**Linux 系统详解**

目录

[ 设备管理](#_Toc701646690_WPSOffice_Level1) [3](#_Toc701646690_WPSOffice_Level1)

[1. 设备文件（设备节点）](#_Toc1052552569_WPSOffice_Level1) [3](#_Toc1052552569_WPSOffice_Level1)

[2. sysfs设备路径](#_Toc620839754_WPSOffice_Level1) [4](#_Toc620839754_WPSOffice_Level1)

[3. dd 命令和设备](#_Toc16083606_WPSOffice_Level1) [4](#_Toc16083606_WPSOffice_Level1)

[4. 创建设备文件](#_Toc1415232976_WPSOffice_Level1) [5](#_Toc1415232976_WPSOffice_Level1)

[5. udev](#_Toc4730865_WPSOffice_Level1) [5](#_Toc4730865_WPSOffice_Level1)

[第一代静态管理系统：由mknode 命令创建，放在/dev 目录下](#_Toc1052552569_WPSOffice_Level2) [5](#_Toc1052552569_WPSOffice_Level2)

[第二代动态：基于内核的动态设备文件系统 devfs；（弊端很大）](#_Toc620839754_WPSOffice_Level2) [5](#_Toc620839754_WPSOffice_Level2)

[第三代：udev](#_Toc16083606_WPSOffice_Level2) [5](#_Toc16083606_WPSOffice_Level2)

[6. SCSI和Linux内核](#_Toc1273174356_WPSOffice_Level1) [6](#_Toc1273174356_WPSOffice_Level1)

[① 磁盘的框架](#_Toc1415232976_WPSOffice_Level2) [7](#_Toc1415232976_WPSOffice_Level2)

[② USB的框架](#_Toc4730865_WPSOffice_Level2) [7](#_Toc4730865_WPSOffice_Level2)

[③ 通用设备框架](#_Toc1273174356_WPSOffice_Level2) [8](#_Toc1273174356_WPSOffice_Level2)

# 设备管理

## 设备文件（设备节点）

设备文件存放在/dev目录下面，可以使用 ls /dev来查看

设备类型如下

|  |  |  |  |
| --- | --- | --- | --- |
| 设备类型 | 现实例子 | 备注 | 解释 |
| 块设备 | 磁盘 | 按固定块的大小读取数据 | sda是一个磁盘设备，是块设备的一种，我们可以很轻松将磁盘划分成数据区块，进程可以通过内核访问磁盘上的任意区块 |
| 字符设备 | 打印机 | 处理流数据只能对字符设备读取和输入字符，字符设备没有固定容量 | 内核在流数据送达设备进程后不会再备份和再次验证； |
| 管道设备 | 进程通信通道 | 和字符设备类似不同的是输入和输出端不是内核而是进程 |  |
| 套接字设备 | 是跨进程通信用到的特殊接口 | 套接字文件代表着UNIX域套接字，存放在/dev 目录之外的地方 |  |

设备名称总结

|  |  |  |
| --- | --- | --- |
| 设备名称 | /dev展示 | 解释 |
| 硬盘 | /dev/sda | sda代表整块硬盘 |
| /dev/sda1 | sda1表示硬盘的分区 |
| 音频设备 | /dev/audio |  |
| CD/DVD | /dev/sr\* | 从光盘上只能读取数据 |
| PATA 硬盘 | /dev/hd\* |  |
| 终端设备 | /dev/tty | cmd命令窗口 |

sd代表SCSI(Small Computer System Interface)disk;USB也是使用的SCSI协议

## sysfs设备路径

Sysfs 是Linux 2.6所提供的一种基于内存的虚拟文件系统。这个文件系统不仅可以把设备（devices）和驱动程序（drivers）的信息从内核输出到用户空间，也可以用来对设备和驱动程序做设置。

Linux 内核通过一个文件和目录系统提供sysfs界面，旨在基于硬件属性统一显示设备信息相关信息；设备以/sys/devices/为根路径；/dev/sda这个磁盘在sysfs中的路径是

/sys/devices/pci0000:00/0000:00:1f.2/host0/target0:0:0/0:0:0:0/block/sda

可以发现sysfs系统下的/sys/devices/,提供的设备信息,比/dev/提供的更详尽；/dev/下的文件是供用户进程访问使用的，/sys/devices/的设备文件是为供内核程序查看设备信息和管理设备用的；下面是查看设备信息的方法：

