快速成　Linux系统管理员教程

————*Linux学习教程*

里面的好些Linux内容都是需要学习者去了解掌握的，本书籍的内容仅供学习参考。让你对Linux系统的管理更加的到位。

内容基础，语言简短简洁

红联Linux论坛是致力于Linux技术讨论的站点，目前网站收录的文章及教程基本能满足不同水平的朋友学习。

红联Linux门户： www.linux110.com
红联Linux论坛： www.linuxdiyf.com/bbs
下载:Linux电子书籍：
http://www.linux286.com/linux/linuxdzsj.htm

目录

速成Linux系统管理员

Linux 系统概述

发布时间 :2007-01-27 23:10:37

本章概述Linux系统。首先描述操作系统提供的主要服务。然后说明实现这些服务的程序 with a considerable lack of detail。本章的目的是理解系统整体，以后再详细说明每个部分。

一个操作系统的不同部分

一个UNIX操作系统包括一个核心程序kernel和一些系统程序system programs。还有一些做某件事情的应用程序application programs。核心是操作系统的心脏 。它跟踪磁盘上的文件，启动程序并运行它们，给不同的进程分配内存和其他资源，从网络接收和向网络发送包，等等。核心很少自己干活，但它提供工具，用这些工具可以建立所有服务。它还阻止任何人直接存取硬件，强制每个人使用它提供的工具。这样，核心给每个用户与其他用户之间提供保护。核心提供的工具通过系统调用system calls使用；要了解关于这的更多的信息，看手册页第二节。

系统程序使用核心提供的工具实现操作系统要求的不同的服务。系统程序和所有其他程序运行在核心之上，叫做用户模式user mode。系统程序和应用程序的区别是目的：应用程序意图完成一些有用的工作(或游戏)，而系统程序则为系统工作而需要。字处理器是应用；telnet 是系统程序。区别经常有些模糊，也许，只是为了强制分类。

虽然编程语言不必是操作系统的一部分，操作系统也可以包括编译器及其相关的库(Linux下是gcc和C库)。文档、有时甚至游戏，可以是操作系统的一部分。传统上，操作系统被定义为安装磁带或磁盘上的内容；对于Linux就更不清楚，因为它分布在全世界的FTP站点上。

核心的重要部件

Linux核心包括几个重要部分：进程管理、存储器管理、硬件设备驱动、文件系统驱动、网络管理和其他不同的部分。图2.1显示了它们的一些。

可能核心最重要的部分(没有它们什么也不能工作)是存储器管理和进程管理。存储器管理负责分配进程的存储器区域和对换空间区域、核心的部件及buffer cache。进程管理产生进程，用切换处理器上的活动进程来实现多任务。

在最低级，核心对它支持的每种硬件包含一个硬件设备驱动。因为世界上存在大量不同的硬件，硬件设备驱动的数量极大。有许多不同的硬件，因此软件控制方法不同。但其相似性可能分类驱动，支持相似的操作；每类的每个成员有相同的与核心其他部分接口，但具体实现是不同的。例如，所有的硬盘驱动与核心其他部分接口相同，即他们都有初始化驱动器、读N扇区、写N扇区。

核心自己提供的有些软件服务有类似的抽象属性，因此可以抽象分类。例如，不同的网络协议已经被抽象为一个编程接口：BSD socket库。另一个例子是虚拟文件系统virtual filesystem(VFS)层，它从文件系统操作实现中抽象出文件系统操作。每个文件系统类型提供了每个文件系统操作的实现。当一些实体企图使用一个文件系统时，请求通过VFS送出，它将请求发送到适当的文件系统驱动。

UNIX系统的主要服务

本节说明一些最重要的UNIX服务，但不太详细。以后的章节中将更详细地说明。

init

UNIX系统里最重要的服务是由init 提供的。 init 是每个UNIX系统在核心引导最后启动的第一个进程。 init 启动后，它做不同的启动工作继续引导过程 (检查和mount文件系统、启动守侯程序等)

init 做的具体事情的列表依赖于用户意愿； init 有多个启动选择。通常提供单用户模式single user mode的概念，此时没有用户能登录，root 在控制台使用一个shell；通常的模式叫多用户模式multiuser mode。一般说来这称为运行级run levels；单用户和多用户模式可以理解为2个运行级，还可以有其他运行级，例如，在控制台上运行X。

在普通操作中，init 确认getty 正常运行(允许用户登录)，并收养孤儿进程(父进程已死的进程；UNIX中，所有进程 必须在一棵树中，因此孤儿进程必须收养)。

当系统关闭时，init 负责杀死所有其他进程，unmount所有文件系统并停止处理器，根据设置。

从终端登录

从终端登录(通过串行线)和控制台(当不运行X时)是由getty 程序提供的。 init 为每个允许登录的终端启动一个单独的getty 实例 getty 读用户名并运行login 程序，由它读口令。如果用户名和口令正确，login 就运行shell 。当 shell终止时，即用户注销，或login 因用户名和口令不对而终止时， init 知道并启动一个新的getty 实例 。核心没有登录的概念，这都由系统程序处理。

Syslog

核心和许多系统程序会产生错误、警告和其他信息。这些信息在以后能看经常是很重要的，甚至很久以后，所以它们应该被写到一个文件。这个程序是syslog 。它能设置成根据输出信息的程序或重要程度将信息排序到不同的文件。例如，核心信息经常与其他信息分开，单独定向到一个分离的文件，因为核心信息经常更重要且需要有规律地阅读以确定问题。

定时执行命令: cron 和at

用户和系统管理员经常需要定时运行命令。例如，系统管理员可能想运行一个程序从老文件中清除暂存文件的目录 (/tmp 和/var/tmp)，以免磁盘满，因为并非所有程序都正确地清除自己的暂存文件。

cron 服务是做这个的。每个用户有个crontab ，在这里列出他要执行的命令和想执行的时间。 cron 守侯进程负责在特定的时间启动命令。

at 服务与cron 类似，但它只执行一次：命令在给定的时间执行，但不可自动重复。

图形用户接口GUI

UNIX和Linux不将用户接口合在核心中，而是用用户级程序实现。用户接口同时提供文本和图形环境。

这样的安排使系统更灵活，但有容易对每个程序实现不同的用户接口的缺点，使系统较难学。

Linux使用的主要的图形环境叫X Window系统(简称X)。 X也不实现用户接口；它只实现一个窗口系统，即可以实现图形用户接口的工具。 3种最流行的基于X实现的用户接口风格是Athena、Motif和Open Look。

网络

网络连接2台或更多的计算机使之能互相通信。连接和通信的实际方法有些复杂，但结果非常有用。

UNIX操作系统具有许多网络特征。最基本的服务：文件系统、打印、备份等都可以通过网络完成。这可使系统管理更简单，因为它允许集中管理，同时获得小型机和分布计算的优点，例如降低成本和更好的容错能力。

然而，本书只概述网络；更多的信息请见《Linux网络管理员指南》，包括一个网络如何操作的基本说明。

网络登录

网络登录与普通登录有一点不同。可以登录的每个终端各有一条单独的物理串行线。从网络登录的每个人，有一条单独的虚拟网络连接，并且可以有任意数量 。因此不可能为每个可能的虚拟连接运行单独的getty 。通过网络登录有若干不同的方法， telnet 和rlogin 是TCP/IP网络中的主要方法。

网络登录为每种登录方法提供一个单独的守侯程序(telnet 和rlogin 使用不同的守侯程序)，而不是使用一群getty ，来侦听所有的输入的登录企图。当发现一个登录企图，就启动一个自己的新实例来处理这个企图；原来的实例继续侦听其他企图。新实例的工作和getty 类似。

网络文件系统

网络服务的一个最有用的东西是通过网络文件系统network file system共享文件。这个服务一般用Sun公司开发的网络文件系统Network File System，或NFS。

通过网络文件系统，任何文件操作可以由一台机器的一个程序通过网络发送到其他任何机器。这愚弄了程序，使它以为其他机器上的所有文件是在程序运行的机器上。这极大地简化了信息共享，因为它对程序无须任何修改。

邮件

电子邮件通常是通过计算机通信的最重要的方法。一封电子信件用特定的格式存储在一个文件中，使用特定的邮件程序来收发邮件。

每个用户有一个收件箱incoming mailbox(一个特定格式的文件)，所有新邮件存在这里。当什么人发送邮件时，邮件程序定位收件人的邮箱，并在邮箱文件中添加信件。如果收件人的邮箱在另一台机器上，信件就被发送到那台机器，由它用它觉得最合适的方法来投递邮箱。

邮件系统包括很多程序。投递邮件到本地或远程邮箱使用一个程序(mail transfer agent或MTA，例如，sendmail 或smail)，而用户使用的则有很多不同的程序(mail user agent或MUA，例如pine 或elm)。邮箱一般存在/var/spool/mail 中。

打印

同时只能有一个人使用某一台打印机，但各用户不共享打印机是不经济的。因此打印机由软件来管理，实现一个打印队列print queue：所有的打印任务放进一个队列，打印机完成一个任务后，自动再打印下一个。这无须用户来组织打印队列，回避了直接控制打印机。

打印队列软件也把打印输出spools到磁盘，即，当任务在队列中时，打印输出存在一个文件中。这允许应用程序快速地完成一个打印任务到打印队列程序，应用程序无须等到打印任务真正完成就可以继续下去。这真的很方便，因为它允许打印出一个版本，无须打印完成，就继续修改一个新版本。

文件系统布局

文件系统分为许多部分，通常从根文件系统有： /bin , /lib , /etc , /dev , 及一些其他； /usr 文件系统包含程序和不改变的数据； /var 文件系统包含改变的数据(例如log文件)； /home 文件系统包含每个用户的个人文件。依赖于硬件配置和系统管理员的决定，方法不同，甚至所有东西可能在一个文件系统中。

3章将更详细地说明文件系统布局，Linux文件系统标准(Linux Filesystem Standard)中有更详细的说明。

Linux 目录树概述

发布时间 :2007-01-27 23:11:02

本章说明标准Linux目录树的重要部分，基于FSSTND文件系统标准。概述根据不同的目的和给定的要求将目录树分为若干分离的文件系统的一般方法。也说明一些其他方法。

背景

本章松散地基于Linux文件系统标准FSSTND版本1.2(见参考书目[Qui95])，它意图建立一个如何组织Linux系统目录树的标准。这样一个标准具有易于写或 port(移植?)Linux软件、管理Linux系统的优点，因为所有东西都将在他们的一般地方。此标准没有强制所有人遵从的权威，但它有最多的Linux distributions的支持。如果没有什么特殊的理由，不遵从FSSTND不是个好主意。 FSSTND意图遵从Unix传统和当前趋势，使熟悉其他Unix系统的人对Linux系统更容易接受(反之亦然)。

本章并非如FSSTND那么详细。一个系统管理员应该阅读FSSTND以得到全部的理解。

本章不详细解释所有文件。其意图并非说明每个文件，而是从文件系统的视角给出系统的一个概览。每个文件的更多的信息在本手册或man页的其他地方。

有意将全目录树可以分为小的部分，每个部分可以在自己的磁盘或分区上，以能为磁盘容量所容纳，并易于备份及其他系统管理。主要部分是根、/usr 、/var 和 /home 文件系统。每个部分有不同的目的。目录树已被设计成能在Linux机器的网络中很好地工作，可以通过只读设备(如CDROM)或NFS网络共享文件系统的一些部分。

下面说明目录树不同部分的任务。

每台机器都有根文件系统(一般在本地盘中，当然也可以在RAM盘或网络盘中)，它包含系统引导和使其他文件系统得以mount所必要的文件，根文件系统应该有单用户状态所必须的足够的内容。还应该包括修复损坏系统、恢复备份等的工具。

/usr 文件系统包含所有命令、库、man页和其他一般操作中所需的不改变的文件。 /usr 应该没有对给定机器特定的文件，也不应该有一般使用中要修改的文件。这样允许此文件系统中的文件通过网络共享，这样可以更有效，因为这样节省了磁盘空间(/usr 很容易是数百兆)，且易于管理(当升级应用时，只有主/usr 需要改变，而无须改变每台机器) 即使此文件系统在本地盘上，也可以只读mount，以减少系统崩溃时文件系统的损坏。

/var 文件系统包含会改变的文件，比如spool目录(mail、news、打印机等用的)， log文件、formatted manual pages和暂存文件。传统上/var 的所有东西曾在 /usr 下的某个地方，但这样/usr 就不可能只读安装了。

/home 文件系统包含用户家目录，即系统上的所有实际数据。将家目录分到自己的目录树或文件系统中易于备份，其他部分经常不必备份，至少不必经常备份(它们很少改变)。一个大的/home 可能要分为若干文件系统，需要在/home 下加一级名字，如/home/students 、/home/staff 等。

虽然上面将不同的部分称为文件系统，但它们不必是真的分离的文件系统。如果系统是小的单用户系统，而用户希望简单化，可以很容易地放在一个文件系统中。根据磁盘容量和不同目的所需分配的空间，目录树也可以分到不同的文件系统中。重要的是使用标准的名字，即使/var 和/usr 在同一分区上，名字/usr/lib/libc.a 和/var/adm/messages 必须能工作，例如将/var 下的文件移动到/usr/var ，并将/var 作为/usr/var 的符号连接。

Unix文件结构根据目的来分组文件，即所有的命令在一个地方，所有的数据在另一个地方，所有的文档又在一个地方，等等。另一个方法是根据属于的程序分组文件，即所有Emacs文件在一个目录中，所有TeX文件在另一个中，等等。后一种方法的问题是文件难于共享(程序目录经常同时包含静态可共享的和动态不可共享的文件)，有时难于查找 (例如man页在极大数量的地方，使man程序查找它们极其困难)。

根文件系统

根文件系统一般应该比较小，因为包括严格的文件和一个小的不经常改变的文件系统不容易损坏。损坏的根文件系统一般意味着除非用特定的方法(例如从软盘)系统无法引导，所以不应该冒这个险。

根目录一般不含任何文件，除了可能的标准的系统引导映象，通常叫/vmlinuz 。所有其他文件在根文件系统的子目录中。

/bin
引导启动所需的命令或普通用户可能用的命令(可能在引导启动后)。

/sbin
类似/bin ，但不给普通用户使用，虽然如果必要且允许时可以使用。

/etc
特定机器的配置文件。

/root
root用户的家目录。

/lib
根文件系统上的程序所需的共享库。

/lib/modules
核心可加载模块，特别是那些恢复损坏系统时引导所需的(例如网络和文件系统驱动)。

/dev
设备文件。

/tmp
临时文件。引导启动后运行的程序应该使用/var/tmp，而不是/tmp，因为前者可能在一个拥有更多空间的磁盘上。

/boot
引导加载器(bootstrap loader)使用的文件，如LILO。核心映象也经常在这里，而不是在根目录。如果有许多核心映象，这个目录可能变得很大，这时可能使用单独的文件系统更好。另一个理由是要确保核心映象必须在IDE硬盘的前1024柱面内。

/mnt
系统管理员临时mount的安装点。程序并不自动支持安装到/mnt 。 /mnt 可以分为子目录(例如/mnt/dosa 可能是使用MSDOS文件系统的软驱，而/mnt/exta 可能是使用ext2文件系统的软驱)。

/proc , /usr , /var , /home
其他文件系统的安装点。

/etc目录
/etc 目录包含很多文件。下面说明其中的一些。其他的你应该知道它们属于哪个程序，并阅读该程序的man页。许多网络配置文件也在/etc 中，它们在《网络管理指南》中说明。

