



文件和文件系统

课程介绍

- 本章介绍了 Linux文件系统、文件、目录的基本概念以及如何管理文件系统及对文件及目录的操作
- 学完本课程后，您将能够：理解Linux文件系统的概念；管理自己的文件及目录

目录

CONTENT

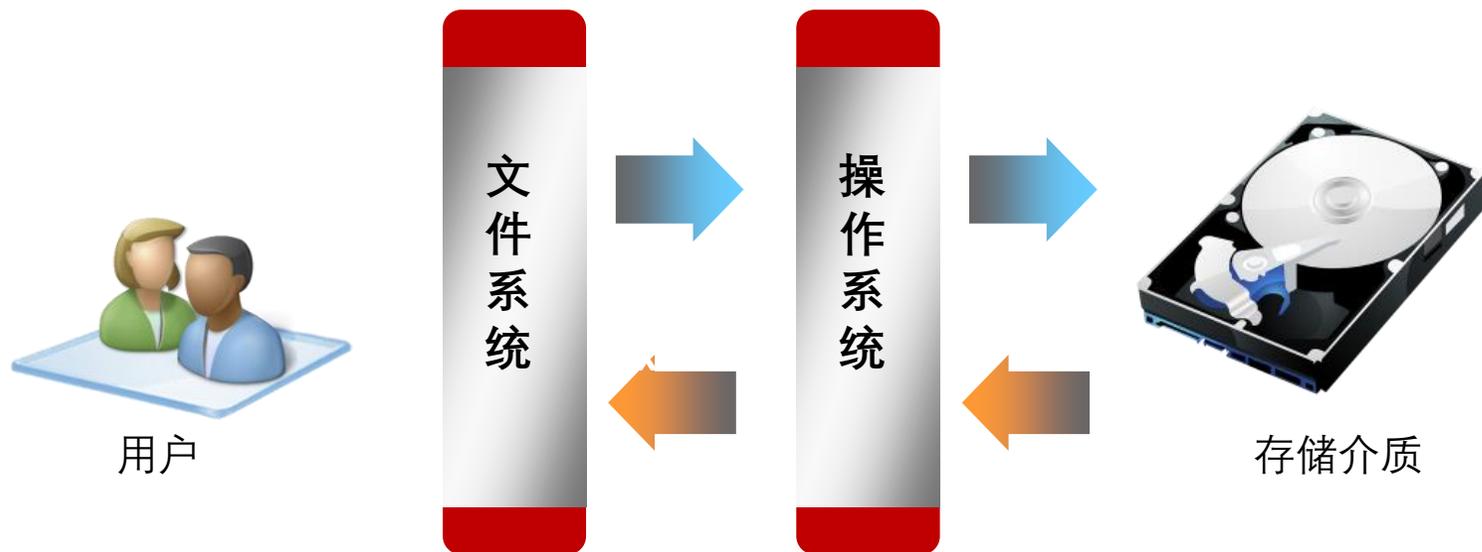
01 linux文件系统

02 管理linux文件系统

03 Linux文件和目录管理

文件系统的概念

文件系统是操作系统用于明确存储和组织计算机数据的方法，它使得对数据的访问和查找变得容易。



文件系统是对一个存储设备上的数据和元数据进行组织的机制。它的最终目的是把大量数据有组织的放入持久性(persistent)的存储设备中，比如硬盘和磁盘。文件系统(file system)是就是文件在逻辑上组织形式，它以一种更加清晰的方式来存放各个文件。