$udevadmin info --query=all --name=/dev/sda

## dd 命令和设备

dd命令对于块设备和字符设备非常有用，主要的功能是从输入文件或者输入流读取数据后写入输出文件和输出流；

$dd if=/dev/zero of=new\_file bs=1024 count=1

if ：输入文件 of:输出文件 bs:数据块大小，在输入和输出一样大小的情况下使用bs;

ibs 表示输入数据大小 obs表示输出数据大小；count 表示复制块数的大小，在处理大数据和无限流数据的时候，需要在某个时间停止dd复制，防止消耗大量的空间和cPu时间；

dd 命令非常强大，但也容易损坏设备文件，要小心使用；

## 创建设备文件

在现代的Linux系统中，你不需要创建自己的设备文件，这项工作是由devtmpfs和udev来完成的。

mknode命令可以用来创建设备。你必须知道设备名字以及主要和次要编号

$mknode /dev/sda1 b 8 2

mknode命令用来创建临时命名管道文件很方便，也可以用于系统恢复的时候创建丢失的设备文件。

## udev

设备的管理系统：

第一代静态管理系统：由mknode 命令创建，放在/dev 目录下

第二代动态：基于内核的动态设备文件系统 devfs；（弊端很大）

第三代：udev

Linux 系统内核发现新设备，会向用户空间进程发送消息（udeved)；用户进程会验证设备的属性，创建设备文件，执行初始化；

系统启动前期就需要设备文件，所以udevd这个系统服务必须优先启动，以免拖延整个系统；

devtmpfs文件系统，正是为了解决以上问题开发的；类似于老的devfs系统，但是更简单。内核需要创建设备文件，并且在新设备可用的时候通知udevd这个系统服务；udevd收到通知以后只是进行设备初始化和发送消息通知，并不创建设备文件；此外还在/dev/目录中创建设备符号链接文件；

udevd守护进程，这项服务是这样工作的，内核通过一个内部网络链接向udevd发送一个通知事件称为uevent;udevd服务加载该uevent的属性信息，udevd通过规则分析来决定执行那些操作和增加那些属性信息；

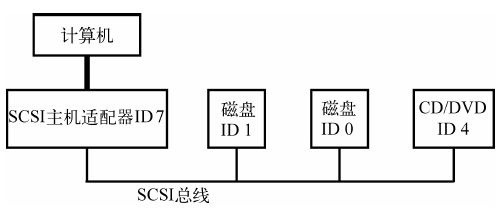
udevadm是udevd的管理工具，可以用来重新加载规则，触发消息。它的强大之处是在于搜寻和浏览系统设备以及监控udevd从内核接收的信息；

以下命令用于设备监控

$udevadm monior

## SCSI和Linux内核

SCSI硬件设置是通过SCSI总线链接设备到主机适配器，主机适配器和设备都有一个SCSI ID,每个总线有8到16个ID;



（0007.jpeg)

计算机必须通过主机设备和设备通信；SCSI命令集与设备进行一对一通信，设备向其发送相应信息；大部分计算机中并没有真正意义上的SCSI设备。更多是那些使用SCSI命令的USB设备，支持ATAPI的设备（也是用某个版本的SCSI命令集）

SATA 硬盘在系统中通常由一个处于libata层的转换机制呈现为SCSI设备；

$lsscsi

[0:0:0:0] disk ATA WDC WD3200AAJS-2 01.0 /dev/sda

[1:0:0:0] cd/dvd slimtype DVD A DS8A5SH XA15 /dev/sr0

[2:0:0:0] disk usb2.0 CardReader CF 0100 /dev/sdb

[2:0:0:1] disk usb2.0 CardReader SM XD 0100 /dev/sdc

[3:0:0:0] disk FLASH Drive UT\_USB20 0.00 /dev/sdf

方括号内的数字分别是SCSI主机适配器编号 SCSI总线编号，设备SCSI ID,设备的逻辑元件编号；上面的例子一共有四个SCSI主机适配器，每一个都有一个单独的总线（因此总线编号为0），每一个总线上有一个设备（设备SCSI ID为0））；对于编号为2的SCSI主机适配器，USB有两个逻辑单元，每一个单元代表可以插入一个闪盘；

SCSI系统的组成:

最顶层负责处理某一类设备。sd（SCSI硬盘）驱动负责将来自内核块设备接口的请求消息翻译为SCSI协议中硬盘的相关命令；

中间层在上线层之间调控和分流SCSI消息，负责管理所有的总线和设备

最底层负责处理硬件相关操作。该层中的 驱动程序向特定主机适配器发送scsi协议，并且从硬件提取发送过来的消息；

### 磁盘的框架

用户进程

内核

块设备接口 /dev/sda

scsi子系统

sd磁盘驱动程序

SCSI协议和主机管理

ATA桥

libata翻译器

将SCSI系统连接到ATA/SATA接口和设备

SATA主机驱动程序

硬件

SATA硬盘

ATA 在SATA(Serial ATA)之后改名为PATA(Parall ATA)