/etc/rc or /etc/rc.d or /etc/rc.d
启动、或改变运行级时运行的scr ī pts或scr ī pts的目录，更详细的信息见关于init 的章。

/etc/passwd
用户数据库，其中的域给出了用户名、真实姓名、家目录、加密的口令和用户的其他信息。格式见passwd 的man页。

/etc/fdprm
软盘参数表。说明不同的软盘格式。用setfdprm 设置。更多的信息见setfdprm 的man页。

/etc/fstab
启动时mount -a命令(在/etc/rc 或等效的启动文件中)自动mount的文件系统列表。 Linux下，也包括用swapon -a启用的swap区的信息。见4.8.5节和mount 的man页。

/etc/group
类似/etc/passwd，但说明的不是用户而是组。见group 的man页。

/etc/inittab

init 的配置文件。

/etc/issue
getty 在登录提示符前的输出信息。通常包括系统的一段短说明或欢迎信息。内容由系统管理员确定。

/etc/magic
file 的配置文件。包含不同文件格式的说明，file 基于它猜测文件类型。见magic 和file 的man页。

/etc/motd
Message Of The Day，成功登录后自动输出。内容由系统管理员确定。经常用于通告信息，如计划关机时间的警告。

/etc/mtab
当前安装的文件系统列表。由scr ī pts初始化，并由mount 命令自动更新。需要一个当前安装的文件系统的列表时使用，例如df 命令。

/etc/shadow
在安装了影子口令软件的系统上的影子口令文件。影子口令文件将/etc/passwd 文件中的加密口令移动到/etc/shadow 中，而后者只对root可读。这使破译口令更困难。

/etc/login.defs
login 命令的配置文件。

/etc/printcap
类似/etc/termcap ，但针对打印机。语法不同。

/etc/profile , /etc/csh.login , /etc/csh.cshrc
登录或启动时Bourne或C shells执行的文件。这允许系统管理员为所有用户建立全局缺省环境。各shell见man页。

/etc/securetty
确认安全终端，即哪个终端允许root登录。一般只列出虚拟控制台，这样就不可能(至少很困难)通过modem或网络闯入系统并得到超级用户特权。

/etc/shells
列出可信任的shell。chsh 命令允许用户在本文件指定范围内改变登录shell。提供一台机器FTP服务的服务进程ftpd 检查用户shell是否列在 /etc/shells 文件中，如果不是将不允许该用户登录。

/etc/termcap
终端性能数据库。说明不同的终端用什么"转义序列"控制。写程序时不直接输出转义序列(这样只能工作于特定品牌的终端)，而是从/etc/termcap 中查找要做的工作的正确序列。这样，多数的程序可以在多数终端上运行。

见termcap 、 curs_termcap 和terminfo 的man页。

/dev目录
/dev 目录包括所有设备的设备文件。设备文件用特定的约定命名，这在设备列表中说明 (见[Anv])。设备文件在安装是产生，以后可以用 /dev/MAKEDEV 描述。 /dev/MAKEDEV.local 是系统管理员为本地设备文件(或连接)写的描述文稿 (即如一些非标准设备驱动不是标准MAKEDEV 的一部分)。

/usr文件系统
/usr 文件系统经常很大，因为所有程序安装在这里。 /usr 里的所有文件一般来自Linux distribution；本地安装的程序和其他东西在/usr/local 下。这样可能在升级新版系统或新distribution时无须重新安装全部程序。 /usr 的有些子目录在下面列出(一些不太重要的目录省略了，更多信息见FSSTND)。

/usr/X11R6
X Window系统的所有文件。为简化X的开发和安装，X的文件没有集成到系统中。 X自己在/usr/X11R6 下类似/usr 。

/usr/X386
类似/usr/X11R6 ，但是给X11 Release 5的。

/usr/bin
几乎所有用户命令。有些命令在/bin 或/usr/local/bin 中。

/usr/sbin
根文件系统不必要的系统管理命令，例如多数服务程序。

/usr/man , /usr/info , /usr/doc
手册页、GNU信息文档和各种其他文档文件。

/usr/include
C编程语言的头文件。为了一致性这实际上应该在/usr/lib 下，但传统上支持这个名字。

/usr/lib
程序或子系统的不变的数据文件，包括一些site-wide配置文件。名字lib来源于库(library); 编程的原始库存在/usr/lib 里。

/usr/local
本地安装的软件和其他文件放在这里。

/var文件系统
/var 包括系统一般运行时要改变的数据。每个系统是特定的，即不通过网络与其他计算机共享。

/var/catman
当要求格式化时的man页的cache。man页的源文件一般存在/usr/man/man* 中；有些man页可能有预格式化的版本，存在/usr/man/cat* 中。而其他的man页在第一次看时需要格式化，格式化完的版本存在/var/man 中，这样其他人再看相同的页时就无须等待格式化了。 (/var/catman 经常被清除，就象清除临时目录一样。)

/var/lib
系统正常运行时要改变的文件。

/var/local
/usr/local 中安装的程序的可变数据(即系统管理员安装的程序)。注意，如果必要，即使本地安装的程序也会使用其他/var 目录，例如/var/lock 。

/var/lock
锁定文件。许多程序遵循在/var/lock 中产生一个锁定文件的约定，以支持他们正在使用某个特定的设备或文件。其他程序注意到这个锁定文件，将不试图使用这个设备或文件。

/var/log
各种程序的Log文件，特别是login (/var/log/wtmp log所有到系统的登录和注销) 和syslog (/var/log/messages 里存储所有核心和系统程序信息。 /var/log 里的文件经常不确定地增长，应该定期清除。

/var/run
保存到下次引导前有效的关于系统的信息文件。例如， /var/run/utmp 包含当前登录的用户的信息。

/var/spool
mail, news, 打印队列和其他队列工作的目录。每个不同的spool在/var/spool 下有自己的子目录，例如，用户的邮箱在/var/spool/mail 中。

/var/tmp
比/tmp 允许的大或需要存在较长时间的临时文件。 (虽然系统管理员可能不允许/var/tmp 有很旧的文件。)

/proc文件系统
/proc 文件系统是一个假的文件系统。它不存在在磁盘某个磁盘上。而是由核心在内存中产生。用于提供关于系统的信息(originally about processes, hence the name)。下面说明一些最重要的文件和目录。 /proc 文件系统在proc man页中有更详细的说明。

/proc/1
关于进程1的信息目录。每个进程在/proc 下有一个名为其进程号的目录。

/proc/cpuinfo
处理器信息，如类型、制造商、型号和性能。

/proc/devices
当前运行的核心配置的设备驱动的列表。

/proc/dma
显示当前使用的DMA通道。

/proc/filesystems
核心配置的文件系统。

/proc/interrupts
显示使用的中断，and how many of each there have been.

/proc/ioports
当前使用的I/O端口。

/proc/kcore
系统物理内存映象。与物理内存大小完全一样，但不实际占用这么多内存；it is generated on the fly as programs access it. (记住：除非你把它拷贝到什么地方，/proc 下没有任何东西占用任何磁盘空间。)

/proc/kmsg
核心输出的消息。也被送到syslog 。

/proc/ksyms
核心符号表。

/proc/loadavg
系统"平均负载"；3个没有意义的指示器指出系统当前的工作量。

/proc/meminfo
存储器使用信息，包括物理内存和swap。

/proc/modules
当前加载了哪些核心模块。

/proc/net
网络协议状态信息。

/proc/self
到查看/proc 的程序的进程目录的符号连接。当2个进程查看/proc 时，是不同的连接。这主要便于程序得到它自己的进程目录。

/proc/stat
系统的不同状态，such as the number of page faults since the system was booted.

/proc/uptime
系统启动的时间长度。

/proc/version
核心版本。

注意所有上述文件给出易读的文本文件，有时可能是不易读的格式。有许多命令做了些格式化以更容易读。例如，free 程序读/proc/meminfo 并将给出的字节数转换为千字节(并增加了一些信息)。

磁盘和其他存贮介质的使用

发布时间 :2007-01-27 23:16:28

安装和升级系统时，需要对硬盘做很多工作。必须在硬盘上做文件系统，使文件能存在其上，并为系统不同的部分保留空间。

本章说明所有这些初始化工作。通常，一旦你建立了系统，就不必再做这些工作(除了使用软盘)。如果你要增加一个新硬盘或更好地调整你的硬盘的使用，那么可能回到这一章。

管理磁盘的基本任务有：

格式化磁盘。这为磁盘进入使用做一些工作，比如检查坏扇区。(现在多数硬盘无须格式化。)

给硬盘分区，如果想用于互相不干扰的几件事。分区的一个原因是要在一个硬盘上存不同的操作系统。另一个原因是将用户文件和系统文件分开，以简化备份并在系统崩溃时有助于保护系统文件。

在每个磁盘或分区上建立合适类型的文件系统，然后文件就可以在其上产生和存取。在你建立文件系统前，磁盘对Linux没有意义。

将不同的文件系统安装起来形成一个单独的树结构，按需要可以自动或手工完成。 (手工安装的文件系统通常还要手工unmount)

5章包括虚拟内存和磁盘cache的信息，使用磁盘应该知道这些。

本章说明对硬盘、软盘、CDROM和磁带机应该知道什么。

2种设备

UNIX及Linux，识别2类设备：随机存取的块设备(如磁盘)和字符设备(如磁带和串行线)，有些是串行的，有些是随机存取的。文件系统支持的每种看来是个设备文件。当读写设备文件时，数据与设备联系。这样没有必要为存取设备编制特别的程序(程序不直接获取中断或读取串口)，例如，发送文件到打印机，只需：

```
$ cat filename > /dev/lp1
$
```

文件内容就被打印了(当然，文件必须是打印机能理解的格式)。当然，因为不应该让多人同时cat文件到同一打印机，一般用特定的程序发送文件去打印(通常是lpr)。这个程序能确保同时只有一个文件被打印，并自动在完成后发送下一个。多数设备有类似需要。实际上，根本很少需要关心设备文件。

因为设备被视为文件系统中的文件(在/dev 目录中)，很容易看到存在哪些设备文件，使用ls 或其他的适当的命令即可。在ls -l 的输出中，第一列包含文件类型和权限。例如，查看我系统上的一个串行设备：

```
$ ls -l /dev/cua0
crw-rw-rw- 1 root uucp 5, 64 Nov 30 1993 /dev/cua0
$
```

第一列第一个字符，即crw-rw-rw-中的c告诉用户文件的种类，这是一个字符设备。一般文件的第一个字符是"-"，目录是"d"，块设备是"b"；更多的信息见ls man页。

注意即使设备没有安装，一般所有设备文件都存在。因此有/dev/sda 文件并不意味着你真的有个SCSI硬盘。有所有的设备文件使安装程序更简单，也易于增加新硬件(无须再为产生新设备的设备文件找出正确的参数)。

硬盘

本节介绍有关硬盘的术语。如果你已经知道这些项目和内容，可以跳过本节。

硬盘包括一到数片盘片platters， 其一个或两个面surfaces涂有磁性材料用于记录数据。每面有一个读写头read-write head用于读写数据。盘片有一个共同的轴，典型的旋转速度是每分钟3600转，高性能的硬盘转速可能更高。磁头可沿着盘片的半径移动，磁头移动加上盘片旋转可以使词头存取磁盘表面的任何一个位置。

处理器(CPU)和实际磁盘通过磁盘控制器disk controller通讯。这使计算机其他部分不必知道如何使用驱动器，因为不同磁盘的控制器可以做成对计算机其他部分相同的接口。这样，计算机只要说"嗨，磁盘，给我我要的东西"，而不是用一串长而复杂的电信号来移动磁头到正确的位置，并等正确的位置到了磁头下后再做那些不愉快的工作。 (实际上，到控制器的接口仍然很复杂，但比没有好多了。) 控制器还可以做一些其他的事，比如缓冲，或自动坏扇区替换等。用电信号控制操作机械部件，

以上只是理解硬件所需的。还有其他好多工作，比如马达旋转磁盘、移动磁头，但这都与理解硬盘工作原理无关。

磁盘表面通常被分为同心圆环，叫磁道tracks，磁道又被分为扇区sectors。用这样分来将磁盘定位，用于为文件定位磁盘空间。要在硬盘上找到给定的位置，可能?quot;3面5道7扇区"。通常所有磁道有相同的扇区数，但也有硬盘在外圈磁道放较多的扇区(所有扇区用同样大小的物理空间，这样在较长的外圈磁道可以容纳更多的数据)。一般一个扇区容纳512字节数据。磁盘不能处理比一个扇区更小的数据量。

每个面以相同的方式分为磁道和扇区。这意味着当一个磁头在某个磁道时，其他磁头也在相应的位置，所有相同位置的磁道组成柱面cylinder。磁头从一个磁道(柱面)移动到另一个需要花时间，所以将经常要在一起存取的数据(如一个文件)放在一个柱面里。这改善了性能。当然不可能完全作到，文件被放在几个相分离的位置

叫碎片fragmented。

磁盘的面(或头，实际是一样的)、柱面、扇区数各不相同，硬盘这些数目叫硬盘参数geometry。硬盘参数通常存在一个特定的、由电池供电的存储区中，叫CMOS RAM，操作系统在引导启动或驱动器初始化时可以从那里得到硬盘参数。

不幸的是，BIOS 有一个设计限制，就是不能在CMOS RAM中定义大于1024的磁道数，这对大硬盘来说就太小了。为了克服这个问题，硬盘控制器在磁盘参数上做了一个欺骗，用地址转换translates the addresses使计算机接受。例如，一个硬盘可能有8个磁头，2048个磁道，每磁道35个扇区。其控制器可以对计算机谎称它有16个磁头，1024个磁道，每磁道35个扇区，这样就没有超过磁道数的限制，地址转换将磁头数减半，磁道数加倍后传给硬盘。实际的算法可能更复杂，因为数量可能不象我们在这里假设的这么好(但这不影响我们理解原理)。这个转换在操作系统来看产生了错觉，并可能影响操作系统对把所有数据存在相同柱面的企图受到影响。

转换只是IDE硬盘的问题。SCSI硬盘使用连续的扇区号(即控制器将连续的扇区好转换成磁头、柱面、扇区的三参数组),对CPU与控制器的通信使用完全不同的方法，因此不会有这个问题。注意，计算机可能根本不知道一个SCSI硬盘的实际参数。

由于Linux经常不知道一个硬盘的真正参数，其文件系统也不试图将文件存在一个柱面里。而是争取给一个文件分配连续编号的山区，这样能得到类似的性能。对于控制器上有cashe或控制器能自动预取的硬盘，情况将更复杂。

每个硬盘表现为一个单独的设备文件。通常只能有2-4个IDE硬盘。这就是 /dev/hda , /dev/hdb , /dev/hdc , 和 /dev/hdd 。 SCSI是 /dev/sda , /dev/sdb , 等等。其他硬盘类型有类似的命名约定，更多的信息见[Anv]。注意硬盘的设备文件给出整个硬盘的存取，而不是分区(下面讨论的)，因此如果不小心可能搞乱分区或数据。硬盘的设备文件只在存取主引导扇(也将在下面讨论)时使用。

软盘

软盘的一面或两面涂有和硬盘类似的磁性介质。软盘自己没有读写头，读写头在驱动器上。软盘相当于硬盘的一张盘片，但可移动，一个驱动器可以存取不同的软盘，而硬盘则是一个独立的单元。

