精通Linux丛书

Linux的管理与配置

雷澍 等编著

关于本书

- 详细介绍了Linux系统的管理与配置方法
- 提供实现Linux网络应用的完整方案
- 书中实例是作者多年使用Unix/Linux的经验总结
- 着重讲述了Linux在安全方面要注意的问题
- 语言通俗易懂，读来轻松自如

机械工业出版社
China Machine Press

精通 Linux 丛书

Linux 的管理与配置

雷 澍 谭洪湘 马艾岩
欧阳定恒 楼玲玲 编著

机械工业出版社

本书是精通 Linux 丛书的第二本。本丛书共 3 本，另外两本是《Linux 的安装与使用》和《Linux 的内核与编程》。

本书全面介绍了 Linux 操作系统的管理与配置方法。全书分两部分共 13 章，分别介绍 Linux 操作系统在机器管理、文件系统管理、打印管理、备份、网络技术基础、DNS 服务、SLIP 和 PPP、Telnet 和 FTP、Web Server、SMB、BBS、防火墙和安全等方面的内容，可供需要管理 Linux 系统或使用 Linux 网络功能的中、高级用户使用。

本书内容深入浅出，即使非专业人士也能很容易地理解一些十分专业的概念。

图书在版编目（CIP）数据

Linux 的管理与配置 / 雷澍等编著. —北京：机械工业出版社，2000. 7
（精通 Linux 丛书）

ISBN 7-111-08166-8

Ⅰ. Linux　Ⅱ. 雷…　Ⅲ. Linux 操作系统　Ⅳ. TP316.81

中国版本图书馆 CIP 数据核字（2000）第 64558 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）

责任编辑：余茂祚　　封面设计：艾　迪

责任印制：何全君

中国农业出版社印刷厂印刷·新华书店北京发行所发行

2000 年 7 月第 1 版第 1 次印刷

787mm×1092mm 1/16 · 14.25 印张 · 343 千字

0 001— 5 000 册

定价：23.00 元

前　言

Linux 是 Unix 在微机上的完整实现，是芬兰的 Linus Torvalds 于 1991 年独立发展起来的，Linux Torvalds 堪称一代高手！由于 Linux 免费提供源代码和可执行文件，并且公布在互联网上，因此从一开始就吸引了世界各地的 Unix 行家为 Linux 编写了大量的驱动程序和应用软件，在短短几年时间里，Linux 就发展成为一个相当完善的操作系统，成为 Unix 世界的一朵奇葩。

Linux 的特点

1. 可完全免费得到

Linux 操作系统可以从互联网免费下载使用，只要有快速的网络连接就行；而且，Linux 上运行的绝大多数应用程序也是免费可得的。用了 Linux 就再也不用背“使用盗版软件”的黑锅了。

2. 可以运行在 386 以上的 X86 机器上

Linux 专门为微机环境而设计，充分利用了 X86 CPU 的任务切换能力，使 X86 CPU 的效能发挥得淋漓尽致，而这一点 Windows 都没有做到。

3. Linux 是 Unix 的完整实现

Unix 上的绝大多数命令都可以在 Linux 里找到并有所加强。Unix 的可靠性、稳定性以及强大的网络功能也在 Linux 上一一体现。

4. 真正的多任务多用户

只有很少的操作系统能提供真正的多任务能力，尽管许多操作系统声明支持多任务，但并不完全准确，如 Windows。而 Linux 则充分利用了 X86 CPU 的任务切换机制，实现了真正多任务、多用户环境，允许多个用户同时执行不同的程序，并且可以给紧急任务以较高的优先级。

5. 完全符合 POSIX 标准

这使 Unix 下的许多应用程序可以很容易地移植到 Linux 下，相反也是如此。

6. 具有图形用户界面

Linux 的图形用户界面是 X Window 系统。X Window 可以做 MS Windows 下的所有事情，而且更有趣、更丰富，甚至可以在几种不同风格的窗口之间来回切换。

7. 具有强大的网络功能

实际上，Linux 就是依靠互联网才迅速发展了起来，Linux 具有强大的网络功能也是自然而然的事。它可以轻松地与 TCP/IP、LAN Manager、Windows for Workgroups、Novell Netware 或 Windows NT 网络集成在一起，还可以通过以太网卡或调制解调器连接到 Internet 上。

Linux 不仅能够作为网络工作站使用，更可以胜任各类服务器，如 X 应用服务器、文件服务器、打印服务器、邮件服务器、新闻服务器等等。可以说，Linux 操作系统的网络功能胜过了其他任何一种操作系统，连 Windows NT 也不是它的对手。

8. 是完整的 Unix 开发平台

Linux 支持一系列的 Unix 开发工具，几乎所有的主流程序设计语言都已移植到 Linux 上并可免费得到，如 C、C++、Fortran 77、ADA、PASCAL、Modual 2 和 3、Tcl/TkScheme、SmallTalk/X 等。