存储在介质中的数据有三个因素

文件名：在文件系统层面上，文件名是用于定位存储位置

数据：文件的具体内容，如txt文档中的内容

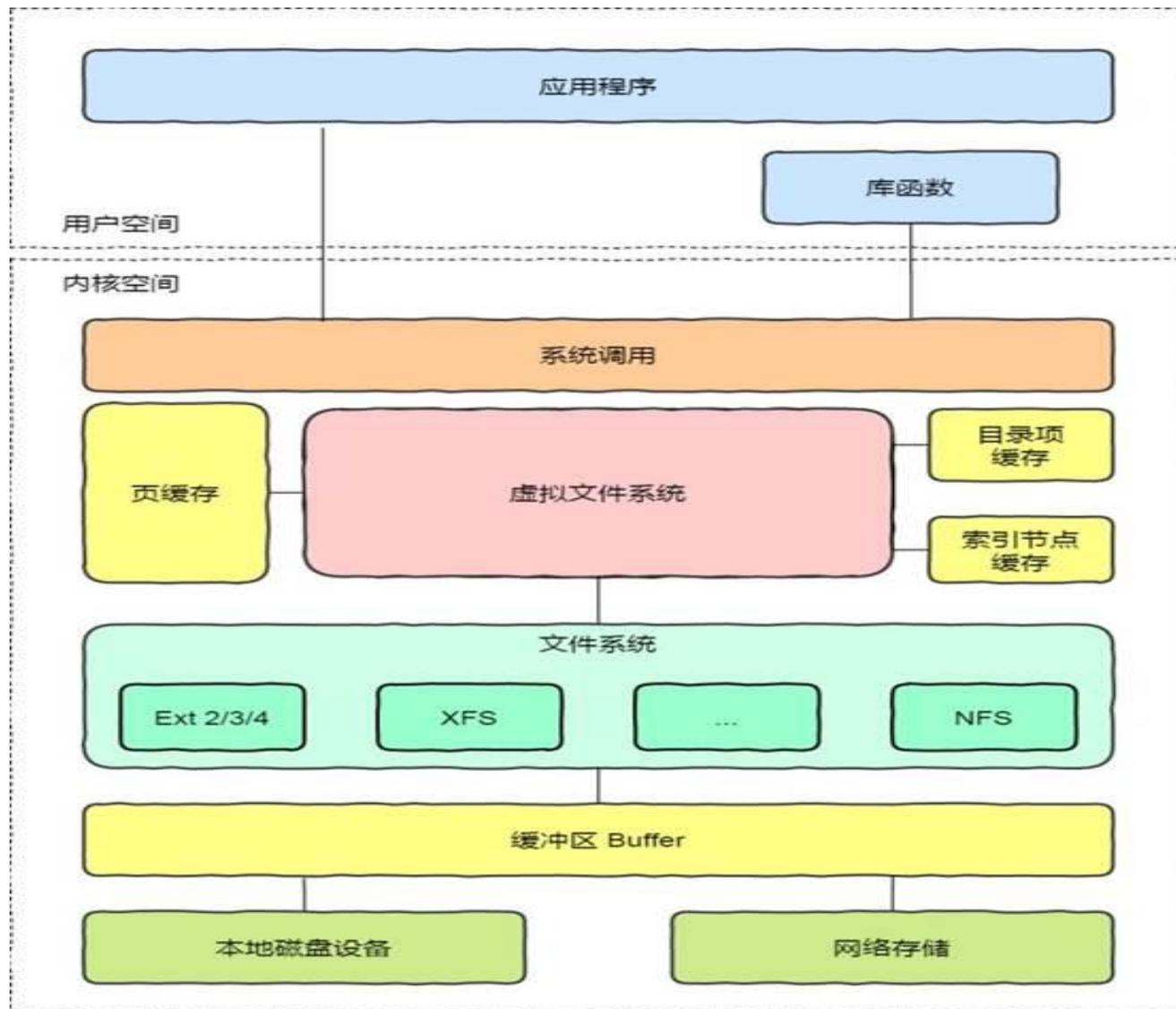
元数据 (meta-data)：文件有关的信息，如权限、所有者、修改时间等

文件系统的概念

文件系统的种类众多，而操作系统希望对用户提供一个统一的接口，于是在用户层与文件系统层引入了中间层，这个中间层就称为虚拟文件系统（Virtual File System, VFS）。

VFS 定义了一组所有文件系统都支持的数据结构和标准接口，这样程序员不需要了解文件系统的工作原理，只需要了解 VFS 提供的统一接口即可。

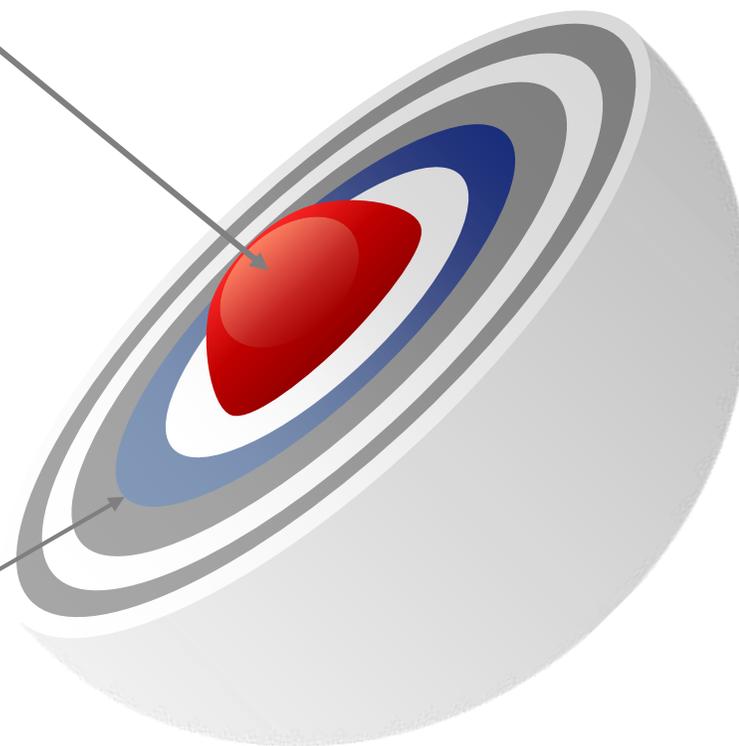
在 Linux 文件系统中，用户空间、系统调用、虚拟机文件系统、缓存、文件系统以及存储之间的关系如右图：