如同硬盘，一张软盘也分为磁道和扇区(软盘2面上的相同的磁道组成柱面)，但数量要比硬盘少得多。

软驱通常可以使用几中不同的盘片，例如，一个3.5'软驱可以使用720KB和1.44MB的软盘。因为软驱操作有些不同，而操作系统必须知道软盘的容量，所以软驱有许多设备文件，每个都与软驱和软盘种类有关。因此，/dev/fd0H1440 是第一个软驱(fd0)，必须是3.5'软驱，使用3.5'高密度软盘(H)，容量是1440KB(1440)，即普通的3.5'HD软盘。软盘设备的命名约定见[Anv]。

软驱的名字是复杂的，因此Linux有一个特定的软驱设备类型，能自动检测软驱中软盘的种类。它使用不同的软盘类型试图读取新插入的软盘的第一个扇区，直到找到正确的一个。这自然要求软盘是已经格式化过的。自动设备叫/dev/fd0 、/dev/fd1 等。

存取软盘的自动设备的参数可用程序setfdprm 设定。这可使你使用不是通常容量的软盘，例如有非标准扇区数的软盘，或自动检测由于某种原因失败或适当的设备文件丢失。

Linux除了所有标准的，还能处理许多非标准的软盘格式。这有时需要特殊的格式化程序。我们现在先跳过这些软盘格式，同时你可以查看/etc/fdprm 文件。它定义了setfdprm 识别的设定。

操作系统必须知道软驱何时换了软盘，例如，以免使用上一张软盘的cache数据。不幸的是，当用于此的信号线断了或不好时，当在MSDOS中使用时，这并不总有效。如果你曾遇到过软驱的这种怪异的问题，可能是这个原因。解决这个问题的唯一方法是修理软驱。

CD-ROM

CD-ROM驱动器使用一个光学可读的塑料涂布的盘片。信息记录在盘片表面 的从中心的边沿的螺旋型小坑上。驱动器发出一束激光来读盘。当激光射到小坑上，激光以一种方式反射；当它射到光滑表面上，它以另一种方式反射。这很容易地编码成bit，组成信息。其他很容易，不过是机械。

CD-ROM驱动器比硬盘慢。典型的硬盘的平均寻道(seek)时间小于15毫秒，而快速的CD-ROM驱动器要花零点几秒。实际数据传输率则相当快，在数百KB/s。速度慢使CDROM驱动器不能代替硬盘使用 (有些Linux distributions提供"live" CD-ROM文件系统，使之不必拷贝文件到硬盘，使安装简单并节约了许多硬盘空间)，虽然是可能的。要安装新软件，CD-ROM很好，因为在安装时速度并非最重要的。

有多种方法在CDROM上安排数据。最流行的是国际标准化组织定义的ISO9660。这个标准定义了一个最小的文件系统，甚至比MSDOS更粗糙。这样，由于它是这么小，所有操作系统都可以将它映射到自己的系统。

不同UNIX不能使用ISO9660文件系统，因此开发了对这个标准的一个增强，叫Rock Ridge增强。 Rock Ridge允许长文件名、符号连接和许多其他优点，使CD-ROM更象UNIX文件系统。同时，Rock Ridge文件系统仍然是一个有效的ISO9660文件系统，使非UNIX一样可以使用。 Linux同时支持ISO9660和Rock Ridge增强，增强被自动识别和使用。

文件系统只是一部分，许多CD-ROM包含的数据需要特定的程序存取，而多数程序不能运行在Linux下 (当然，可能运行在Linux的MSDOS仿真器dosemu下)。

CD-ROM驱动器通过相关的设备文件存取。有多种方法将CDROM连接到计算机：SCSI、声卡或EIDE。要完成这的硬件hacking工作超出了本书的范围，但连接方法决定了设备文件。指导见[Anv]

磁带

磁带驱动器使用磁带，类似 音乐用的盒带。磁带是串行的，即如果要得到给定部分的数据，必须经过所有部分。磁盘可以随机存取，即可以直接跳到磁盘上的某个部分。串行存取的磁带当然慢了。

另外一方面，磁带相当便宜，因为无须快速。也容易做得很长，因此可以容纳大量的数据。这使磁带很适于如归档、备份等无须高速的、但需要低成本和大容量的事情。

格式化

格式化在磁介质上写用于标记磁道和扇区的标志的过程。磁盘格式化前，其磁表面是完成的一块。格式化后，混沌变为秩序，建立的磁道，划分了扇区。实际细节并非准确地这样，但重要的是：磁盘不经过格式化是不能使用的。

这里术语有些模糊：MS-DOS中，格式化(format)这个词还包括了产生文件系统的过程(下面将讨论的)。这两个过程经常一起使用，尤其是软盘。当必须区分时，真正的格式化被称为低级格式化low-level formatting，而建立文件系统被成为高级格式化high-level formatting。在UNIX圈中，这两者叫格式画format和建立文件系统make a filesystem，本书中也这样称。

IDE硬盘和一些SCSI硬盘实际上厂商已经做了格式化，并无须重复；因为多数人无须关心它。实际上，格式化硬盘可能反而不好，比如因为硬盘可能需要用特定的方法格式化使坏扇区被自动替换。

磁盘经常需要特定的程序来格式化，因为驱动器的格式化逻辑的接口每个驱动器都不一样。格式化程序经常在控制器BIOS上，或用MSDOS程序提供，这都不太容易在Linux中使用。

格式化中可能会发现磁盘的坏点，叫坏块bad blocks or bad sectors。这有时由驱动器自己处理。但有时，如果坏块太多，需要一些工作来避免使用磁盘的这部分。 The logic to do this is built into the filesystem; 下面将说明如何增加这些信息到文件系统。另外，产生一个只覆盖这些坏的部分的小分区也是一个办法。如果坏区较大，这可能是个好办法，因为文件系统有时难以处理大量的坏区。

软盘格式化使用fdformat 。软盘设备使用给定的参数，例如下面的命令在第一个软驱中格式化一张高密度3.5'软盘：

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
$
```

注意，如果想使用自动检测设备(如/dev/fd0)， 必须用先setfdprm 设定参数。要得到与上面一样的结果，可以这样：

```
$ setfdprm /dev/fd0 1440/1440
$ fdformat /dev/fd0
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
```

```
$
```

选择与软盘类型相符的正确的设备文件通常更方便。注意，比软盘设计格式化更多的信息容量是没有意义的。

fdformat 也将验证软盘，例如检查坏块。它在坏块试验几次(你通常能听到，驱动器的噪声很明显)。 If the floppy is only marginally bad (due to dirt on the read/write head, some errors are false signals), fdformat 可能没事，而真正的错误可能退出有效过程。核心把发现的每个I/O错误打印log信息，送到控制台，或者，如果使用了syslog ， 也送到/usr/adm/messages 文件。fdformat 自己不说明哪里出错(也不必考虑，软盘很便宜，坏了就扔)。

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... read: Unknown error
$
```

badblocks 命令可用于查找任何磁盘或分区的坏块(包括软盘)。它不格式化磁盘，因此可以用于检查存在的文件系统。下面的例子检查出一张3.5'软盘上的2个坏块：

```
$ badblocks /dev/fd0H1440 1440
718
719
$
```

badblocks 输出发现的坏块的块号。多数文件系统可以避免这样的坏块。他们维护一个已知的坏块列表，在文件系统建立时初始化，并可以在以后修改。初始的坏块查找可由mkfs 命令完成(它初始化文件系统)，以后可以用badblocks 来检查，新的块可以用fsck 加入。后面我们将说明mkfs 和fsck 。

许多新型的硬盘自动发现坏块，并企图用一个特定的、保护的好块来代替它。这对操作系统是不可见的。这种特征应该在硬盘手册的文档中，如果你好奇的话。但即使这样的硬盘也可能失败，如果坏块数量太大的话，虽然如果这样，那硬盘就基本上不能用了。

分区

一个硬盘可分为几个分区。每个分区好象是单独的硬盘。这样，你如果只有一个硬盘，却想安装2个操作系统，你可以把这个硬盘分为2个分区。每个操作系统任意使用自己的分区而不干扰另一个。这种方法，2个操作系统可以在同一硬盘上和平共处。如果没有分区，你只能为每个操作系统购买一个硬盘。

软盘不分区。这没有技术原因，只因为太小，没有必要。CDROM一般也不分区，因为作为一个大盘更易于使用，而且很少有多操作系统的需要。

MBR(主引导记录), 启动扇区和分区表

一个硬盘如何分区的信息存在它的第一个扇区(即第一面第一道第一扇区)。这个第一扇区是硬盘的主引导记录(MBR)；这是计算机启动时BIOS读入和启动的扇区。主引导记录包括一段小程序，读入分区表，检查哪个分区是活动分区(即启动分区)，并读入活动分区的第一个扇区：该分区的启动扇区(MBR也是启动扇区，只不过因为其特殊地位，所以使用特殊的名字)。这个启动扇区包括另一个小程序，读入这个分区(假设是可启动的)上操作系统的第一个部分，然后启动它。

这个分区方案不是内置于硬件和BIOS的，只是许多操作系统遵循的约定。并非所有的操作系统都遵循这个约定，也有例外。有些操作系统支持分区，但他们占领硬盘上的一个分区，然后使用他们自己的内部分区方法管理这个分区。较新的操作系统可以和其他操作系统和平共处(包括Linux)，而无需特殊的措施，但不支持分区的操作系统无法在同一硬盘上与其他操作系统共存。

为安全预防，最好先在纸上写下分区表，这样在错误发生时不会丢失你的文件。(可以使用fdisk 修复坏的分区表)。)相关信息可用fdisk -l 命令给出:

```
$ fdisk -l /dev/hda


Disk /dev/hda: 15 heads, 57 sectors, 790 cylinders
Units = cylinders of 855 * 512 bytes


Device Boot Begin Start End Blocks Id System
/dev/hda1 1 1 24 10231+ 82 Linux swap
/dev/hda2 25 25 48 10260 83 Linux native
/dev/hda3 49 49 408 153900 83 Linux native
/dev/hda4 409 409 790 163305 5 Extended
/dev/hda5 409 409 744 143611+ 83 Linux native
/dev/hda6 745 745 790 19636+ 83 Linux native
$
```

扩展和逻辑分区

PC硬盘的最初的分区方案只允许4个分区。实际使用中这太少了，比如有人想装多于4个操作系统 (Linux, MS-DOS, OS/2, Minix, FreeBSD, NetBSD, Windows/NT等),或有时一个操作系统有多个分区更好，例如由于速度的原因，Linux的对换区最好单独使用自己的分区，而不是在主 Linux分区中(下文详述)。

为克服这个设计问题，发明了扩展分区。这个方法允许将基本分区分为若干子分区，因而被子分区的基本分区称为扩展分区，而子分区称为逻辑分区，他们的表现类似基本分区 ，但产生方法不同。他们之间没有速度差别。

硬盘的分区结构可能类似。这个硬盘被分为3个基本分区，第二个被分为2个逻辑分区。部分硬盘根本没有分区。硬盘是一个整体，每个基本分区有一个启动扇区。

分区种类

分区表(MBR和扩展分区里都有)中，对每个分区，有一个字节指出分区种类。这试图确定使用该分区的操作系统，或用于何操作系统。其目的是避免2个操作系统使用同一分区。可实际上，操作系统并不真的注意分区种类字节；例如，Linux根本不管它是什么。较坏的情况是，有些操作系统错误地使用它：例如有些版本的DR-DOS忽略了它的最高位(MSB)，而其他一些系统则不是。

没有一个标准化组织定义分区种类字节每个值的意义，但一些共同接受的值包括在表 4.1中。相同的列表可以通过Linux的fdisk 命令得到。

给硬盘分区

有许多产生和删除分区的程序。许多操作系统自带，最好使用其自带的，除非要做一些它不能作到的。许多这种程序叫fdisk , 包括Linux, 或其变种。 Linux fdisk 的使用细节可见其Man手册。 cfdisk 命令类似fdisk , 但有更好的用户界面(全屏的)。

使用IDE硬盘时，启动分区(带可启动核心映象文件的分区)必须全在前1024个柱面内。这是因为硬盘通过BIOS启动(在系统进入保护模式前)，而BIOS不能处理多于1024柱面。有时也可能使用部分在前1024柱面的启动分区，这要求所有用BIOS读入的文件都在前1024柱面内。由于这难与安排，因此这是个很差的主意；你不可能知道什么时候核心升级或磁盘碎片整理会导致系统无法启动。因此，应该确认你的启动分区完全在前1024柱面内。

事实上，一些新版的BIOS和IDE硬盘可以处理多于1024柱面。如果你有这样一个系统，你可以忘却这个问题；如果你不能确认，还是把启动分区放在前1024柱面内。

每个分区拥有一块连续的扇区。因为Linux文件系统使用1 kB的块，即2个扇区，所以奇数个扇区会导致最后一个扇区不能使用，这不会有什么问题，但不好，有些版本的 fdisk 会对此给出警告。

改变分区大小一般要求首先备分此分区想保留的所有东西(为防万一，最好备分整个硬盘)，然后删除此分区，产生新分区，最后回存所有东西到新分区。如果是扩大分区，你可能需要调整相邻分区的大小(并备分、回存)。

由于改变分区大小是如此痛苦，最好一次就确定。或拥有一个有效而易用的备分系统。如果你通过无须太多人工干预的介质安装(例如CDROM，而不是软盘)，那么开始可以比较容易地玩玩各种设置。因为你无须备分什么数据，改几次分区大小不会太过痛苦。

有个MSDOS的程序叫fips , 可以无须备分和回存地改变MSDOS分区的大小, 但对其他文件系统，备分回存还是必须的。

设备文件和分区

每个分区和扩展分区有自己的设备文件。这些文件的命名规定是在整个盘的名字加分区号，并约定1-4是基本分区(不管真的有几个基本分区)，5-8是逻辑分区(不管它在哪个基本分区中)。例如，/dev/hda1 是第一个IDE硬盘的第一个基本分区，而 /dev/sdb7 是第二个SCSI硬盘的第三个扩展分区。设备列表 [Anv]给出更详细的信息。

文件系统

什么是文件系统?

文件系统是操作系统用于明确磁盘或分区上的文件的方法和数据结构；即在磁盘上组织文件的方法。也指用于存储文件的磁盘或分区，或文件系统种类。因此，可以说"我有2个文件系统"意思是他有2个分区，一个存文件，或他用 "扩展文件系统"，意思是文件系统的种类。

磁盘或分区和它所包括的文件系统的不同是很重要的。少数程序(包括最有理由的产生文件系统的程序)直接对磁盘或分区的原始扇区进行操作；这可能破坏一个存在的文件系统。大部分程序基于文件系统进行操作，在不同种文件系统上不能工作。

一个分区或磁盘能作为文件系统使用前，需要初始化，并将记录数据结构写到磁盘上。这个过程就叫建立文件系统。

大部分UNIX文件系统种类具有类似的通用结构，即使细节有些变化。其中心概念是超级块superblock, i节点inode, 数据块data block,目录块directory block, 和间接块indirection block。超级块包括文件系统的总体信息，比如大小(其准确信息依赖文件系统)。 i节点包括除了名字外的一个文件的所有信息，名字与i节点数目一起存在目录中，目录条目包括文件名和文件的i节点数目。 i节点包括几个数据块的数目，用于存储文件的数据。 i节点中只有少量数据块数的空间，如果需要更多，会动态分配指向数据块的指针空间。这些动态分配的块是间接块；为了找到数据块，这名字指出它必须先找到间接块的号码。

UNIX文件系统通常允许在文件中产生孔(hole) (用lseek ; 请看手册), 意思是文件系统假装文件中有一个特殊的位置只有0字节，但没有为这文件的这个位置保留实际的磁盘空间(这意味着这个文件将少用一些磁盘空间)。这对小的二进制文件经常发生， Linux共享库、一些数据库和其他一些特殊情况。 (孔由存储在间接块或i节点中的作为数据块地址的一个特殊值实现，这个特殊地址说明没有为文件的这个部分分配数据块，即，文件中有一个孔。)

孔有一定的用处。在笔者的系统中，一个简单的测量工具显示在200MB使用的磁盘空间中，由于孔，节约了大约4MB。在这个系统中，程序相对较少，没有数据库文件。有关这个测量工具的细节请看附录 A.

