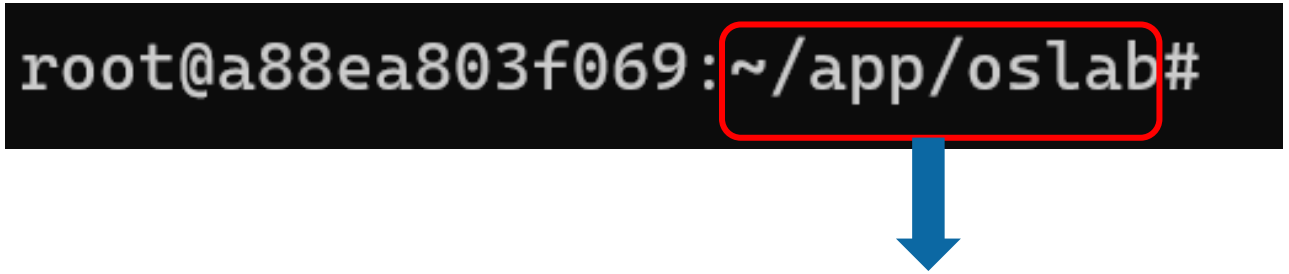
命令提示符

Linux中shell的命令提示符



命令提示符

Linux中shell的命令提示符



当前工作目录

命令提示符

Linux中shell的命令提示符

```
root@a88ea803f069:~/app/oslab#
```

当前工作目录

~ 表示 用户的home目录

~/app/oslab 是一个相对路径

Understanding Linux / UNIX Relative Pathname

1. In Linux, macOS, *BSD and Unix-like systems, a relative pathname is a path that specifies the location of a file or directory relative to the current working directory.
2. It doesn't begin with a slash (`/`), which signifies the root directory.
3. The relative path begins with a dot (`.`) or two dots (`..`) or tilde (`~`), which represent the current directory or the parent directory, and home directory respectively.

pwd命令

pwd命令：输出当前工作目录的绝对路径

```
root@a88ea803f069:~/app/oslab# pwd
/root/app/oslab
root@a88ea803f069:~/app/oslab#
```

cd命令：切换工作目录

```
root@a88ea803f069:~/app/oslab# cd ~
root@a88ea803f069:~# pwd
/root
root@a88ea803f069:~# cd app/oslab
root@a88ea803f069:~/app/oslab# pwd
/root/app/oslab
root@a88ea803f069:~/app/oslab#
```

mkdir命令：创建新目录

```
root@a88ea803f069:~/app/oslab# ls
lab4  misc
root@a88ea803f069:~/app/oslab# mkdir lec18
root@a88ea803f069:~/app/oslab# ls
lab4  lec18  misc
root@a88ea803f069:~/app/oslab#
```

```
root@a88ea803f069:~/app/oslab# mkdir /root/app/oslab/lec19
root@a88ea803f069:~/app/oslab# ls
lab4  lec18  lec19  misc
root@a88ea803f069:~/app/oslab#
```

tree命令：显示目录结构

```
root@a88ea803f069:~/app/oslab# apt install tree
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  tree
0 upgraded, 1 newly installed, 0 to remove and 12 not upgraded.
Need to get 47.9 kB of archives.
```

```
root@a88ea803f069:~/app/oslab# tree -L 2
.
|-- lab4
|   |-- cunit-demo
|   |-- democode
|   `-- democode-with-multiple-def-error
|-- lec18
|-- lec19
`-- misc
    `-- c-file-io

8 directories, 0 files
```

Linux文件权限信息

```
root@a88ea803f069:~/app/oslab/lab4/democode# ls -la
total 28
drwxr-xr-x 1 root root   512 May 27 19:51 .
drwxr-xr-x 1 root root   512 May 27 12:48 ..
-rw-r--r-- 1 root root   227 May 27 13:15 Makefile
-rw-r--r-- 1 root root   164 May 27 13:14 ipc.c
-rw-r--r-- 1 root root   315 May 27 13:12 ipc.h
-rw-r--r-- 1 root root  1336 May 27 13:14 ipc.o
-rwxr-xr-x 1 root root 15944 May 27 13:17 ipc_test
-rw-r--r-- 1 root root   176 May 27 19:51 ipc_test.c
-rw-r--r-- 1 root root  1232 May 27 13:16 ipc_test.o
-rw-r--r-- 1 root root    48 May 27 12:32 producer.c
-rw-r--r-- 1 root root  1232 May 27 13:12 producer.o
root@a88ea803f069:~/app/oslab/lab4/democode#
```

一些讨论

采用树状目录结构，相比于其他目录结构，有何好处?

一些讨论

采用树状目录结构，相比于其他目录结构，有何好处?

可以支持灵活的文件分组（grouping）

支持高效率的文件查找

一些讨论

绝对路径和相对路径，是否已清楚？

.

..

/home/alice/test.c

一些讨论

操作系统提供文件系统加载机制，为什么是必要的?

一些讨论

Linux下的文件权限设置与修改，可以用怎样的命令实现?

一些讨论

Linux下的文件权限设置与修改，可以用怎样的命令实现?

chmod

谢 谢!
Thank you!