文件系统的分类

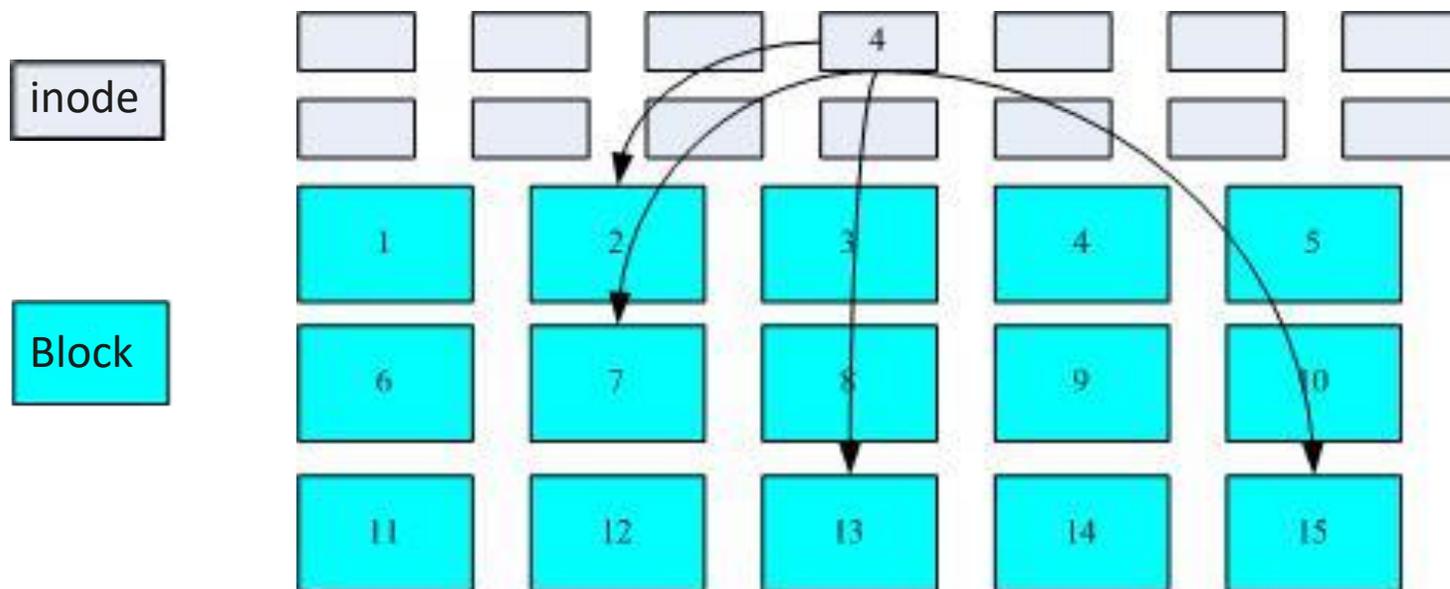
是否有日志 ?
传统型文件系统
日志型文件系统

如何查找数据 ?
索引式文件系统
非索引式文件系统



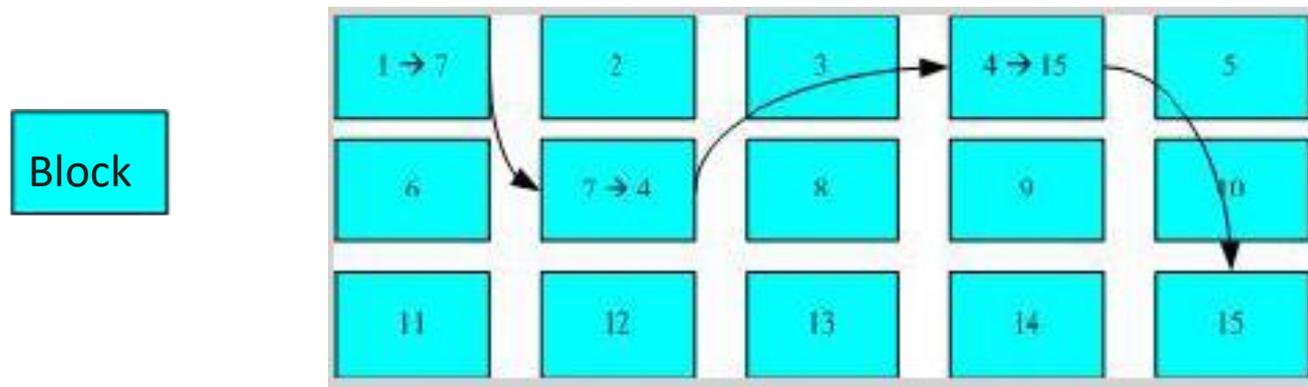
文件系统的分类

索引式文件系统：将文件属性数据和实际内容分别存放在不同的区块，通过属性数据，可以一下子找到实际数据所在。



文件系统的分类

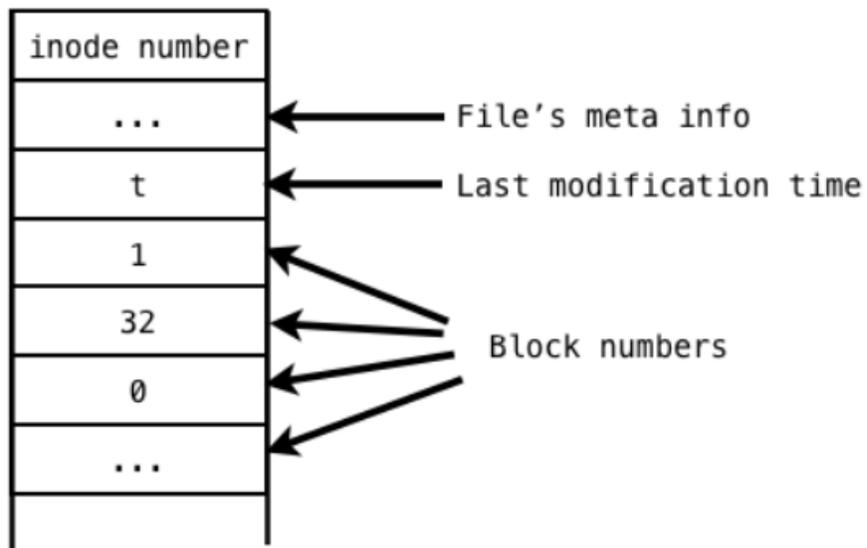
非索引式文件系统：只有block存在，读取数据时，需要一个block一个block读取，效率较低。



文件系统inode

inodes 是实现文件存储的关键。在 Linux 中，文件系统中管理的每个对象（文件或目录）表示为一个 inode。inode 包含管理文件系统中的对象所需的所有元数据（包括可以在对象上执行的操作）。在 Linux 系统中，一个文件可以分成几个数据块存储在分区内。为了搜集各数据块，我们需要该文件对应的inode。每个文件对应一个 inode。这个 inode 中包含多个指针，指向属于该文件各个数据块。当操作系统需要读取文件时，只需要找到对应 inode，收集分散的数据块，就可以收获我们的文件了。

inode structure



inode既可以表示普通文件，也可以表示目录，那么肯定要有方法来区分它到底是普通文件还是目录。这个就是通过inode中的数据块来区分的。

普通文件的inode的数据块是指向普通文件自己的数据的

目录的inode的数据块是指向位于该目录下的目录项的

读取文件

Unix/Linux系统中，目录（directory）也是一种文件。打开目录，实际上就是打开目录文件。目录文件的结构非常简单，就是一系列目录项（dirent）的列表。每个目录项，由两部分组成：所包含文件的文件名，以及该文件名对应的inode号码。