Filesystems galore

Linux支持多种文件系统。下面是最重要的几个：

minix

最老的，相信是最可靠的，但缺少特色(有些没有时间标记，文件名最长30个字符)，能力有局限(每个文件系统最多64MB)。

xia
minix文件系统的一个修正版本，提升了文件名和文件系统大小的局限，但没有新的特色。不太流行，但据说工作得很好。

ext2
最好的Linux自己的文件系统，也是当前最通用的。其设计易于向上兼容，所以新版的文件系统代码无需重做已有的文件系统。

ext
ext2的老版，且不向上兼容。难于用新版安装程序安装，大部分人都改用ext2。

另外，支持多种其他现存的外围文件系统，很容易与其他外围文件系统交换文件。这些外围文件系统好象是自己的一样，除了可能缺少一些一般UNIX的特征，或有些不同的局限。

msdos
与MSDOS、OS/2等的FAT文件系统兼容。

umsdos
Linux下的扩展msdos文件系统驱动，支持长文件名、所有者、允许权限、连接和设备文件。允许一个普通的msdos文件系统用于Linux，而无须为Linux建立单独的分区。

iso9660
标准CDROM文件系统，通用的Rock Ridge增强，允许长文件名。

nfs
网络文件系统，允许多台计算机之间共享文件系统，易于从所有这些计算机上存取文件。

hpfs
OS/2文件系统。

sysv
SystemV/386, Coherent, 和Xenix文件系统。

根据情况选择文件系统。如兼容性或其他原因必需使用非Linux文件系统，那就必须用。如果可以自由选择，可能最明智的选择是ext2，因为它拥有全部特征而无须忍受性能缺陷。

还有proc文件系统, 一般在/proc 目录, 它不是一个真正的文件系统，虽然好象是。proc文件系统使用户易

于存取全部核心数据结构，比如进程列表。它使这些数据结构看起来象个文件系统，且此文件系统可以用所有一般的文件工具操作。例如，要得到所有进程的列表，可以使用命令

```
$ ls -l /proc
total 0
dr-xr-xr-x 4 root root 0 Jan 31 20:37 1
dr-xr-xr-x 4 liw users 0 Jan 31 20:37 63
dr-xr-xr-x 4 liw users 0 Jan 31 20:37 94
dr-xr-xr-x 4 liw users 0 Jan 31 20:37 95
dr-xr-xr-x 4 root users 0 Jan 31 20:37 98
dr-xr-xr-x 4 liw users 0 Jan 31 20:37 99
-r--r--r-- 1 root root 0 Jan 31 20:37 devices
-r--r--r-- 1 root root 0 Jan 31 20:37 dma
-r--r--r-- 1 root root 0 Jan 31 20:37 filesystems
-r--r--r-- 1 root root 0 Jan 31 20:37 interrupts
-r-------- 1 root root 8654848 Jan 31 20:37 kcore
-r--r--r-- 1 root root 0 Jan 31 11:50 kmsg
-r--r--r-- 1 root root 0 Jan 31 20:37 ksyms
-r--r--r-- 1 root root 0 Jan 31 11:51 loadavg
-r--r--r-- 1 root root 0 Jan 31 20:37 meminfo
-r--r--r-- 1 root root 0 Jan 31 20:37 modules
dr-xr-xr-x 2 root root 0 Jan 31 20:37 net
dr-xr-xr-x 4 root root 0 Jan 31 20:37 self
-r--r--r-- 1 root root 0 Jan 31 20:37 stat
-r--r--r-- 1 root root 0 Jan 31 20:37 uptime
-r--r--r-- 1 root root 0 Jan 31 20:37 version
$
```

(可能有些文件与进程不符。上面的例子被简短了。)

注意虽然叫文件系统，proc文件系统没有一个部分与磁盘有关，它只在核心映象中存在。任何人任何时候想看proc文件系统的任何部分,核心使它看起来好象这部分在什么地方存在(虽然没有)。因此，虽然/proc/kcore文件有好多兆字节，但它根本没用任何磁盘空间。

应该用哪个文件系统?

一般没有什么理由用许多不同的文件系统。当前，ext2fs是最流行的，可能是最明智的选择。根据记录结构、速度、(感觉的)可靠性、兼容性和其他不同的理由，适当地使用其他文件系统。个别情况需要个别决定。

建立文件系统

用mkfs 命令建立文件系统，即初始化。实际上，对每个不同种类的文件系统有一个单独的程序。 mkfs 只是为了建立不同文件系统种类确定运行不同程序的一个前端。用-t fstype选项选择种类。

被mkfs 调用的程序有不同的命令行接口。最通用和最重要的选项如下，细节请看手册。

-t fstype
选择文件系统种类。

-c
查找坏块，初始化坏块列表。

-l filename
从文件filename读入坏块列表。

用如下命令在软盘上产生ext2文件系统：

```
$ fdformat -n /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
$ badblocks /dev/fd0H1440 1440 > bad-blocks
$ mkfs -t ext2 -l bad-blocks /dev/fd0H1440
mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done
$
```

首先，格式化软盘(-n选项不进行确认，即坏块检查)。然后用badblocks 查找坏块, 输出定向到文件bad-blocks。最后，产生文件系统，坏块列表由文件badblocks 初始化。
-c选项可以与mkfs 一起使用，而无须badblocks 和一个单独的文件。如下：

```
$ mkfs -t ext2 -c /dev/fd0H1440
mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group
```

```
Checking for bad blocks (read-only test): done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
$
```

使用-c比用单独使用badblocks 更方便, 但建立文件系统后检查，badblocks 是必要的。
在硬盘或分区上准备文件系统的过程和软盘是一样的，除了无须格式化。

Mount和unmount

一个文件系统可以使用之前，必须mount。操作系统然后做一些记录以确认正常。因为UNIX所有的文件在一个目录树中， mount操作的结果使新的文件系统的内容好象在某个已经mount的文件系统的一个已经存在的子目录中。

上面的mount可能使用如下命令：

```
$ mount /dev/hda2 /home
$ mount /dev/hda3 /usr
$
```

mount 命令使用2个参数。第一个是与包括文件系统的磁盘或分区相关的设备文件。第二个是要mount到的目录。 mount以后，这2个文件系统的内容好象是/home 和/usr 目录。这样就可以说：/dev/hda2 被mount到 /home , /usr 也同样。要看每个文件系统，就看其被安装的目录，好象它就是在那里。注意设备文件的区别, /dev/hda2 , 和安装到的目录/home 。设备文件给出硬盘原始内容的存取，安装到的目录给出磁盘上文件的存取。安装到的目录叫安装点。

Linux支持许多文件系统。mount 会试着猜测文件系统种类。也可以使用-t fstype 选项直接定义种类；这有时是必要的，因为自检测mount 并非总能成功。例如要mount一个MSDOS软盘，可以用如下命令：

```
$ mount -t msdos /dev/fd0 /floppy
$
```

安装点目录不必是空的，但必须存在。其中的所有文件当文件系统mount后将不可用名字存取(已经打开的文件将继续可存取。有其他目录硬连接的文件可以通过那些名字存取)。这没有坏处，反而可能更有用。例如, 有人喜欢将/tmp 和/var/tmp 作为同义，将/tmp 作为/var/tmp 的符号连接。系统启动时，在/usr 文件系统被mount之前，使用驻留在根文件系统的 /var/tmp 目录。当/usr 被mount上以后，根文件系统上的/var/tmp 将不可用，如果根文件系统上不存在 /var/tmp ，那么在mount上/var 之前将不可能使用暂存文件。

如果不打算在一个文件系统上写任何东西，可以使用mount 的-r开关做一个只读mount。这将使核心停止任何对此文件系统的写要求，也将停止核心的对i节点的文件存取时间的更新。只读mount对不可写介质是必要的，例如CDROM。

细心的读者可能已经注意到一个小的逻辑问题。第一个文件系统(叫根文件系统，因为它包含根目录)如何mount，因为很明显，它不能mount到另一个文件系统? Well, the answer is that it is done by magic. The root filesystem is magically mounted at boot time, and one can rely on it to always be mounted-- 如果根文件系统

不能mount，系统将不能启动。 The name of the filesystem that is magically mounted as root 被编译进核心，或用LILO或rdev 设置。

根文件系统通常先被只读mount。然后启动手稿运行fsck 校验它的有效性，如果没有问题，将re-mount它，使之可写。fsck 不能运行于一个已mount的文件系统，因为fsck 运行时，任何文件系统的改变将导致错误。因为根文件系统在被检查时是只读，fsck 可以无虑地修复任何问题，因为re-mount 操作将刷新文件系统在内存中的所有数据。

在有其他文件系统的许多系统中，启动时要自动mount，可以在/etc/fstab 文件中定义：文件格式细节请参考fstab 的手册页。 mount特别的文件系统的特别细节依赖于许多因素，可以根据需要由每个管理员设置。When the chapter on booting is finished, you may read all about it there.

当一个文件系统不需要再mount着，可以用umount . umount 加一个参数unmount它，参数可以是设备文件或安装点。例如，要unmount上面例子中的目录，可以用：

```
$ umount /dev/hda2
$ umount /usr
$
```

要了解使用这个命令的更多的说明，参阅手册。注意：记住unmount已经mount的软盘， 而不能仅仅将软盘弹出软驱!由于磁盘缓冲，在你unmount软盘之前无须回写，因此过早取出软盘将导致内容不正确。只从软盘上读还不要紧，如果写，就可能发生灾难性的损失。

mount和umount需要超级拥护特权，即只有root 用户可以做。原因是：如果任何用户都可以mount软盘到任何目录，那么很容易用软盘做，比如，用特洛伊木马替换/bin/sh , 或者其他常用的程序。但是允许用户使用软盘经常又是必要的，有几种方法：

给用户root 口令，很明显这对安全不利，但是最简单的方法。如果没有安全要求，这个方法很好，比如在非网络的、个人系统上。

使用一个程序比如sudo 允许拥护使用mount。这同样对安全不利，但没有直接给任何人超级用户特权。

让用户使用mtools , 这是一个利用MSDOS文件系统的软件包，无须mount。如果是MSDOS软盘这样做很好，否则不好。

在/etc/fstab 中用合适的选项列出软驱设备和允许的安装点。

最后一个选择可以在/etc/fstab 文件中加类似下面的一行来完成：

/dev/fd0 /floppy msdos user,noauto 0 0

各列分别是：要mount的设备文件，要安装到的目录，文件系统类型，选项，备份频率(用于dump) 和fsck次序(定义启动时文件系统被检查的次序，0表示不检查)。

noauto选项使系统启动是不自动mount(即, it stops mount -a from mounting it)。 user允许任何用户mount这个文件系统，并且，由于安全原因，不允许执行程序(normal or setuid) and interpretation of device files from the mounted filesystem。这样，任何用户都可以用如下命令mount一个msdos文件系统的软盘：

```
$ mount /floppy
$
```

软盘可以用相关的umount 被unmount。

如果想提供多种软盘的存取，需要给出多个安装点。对每个安装点的设置可以不同。例如，提供MSDOS和 ext2文件系统的存取，可以在/etc/fstab 文件中加如下行:

```
/dev/fd0 /dosfloppy msdos user,noauto 0 0
/dev/fd0 /ext2floppy ext2 user,noauto 0 0
```

对于MSDOS文件系统(不仅是软盘)，可能需要用uid, gid,和umask 文件系统选项来限制存取权限，请看mount 手册页。如果不小心，mount一个MS-DOS文件系统将给予任何用户至少是读权限，这可不是一个好主意。

用fsck检查文件系统完整性

文件系统很复杂，因此易于发生错误。可以用fsck 命令检查文件系统是否正确和有效。它可以根据指令修复找到的小错误，并将未修复错误报告用户。幸运的是，文件系统的代码非常有效，所以根本极少出现问题，并且问题通常原因是电源失败、硬件失败、或操作错误，例如没有正常关闭系统。

大多数系统设置为启动时自动运行fsck ，因此任何错误将在系统使用前被检测到(并根据希望修正)。使用有错误的文件系统可能使问题变得更坏：如果数据结构有问题，使用这个文件系统可能使之更糟，导致更多的数据丢失。当然，在大的文件系统上运行fsck 会花一定的时间，如果系统正常关闭，几乎从不发生错误，因此有一些方法可以不进行检查。如果文件/etc/fastboot 存在，就不检查。另外，如果ext2文件系统在超级快中有一个特定的标记告知该文件系统在上次mount后没有正常unmount. 如果标记指出unmount正常完成(假设正常unmount指出没问题)，e2fsck (fsck 的ext2文件系统版) 就不检查系统。/etc/fastboot 是否影响系统依赖于你的启动手稿，但ext2标记则在你使用e2fsck 时发生作用--基于一个e2fsck 选项(参阅e2fsck 手册页)

自动检查只对启动时自动mount的文件系统发生作用。使用fsck 手工检查其他文件系统，比如软盘。

如果fsck 发现为修复的问题，你需要深入了解文件系统的一般工作原理和有问题的文件系统的细节，或好的备份。最后一个办法容易(虽然冗长)安排，如果你自己不知道，有时可以通过朋友、Linux新闻组、电子邮件列表或其他支持源安排。我很想告诉你更多，但我对这的学习和实践也并不多。Theodore T'so的debugfs 程序

应该有用。

fsck 只能运行于未mount的文件系统，不要用于已mount的文件系统(除了启动时的只读根文件系统)。这是因为它存取原始磁盘，在操作系统不知道的情况下修改文件系统。 There will be trouble, if the operating system is confused.