### USB的框架

用户进程

内核

SCSI子系统

CD/DVD驱动程序（sr)

SCSI协议和主机管理

USB存储桥

USB存储驱动程序

USB核心

USB主机驱动程序

硬件

USB闪存驱动

USB读卡器

### 通用设备框架

用户进程

内核

SCSI子系统

通用驱动程序

SCSI协议和主机管理

底层驱动程序

硬件

光驱软件

# 磁盘和文件系统

## 为磁盘设备分区

访问磁盘的流程

用户进程

Linux 内核

系统调用

文件系统

块设备接口和分区映射

scsi子系统和其他驱动程序

硬件

存储设备

分区表有两种一种叫主引导记录（Master Boot Record MBR),另一种是全局唯一标识符分区表（Globally Unique Identifier Partition Table GPT)；分区工具有一下几种：

parted：一个文本命令工具支持MBR和GPT

gparted： parted的图形版本

fdisk ：Linux传统的文本命令分区工具，不支持GPT;

gdisk ：fdisk的另一个版本，支持GPT,但不支持MBR

本书重点介绍parted,parted不要用于操作文件系统，分区和文件操作还是不同的；

$parted -l 查看系统分区表

MBR表包含主分区（primary），扩展区(extended)，逻辑分区(logical)

Linux内核在初始化读取MBR表的时候，会显示一下调试信息：

sda: sda1 sda2 <sda5>

/dev/sda2是一个扩展分区，包含一个逻辑分区sda5

fdisk工具和parted分区有很大的区别，fdisk 让你先设计好分区表，然后在退出fdisk之前才做实际的修改。parted则是在你运行命令的同时直接创建，更改和删除操作。二者都是在用户空间对分区做修改的，没必要为二者提供内核支持；内核必须确保读取分区表并将分区呈现为块设备。fdisk 是在更改分区以后向内核发送一个磁盘调用，告诉磁盘需要重新读取分区表。parted是在分区修改以后向内核发送信号；分区的修改可以使用一下方式查看：

$ udevadm monitor 查看内核消息关于分区变动的

/proc/partitions 查看完整的分区信息

/sys/block/device 查看更改分区后的系统接口信息

/dev/查看分区修改以后的分区设备

如果对磁盘分区不好，会影响磁盘的性能；

## 文件系统

文件系统通常是内核和用户空间之间联系的最后一环，文件系统是一个数据库，它将简单的块设备映射为用于易于理解的树状文件目录结构；Plan9这一特性使Linux上实现用户空间文件系统成为可能；虚拟文件系统（VFS)负责文件系统的具体实现，类似于SCSI系统将设备之间和设备与内核之间通信标准化一样，VFS为用户空间进程访问不同文件提供了标准接口，VFS使得LINUX可以支持不同的文件系统。

### 文件系统类型

ext4 Linux原生的文件系统

iso 9660 CD\_ROM使用该版本

FAT 文件系统（msdos vfat umsdos) 是微软的文件系统

HFS+ 苹果计算机的文件系统

创建文件系统

$mkfs -t ext4 /dev/sdf2

挂载文件系统

$mount

/dev/sda1 on / type ext4 (rw errors=remount-ro)

proc on /proc type proc

sysfs on /sys type sysfs

devpts on /dev/pts type devpts

以上命令可以查看当前文件系统的状态

$mount -t type /dev/sdf2 /home/extra

$unmount /home/xu

文件系统UUID

挂载系统可以使用设备名，但是设备名称会根据内核发现设备的顺序而改变，因此你可以使用文件系统的通用唯一标识来挂载；

$blkid 查看设备以及对应的文件系统以及UUID

$mount UUID=a9011c2b-1c03-4288-b3ef /home/xu

### 磁盘缓冲区 缓存和文件系统

Linux 将要写入磁盘的数据先写到缓存区，这意味着如果内核有修改请求不需要直接改写到文件系统，而是将更改保存到RAM中，直到内核可以便捷写到磁盘；这个缓存机制能够带来性能上的提升；

当你unmount文件系统的时候，内核自动和磁盘同步。你可以随时通过sync命令强制内核将缓存区数据写到磁盘；如果因为种种原因无法卸载文件系统，请务必运行sync命令。