不管是普通文件还是目录文件，它总会存在于某个目录中，所有的文件都位于根目录 / 之下。我们平时使用的 ls 命令就是目录项的外在表现

在目录项中会记录该文件的类型，是属于普通文件，还是属于一个目录。

下面是目录项的一个示意图

567272	.	目录
171460	..	目录
567273	php	目录
567320	Php.ini	普通文件

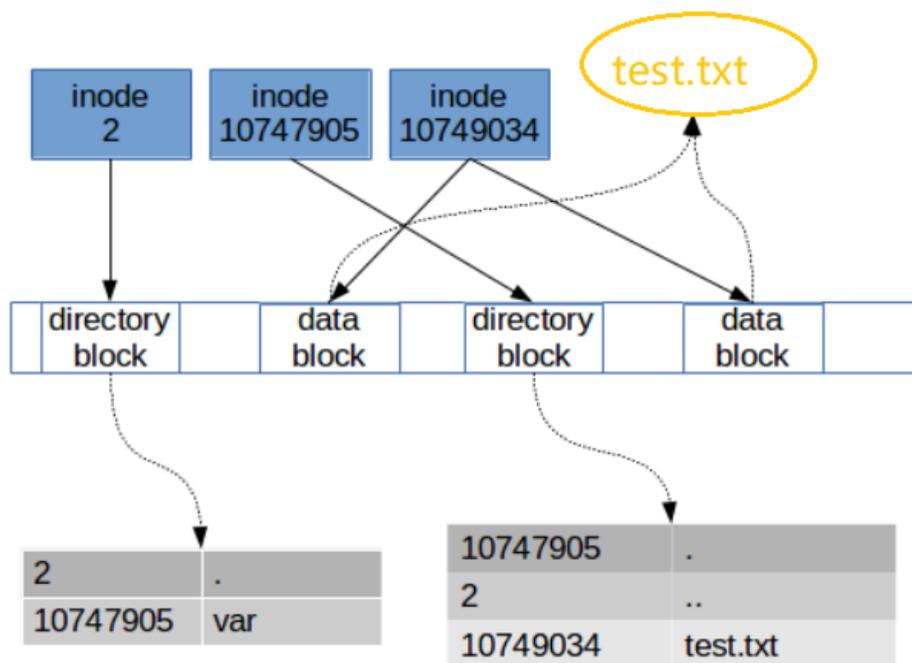
文件系统的概念

在Linux中，我们通过解析路径，根据沿途的目录文件来找到某个文件。目录中的条目除了所包含的文件名，还有对应的inode编号。当我们输入`$cat /var/test.txt`时，Linux将在根目录文件中找到`var`这个目录文件的inode编号，然后根据inode合成`var`的数据。随后，根据`var`中的记录，找到`test.txt`的inode编号，沿着inode中的指针，收集数据块，合成`test.txt`的数据。整个过程中，会参考三个inode：

根目录文件的 inode: 2, 用于找到 var 的 inode id

var 目录文件的 inode: 10747905, 用于找到 test.txt 的 inode id

test.txt 文件的 inode: 10749034, 用于找到 data blocks



因此，当我们读取一个文件时，实际上是在目录中找到了这个文件的inode编号，然后根据inode的指针，把数据块组合起来，放入内存供进一步的处理。当我们创建一个文件时，是分配一个空白 inode 给该文件，将其 inode 编号记入该文件所属的目录，然后选取空白的数据块，让 inode 的指针指向这些数据块，并放入内存中的数据。

获取inode中的信息

获取inode中的信息：stat 文件名，举例如下：

```
[root@localhost etc]# stat zabbix
File: zabbix
Size: 4096      Blocks: 8      IO Block: 4096  directory
Device: fd00h/64768d  Inode: 266226  Links: 3
Access: (0755/drwxr-xr-x)  Uid: (  0/   root)  Gid: (  0/   root)
Context: system_u:object_r:etc_t:s0
Access: 2022-05-26 15:27:54.345831480 +0800
Modify: 2022-02-16 15:42:49.396000000 +0800
Change: 2022-02-16 15:42:49.396000000 +0800
Birth: 2022-02-16 15:42:49.396000000 +0800
```

查看硬盘分区的inode信息

查看每个硬盘分区的inode总数和已经使用的数量： `df -i`，举例如下

```
[root@localhost ~]# df -i
Filesystem          Inodes IUsed   IFree IUse% Mounted on
devtmpfs            4171856  441  4171415  1% /dev
tmpfs               4176195   2  4176193  1% /dev/shm
tmpfs               819200  1672  817528  1% /run
tmpfs               1024    18    1006   2% /sys/fs/cgroup
/dev/mapper/openeuler-root 8519680 175020 8344660 3% /
tmpfs              409600   18  409582  1% /tmp
/dev/sdb2          65536   171   65365  1% /boot
/dev/sdb1          0        0     0    - /boot/efi
/dev/mapper/openeuler-home 1226926080 5062 1226921018 1% /home
tmpfs             13363824   20  13363804  1% /run/user/994
tmpfs             13363824   21  13363803  1% /run/user/0
```

查看文件名对应的inode号码

查看文件名对应的inode号码: ls -li 文件名, 举例如下

```
[root@localhost etc]# ls -li zabbix/  
266227 zabbix_agentd.d
```

Linux 文件系统类型-ReiserFS

ReiserFS 是一种文件系统格式。Linux内核从2.4.1版本开始支持ReiserFS。ReiserFS原先是Novell公司的SuSE Linux Enterprise采用的默认文件系统，直到2006年10月12日其宣称将在未来的版本改采 ext3 为默认。和同样在 Linux Kernel 2.4 版本下的 ext2 及 ext3 相比较，处理 4KB 以下的小文件时（tail packing enable），ReiserFS 的速度快了 10 到 15 倍[3]。但是，有些目录的操作在 ReiserFS 上面并不同步，（包括像 unlink(2)），可能会导致一些重度依赖文件锁（file-based lock）机制的应用程序上面数据的毁损。ReiserFS 在一个单一复合B+树中存储文件的亚数据信息（stat item）、目录文件信息（directory items）、索引节点中的块列表（indirect items），这些信息都有唯一的标识号作为B+树的索引值。