用badblocks检查磁盘错误

应该周期性地用badblocks 命令检查坏块它输出找到的所有坏块的编号的列表。列表给fsck 记录在文件系统数据结构中，使操作系统存储数据时不使用这些坏块。举例：

```
$ badblocks /dev/fd0H1440 1440 > bad-blocks
$ fsck -t ext2 -l bad-blocks /dev/fd0H1440
Parallelizing fsck version 0.5a (5-Apr-94)
e2fsck 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Check reference counts.
Pass 5: Checking group summary information.

/dev/fd0H1440: ***** FILE SYSTEM WAS MODIFIED *****
/dev/fd0H1440: 11/360 files, 63/1440 blocks
$
```

如果badblocks报告一个块已经使用，e2fsck 将试着将此块移到其他地方。如果该块真的坏了，而不是在坏块边缘，文件内容可能丢失。

Fighting fragmentation

文件写到磁盘时，不一定在连续的块中。没在连续块中的文件叫碎片。因为磁盘的读写头回更多地移动，读碎片文件会花较长的时间。虽然如果有好的读前缓冲系统不会有什么问题，但最好还是避免碎片。

Ext2文件系统试图使碎片最少，即使不能将一个文件的所有块存在连续扇区中，也尽量靠近。 Ext2通常有效地安排里文件其他块最近的空闲块，因此很少需要关心碎片问题。 Ext2文件系统有一个消除碎片程序，请看参考书目中的 [TV]。

有许多MSDOS消碎片程序在文件系统中移动块以消除碎片。其他一些文件系统，消碎片必须通过备份-重产生-回存文件系统来完成。对于所有文件系统，消碎片应该备份文件系统，因为很多原因可能在消碎片过程中导致错误。
对所有文件系统的其他工具

一些其他工具对管理文件系统有用。 df (Disk Free)显示一个或多个文件系统的空闲磁盘空间。 du (Disk Usage)显示一个目录和其内的所有文件使用了多少磁盘空间。这用于发现磁盘空间浪费。

sync 强制将磁盘缓冲的所有未写块写入磁盘(见 5.6)。这一般无须手工完成，由守护进程update 自动完成。这在有些情况下很有用，例如，如果update 或其辅助进程bdflush 死了，或你不能等运行update 必须马上关闭电源。

对ext2文件系统的其他工具

除了产生文件系统的mke2fs 和检查文件系统的e2fsck 直接或通过与文件系统类型无关的前端存取外，Ext2文件系统还有几个有用的工具。

tune2fs 调整文件系统参数。一些有趣的参数有：

最大mount数。当文件系统被mount过多少次以后，即使标志是干净的，e2fsck 强制检查。对用于开发或测试的系统，应该降低这个限制数。

最大检查间隔。到达这个间隔时间，即使标志是干净的，e2fsck 强制检查。如果文件系统不是经常mount，可以不使能这个功能。

保留给root的块数。Ext2给root保留一些块，这样如果文件系统满了，还可能无须删除任何东西做系统管理。保留量确省是5%，这在大多数磁盘上不会造成浪费。当然，软盘没有理由保留块。

dumpe2fs 显示一个ext2文件系统的信息，大部分来源于超级块。有些输出信息是技术性的，要求对文件系统工作的理解(见附录), 但许多即使是一般管理员是也易于理解的。

debugfs 是一个文件系统调试器。它允许直接存取磁盘上的文件系统数据结构，可用于修复fsck 不能自动修复的磁盘。它也可用于恢复被删除的文件。但是，debugfs 非常要求你理解你所干的事，错误的理解和操作将破坏你的所有数据。

dump 和restore 可用于备份一个ext2文件系统。它们是传统UNIX备份工具的ext2版。关于更有关备份的信息见 9章。

没有文件系统的磁盘

并非所有磁盘或分区都作为文件系统使用。例如对换分区，就没有文件系统。许多软盘作为磁带仿真使用，所以tar 或其他文件可以直接写到原始磁盘，而不是文件系统。Linux启动软盘不包括文件系统，只是原始核心。

不用文件系统的优点是有更多的磁盘可用空间，因为文件系统需要一些记录。也更容易与其他系统兼容；例如tar 文件格式在所有系统上相同，而文件系统则在大多数系统上不同。如果需要，你会很快使用没有文件系统的磁盘。可启动的 Linux软盘无需文件系统，虽然有也可能。

使用原始磁盘的一个原因是做映象拷贝。比如，如果磁盘包含部分损坏的文件系统，那么在修复前做一个完全拷贝是个好主意，因为如果你修错了，可以重来。做映象拷贝的一个方法是用 dd :

```
$ dd if=/dev/fd0H1440 ō f=floppy-image
2880+0 records in
2880+0 records out
$ dd if=floppy-image ō f=/dev/fd0H1440
2880+0 records in
2880+0 records out
$
```

第一个dd 给软盘做了一个完全映象到文件 floppy-image , 第二个把映象写到软盘。 (假设用户在第二个命令前换了软盘。否则这个命令对可能没用。)

分配磁盘空间

分区概要

用最好的方式给磁盘分区不容易，而且，没有一个通用的正确方法，这包括很多因素。

传统的方法是有个(相对)小的根文件系统，包括 /bin , /etc , /dev , /lib , /tmp , 和其他系统启动和运行需要的东西。这种方法，根文件系统(在它自己的分区或硬盘上)是所有系统启动需要的东西。理由是如果根文件系统小而不常用，系统崩溃时它就不太容易损坏，而且崩溃时也易于修复。然后给/usr 目录树、用户主目录(经常在/home)、对换空间产生单独的分区或使用单独的硬盘。分离的用户主目录(存用户文件)在其自己的分区中易于备份，因为一般无须备份程序(/usr 中)。网络环境中，这样可以使多台计算机共享/usr (例如使用NFS) 这样每台机器可以节约数十、数百兆的磁盘空间。

多分区的问题是将整个磁盘的空闲空间分割成若干小片。现在，由于磁盘和操作系统已经很可靠，许多人更倾向与一个分区存所有文件。当然，这样可能比备份、回存小分区痛苦些。

对于小硬盘(假设你不做核心开发)，最好的方法可能是只要一个分区。对于大硬盘，分几个大分区可能更好。尤其在某种情况下出现错误时。 (注意这里说的小和大是相对的，根据你对磁盘空间的需求而言。)

如果你有多个硬盘，你可能想让根文件系统(包括/usr) 在一个上，而用户主目录在另一个上。

最好准备尝试几个不同的分区方案(over time, not just while first installing the system)。这有些工作量，因为这其实是从头安装系统若干遍，但这是确认正确的唯一方法。

空间要求

你安装的Linux给出一些对不同配置所需磁盘空间的指示。单独安装的程序可能也是。这能帮助你计划你的磁盘使用，但你应该为以后可能的需求保留一些额外空间。

拥护文件总量基于你的用户希望。许多人好象想要他们所有可能的文件量，但多多益善。有些人只有很少的文字处理，也许几兆就够，而有些人可能需要上GB的空间做图象处理。

顺便说一句，用KB或MB比较文件大小和用MB给出的磁盘空间时，应当注意这2者的可能的不同。一些硬盘制造商喜欢称1000字节为1KB，1000KB为1MB，而计算机世界的其他地方都以1024为因数。因此我的MB硬盘实际只是　330MB硬盘。

对换空间在5.5章讨论。

硬盘分配举例

我原来有个109MB的硬盘，现在我用一个330MB硬盘。我解释一下我如何分区这些硬盘并说明为什么。

当我的需要和操作系统变化时，我用不同的方法分区109MB硬盘。我说明2种方案。首先，我曾和Linux一起运行MSDOS，为此，我需要大约20MB给MSDOS、C编译器、编辑器、一些其他工具、我工作的程序、和足够的空闲空间。给Linux开了10MB对换分区，其他79MB作为一个分区给Linux。我曾试验给出单独的根，/usr , 和/home ， 但这样就没有什么空闲空间干什么有趣的事了。

当我不再需要MSDOS，我重新分区，12MB对换分区，其他是一个单独的文件系统。

从头分区是为了玩玩要求自己分区的一些东西，例如试试不同的Linux，或比较文件系统的速度。当没有这些需要后，就把它作为对换区(我喜欢打开好多窗口)。

给Linux增加更多的磁盘空间

给Linux增加更多的磁盘空间很容易，至少在硬件都安装好后(硬件安装不在本书所述的范围)。如果需要，先格式化，然后产生分区和上面说过的文件系统，在/etc/fstab 中加入正确的行使之能自动mount。

节约磁盘空间的提示

节约磁盘空间的最好提示是不要安装不必要的程序。许多Linux distributions给出安装其所带软件包某些部分的选择，分析你的需求你可能发现好多你并不需要。这会节约很多磁盘空间，因为许多程序需要很大空间。即使你需要某部分包或程序，也不一定需要其全部。例如有些在线文档可能不必要，有些GNU Emacs的Elisp文件　, 有些X11的字体，或者有些编程库。

如果你不能卸装包，你可以压缩。如gzip 或zip 的压缩程序可以压缩/解压文件或文件群。gzexe 系统可以对用户透明地压缩/解压程序 (没用的程序被压缩，当被使用时解压)。实验中的DouBle 系统对程序透明地压缩文件系统中的所有文件。(如果你熟悉例如Stacker for MS-DOS等产品，原理是一样的。

内存管理

发布时间 :2007-01-27 23:12:22

本章说明Linux的内存管理特征，即虚拟内存和磁盘缓存。描述系统管理员应该考虑的东西、工作和目的。

什么是虚拟内存?

Linux支持虚拟内存, 就是使用磁盘作为RAM的扩展，使可用内存相应地有效扩大。核心把当前不用的内存块存到硬盘，腾出内存给其他目的。当原来的内容又要使用时，再读回内存。这对用户全透明：运行于Linux的程序只看到大量的可用内存而不甘心哪部分在磁盘上。当然，读写硬盘比真的内存慢(慢千倍)，所以程序运行较慢。用做虚拟内存的这部分硬盘叫 对换空间。

Linux可以使用文件系统中的普通文件或单独的分区作为对换空间。对换分区更快，但对换文件更易于改变大小(无须对硬盘重分区)。如果知道要多少对换空间，应该用对换分区；如果不能确认，可以先用对换文件，用一段时间后再根据所需空间建立对换分区。

Linux允许同时使用多个对换分区和/或对换文件。即如果偶尔需要更多的对换空间，可以随时建立一个额外的对换文件。

产生对换空间

对换文件是普通文件，对核心没有什么特别的。唯一不同是它没有孔，用 mkswap 准备。必须在本地盘上，不能在通过NFS mount的文件系统中。

关于孔，是重要的。对换文件保留了磁盘空间，使核心能快速对换出一页，而不必经过如文件的定位磁盘扇区的全部事情。核心只用分配给这个文件的所有扇区。由于文件中的孔意味着没有为文件中这个位置分配磁盘扇区，这对核心使用不利。

产生没有孔的对换文件的一个好办法是通过如下命令：

```
$ dd if=/dev/zero  ō f=/extra-swap bs=1024 count=1024
1024+0 records in
1024+0 records out
$
```

/extra-swap 是对换文件名，大小由count=给出. 大小最好是4的倍数，因为核心写出的内存页是4KB。如果不是4的倍数，最后那几KB将不可用。

对换分区也没什么特别。就象产生其他分区一样产生；唯一的不同是它作为原始分区使用，即没有任何文件系统，最好将对换分区标记为类型82(Linux swap)，虽然这对核心没有影响，但这使分区列表更清晰。

产生对换文件或对换分区后，需要写个标记起用它，这包括核心要用的一些管理信息。命令是 mkswap，用法如下：

```
$ mkswap /extra-swap 1024
Setting up swapspace, size = 1044480 bytes
$
```

注意对换空间现在还没用，它存在，但核心还没用它提供虚拟内存。

请一定小心使用mkswap , 因为它不检查文件或分区是否被其他东西使用。 你可能用mkswap 很容易地覆盖了重要文件和分区! 幸好，你只需在你安装系统时使用mkswap 。

Linux内存管理限制了每个对换空间约为127MB(由于技术原因，实际限制是127.6875MB)。 可以同时使用最多16个对换空间，总计差不多2GB。

使用对换空间

用swapon 将一个初始化的对换空间可用。此命令告诉核心对换空间可以用了，对换空间的路径作为参数，启动一个临时对换文件可以用如下命令：

```
$ swapon /extra-swap
$
```

对换空间如果列入/etc/fstab ，就可自动使用。

```
/dev/hda8 none swap sw 0 0
/swapfile none swap sw 0 0
```

启动手稿运行命令swapon -a, 它将启动/etc/fstab 中所列的所有对换空间。因此swapon 命令只有在启动额外的对换空间时才使用。

可以用free 监视对换空间的使用，它将给出所有使用的对换空间。

```
$ free
total used free shared buffers
Mem: 15152 14896 256 12404 2528
```

-/+ buffers: 12368 2784
Swap: 32452 6684 25768
$

前一行输出(Mem 显示物理内存。 Total列不显示核心使用的物理内存(通常大约1MB)。Used列显示被使用的内存总额(第二行不计缓冲)。 Free列显示全部没使用的内存。Shared列显示多个进程共享的内存总额。 Buffers列显示磁盘缓存的当前大小。
后一行(Swap 对对换空间，显示的信息类似上面。如果这行为全0，那么没使用对换空间。

通过top ，或使用proc文件系统的/proc/meminfo 文件可以得到相同的信息。得到某个对换空间的使用信息目前还比较困难。

可用swapoff 取消对换空间，一般不必这样，除非是临时对换空间。对换空间中的要用的页被换入(swap->RAM)，如果没有足够的物理内存，就被换出(RAM->swap，到其他对换空间)。如果没有足够的虚拟内存放进所有页面，Linux将开始震荡(thrash); 很长时间以后应该能恢复，但此时系统不可用。取消一个对换空间前，应该检查(例如用free)是否有足够的物理内存。

用swapon -a自动使用的所有对换空间可以用swapoff -a取消。它查看文件/etc/fstab 得知要取消什么。任何手工起用的对换空间将依然使用着。

即使有许多空闲的物理内存，有时许多对换空间也被使用着。这种情况是由于在某个时间需要对换，但后来一个占用大量物理内存的大进程终止并释放了内存。直到被换出的数据要被使用之前它们并不自动换入。不必顾虑这种情况，但知道为什么会发生这种情况会更安心。

与其他操作系统共享对换空间

许多操作系统内置虚拟内存。由于他们只需在运行时使用，即，不会同时，那么除了当前运行的，其他所有对换空间都浪费着。如果他们共享同一个对换空间将更有效。这是可能的但需要一些Hacking工作。 Tips-HOWTO包含了一些如何完成这项任务的忠告。

分配对换空间

也许有人告诉你，应该分配2倍于物理内存的对换空间，但这是个虚假的规律。下面说明如何正确：

估计你的全部内存需求。这是你可能需要的最大量，即你要同时运行的所有程序所需的内存要求的总和。你可以同时运行你可能同时运行的所有程序试试。

例如，如果你想运行X，你得分配8MB给他，gcc要求数MB(有些文件偶尔可能需要很大量，数十MB，但一般4MB差不多)，等等。核心自己使用1MB，Shell和一些小工具可能需要几百KB(或说，总共1MB)。不必太精确，粗略估计就行，但可以较悲观地考虑。

记得如果将有多人同时使用系统，他们将都消耗内存。如果2个人同时运行相同的程序，总内存消耗一般并非加倍，因为代码页和共享库是单一的。

free 和ps 命令对估计内存需求很有用。

第一步的估计加上一些安全量。因为对程序大小的估计很可能是错误的，因为你可能忘了一些要运行的程序，并确定你有一些额外空间。应该有数MB。(分配太多对换空间比分配太少好，但不必过分，因为不使用的对换空间是浪费；见后文：关于增加对换空间。) Also,since it is nicer to deal with even numbers, you can round the value up to the next full megabyte.