内核有一系列的机制使用RAM自动缓存从磁盘读取的数据块，对于重复访问同一个文件的多个进程来说，内核不需要反复读取磁盘，只需要从缓存中读取；

### 文件系统挂载选项

-t 指定文件类型

-o norock 在ISO9660文件系统上关闭Rock Ridge扩展，但是对于其它文件类型无效

-r 以只读的形式挂载文件系统

-n 确保mount命令不会更新系统运行时候的挂载数据库/etc/mtab

### /etc/fstab

为了降低mount命令的使用和系统启动的时候挂载文件系统，linux 在/etc/fstab 中永久保存了文件系统和选项列表；

proc /proc nodev noexec nosuid 0 0

UUID=70ccd6e7-6ae4-44f6-812c-51aab830d39 / ext4 errors=remount-ro 0 1

UUID=59dcgdff-3kkdll-3445-890c-dddsevvbgv none swap sw 0 0

/dev/sr0 /cdrom iso9660 ro user nosuid noauto 0 0

每一行是6个字段分别是

设备或者UUID

挂载点

文件系统类型

选项

提供给dump命令使用的备份信息

文件系统完整性测试顺序：

为了确保fsck 总是第一个在root上运行，对root文件系 统总是设置问1，硬盘上的其他文件系统设置为2，使用0 表示禁止其他启动检查，包括CD\_ROM,交换分区，/proc文件系统；

$mount -a

以上命令，你可以挂载/etc/fstab 中所有未标识为noauto的设置

选项介绍

errors 是ext2的相关参数；errors=continue挂载发生错误继续运行；errors=re

mount-ro 以只读形式重新挂载；errors=panic发生错误停止挂载；

noauto 该选项让mount -a 命令忽略本行设备；使用该选项可以防止系统启动的时 候挂载可移动设备例如CD\_ROM和软盘

user选项是让没有权限的用户可以对某一行设备运行mount命令；

一直以来我们都是使用/etc/fstab 管理文件系统和挂载点，另一种方式是systemd及其单元，也是基于/etc/fstab生成的；

一下命令用于查看文件系统的容量和使用量

$ds

### 重新挂载文件系统

有时候需要更新挂载选项而在同一挂载点挂载文件系统，比较常见的情况系统崩溃的时候，你只需要将只读文件系统改为可写；

$mount -n -o remount /

该命令假定设备在目录/etc/fstab中，否则你需要指定设备；上面命令以读写方式重新挂载root (mount 命令在root只为读的情况下无法挂载数据库 必须需要 -o)

### 检查和修复文件系统

fsck是检查文件系统的工具，对于mkfs来说fsck对每个Linux支持的文件系统都有一个对应的版本；

$fsck /dev/sda

以上命令可以指定设备或者挂载点作为参数

面对严重的磁盘故障你可以有下面的选择

使用dd 尝试从磁盘提取出整个文件系统的映像，然后将它转移到另一个磁盘的相 同大小的分区中；

你可以在只读模式挂载文件系统，在想办法修复；

debugfs 遍历文件系统中的文件，将它们复制到其他地方；

### 特殊用途文件系统

proc 挂载在/proc 是进程的缩写；/proc目录中的 子目录以系统的PID命名，子目录中的文件代表进程的各种状态；/proc/self表示当前进程；Linux系统的proc文件系统包括大量的内核和硬件系统信息；

sysfs (前面已经介绍）

tmpfs 挂载在/run/ 和其他位置；通过tmpfs可以将物理内存和交换空间作为临时存储；

## 交换空间（swap space)

系统可以使用磁盘以便扩从内存容量；当内存不足的时候，Linux内存系统会自动将内存的进程（闲置进程）移出到磁盘，以及激活进程从磁盘移入内存；

### 可以使用磁盘分区或者文件作为交换空间

磁盘分区作为交换空间的步骤

确保分区为空

mkswap dev //dev 是分区设备

swapon dev //向内核注册

以文件作为交换空间

dd if=/dev/zero of=swap\_file bs=1024k count=num\_mb

mkswap swap\_file

swapon swap\_file

最好交换空间是内存的2倍以上；

## 传统文件系统

传统的文件系统有两个基础组件： 一个用来存储数据的数据块池和一个用来管理数据池的数据库系统。这个数据库是inode数据结构的核心。inode是一组描述文件的数据，包括文件类型，权限，文件数据所在数据池；

文件和目录也是通过inode来实现的，目录inode包含一个文件名列表以及对应的指向其他inode链接；

inode表 包含若干