Linux 文件系统类型-ext

ext2 文件系统（也称为第二扩展文件系统）旨在克服早期 Linux 版本中使用的 Minix 文件系统的缺点。多年来，该文件系统一直广泛应用于 Linux。但 ext2 中没有日志，现在基本上已被 ext3 和最新的 ext4 所取代。

ext3 文件系统向标准 ext2 文件系统添加了日志功能，因此是一个非常稳定的文件系统的演化发展。它在大多数情况下提供合理的性能并且仍旧在改进。由于它在可靠的 ext2 文件系统上添加了日志功能，因此可以将现有 ext2 文件系统转换为 ext3 文件系统，并且在必要时还可以转换回来。

ext4 是作为 ext3 的扩展来启动的，它通过增加存储限制和提高性能来满足更大文件系统的需求。为了保留 ext3 的稳定性，在2006年6月，该扩展被拆分成一个新的文件系统，即 ext4。ext4 文件系统在 2008 年 12 月正式发布，包含在 2.6.28 内核中。

Linux 文件系统类型-其他

vfat 文件系统

vfat 文件系统（也称为 FAT32）没有日志功能，且缺乏完整的 Linux 文件系统实现所需的许多特性。它可用于在 Windows 和 Linux 系统之间交换数据，因为 Windows 和 Linux 都能读取它。不要将这个文件系统用于 Linux，除非要在 Windows 和 Linux 之间共享数据。

XFS 文件系统

XFS 文件系统拥有日志功能，包含一些健壮的特性，并针对可伸缩性进行了优化。XFS 通常是相当快的。在大文件操作方面，XFS 在所有测试中一直处于领先地位。XFS 的性能非常接近 ReiserFS，并在大多数测试指标上都超过了 ext3。

IBM JFS 文件系统

IBM 的 Journaled File System (JFS)，目前用于 IBM 企业服务器，专为高吞吐量服务器环境而设计。它可用于 Linux，包含在几个发行版中。要创建 JFS 文件系统，使用 `mkfs.jfs` 命令。

创建文件系统

Linux 使用 `mkfs` 命令来创建文件系统，使用 `mkswap` 命令创建交换空间。`mkfs` 命令实际上是几个特定文件系统的命令的前端，比如面向 `ext3` 的 `mkfs.ext3`，面向 `ext4` 的 `mkfs.ext4` 以及面向 `ReiserFS` 的 `mkfs.reiserfs`。你的文件系统上安装的是什么文件系统支持？使用 `ls /sbin/mk*` 命令即可得到答案或者 `mkfs`，敲 `tab` 键。

Linux系统磁盘分区、文件系统和目录

Linux系统磁盘分区、文件系统和目录的关系如下：

- 文件系统定义了磁盘上储存文件的方法和数据结构，磁盘上的每个分区都对应一个文件系统。
- 目录是逻辑上的区分，分区是物理上的区分。
- 磁盘Linux分区都必须挂载到目录树中的某个具体的目录上才能进行读写操作。
- 根目录是所有Linux的文件和目录所在的地方，需要挂载上一个磁盘分区。

Linux文件类型

Linux常见基本文件类型：

普通文件

语言元代码、SHELL脚本、二进制的可执行文件等。分为纯文本和二进制。

目录文件

目录，存储文件的唯一地方。

链接文件

指向某个文件或目录的文件。分为硬链接和软链接。

随堂测

- 1. 将分区/dev/hdb6格式化的命令是哪个？（单选题）
 - A. `mkfs -t ext4 /dev/hdb6`
 - B. `format -t ext4 /dev/hdb6`
 - C. `format /dev/hdb6`
 - D. `makefile -t ext4 /dev/hdb6`
- 2. Linux 使用 `mkfs` 命令来创建文件系统。（判断题）

目录

CONTENT

01 linux文件系统

02 管理linux文件系统

03 Linux文件和目录管理

查看分区使用情况

df命令查看文件系统的磁盘空间占用情况，主要参数：

- h：以容易理解的格式印出文件系统大小，例如136KB、24MB、21GB。
- i：显示inode信息而非块使用量。

```
[root@localhost etc]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        255G   0 255G  0% /dev
/dev/mapper/openeuler-root 128G  70G  52G  58% /
/dev/sdb2        976M 146M 763M 17% /boot
/dev/sdb1        599M  6.4M 593M  2% /boot/efi
/dev/mapper/openeuler-home 73T 308G 69T  1% /home
[root@localhost etc]# df -i
Filesystem      Inodes IUsed   IFree IUse% Mounted on
devtmpfs        4171856  441 4171415  1% /dev
/dev/mapper/openeuler-root 8519680 175025 8344655  3% /
/dev/sdb2        65536  171  65365  1% /boot
/dev/sdb1         0    0    0  - /boot/efi
/dev/mapper/openeuler-home 1226926080 5062 1226921018  1% /home
```

查看分区使用情况 (续)

du命令查询文件或目录的磁盘使用空间，主要参数

- a: 显示全部目录和子目录下的每个档案所占的磁盘空间
- s: 只显示大小的总合 (summarize)
- h: 以容易理解的格式输出文件大小信息 (例如, 234M 、 2G)

```
[root@localhost etc]# du * -sh
12K  acpi
4.0K adjtime
4.0K aliases
20K  alsa
4.0K alternatives
4.0K anacrontab
88K  ansible
4.0K asound.conf
0    at.allow
```

查看打开的文件

lsof即可显示系统打开的文件，必须以 root 用户的身份运行它能够充分地发挥其功能。

直接运行lsof命令时默认将显示所有进程打开的所有文件。

```
MMSC_DOM:/home/smc # lsof /home/smc
COMMAND  PID USER  FD   TYPE DEVICE SIZE  NODE NAME
bash     9827 root   cwd   DIR   8,2  4096 2610172 /home/smc
lsof    11046 root   cwd   DIR   8,2  4096 2610172 /home/smc
lsof    11047 root   cwd   DIR   8,2  4096 2610172 /home/smc
MMSC_DOM:/home/smc # lsof -i:21
COMMAND  PID USER  FD   TYPE DEVICE SIZE  NODE NAME
xinetd   5041 root   5u   IPv4  13328          TCP *:ftp (LISTEN)
MMSC_DOM:/home/smc # lsof -u smc
COMMAND  PID USER  FD   TYPE DEVICE  SIZE  NODE NAME
clustermn 10552  smc   cwd   DIR    8,2    4096 2610176 /home/smc/bin
clustermn 10552  smc   rtd   DIR    8,2    4096      2 /
clustermn 10552  smc   txt   REG    8,2 1423660 2610181 /home/smc/bin/clustermng
```