基于以上计算，你知道了你总共需要多少内存。减去你的实际物理内存，就是对换空间。 (有些版本的UNIX中，你还需要分配物理内存的映象空间，所以第二布中计算的你所需的空间就不能减)

如果你计算的对换空间比你的实际物理内存大得多(大于好几倍以上)，那么你也许需要更多的物理内存，否则系统性能将太低。

即使计算显示你无须对换空间，最好还是至少有一些。Linux有些侵略性地使用对换空间，这样保持一定的空闲物理内存。即使内存还不为什么程序所需，Linux也会换出一些不用的内存页，这样在需要的时候就可以避免因对换的等待--即对换可以在硬盘空闲的时候提早完成。

对换空间可以分在几个硬盘中，这有时可以提高性能，依赖于这些盘的相对速度和存取模式。你可以尝试几中方案，但要知道正确地尝试是很困难的。不要相信某种方案比其他方案好的断定，因为它不会总是对的。

高速缓存

与存取(真正的)内存相比，从磁盘读是很慢的 另外，在相对短的一端时间里，多次读硬盘相同的部分是很常见的。例如，你可能先读了一封电子邮件，然后回复时又将它读入编辑器，然后复制它到一个文件夹时又用邮件程序读它。或者，考虑命令ls 可能被系统上的很多用户多么频繁地使用。只从磁盘读一次信息，并保持在硬盘中，知道不再需要，除了第一次读，其他都会较快。这就叫磁盘缓存disk buffering，用于此目的的内存叫buffer cache。

不幸的是，由于内存是有限且缺乏的资源，buffer cache一般不会足够大(大到能够装下所有人可能用到的数据)。当cache满时，最长时间不用的数据将被丢弃，内存释放给最新的数据。

磁盘缓冲也用于写操作。要写的数据经常马上又被读(例如一个源代码文件保存到文件中后又被编译器读出)，所以将要写的数据放在缓冲里是个好主意。另外，只将数据放如cache而不马上写到磁盘，写操作的程序执行速度更快。写操作然后可以在后台完成，而不降低其他程序的速度。

许多操作系统有buffer caches (即使名称不同),但并非都根据上述原理。有些是透写write-through: 数据马上写到磁盘(当然也同时写到cache) 不马上写的cache叫回写write-back。回写比透写更有效，但也更容易出错：如果系统崩溃，或电源突然掉电，或软盘在cache回写前被取出，那么cache中改变的数据将丢失。这可能意

味着文件系统is not in full working order, 可能由于未写数据包含了系统记录信息的重要的变化。

因此，千万不要不经过正常的关闭过程直接关闭电源(见6章), 或没有unmount就取出软盘(如果是mount的)，或什么程序还在用着软盘，或软盘灯还在闪。 sync 命令刷新缓冲，即强制将所有未写数据写回磁盘，如果要确保所有数据安全回写，可以用它。传统的UNIX系统中，有个update 程序在后台运行，它每30秒运行一次 sync ，所以通常无须使用sync 。 Linux有一个另外的守侯程序bdflush ，它克服了sync 有时因磁盘I/O负荷太重(因为频繁的操作)而导致有时系统突然呆住的问题。

Linux下，bdflush 由update 启动。一般无须考虑它，但如果bdflush 偶尔因为什么原因死了，核心会给出警告，此时应该手工启动它(/sbin/update)。

cache并不真正缓冲文件，而是块，就是磁盘I/O的最小单元(Linux下，一般是1kB)。这样，所有的目录、超级块、其他文件系统记录数据和无文件系统磁盘都可以被缓冲。

cache的效果决定于其大小。太小的cache几乎无用；它只能cache很少的数据，而可能在被重用前就被清除了。大小有赖于有多少数据被读写，相同的数据的存取频度。唯一的方法是实验。

如果cache是固定大小，那么不应该太大，否则，会由于空闲内存空间太小而使用swap(也很慢)。为了最有效地使用真实内存，Linux自动使用所有空闲内存作为buffer cache，当程序需要更多内存时，自动减少cache。

Linux下，对cache使用无须做任何工作，它完全是自动的。除了要正常关闭系统和取出软盘，无须关心cache。

引导和关机

发布时间 :2007-01-27 23:12:46

本节说明当Linux系统引导和关机时发生了什么,应该任何正确完成. 如果没有遵循正确的过程, 文件可能损坏或丢失.

引导和关机概述

开启计算机并导致其操作系统被加载的过程 叫引导. The name comes from an image of the computer pulling itself up from its bootstraps, but the act itself slightly more realistic.
启动过程中,计算机首先加载了一小段叫 bootstrap loader的程序,它依次加载和启动操作系统, bootstrap loader通常存储在硬盘或软盘的固定的位置 . 这2步过程的理由是操作系统大而复杂,而计算机加载的第一段代码很小(几百字节),以免使固件不必要地复杂化.

不同的计算机的bootstrap不同. 对于PC, 计算机(它的BIOS)读软盘或硬盘的第一个扇区(叫 引导扇). bootstrap loader包含在这个扇区中. 它加载位于磁盘(和其他)的其他地方的操作系统.

Linux加载后, 它创始化硬件和设备驱动, 然后运行 init . init 启动其他进程以允许用户登录和做其他事情. 这部分的细节在下面讨论.

为了关闭一个Linux系统, 首先所有进程被告知结束(这使他们关闭所有文件, 完成必要的其他事情, 使之整齐地结束), 然后unmount文件系统和对换区, 最后打印可以关掉电源的信息到控制台. 如果没有遵循正确的过程, 可怕的事情可能发生 . 最重要的, 文件系统缓冲cache可能没有回写, 这意味着其中的所有数据将丢失, 磁盘上的文件系统不完整, 并可能不可用.

近观引导过程

可以从软盘或硬盘引导Linux. 安装和开始指南的安装一节 ([Wel]) 告诉你如何安装Linux, 并按你希望的方式引导.

当PC引导后, BIOS做一些测试保证一切正常, 然后开始真正的引导. 它选择一个磁盘(通常是第一个软驱, 如果有软盘的话, 否则就是第一个硬盘, 如果安装了的话; 顺序是可设置的). 然后读第一个扇区, 这叫引导扇; 对于硬盘, 也叫主引导记录, 因为硬盘可以包含多个分区, 每个分区都有自己的引导扇.

引导扇包含一个小程序(小到可以存入一个扇区), 它的责任是从磁盘读入真正的操作系统并启动之. 从软盘启动Linux时, 引导扇包含的代码只读前数百个数据块(当然, 依赖于核心的大小)到预定的内存位置. Linux引导软盘上, 没有文件系统, 核心存在连续的扇区中, 因为这样简化了引导过程. 当然, 使用LILO(LInux LOader)可以从

文件系统引导.

从硬盘引导, 主引导记录的代码检查分区表(也在主引导记录扇区中), 确认活动分区(标记为可引导的分区), 从该分区读引导扇区 , 然后启动该引导扇区的代码. 该分区的引导扇区的代码做与软盘所做的相同: 从该分区读入核心并启动. 但细节不同, 因为一般只给核心映象做一个单独的分区是没什么用的, 所以分区引导扇中的代码不能只顺序地读磁盘, 它必须找到文件系统把它们放在哪些扇区中. 有几个方法解决这个问题, 但最通常的方法是使用LILO. (关于如何做的细节与这里的讨论无关; 更多的信息请看LILO文档, 它很全面)

用LILO引导时, 它读入并引导缺省核心. 也可以设置LILO, 使之能引导若干个核心之一, 甚至其他操作系统, 也可以在引导时让用户选择引导哪个核心或操作系统 . LILO可以设置为如果有人在引导时按住 alt, shift, or ctrl键 (LILO启动时), LILO将不立即引导缺省的而问用户引导哪个. LILO可以设置为带一个timeout选项并询问, 当超时时 , 就引导缺省核心.

META: 除了LILO还有其他的引导载入程序, 如loadlin, 它们的信息将在下一版本中给出.

从软盘和硬盘启动各有优势, 但通常从硬盘启动更好, 因为这避免了关于软盘的争论. 而且快. 然而, 安装相同从硬盘启动可能有更多的麻烦, 因此很多人先用软盘引导, 然后当相同工作很好后, 再安装LILO从硬盘引导.

Linux核心被读入内存后, 才真正启动了, 概述如下:

Linux核心是被压缩安装的, 所以它首先得解压自己. 核心映象开头包括一个解压的小程序.

如果你有Linux可识别的super-VGA卡, 且支持一些特殊的文本模式(如100列40行), Linux会问你要用哪个模式. 编译核心时, 可能预定了一个视频模式, 就不会问了. 这也可以用LILO或 rdev 完成.
然后, 核心检查还有什么其他硬件(硬盘, 软盘, 网卡...), 并配置适当的设备驱动; 同时, 输出查找结果的信息. 例如, 我引导时, 得到类似如下信息:

```
LILO boot:
Loading linux.
Console: colour EGA+ 80x25, 8 virtual consoles
Serial driver version 3.94 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16450
tty01 at 0x02f8 (irq = 3) is a 16450
lp_init: lp1 exists (0), using polling driver
Memory: 7332k/8192k available (300k kernel code, 384k reserved, 176k data)
Floppy drive(s): fd0 is 1.44M, fd1 is 1.2M
Loopback device init
Warning WD8013 board not found at i/o = 280.
Math coprocessor using irq13 error reporting.
Partition check:
hda: hda1 hda2 hda3
VFS: Mounted root (ext filesystem).
Linux version 0.99.pl9-1 ( root@haven) 05/01/93 14:12:20
```

精确的文本在不同系统上不同, 依赖硬件, Linux版本, 及其配置.
然后核心试图mount根文件系统. 位置可在编译时设置, 或在任何时候使用 rdev 或LILO. 文件系统类型自动检测. 如果根文件系统mount 失败, 例如因为你忘了在核心中包含相关的文件系统驱动, 核心将失败, 系统停止(此时没什么可做了).

根文件系统通常被只读mount(这可用与位置相同的方法). 这可使文件系统在mount上时检查; 检查一个可读写的已mount的文件系统可不是个好主意.

然后, 核心在后台启动程序 init (位于/sbin/init) (它的进程号是1). init 做许多启动工作. 确切的事依赖于设置; 参见章了解更多信息. 它至少要启动一些必要的后台守候程序.

init 然后切换到多用户模式并启动getty ,提供虚拟控制台和串行线. getty 是一个让用户通过虚拟控制台和串行终端登录的程序. init 还可能启动一些其他程序, 基于设置.

至此, 引导完成, 系统启动并正常运行.

关于关机的更多信息

关闭Linux系统时，遵循正确的过程是很重要的。否则，文件系统可能成为废物，文件可能变成杂乱的。这是因为Linux使用磁盘缓存，并不立即将数据写到磁盘，而是间歇地回写。这极大地改善了性能，但同时也意味着如果你只是关闭电源，cache可能保留着大量数据，而磁盘上的数据可能不是一个全部的正在工作的文件系统(因为有些数据已经回写到硬盘，而有些没有)。

另一个不能直接关闭电源的原因是：在多任务系统中，后台可能运行着很多东西，关闭电源可能损失惨重。使用正确的关机顺序，可以保证所有的后台进程得以保存他们的数据。

正常关闭Linux系统的命令是shutdown 。它通常使用2种方法之一。

如果系统只有你一个用户，使用shutdown 的通常方法是退出所有运行程序，从所有虚拟控制台注销，用root登录(如果你已经是root，当然不必再注销、登录，但应该换到根目录，以免由于unmount出现问题)，然后运行命令shutdown -h now (虽然单用户时一般不必要，但如果需要一个延时，用一个加号加一个表示分钟的数目代替now)

如果系统是多用户，使用命令shutdown -h +time message， time是到系统停止的分钟数，message是告知所有用户系统关机原因的短信息。

```
# shutdown -h +10 'We will install a new disk. System should
> be back on-line in three hours.'
#
```

上面的命令警告所有用户，系统将在10分钟后关闭，他们最好保存信息，否则将丢失。警告将显示在所有登录的终端上，包括所有的xterm 上:
Broadcast message from root (ttyp0) Wed Aug 2 01:03:25 1995...

We will install a new disk. System should
be back on-line in three hours.
The system is going DOWN for system halt in 10 minutes !!

警告在系统关闭前将自动重复数遍，随着时间流逝，间隔越来越短。

当延时之后关闭系统真正开始时，所有文件系统(除了根)被unmount，所有用户进程(如果有人还未注销)被终止，守侯进程被关闭，所有东西都停下来。此后，init 打印出一条信息告知你可以关掉电源了。此时，也只有在此时，你才可以关闭电源。

有时(虽然在任何好的系统上极少)，系统可能不能正常关闭。例如，核心紊乱、崩溃等不正常情况，可能无法键入任何命令，因此正常关机可能有些困难，这是只能直接关机。问题可能没那么严重，比如，有人误动了你的键盘，核心和update 程序还在正常运行，等待一些时间可能是个好建议，这能使update 有机会将缓冲cache 中的数据回存硬盘，然后再直接关机。

有人喜欢用sync 三遍来关闭系统，等到磁盘I/O停止，然后在关闭电源。如果没有什么程序运行着，这和用shutdown 等效。然而，它不unmount任何文件系统，可能导致ext2fs的"干净文件系统"标志出问题。这种3遍sync的方法是不推荐使用的。

(In case you're wondering: the reason for three syncs is that in the early days of UNIX, when the commands were typed separately, that usually gave sufficient time for most disk I/O to be finished.)

重启动

重启动就是完全关闭系统，关掉电源，然后再打开。简单方法是用shutdown 重启动系统而不是仅停止系统。这要使用shutdown的 -r选项，例如命令shutdown -r now。

许多Linux系统在按ctrl-alt-del键时运行shutdown -r now。这是可设置的，比如在多用户系统中设置一定的延时也许更好。如果是谁都能接触到的系统，那么最好设置为按ctrl-alt-del什么也不干。

单用户模式

shutdown命令也可用于切换到单用户模式，这种模式谁也不能登录，只有root可以使用控制台。这对系统一般运行时不能做的系统管理任务很有用。单用户模式将在章详细讨论。

紧急引导(软)盘

并非总可以从硬盘引导。例如，LILO设错了，系统可能就无法引导。这时，需要另一个总能引导的方法。对于典型的PC，可能是软驱。

许多Linux distributions允许在安装时产生一张紧急引导盘emergency boot floppy。应该做。然而，有些这样的引导盘只包含核心，and assume you will be using the programs on the distribution's installation disks to fix whatever problem you have。有时这些程序是不够的：例如你可能需要回存你的备份，而备份/回存软件在Linux安装盘里没有。

因此，可能需要自己产生root盘。 Graham Chapman写的Bootdisk HOWTO([Cha]) 包含关于此的指导。当然，你必须记得使你的紧急引导盘和root盘最新。

root盘被mount上时，不能用软驱干其他任何事，因此如果你只有一个软驱可能不太方便。然而，如果你有足够的内存，可以设置引导盘将root盘加载到RAM盘上(为此，引导盘的核心需要特殊设置)。一旦root盘被加载到RAM盘中，软驱就可以用于mount其他盘了。

登录和注销

发布时间 :2007-01-27 23:13:06

说明当一个用户登录和注销时发生了什么。较详细地说明后台进程的各种交互、log文件、配置文件等

通过终端登录

首先，init 确认有一个getty 程序提供给终端连接(或控制台)。 getty 侦听终端等候用户告知它要登录 (这通常意味着用户必然键入些什么)。当它注意到一个用户，getty 输出一个欢迎信息(存在/etc/issue 中)，并提示用户名，最后运行login 程序。 login 作为一个参数得到用户名，并提示用户输入口令。如果正确，login 启动给此用户设置的shell；否则退出并终止进程 (可能在再给用户一个机会输入用户名和口令之后)。 init 注意到进程终止，就给这个终端启动一个新的getty 。

注意唯一的新进程是由init 产生的(用fork 系统调用)； getty 和login 只是替代进程运行的程序 (使用exec 系统调用)。

为注意用户，串行线需要一个单独的程序，因为终端活动时可以(传统上也是)变得复杂。 getty 也适应连接的速度和其他设置，这对拨号连接特别重要，因为连接和连接的参数可能不同。

getty 和init 有多个版本在使用，各有优缺点。学习你的系统的版本也了解其他版本是个好主意(你可以用Linux Software Map来找。)如果你没有拨入，可能不必考虑 getty ，但 init 仍然很重要。