9．开放的源代码

Linux 的源代码完全开放，而且任何人可以修改这些源代码，修改后的源代码可以自己使用，也可以发布到网上让所有计算机用户使用。

丛书内容简介

丛书分为三本，分别是《Linux 的安装与使用》、《Linux 的管理与配置》和《Linux 的内核与编程》。丛书全面阐述了 Linux 的方方面面，通俗易懂，是笔者多年使用 Unix/Linux 经验的结晶。

《Linux 的安装与使用》主要面向 Linux 操作系统的初学者，从安装和使用两方面详细讲解了 Linux 的基础知识。

《Linux 的管理与配置》主要面向 Linux 操作系统的中、高级读者，详细说明了 Linux 所完成的各个功能的配置方法，尤其适合系统管理员或者网络管理员参考。

《Linux 的内核与编程》主要面向 Linux 操作系统的中、高级读者，从操作系统原理开始，以各个方面的编程方法结束，对计算机专业的学生和程序员都由很大的帮助。

本书内容简介

本书面向 Linux 操作系统的中、高级读者，要求读者对操作系统原理或软件开发有一定了解，而不一定熟悉 Unix/Linux 环境。

本书分为两大部分。

第一部分： Linux 系统管理

本部分介绍 Linux 系统管理方面的一些知识。

第 1 章主要介绍 Linux 机器的管理与维护，包括系统的引导、启动配置、关闭与登录等。

第 2 章详细描述描述了什么是文件系统及保持 Linux 处于最佳状态的文件系统和硬盘所应采取的行动。

第 3 章介绍 Linux 在打印管理方面的内容，包括打印设备，打印作业，打印机的安装、使用、共享及配置等。

第 4 章讲到了 Linux 的备份与恢复，包括备份的目的、介质以及工具，怎样备份以及如何恢复等内容。

第二部分：网络应用

这里介绍 Linux 网络应用的各个方面。

第 5 章介绍 Linux 是如何实现计算机之间的互联的，包括网络设备、网络概念、TCP/IP 等方面的内容，这一章是下面章节的基础和准备。

第 6 章介绍一个重要的网络概念：域名服务，以及 Linux 系统中 DNS 的配置、启动、测试和维护方面的知识。

第 7 章介绍一种标准远程接入协议：点对点协议以及 PPP 与 TCP/IP 的无缝连接，这样就可以通过电话网络在计算机之间传递数据。

第 8 章介绍操作远程计算机的两种重要方法，使用 Telnet 可以登录远程计算机，就像使用本地计算机一样使用远程计算机；使用 Ftp 则可以在远程计算机之间传递各种文件。

第 9 章介绍怎样建立基于 Linux 的 Web 服务器，包括 Apache 的配置以及 Web 的安全问题解答。

第 10 章介绍怎样使用 SMB 协议实现 Linux 系统与 MS Windows 系统之间的信息共享。

第 11 章介绍另一种重要的网络资源 BBS 的 Linux 实现方法，这里主要根据 Firebird BBS 进行讲解。

第 12 章描述保护自己网络的一种方法：构建防火墙，只有很好地使用了 Firewall 技术，内部网络才可以高枕无忧。

第 13 章介绍一种应用广泛的网络文件系统 NFS，它是由 SUN 公司首先开发的，使用它，可以像操作本地数据一样方便地操作远程计算机上的数据和文件。

附录部分通过三方面来解释用户使用 Linux 时，在安全方面要注意的问题，它们是系统管理员安全问题、程序员安全问题和用户安全问题。

目　录

第一部分　Linux系统管理

第1章　机 器 管 理

本章主要介绍机器的管理与维护，包括系统的引导、启动配置、关闭与登录等。

1.1　引导过程

引导过程也叫自举过程，意味着装载并执行可引导操作系统的过程。 大部分的计算机中，引导过程都有三个基本步骤。

第一阶段：或者打开机器，或者通过硬件重置按钮(reset)、shutdown或init命令，进入 ROM方式，在加电时，引导过程一般是自动开始的：装入一个小的ROM引导程序并执行它，然后进入第二阶段。

第二阶段：该ROM引导程序装入并执行一个更大些的宏引导程序。

第三阶段：该引导程序装入并执行可引导操作系统。这是Unit系统的起始点，此时必要的文件系统已安装好，运行init让系统进入在/sbin/inittab中规定的initdefault状态。boot再加载可引导的操作系统。