修复文件系统

fsck命令检查文件系统并尝试修复错误

[语法] : fsck [-参数] 设备名称

```
MMSC_DOM:/home/smc # fsck -f /dev/sda6
fsck 1.38 (30-Jun-2005)
e2fsck 1.38 (30-Jun-2005)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sda6: 12/22088 files (0.0% non-contiguous), 6924/88324 blocks
MMSC_DOM:/home/smc # █
```

修复文件系统（续）

e2fsck可检查和修复ext2和ext3文件系统

文件系统损坏可能是由于superblock损坏导致，dumpe2fs命令可以查看superblock的备份位置。

例如

```
dumpe2fs /dev/sdb1 | grep superblock
```

可使用备份的superblock来恢复数据

例如

```
e2fsck -f -b 32768 /dev/sda6
```

设置文件系统自检周期

tune2fs可以调整和查看ext2/ext3文件系统的参数，自行定义自检周期及方式。

```
MMSC_DOM:/home/smc # tune2fs -l /dev/sda6 | grep check
Last checked:          Mon Dec 30 02:28:09 2013
Next check after:     Sat Jun 28 02:28:09 2014
MMSC_DOM:/home/smc # tune2fs -i 30d /dev/sda6
tune2fs 1.38 (30-Jun-2005)
Setting interval between checks to 2592000 seconds
MMSC_DOM:/home/smc # tune2fs -l /dev/sda6 | grep check
Last checked:          Mon Dec 30 02:28:09 2013
Next check after:     Wed Jan 29 02:28:09 2014
MMSC_DOM:/home/smc # tune2fs -i 6m /dev/sda6
tune2fs 1.38 (30-Jun-2005)
Setting interval between checks to 15552000 seconds
MMSC_DOM:/home/smc # tune2fs -l /dev/sda6 | grep check
Last checked:          Mon Dec 30 02:28:09 2013
Next check after:     Sat Jun 28 02:28:09 2014
```

创建文件系统

可在root权限下通过mkfs命令创建文件系统。

```
mkfs [option] lvname
```

其中：

option：命令参数选项。常用的参数选项有：

-t：指定创建的linux系统类型，如ext2，ext3，ext4等等，默认类型为ext2。

lvname：指定要创建的文件系统对应的逻辑卷设备文件名。

示例：在逻辑卷/dev/vg1/lv1上创建ext4文件系统。

```
# mkfs -t ext4 /dev/vg1/lv1
```

手动挂载文件系统

手动挂载的文件系统仅在当时有效，一旦操作系统重启则会不存在。

可在root权限下通过mount命令挂载文件系统。

```
mount lvname mntpath
```

其中：

lvname：指定要挂载文件系统的逻辑卷设备文件名。

mntpath：挂载路径。

示例：将逻辑卷/dev/vg1/lv1挂载到/mnt/data目录。

```
# mount /dev/vg1/lv1 /mnt/data
```

自动挂载文件系统

手动挂载的文件系统在操作系统重启之后会不存在，需要重新手动挂载文件系统。但若在手动挂载文件系统后在root权限下进行如下设置，可以实现操作系统重启后文件系统自动挂载文件系统。

- 执行blkid命令查询逻辑卷的UUID，逻辑卷以/dev/vg1/lv1为例。

```
# blkid /dev/vg1/lv1
```

查看打印信息，打印信息中包含如下内容，其中 uuidnumber 是一串数字，为UUID， fstype 为文件系统。

```
/dev/vg1/lv1: UUID=" uuidnumber " TYPE=" fstype "
```

- 执行vi /etc/fstab命令编辑fstab文件，并在最后加上如下内容。

```
UUID=uuidnumber mntpath          fstype defaults    0 0
```

fstab文件详解

fstab文件的作用

文件/etc/fstab存放的是系统中的文件系统信息。当正确的设置了该文件，则可以通过mount /directoryname命令来加载一个文件系统，每种文件系统都对应一个独立的行，每行中的字段都有空格或tab键分开。同时fsck、mount、umount的等命令都利用该程序。

fstab文件内容说明如下：

第一列：设备名或者设备卷标名或者逻辑卷UUID

第二列：文件系统的挂载目录 mntpath 。

第三列：文件系统的文件格式（ext3或者vfat等等）

第四列：挂载选项，默认“defaults”；

第五列：备份选项，设置为“1”时，系统自动对该文件系统进行备份；设置为“0”时，不进行备份。此处以“0”为例；

第六列：指明自检顺序。（0为不自检，1或者2为要自检，如果是根分区要设为1，其他分区只能是2）

随堂测

1.下面那个命令显示inode信息而非块使用量？（单选题）

- A. df -i
- B. df -h
- C. du -a
- D. du -h

2. lsof即可显示系统打开的文件。（判断题）

目录

CONTENT

01 linux文件系统

02 管理linux文件系统

03 Linux文件和目录管理

文件的概念

文件是具有符号名的，在逻辑上具有完整意义的一组相关信息项的序列。
文件名是由字母、数字和其他符号组成的一个字符串，其格式和长度因系统而异

文件名

文件命名一般包括文件名和扩展名：前者用于识别文件，后者用于标识文件特性，两者之间用圆点隔开

文件的分类

按用途可分成：系统文件、库文件、用户文件按保护级别可分成：只读文件、读写文件、不保护文件

按信息时限可分成：临时文件、永久文件、档案文件

按设备类型可分成：磁盘文件、磁带文件、光盘文件、软盘文件，

还可以按文件的逻辑结构或物理结构分类

引入文件的优点

用户使用方便
使用者无需记住信息存放在辅助存储器中的物理位置，也无需考虑如何将信息存放到存储介质上，只要知道文件名，给出有关操作系统要求便可存取信息，实现了“按名存取”