通过网络登录

一个网络中的2台计算机通常通过一个物理电缆连接。当他们通过网络通信是，参与通信的每个计算机里的程序通过虚拟连接virtual connection通信，即一些虚构的电缆。虚拟连接的每端的程序，独占自己的(虚拟)电缆。然而，因为这电缆不是真的，只是虚构的，所有计算机的操作系统可以在同一物理电缆上有多条虚拟 印Ｕ庋 挥靡惶醢缋拢 喔龇绦蚩梢圆槐乜悸瞧渌 ㄐ哦 嗷ネㄐ拧Ｊ褂猛 坏缋率苟嗵ḿ扑慷 强赡芳模 2台计算机间存在的虚拟连接，其他计算机会忽略他们不参加的连接。

那是一个复杂和抽象的真实描述。但可能足够理解网络登录与普通登录的不同的重要原因。不同计算机上的2个程序要通信时，虚拟连接建立。由于理论上可能从网络上的任何一台计算机登录到任何一台计算机，因此可能有极大数量的潜在的虚拟通讯。因此，为每个潜在的login启动一个getty 是不现实的。

有一个进程inetd(与getty 协同)处理所有的网络登录。当它发现一个进来的网络登录(即发现某台其他计算机来的新的虚拟连接), 它启动一个新进程来处理那个登录。原来的进程继续侦听新的登录。

更复杂的是，网络登录有多个通讯协议。2个最重要的协议是 telnet 和rlogin 。除了登录，还有许多其他虚拟连接可能建立(为FTP、Gopher、HTTP和其他网络服务)。为要侦听的每种类型的连接提供一个进程不是很有效，因此，只用一个侦听器来识别连接的种类，能启动正确的程序来提供服务。这个侦听器叫inetd ；更多的信息请见《Linux网络管理指南》。

login干了些什么

login 程序负责认证用户(确认用户名和口令相配)，并建立串行线，启动shell，建立用户的初始环境。

部分初始化设置是输出文件/etc/motd (每天的短信息)的内容，并检查电子邮件。可以在用户家目录中产生一个叫.hushlogin 的文件来是上面所述的失效。

如果存在文件/etc/nologin ，就不允许登录。这个文件一般由shutdown 及其相关的东西产生。 login 检查这个文件，如果这个文件存在，就拒绝接受登录。如果这个文件确实存在，login 就会在退出之前，将它的内容输出到终端。

login 将所有失败的登录企图登记在系统log文件中 (通过syslog)。它也登记所有的root的登录。这些都对跟踪入侵者有用。

当前登录着的用户列在/var/run/utmp 中。这个文件直到系统下次启动或关机前有效。系统刚启动时它被清空。它列出了每个用户和用户使用的终端(或网络连接)，及一些有用的信息。 who 、w 及其他类似的命令查看utmp 文件得到都有谁登录着。

所有成功的登录记录在/var/log/wtmp 中。这个文件将无限制地增大，所以必须有规律的清除，例如有个每周的cron 任务来清除它。 last 命令浏览wtmp 文件。

utmp 和wtmp 都是二进制格式 (见utmp 的man页)；不幸的是，没有特殊的程序无法查看它们。

X和xdm

META: X implements logins via xdm; also: xterm -ls

存取控制

用户数据库传统上包含在/etc/passwd 文件中。有些系统使用影子口令shadow passwords，并把口令移到/etc/shadow 中。许多计算机的场所可以用NIS或其他存储用户数据库的方法共享帐户；它们可能也自动从中心位置复制数据库到所有其他计算机。

用户数据库不仅包含口令，还包括有用户的其他信息，比如其真实姓名、家目录、登录 shell等。这其他信

息需要公用，使所有人都能读。因此口令是加密保存的。这有缺点，任何人取得加密的口令，可以用不同的加密方法猜试口令，而不用试着真正登录到计算机。影子口令试图用把口令移动到其他文件的办法避免这种情况，只有 root能读(口令还是加密保存的)。 However, installing shadow passwords later onto a system that did not support them can be difficult.

不管有没有口令，确认系统中的所有口令是好的是很重要的，即不易猜。 crack 程序可用于破解口令；任何可以精确地找到的口令都不是好的口令。同时crack 可以为入侵者运行，也可由系统管理员运行以避免坏的口令。好的口令也可以被passwd 程序强制实现；这样对CPU周期来说很有效，因为破解口令需要许多计算。

用户组数据库保存在/etc/group 文件中；有影子口令的系统，是/etc/shadow.group 。

root通常不能通过更多的终端或网络登录，只能通过列在/etc/securetty 文件中的终端登录。这使得必须能够物理存取这其中的一个终端。当然也可能通过任何终端用任何拥护登录，然后使用su 命令变成root。

Shell启动

当一个交互的登录shell启动时，它自动执行一个或更多预定义的文件。不同的shell执行不同的文件；更多的信息见每个shell的文档。

多数shell首先运行一些全局文件，例如，Bourne shell(/bin/sh) 和它引出执行的/etc/profile ；另外，它们执行用户家目录中的.profile 。 /etc/profile 允许系统管理员建立一个公用的用户环境，特别是建立PATH，以包括本地命令目录。另外，.profile 允许用户通过覆盖按照自己的口味客户化环境，如果必要，使用确省环境。

管理用户帐户

发布时间 :2007-01-27 23:13:28

本章解释如何产生新用户帐户，如何修改帐户的属性，如何删除帐户。不同的Linux系统有不同的工具实现。

什么是帐户?

当一台计算机为多人所用时，通常需要区分用户，例如，使个人文件保持个人化。即使计算机同时只为一人所用，这也很重要，如多数微机。 因此，每个用户给定一个单独的用户名，这个名字被用于登录。

用户除了名字还有更多。一个帐户是所有的文件、资源和属于这个用户的信息。这个属于暗示是银行，在一个商业系统中，每个帐户通常与一些钱有关，且这些钱依赖于用户使用系统的多少以不同的速度被花掉。例如，磁盘空间可能有个每MB每天的价格，处理时间也可能有个每秒的价格。

创建用户

Linux核心自己只不过视用户为数字。每个用户用一个单一的整数识别，user id或uid，因为数字对计算机来说比文本名字处理更快更容易。核心之外的一个单独的数据库给每个user id安排了文本的名字，即用户名username。这个数据库还包含一些其他信息。

要产生一个用户，需要给用户数据库增加关于用户的信息，并给他产生家目录。培训用户、建立合适的初始化环境也是必要的。

多数Linux distributions有产生帐号的程序，而且有多个。 adduser 和useradd 是其中2个；可能还有GUI的工具。 Whatever the program, the result is that there is little if any manual work to be done. Even if the details are many and intricate, these programs make everything seem trivial. However, section 8.2.4 describes how to do it by hand.

/etc/passwd和其他信息文件

Unix系统的基本用户数据库是文本文件，/etc/passwd (叫口令文件)，它列出所有有效用户名及其相关信息。文件的每个用户一行，分为用:分隔的7个域：

用户名
加密格式的口令

数字的user id
数字的group id
全名或帐户的其他说明
家目录
登录shell(登录时运行的程序)

详细的格式说明在passwd (5)中。

系统中的任何用户可以读口令文件，因此他们可以得到其他用户的名字。即任何人也可以得到口令(第二个域)。口令文件加密了口令，所以利润上说应该没有问题。但是，加密是可破解的，尤其是口令比较简单时(例如太短，或能在词典中找到的)。因此，口令存在口令文件中并不好。

许多Linux系统有影子口令shadow passwords文件。这种方法将加密的口令存在另一个文件/etc/shadow中，而这个文件只有 root能读。 /etc/passwd 文件在第二个域只有一个special marker。 Any program that needs to verify a user is setuid,那么可以存取影子口令文件。而只使用口令文件其他域的普通程序，不能得到口令。
取得数字的用户和组ID

多数系统不管数字的用户和组ID是什么，但如果使用网络文件系统(NFS)，所有系统必须使用相同的uid和gid。因为NFS也用uid认证用户。如果不使用NFS，可以用帐户产生工具自动取得的uid。

如果用NFS，必须用一个机制来同步帐户信息。一个方法是使用NIS系统 (见[Kir])。

初始环境：/etc/skel

当新用户的家目录产生时，用/etc/skel 目录的文件初始化。系统管理员可以产生/etc/skel 里的文件给用户提供一个好的缺省环境。例如，产生一个/etc/skel/.profile 设定EDITOR环境变量，提供新用户一个友善的编辑器。

然而，通常最好保持/etc/skel 尽量小，因为it will be next to impossible to update existing users' files. 例如，如果友善的编辑器的名字改变了，所有现存用户必须编辑他们的.profile 。系统管理员可以用一个scr ī pt自动完成，但仍可能破坏某个用户的文件。

只要可能，最好把全局设置放在全局文件中，如/etc/profile 。这样可以升级，而避免破坏用户自己的设置。

手工创建用户

按以下步骤手工创建新用户：

用vipw (8)编辑/etc/passwd ，为新用户增加一个新行。注意语法。 不要用编辑器直接编辑! vipw 锁定了这个文件，其他命令这时不能更新它。设定口令域为"*"，这样不能登录。

类似，如果要创建新组，用vigr 编辑/etc/group 。

用mkdir 产生用户的家目录。

将/etc/skel 中的文件复制到新的家目录中。

用chown 和chmod 修改所有者和权限。 -R选项是最有用的。 The correct permissions vary a little from one site to another, but usually the following commands do the right thing:

```
cd /home/newusername
chown -R username.group .
chmod -R go=u,go-w .
chmod go= .
```

用passwd (1)设定口令。

最后一步设定完口令，这个帐户就能用了。不应该在其他所有事做完之前设定口令，否则这个用户可能不允许登录while you're still copying the files.

有时需要产生不为任何人使用的虚假(dummy)帐户 例如，建立一个匿名FTP服务器(这样任何人都可以从它下载文件，无须得到一个帐户)，必须产生一个叫ftp的帐户。这种情况，通常无须随后一步的口令设定。而且，最好不设，这样没有人可以使用这个帐户，除非先变成root，因为root可以变成任何用户。

改变用户属性

有几个改变帐户不同属性的命令(即/etc/passwd 中的相关域)：

chfn
改变全名域。
chsh
改变登录shell。
passwd
改变口令。

超级用户可以用这些口令改变任何帐户的属性。普通用户只能改变自己帐户的属性。有时可能有必要使这些命令对普通用户不可用(用chmod)，例如在一个有许多新手的环境中。

其他任务需要手工完成。例如改变用户名，需要编辑/etc/passwd (记住：用vipw)。同样，要增加或删除用户to more groups，需要编辑/etc/group (用vigr)。这种任务较少，需要小心从事：例如，改变了用户名，电子邮件就不能到达这个用户，除非你同时产生一个邮件别名。

删除用户

要删除用户，必须先删除他的所有文件，然后从/etc/passwd 和/etc/group 删除相关的行。有些Linux distributions带特定的命令，看看有没有 deluser 或userdel 。然而，手工删除也很简单。

临时禁止一个用户

有时需要临时禁止一个用户，而不删除它。例如用户没有付费，或系统管理员怀疑黑客得到了某个帐户的口令。

禁止一个用户的最好方法是将它的shell变到一个特定的只打印出一条信息的程序，用这种方法，任何想登录此帐户的人将无法登录，并得知原因。该信息可以告诉用户与系统管理员联系，以处理任何问题。

也可以改变用户名或口令，但这样用户不知道怎么回事。 Confused users mean more work.

产生上述特定程序的一个简单方法是写"tail scr ī pts":

```
#!/usr/bin/tail +2
This account has been closed due to a security breach.
Please call 555-1234 and wait for the men in black to arrive.
```

前2个字符("#!")告诉核心本行的其他部分是解释本文件要运行的命令。这样tail 命令将输出处理第一行外的所有东西到标准输出。
如果怀疑billg是个安全缺口，系统管理员可以这样做：

```
# chsh -s /usr/local/lib/no-login/security billg
# su - tester
This account has been closed due to a security breach.
Please call 555-1234 and wait for the men in black to arrive.
#
```

su 的目的是此时改变是否工作。
Tail scr ī pts应该放在一个分离的目录中，这样它们的名字不会干扰普通用户的命令。

备份

发布时间 :2007-01-27 23:13:50

硬件不肯定是可靠的
软件肯定是不可靠的
人不肯定是不可靠的
而自然肯定是可靠的

本张说明为什么、如何、何时要做备份，及如何回存备份的东西。

备份的重要

数据是有价值的。重新产生它需要你花费时间和努力，并且要花费金钱或至少伤心和眼泪，有时甚至不可能重新产生，例如一些实验结果。由于数据是一种投资，你必须保护它，并采取措施避免丢失。

丢失数据一般有4个原因：硬件失败、软件曲线、人为因素或自然灾害。 虽然现代硬件已经相当可靠，但仍可能自然损坏。存储数据最决定性的硬件是硬盘，它依赖微小的磁区在充满电噪声的世界上保存数据。现代软件依然不可靠，一个真正可靠的程序是理想、罕见的，而不是规律。人更不可靠，他们很容易犯错误，甚至为某种目的恶意地破坏数据。自然可能不是邪恶的，但也可能造成破坏。一切的一切，希望什么都正常、完美几乎是不可能的。

备份是保护数据投资的方法。有数据的多个拷贝，就不怕某个损坏(所需做的仅仅是从备份中恢复丢失的数据)。

正确的备份是很重要的。正如物理世界中任何东西都与其他相关，备份也迟早会失效。好的备份确保有效，你不希望你的备份无效。 如果你的备份又坏了，这将雪上加霜，如果你只有一个备份，它可能根本是坏的，只留下你和硬盘中冒烟的灰烬。 或者当你恢复时，发现忘了备份一些重要的东西，比如15000个用户站点的用户数据库。 Best of all, all your backups might be working perfectly, but the last known tape drive reading the kind of tapes you used was the one that now has a bucketful of water in it.

When it comes to backups, paranoia is in the job descr ī ption.

选择备份介质

备份所需的最重要的决定是选择备份介质。需要考虑成本、可靠性、速度、可得到、可用性。

成本是很重要的，因为你的数据可能需要多个存储、多个备份。便宜的介质可以用很多。

可靠性是最重要的，因为坏的备份会雪上加霜。备份介质必须能存储数据多年而不损坏。作为备份介质，使用方法影响可靠性。硬盘一般是很可靠的，但作为备份介质并非很可靠，如果它和备份源在同一计算机里的话。

速度通常不太重要，如果备份可以非交互地完成。备份花2个小时无所谓，无须监督，多长时间都没有关系。另一方面，if the backup can't be done when the computer would otherwise be idle, 那么速度也是个问题。

可得到是明显必要的，因为你无法使用不存在的备份介质。不太明显的是要在将来还能得到这种介质，并且能在其他计算机上使用。否则灾害之后，你可能无法恢复你的备份。

可用性是决定备份周期的主要因素。备份越容易使用越好。备份介质不能难以使用。

一般用软盘和磁带。软盘很便宜，还算可靠，不太快，很容易得到，但数据量大时不容易使用。磁带也很便宜，还算可靠，还算快，很容易得到，而且，依赖于磁带的容量，使用很轻松。

还有其他选择。但通常可得性不好，但如果这不成问题，有时也不错。例如，磁光盘同时具有软盘(随机存取，可以快速地恢复单个文件)和磁带(大容量)的优点.

选择备份工具

备份有很多工具，传统的UNIX备份工具是tar 、 cpio 和dump 。另外，还可以使用大量第三方软件包(包括freeware和商业版)。备份介质的选择可能影响工具的选择。

tar 和cpio 类似，从备份来看二者基本等效。都能将文件存到磁带并取出文件。都能使用几乎所有介质，因为核心设备驱动处理低级设备操作，对用户级程序看来所有设备都差不多。有写Unix版本的tar 和cpio 对不是普通文件可能有问题(符号连接、设备文件、极长路径名的文件等等)，但Linux的能正确处理所有文件。

dump 不同，它直接读文件系统，而不通过文件系统。 It is also written specifically for backups; tar 和cpio are really for archiving files, although they work for backups as well.