下面对引导过程做一更详细的说明。

首先是机器上电，以80X86机器为例，开机时， CS(Code Segment)这个寄存器中全部都放着1，而IP (Instruction Pointer)这个寄存器中全部都放着0，换句话说，CS=FFFF而IP=0000，此时，CPU就依据CS 及IP的值，到FFFF0H去执行存放的指令。这时候，由于FFFF0H已经到了高位址的顶端，所以，FFFF0H这个地方，总是会放一个JMP 指令，跳到比较低的位址。接着，ROM BIOS就会做一些检查的动作像内存、键盘等，并在我们俗称的UMB(Upper Memory Block)之中扫描，看看是否有合法的ROM存在(比如SCSI卡上的ROM)。假如有，就到里面去执行一些东西，执行完之后再继续刚才的进程。到了最后，读取硬盘的第一个sector。在这里，假设由硬盘启动，它的第一个sector称为MBR(Master Boot Record，主引导记录)。因为一个sector是512 byte，而MBR这512 byte可分为两个部分，第一个部分为Pre-Boot区，占了446 byte；第二部分是Partition Table，占了66 byte。Pre-Boot区的作用之一，就是去看看哪个Partition被标成Active，然后去读那个Partition的Boot区。

安装Linux的过程中，通常把LILO放在MBR或Superblock中。假如把LILO放在MBR中，当读取到MBR的时候，LILO就被执行，此时，屏幕上会出现boot：接着就进行LoadKernel

的动作。假如把LILO安装在Superblock中，通常还会有一个管理开机的程序，也许是驻在MBR(像OSBS)或者是放在一个单独的Partition(像OS/2的Boot Manager)中，再由这个管理开机的程序去读取LILO，进而做Load Kernel的动作。

Kernel被load到memory中之后，接着进行一连串Probe周边的动作（Kernel Bootstrapping)，像串口、并口、软盘、声卡、硬盘、光驱等并Mount Root Partition。在这之后，Kernel会起动Init这个Process（进程)。Init这个Process的PID（进程标识号）为1，它是所有Process的祖先。

另外Linux有两个Kernel类的Process也开始运行了起来，一个是kflushd，另一个是kswapd：

Process ID 1： init

Process ID 2： kflushd

Process ID 3： kswapd

只有这个init是完全属于user类的Process，后两者是Kernel假借Process之名挂在进程上。

Linux系统启动时，可以选择被称为单用户的方式运行，在这种方式中普通用户不能登录，唯有的进程是Init，Swapper，以及一些由系统管理员从控制台运行的进程。当Linux系统以单用户方式启动时，系统管理员能在允许普通用户登录以前，先检查系统操作，确保系统一切正常，当系统处于这种工作方式（即单用户方式）时，控制台作为超级用户，命令提示符是“#”。

1.2 系统状态（init状态）

系统一旦上电，Linux就在某一个系统状态下操作。不同的系统状态下计算机运行一组特定的活动进程，这样就能很有把握地完成管理和操作功能。

首先弄清楚系统状态个概念。有很多术语都用来表示系统的特定操作状态。系统状态又称为“init状态”、“run level”、“运行状态”、“运行级别”或“运行方式”。从一个系统状态切换到另一个系统状态有多种方法：

- 使用shutdown命令
- 使用powerdown命令
- 使用init命令

前两个命令在执行期间将调用init命令。下面介绍一下init命令。

Init是一个总的进程生成器。它的主要作用是根据文件/sbin/inittab中的信息创建进程。

在任意给定的时刻，系统处于8种可能运行级的任何一级上。一个运行级是系统的一种软件配置，在这种配置下仅存在选中的一组进程。在/sbin/inittab文件中定义了由init为每个运行级生成的进程。Init可以在8个运行级的任何一级上：0-6和S或s（运行级S和s是等同的)。当超级用户运行/sbin/init时，运行级改变。重新引导系统时，这个用户产生的init向由操作系统产生的原始init发送相应的信号，告诉它改变为哪一个运行级。

下面是init命令的实参：

（1）0 关闭机器从而可以安全的切断电源。如果可能，要让机器切断电源。

（2）1 把系统置为系统管理员方式。卸下全部文件系统，仅留下一小组必不可少的核心进程运行。

这种方式是为了完成管理任务，例如安装可任选的应用程序包。系统中所有的文件均可访问，在系统中没有用户注册进入。使系统进入多用户方式。生成全部多用户环境终端进程和精灵进程，这种状态通常称为多用户状态。

（3）2 启动远程文件共享进程和精灵进程。安装并公开远程资源。本运行级扩展了多用户方式，并被公认为远程文件共享状态。

（4）3 可用来定义成另一个可供选择的多用户环境配置。它对系统操作不是必不可少的，一般也不使用它。

（5）4 停止 Linux 系统，进入 ROM 监控程序。

（6）5 停止 Linux 系统，重新引导进入由/sbin/inittab 中 initdefault 记录所定义的状态。

（7）6 重新启动计算机，并进入多用户环境。

（8）A、b、c 仅处理那些具有 a、b 或 c 运行级设置的/sbin/inittab 登记项。这些是伪状态，可以定义它们运行某些命令，但它们不会引起当前运行级的改变。