文件安全可靠
由于用户通过文件系统才能实现文件的访问，而文件系统能提供各种安全、保密和保护措施，故可防止对文件信息的有意或无意的破坏或窃用

文件可备份
可组织转储或备份，在文件使用过程中出现硬件故障时，文件系统可组织重执，提高可靠性

文件可共享
文件系统还能提供文件的共享功能，如不同的用户可以使用同名或异名的同一文件，提高了文件和文件空间的利用率

把数据组织成文件形式加以管理和控制是计算机数据管理的重大进展

绝对路径和相对路径

绝对路径：由根目录 (/) 开始写起的文件名或者目录名称，如/home/test

相对路径：相当于当前路径的文件名或者目录名称写法。

如../../mtserver

. 代表当前目录, ..代表上一级目录

```
[root@localhost test]# pwd
/home/test
[root@localhost test]# cd ../../
[root@localhost /]# pwd
/
[root@localhost /]# cd /home/test
[root@localhost test]# cd ../../etc
[root@localhost etc]# pwd
/etc
[root@localhost etc]#
```

显示当前工作目录

[语法]: pwd

[说明]: 本命令用于显示当前的工作目录 present working dir

[例子]:

```
[root@localhost test]# cd ../../etc  
[root@localhost etc]# pwd  
/etc
```

更改工作目录

[语法]: `cd [目录]`

[说明]: 本命令用于改变当前的工作目录, 无参数时使用环境变量`$HOME`作为其参数, `$HOME`一般为登录时进入的路径。其中路径包括绝对路径和相对路径。

[例子]:

```
[root@localhost ~]# cd /home
[root@localhost home]#
[root@localhost home]# cd /etc/sysconfig/network-scripts/
[root@localhost network-scripts]# pwd
/etc/sysconfig/network-scripts
```

查看文件或目录

[语法]: **ls** [-option] [目录或文件.....]

[说明]: ls命令列出指定目录下的文件, 缺省目录为当前目录, 缺省输出顺序为纵向按字符顺序排列。
常用的命令选项为:

- l: 以长格式列出目录下的文件
- a: 以短格式列出目录下的所有文件 (包含隐含文件)

[注] 选项可混合使用。

[例子]:

```
[root@localhost home]# ls -l
total 32
drwxr-xr-x 4 root root 4096 May 20 10:30 ansible_plug
-rw-r--r--. 1 root root  81 Jan 12 2010 index.html
-rw-r--r--. 1 root root  81 Jan 12 2010 index.html.1
drwx-----. 2 root root 16384 May  5 18:15 lost+found
-rw-r--r--. 1 root root  12 May 11 15:16 test.txt
```

```
[root@localhost home]# ls -a
. .. ansible_plug index.html index.html.1 lost+found test.txt
```

新建文件

[语法]: touch 文件名

[说明]: 本命令touch命令其功能是创建新的空文件, 改变已有文件的时间戳属性。

[例子]:

```
[root@localhost home]# touch test.txt
[root@localhost home]# ll
total 32K
drwxr-xr-x 4 root root 4.0K May 20 10:30 ansible_plug
-rw-r--r--. 1 root root 81 Jan 12 2010 index.html
-rw-r--r--. 1 root root 81 Jan 12 2010 index.html.1
drwx-----. 2 root root 16K May 5 18:15 lost+found
-rw-r--r--. 1 root root 12 May 31 14:32 test.txt
[root@localhost home]# touch test.txt
[root@localhost home]# ll
total 32K
drwxr-xr-x 4 root root 4.0K May 20 10:30 ansible_plug
-rw-r--r--. 1 root root 81 Jan 12 2010 index.html
-rw-r--r--. 1 root root 81 Jan 12 2010 index.html.1
drwx-----. 2 root root 16K May 5 18:15 lost+found
-rw-r--r--. 1 root root 12 May 31 14:37 test.txt
```

新增目录

[语法]: `mkdir [-m 模式] [-p] 目录名`

[说明]: 本命令用于建立目录, 目录的存取模式由掩码 (`umask`)决定, 要求对其父目录具有写权限

-m 按指定存取模式建立目录。

-p 建立目录时建立其所有不存在的父目录, 这样可一次建立多层目录

```
[root@localhost home]# mkdir temp
[root@localhost home]# ls
ansible_plug index.html index.html.1 lost+found temp test.txt
[root@localhost home]# mkdir -m 777 temp/abc
[root@localhost home]# cd temp/
[root@localhost temp]# ls
abc
[root@localhost temp]# mkdir -p a/b/c
[root@localhost temp]# ls
a abc
[root@localhost temp]# cd a
[root@localhost a]# ls
b
[root@localhost a]# cd b
[root@localhost b]# ls
c
[root@localhost b]#
```

复制文件或目录

[语法]: **cp** [option] 源文件或目录 目的文件或目录

[说明]: 本命令用于本地主机复制文件或目录, 要求对其父目录具有写权限

cp命令的重要参数如下:

- a : 复制文件或者目录, 并复制所有特性, 包括属主、所属群组和权限。如果是复制目录, 会递归复制目录中的子目录和文件。
- i : 如果和目的文件或者目录有冲突, 提示是否覆盖目的文件或者目录
- p : 此时cp除复制源文件的内容外, 还将把其修改时间和访问权限也复制到新文件中。
- r : 递归的复制目录和目录中的文件及子目录

```
[root@localhost home]# cp test.txt /opt/  
[root@localhost home]# cd /opt  
[root@localhost opt]# ls  
patch_workspace test.txt
```

复制文件或目录(续)

[语法]: scp [option] 源文件或目录 目的文件或目录

[说明]: 本命令用于网络互通的远程主机复制文件或目录, 要求对其父目录具有写权限

-r: 递归复制整个目录。

[例子]:

```
[root@localhost opt]# scp test.txt root@X.XX.XX.XX:/home
The authenticity of host 'X.XX.XX.XX (X.XX.XX.XX)' can't be established.
ECDSA key fingerprint is SHA256:ibqOuFnU5JeO8GbFx656Xuvm5R1HBosWk2rc0GYAcpY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'X.XX.XX.XX' (ECDSA) to the list of known hosts.

Authorized users only. All activities may be monitored and reported.
root@9.88.35.79's password:
test.txt
```