直接读文件系统有些优点，它可能不考虑time stamps备份所有文件；对于tar 和cpio ，必须先将文件系统只读安装。直接读文件系统更有效，如果所有东西都要备份，因为它使磁头移动最少。它的主要缺点是每个文件系统种类需要特定的备份程序， Linux的dump 程序只理解ext2文件系统。

dump 也直接支持备份级(下面讨论)；对tar 和cpio ，这必须用其他工具实现。

第三方备份工具的比较超出了本书的范围。Linux Software Map列出了许多freeware的。

简单备份

一个简单的备份方案是一次备份所有东西，然后备份上次备份后改变的所有东西。第一个备份叫全备份full backup，后来的叫i增量备份ncremental backups。全备份比增量备份费时费力，因为有更多的东西写到磁带，而且全备份可能不能放如一盘磁带中(更别说软盘了)。回存增量备份比全备份可能要花更多的时间。备份可以这样优化，就是自上次全备份以后，总用增量备份保存所有改过的文件。这样，备份可能需要多一些的工作，但你只需回存一个全备份和一个增量备份。

如果有6盘磁带想每天备份，可以用磁带1做第一个全备份(比如在星期五)，用磁带2-5做增量备份(周一到周四)。然后用磁带6做新的全备份(第二个周五)，然后再用磁带2-5做增量备份。在做完新的全备份之前不要覆盖旧的全备份(磁带1)，一面在做全备份的时候出现问题。有了新的全备份磁带6以后，最好在另一个地方保存磁带1，这样如果有一个全备份磁带在火灾中损失了，还能有一个。当再做下一个全备份是，再用磁带1而保存磁带6。

如果你有多于6盘磁带，可以用多的做全备份。每次做全备份，应该使用最老的磁带。这样你会有最近几周的全备份，对你如果想找到一个现在已经删除的就文件，或一个文件的旧版本很有用。

用tar备份

一个全备份可以很容易地用tar 实现：

```
# tar -create -file /dev/ftape /usr/src
tar: Removing leading / from absolute path names in the archive
#
```

上面的例子使用GNU版本的tar 及其长选项名。传统版本的tar 只理解单字符选项。 GNU版还能处理一盘磁带或一张磁盘不能容纳的备份，及很长的路径名；这不是所有传统的版本能作到的。 (Linux只使用GNU tar 。)

如果你的备份一盘磁带不能容纳，你需要使用-multi-volume (-M)选项：

```
# tar -cMf /dev/fd0H1440 /usr/src
tar: Removing leading / from absolute path names in the archive
Prepare volume #2 for /dev/fd0H1440 and hit return:
#
```

注意开始备份前要格式化所有软盘，或在tar 需要新软盘时用另一个虚拟控制台或虚拟终端格式化它。备份完后，应该检查它是否完好，用-compare (-d)选项：

```
# tar -compare -verbose -f /dev/ftape
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
....
#
```

失败的备份检查意味着如果你丢失了原始数据，备份也无法恢复。

增量备份可用带-newer (-N)选项的tar 来实现：

```
# tar -create -newer '8 Sep 1995' -file /dev/ftape /usr/src -verbose
tar: Removing leading / from absolute path names in the archive
usr/src/
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/modules/
usr/src/linux-1.2.10-includes/include/asm-generic/
usr/src/linux-1.2.10-includes/include/asm-i386/
usr/src/linux-1.2.10-includes/include/asm-mips/
usr/src/linux-1.2.10-includes/include/asm-alpha/
usr/src/linux-1.2.10-includes/include/asm-m68k/
usr/src/linux-1.2.10-includes/include/asm-sparc/
usr/src/patch-1.2.11.gz
#
```

不幸的是，tar 不能知道一个文件的i节点信息变化，例如，文件的权限位变化，或文件名变化。这可用find命令和比较当前文件系统状态和先前备份的文件列表。用于此的scr ī pts和程序可以在Linux FTP站点上找到。

用tar回存

tar 的-extract (-x)选项展开文件：

```
# tar -extract -same-permissions -verbose -file /dev/fd0H1440
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/kernel.h
...
#
```

也可以用命令行只展开特定的文件和目录(及其中的文件和子目录)：

```
# tar xpvf /dev/fd0H1440 usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
#
```

用-list (-t)选项看一个备份卷中有什么文件：

```
# tar -list -file /dev/fd0H1440
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/kernel.h
...
#
```

注意tar 永远是顺序读一个备份卷，因此大的卷会很慢。使用磁带机或其他顺序介质时不可能使用随机存取数据库技术。

tar 不处理删除文件属性。如果你需要从一个全备份和一个增量备份恢复一个文件系统，并且2个备份之间你删除了一个文件，当你恢复完后，这个文件又存在了。如果这个文件包含应该删除的敏感数据，这是个大问题。

多级备份

上面的章节概述了简单备份的方法，对个人使用或小的站点使用。对于多数重负荷的使用，多级备份更适用。

简单备份有2个备份级：全备份和增量备份。通常可以有任意数量的备份级。全备份是0级，不同级别的增量备份是1、2、3...级，每个增量备份级备份同一或上一级别的上次备份后改变的所有东西。

这样多的目的是更便宜地允许更长的备份历史backup history。在前面的例子中，备份历史追溯到上一个全备份。可以增多磁带来扩展备份历史，但每个新磁带扩展一周，这样可能太贵。更长的备份历史是有用的，因为删除或损坏的文件可能长时间未被发现。即使不是一个文件的最新版本，也比没有好。

多级备份可以更便宜地扩展备份历史。例如，如果你有10盘磁带，可用磁带1和2做月备份(每月的第一个周五)，磁带3-6做周备份(其他周五，因为每月最多可能有5个周五，因此需要4盘磁带)，磁带7-10做日备份(周一到周四)。只增加了4盘磁带，就将2周的备份历史扩展到2个月。诚然，我们无法恢复这2个月中每个文件的所有版本，但这样恢复的经常是足够好了。

备份级可使文件系统恢复用最少的时间。如果你有许多只是单调增长级别数的增量备份，要恢复整个文件系统，你需要回存所有备份。而如果级别数不是单调增长，可以减少备份和回存的数目。

为了将回存需要的磁带数据减至最小，可以用小的级别做每个增量磁带。然而，这样做每个增量备份的时间会增加(每个备份拷贝了上次全备份后改变的所有东西)。一个好的方案建议在dump man页中给出，并在表9.2中说明。 Use the following succession of backup levels: 3, 2, 5, 4, 7, 6, 9, 8, 9... 这使备份和回存所用的时间保持较少。 The most you have to backup is two day's worth of work. 恢复所需磁带数有赖于全备份的间隔，但它比简单的方案少。

一个好的方案降低了工作量，并能追寻更多的东西。You must decide if it is worth it.

dump 对备份级有内置的支持。而tar 和cpio 则必须用shell scr ī pts实现。

备份什么?

你可能想尽多备份。主要的例外是容易重安装的软件， 但即使是它们，也有配置文件，对备份很重要，以免对这些软件全部重新配置。另一个主要的例外是/proc 文件系统，因为他们只包含通常由核心自动产生的数据，备份它们绝不是个好主意。特别是/proc/kcore 文件更是不必要，因为它只是你当前物理内存的映象，而且很大。

Gray areas include the news spool, log files, and many other things in /var . 你必须决定重点考虑什么。

备份最明显的是用户文件(/home)和系统配置文件(/etc ，但还可能有散落在文件系统其他地方的其他东西。

压缩备份

备份占用大量空间，要花费大量金钱。为了降低空间需求，备份可以压缩。有几种方法。有些程序内置支持压缩。例如GNU tar 的-gzip (-z)选项，通过管道(pipe)，在写到备份介质前，先用 gzip 压缩程序压缩。

不幸的是，压缩备份可能导致问题。由于压缩工作的原理，如果一个bit错误，可能导致所有其他压缩数据不可用。有些备份程序内置错误校正，但没有办法处理大量的错误。就是说，如果用GNU tar 压缩备份，一个单独的错误回导致整个备份丢失。备份必须可靠，这样的压缩方法不好。

还有一个方法是单独压缩每个文件，这也回导致一个文件的丢失，但不会影响其他文件。丢失的文件可能已经因为什么原因损坏，因此这种情况比不使用压缩差不了多少。 afio 程序(cpio 的一个变种)可以这样。

压缩需要时间，which may make the backup program unable to write data fast enough for a tape drive. 这可以靠输出缓冲来避免 (如果备份程序足够智能，可以内置，否则可以通过其他程序)， but even that might not work well enough. 这只会在慢的计算机上是个问题。

Keeping Time

发布时间 :2007-01-27 23:14:24

本章说明Linux系统如何keeps time，及需要做什么来避免发生问题。通常，你无须对时间做什么，但理解它会更好。

时区

时间测量基于最规则的自然现象，如地球转动导致的昼夜更替。昼夜总时间是恒定的，但昼夜分别的长度是变化的。一个简单的常数是正午。

正午是白天太阳在最高点的时间。由于地球是圆的， 不同地方正午发生在不同的时间。这引出了本地时local time的概念。

硬件时钟和软件时钟

个人计算机有一个电池驱动的硬件时钟。电池保证始终在计算机没电的时候依然能工作。硬件始终能从BIOS设置屏或操作系统的别的地方进行设置。

Linux核心独立于硬件始终跟踪时间。启动时，Linux根据硬件时钟设置自己的时钟。此后，2个始终相互独立运行。因为查看硬件始终慢而复杂，因此Linux管理自己的时钟。

核心始终一直显示通用时间。这样，核心无须知道时区，高可靠的简单结果使更新时区信息更简单。每个进程自己处理时区转换(使用时区包部分里的标准工具)。

硬件始终可以是本地时间或通用时间。通常用通用时间更好，因为这样你无须在夏时制开始或结束时改变硬件时钟 (UTC does not have DST)。不幸的是，有些PC操作系统，包括MSDOS、Windows、OS/2都假设硬件时钟是本地时间。 Linux可处理2种方式，但如果硬件时钟显示本地时间，那么必须在夏时制开始或结束时(否则就不能显示本地时间)。

显示和设置时钟

在Debian系统中，系统时区由符号连接/etc/localtime 决定。连接指向描述本地时区的时区数据文件。时区数据文件存在/usr/lib/zoneinfo 中。其他Linux distributions可能不同。

用户可以用设置TZ环境变量来改变他的私人时区。如果不设置，就假定是系统时区。 TZ变量的语法在tzset (3)man页中说明。

date 命令显示当前日期和时间。 例如：

```
$ date
Sun Jul 14 21:53:41 EET DST 1996
$
```

That time is Sunday, 14th of July, 1996, at about ten before ten at the evening, in the time zone called ``EET DST'' (which might be East European Daylight Savings Time). date 也可用于显示通用时间：

```
$ date -u
Sun Jul 14 18:53:42 UTC 1996
$
```

date 也可用于设置核心的软件始终：

```
# date 07142157
Sun Jul 14 21:57:00 EET DST 1996
# date
Sun Jul 14 21:57:02 EET DST 1996
#
```

更详细的见date man页--syntax is a bit arcane. 只有root能设置时间。虽然每个用户可以有自己的时区，但时钟对每个人都是一样的。

date 只显示或设置软件时钟。 clock 命令同步硬件和软件时钟。用于系统启动时读取硬件时钟和设置软件时钟。如果两个时钟都需要设置，则先用date 设置软件时钟，然后用clock -w 设置硬件时钟。

clock 的-u告诉它硬件时钟是通用时间。 必须正确使用-u选项。否则计算机将困惑到底是什么时间。

时钟必须小心改变。Unix系统的许多部分要求时钟工作正常。例如， cron 守侯程序周期地运行命令。如果改变时钟，它可能迷惑它是否该运行命令。 On one early Unix system, someone set the clock twenty years into the future, and cron wanted to run all the periodic commands for twenty years all at once. 现在版本的 cron 可以正确处理，但仍然要小心。大的前后跳跃比小的更危险。

当时钟错误时

Linux软件时钟不会始终精确。 PC硬件产生的时间中断周期地运行软件时钟。如果系统运行了太多进程，服务于时间中断需要花费太多的时间，软件时钟启动靠后。硬件时钟独立运行并通常更精确。如果你的系统经常启动(比如不是服务器的多数系统)，那么通常时间很精确。

如果需要调整硬件时钟，通常最简单的是重启动，进入BIOS设定屏幕，并在那里完成。这避免了改变系统时间可能导致的所有问题。如果不能通过BIOS，用date 和clock 设定新时间(以此顺序)，但如果系统有部分工作不正常，必须准备重启动。

连网的计算机(即使是通过modem)能通过与其他计算机时间比较来自动检查自己的时钟。如果知道保持很精确时间的其他计算机，那么2台计算机都将保持精确的时间。这可以使用rdate 和netdate 命令来完成。 2个命令都检查远程的计算机(netdate 可处理多台远程计算机)，来同步本地计算机的时间。有规律地运行这样一个程序，你的计算机就可以保持与远程计算机一样精确的时间。

测量孔(Measuring Holes)

本附录包括用于测量文件系统中潜在的孔的程序的有趣的部分。 The source distribution of the book contains the full source code(sag/measure-holes/measure-holes.c).

```
int process(FILE *f, char *filename) {
static char *buf = NULL;
static long prev_block_size = -1;
long zeroes;
char *p;


if (buf == NULL || prev_block_size != block_size) {
free(buf);
buf = xmalloc(block_size + 1);
buf[block_size] = 1;
prev_block_size = block_size;
}
zeroes = 0;
while (fread(buf, block_size, 1, f) == 1) {
for (p = buf; *p == '\0'; )
++p;
if (p == buf+block_size)
zeroes += block_size;
}
if (zeroes > 0)
printf("%ld %s\n", zeroes, filename);
if (ferror(f)) {
errormsg(0, -1, "read failed for `%s'", filename);
return -1;
}
return 0;
}
```

(全文完)

海量Linux技术文章汇集

海量 Linux 技术文章

发布时间 :2006-11-24 16:50:29

下面是linux技术文章快速入口。需要联网：

Linux 技术交流

http://www.linuxdiyf.com/bbs/forum-3-1.html

Linux 应用

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=1

Linux 安装及学习指导

http://www.linuxdiyf.com/bbs/forum-45-1.html

Linux 系统安装

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=45&filter=type&typeid=11

Linux 学习指导

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=45&filter=type&typeid=12

Linux 软件安装

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=45&filter=type&typeid=13

shell

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=3

Linux 壁纸

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=4

红旗

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=5

Redhat

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=6

SuSE

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=7

Linux 认证

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=9

Linux下载分享{酷件、书籍、视频分享 }

http://www.linuxdiyf.com/bbs/forum-6-1.html

服务器应用

http://www.linuxdiyf.com/bbs/forum-7-1.html

数据库应用

http://www.linuxdiyf.com/bbs/forum-8-1.html

Linux 编程与内核

http://www.linuxdiyf.com/bbs/forum-9-1.html

UniX 技术文章

http://www.linuxdiyf.com/bbs/forum-32-1.html

Linux 业界声音、新闻

http://www.linuxdiyf.com/bbs/forum-11-1.html

Linux 人才招聘信息

http://www.linuxdiyf.com/bbs/forum-46-1.html

网络转载，感谢原创作者！
制作：红联Linux论坛
祝您阅读愉快！