移动文件或目录

[语法]: mv [-fiu] 源文件或目录 目的路径

[说明]: 本命令用于移动文件或目录, 要求对其父目录具有写权限

[例子]:

```
[root@localhost opt]# ll
total 8.0K
drwxr-xr-x. 4 root root 4.0K May  5 18:19 patch_workspace
-rw-r--r--  1 root root  12 May 31 14:46 test.txt
[root@localhost opt]# mv test.txt /home/temp
[root@localhost opt]# cd /home/temp
[root@localhost temp]# ll
total 12K
drwxr-xr-x 3 root root 4.0K May 31 14:42 a
drwxrwxrwx 2 root root 4.0K May 31 14:42 abc
-rw-r--r--  1 root root  12 May 31 14:46 test.txt
[root@localhost temp]# ll /opt
total 4.0K
drwxr-xr-x. 4 root root 4.0K May  5 18:19 patch_workspace
```

删除目录

[语法]: `rmdir [-p] 目录名`

[说明]: 本命令用于建立目录, 要求对其父目录具有写权限
-p 删除目录和其父目录, 这样可一次删除多个目录

[例子]:

```
[root@localhost temp]# pwd
/home/temp
[root@localhost temp]# ls
a abc test.txt
[root@localhost temp]# rmdir -p a/b/c
[root@localhost temp]# ls
abc test.txt
```

`rmdir`只能删除空目录, 而且需要一层一层的删除, 如果都是子目录和父目录都是空, 可以加上-p参数来删除。
`rmdir -p a/b/c`相当于 `remdir a/b/c`, `rmdir a/b`, `rmdir a`。

删除文件或目录

[语法]: `rm [-fir] 文件或目录`

[说明]: 本命令用于删除文件或目录, 要求对其父目录具有写权限

[例子]:

rm常用参数如下:

- r : 递归删除目录和目录中的文件
- f : 强制删除, 忽略不存在的文件, 不会出现告警信息
- i : 互动模式, 删除前会询问用户是否确定此操作

```
[root@localhost home]# ll
total 36K
drwxr-xr-x  3 root root 4.0K May 31 15:00 temp
[root@localhost home]# pwd
/home
[root@localhost home]# rm -r temp
rm: descend into directory 'temp'? y
rm: remove regular file 'temp/test.txt'? y
rm: remove directory 'temp/abc'? y
rm: remove directory 'temp'? y
```

查找文件或目录路径

[语法]: find path [-option] [查找条件]

[说明]: 本命令用于在硬盘中查找文件或目录的路径 (速度较慢)

-name 根据文件名查找 (精确查找)

[例子]:

find / -name test.txt 从根目录下开始查找精确匹配test.txt名字的文件路径

find ./ -name "*test*" 从当前目录下开始查找包含test名字的所有文件和目录路径 (*通配任意字符)

```
[root@localhost home]# find / -name *.txt  
/home/test.txt
```

查看文件内容

常用的查看文件内容命令：

cat：直接查阅文件内容，不能翻页

more：翻页查看文件内容

less：翻页阅读，和more类似，但操作按键比more更弹性

head：查看文档的前面几行内容，默认10行

tail：查看文件的最后面几行内容，默认10行

```
[root@localhost home]# tail -5 test.txt
ppp
qqq
rrr
sss
:
[root@localhost home]# head -5 test.txt
xxx
ccc
aaa
bbb
ccc
```

查找文件内容

[语法]: `grep [-cin] '目标字符串' filename`

[说明]: `grep` (global search regular expression(RE) and print out the line,全面搜索正则表达式并把行打印出来)是一种强大的文本搜索工具, 它能使用正则表达式搜索文本, 并把匹配的行打印出来。

- c : 计算找到 '搜寻字符串' 的行数
- i : 忽略大小写的不同, 所以大小写视为相同
- n : 顺便输出行号
- l : 根据文件内容查找文件, 只显示包含该内容的文件名
- r : 根据文件内容递归查找文件, 并打印对应内容

```
[root@localhost home]# grep ccc test.txt  
ccc
```

查找文件内容 (续)

[例子]:

将test.txt文件中包含 ccc 的行取出

```
[root@localhost home]# grep -n ccc test.txt
5:ccc
[root@localhost home]# grep ccc test.txt
ccc
[root@localhost home]# cat test.txt |grep ccc
ccc
```

```
[root@localhost home]# grep -n ccc test.txt
5:ccc
```

根据文件内容递归查找目录

```
# grep 'energywise' *      #在当前目录搜索带'energywise'行的文件
# grep -l 'energywise' *   #在当前目录搜索带'energywise'行的文件,只显示文件名
# grep -r 'energywise' *   #在当前目录及其子目录下搜索'energywise'行的文件
```

管道命令

管道：将一个命令的输出连接到另一个命令的输入

符号：|

例如: `cat /etc/passwd | grep oracle` (通常与grep配合用于过滤查找)

```
MMSC_DOM:/home/smc # cat /etc/passwd | grep oracle
oracle:x:1048:120::/opt/oracle:/bin/bash
MMSC_DOM:/home/smc # cat /etc/passwd | grep smc
smc:x:1051:1001::/home/smc:/bin/csh
smcc21:x:1005:100::/home/smcc21:/bin/csh
smc21:x:1007:100::/home/smc21:/bin/csh
cubasmc:x:1009:100::/home/cubasmc:/bin/csh
MMSC_DOM:/home/smc # █
```



输出重定向

标准文件: stdin, stdout, stderr

对应的文件描述符为0,1,2

输出重定向: > (覆盖导入) , >> (从文件末尾导入)

输入重定向: <

例如:

ls -l > ls.out (将ls -l 命令重定向到文件ls.out中。)

find / -name filename 2> find.txt (将命令错误输出重定向到文件中。)

find / -name filename > find.txt (将命令正确输出重定向到文件中。)

find / -name filename &> find.txt (将命令所有输出重定向到文件中。)

随堂测

1.下面那个命令可以跨主机复制文件？（单选题）

A. cp

B. mv

C. scp

D. copy

2. 输出重定向可以在文件末尾增量导入。（判断题）

Thank you