（9）Q、q 重新检验/sbin/inittab。

（10）S、s 进入单用户方式。这时，执行这条命令的终端变为系统主控制台。这是唯一不要求存在适当的格式化/sbin/inittab 文件的运行级。如果该文件不存在，那么默认 init 能够进入的唯一合法运行级是单用户方式。当系统升入 S 或 s 时，没有安装用户文件的文件系统，仅必不可少的内核进程在运行。系统降至 S 或 s 时，保留所有已安装的文件系统，杀死由 init 启动的所有仅能在多用户方式下运行的进程。另外，还将杀死每一个具有 utmp 记录的进程。最后一个条件保证，将杀死全部由 SAC 启动的端口监控进程及由这些端口监控进程启动的服务，包括 ttymon 注册服务。其他不是由 init 直接启动的进程将继续运行。例如 cron 继续运行。

其中2～5都是multi-user的runlevel，通常runlevel(2～5)越高，所提供的服务也就越多。当系统资源有所变动(例如电力)时，可以用telinit去告知init要变换runlevel(例如原来是runlevel=3，用telinit 2使runlevel降为2)，这样子可以关掉一些网络资源服务；或例如telinit S 与telinit 1都是到Single-user mode(但前者不同的是，telinit S 根本就直接在/dev/console上执行一个/bin/sh给你用；后者会去执行/etc/init.d/rc 1 这个指令)；telinit 6就等于reboot。

注意：由init建立的进程以UID为0运行(root)从/etc/inittab运行的程序也作为root运行，所以系统管理员要确保自己知道/etc/inittab中的程序做什么工作，确保这些程序以及这些程序所在的目录直到/和/etc/inittab除root外无人可写。

Linux系统引导时，调用init,这时出现下述情况。首先init在/sbin/inittab中寻找记录initdefault。如果有该记录，那么通常init将使用其中指定的运行级作为初始进入的运行级。如果/sbin/inittab中没有init default记录，那么init要求用户从虚拟的系统主控台上写入一个运行级。如果写入的是S或s,那么init进入单用户状态。在单用户状态下，虚拟系统主控台被指定为用户的终端，并为读写打开。调用/sbin/su命令，在物理的主控台上产生一条消息，说明虚拟主控台已定位在何处。用init命令发信号给init，改

变系统的运行级。注意，当shell结束（通过一个文件终结符）时，如果文件/sbin/inittab不存在，那么init将只是重新初始化为单用户状态。

如果写入0～6，那么init进入相应的运行级。运行级0、5和6是关闭系统时预定的状态。运行级2、3、4可用于多用户操作状态。

如果这是自开机以来，init首次进入非单用户方式的运行级，那么init首先扫描/sbin/inittab,寻找boot和bootwait记录。假定进入的运行级与记录的运行级相匹配，在对/sbin/inittab进行任何其他加工之前执行这些记录。这样，操作系统的每一项专门的初始化，例如安装文件系统，可以在允许用户进入系统之前完成。随后，init继续扫描/sbin/inittab，并在该运行级执行加工处理的所有其他记录。

1.3 几个重要文件

1.3.1 Inittab 文件

为了生成/sbin/inittab中的每一个进程，init读每一项记录，并为将要重新产生的每一项记录建立一个子进程。

Init一开始就去读/etc/inittab，这个inittab中很清楚地设定各个系统状态要运行哪些rc或spawn。

以下是inittab文件的一个例子：

```
# /etc/inittab:  init(8) configuration。
# 将缺省的运行级设为 3;
id: 3: initdefault:
# 开机的系统设置/初始化脚本 system configuration/initialization script.
# 除了紧急模式，首先运行下一行 This is run first except when booting in emergency
#(-b) mode.
si: sysinit: /etc/init.d/boot

# 如果是单用户模式将执行什么操作呢？请看下一行。sulogin 即为 Single User LOGIN。
~~: S: wait: /sbin/sulogin

# /etc/init.d 执行 the S and K scripts 取决于系统状态的变化。其中 /etc/init.d/rc
#是一个 shell script,
# 后面的 0～6 参数表示要运行该系统状态所应运行的设定 script.
# 如前所述，系统状态 0 表示关机
# 1 表示单用户模式
#状态 2～5 表示多用户模式
```

```
#状态 6 表示重启动。

l0: 0: wait: /etc/init.d/rc 0
l1: 1: wait: /etc/init.d/rc 1
l2: 2: wait: /etc/init.d/rc 2
l3: 3: wait: /etc/init.d/rc 3
l4: 4: wait: /etc/init.d/rc 4
l5: 5: wait: /etc/init.d/rc 5
l6: 6: wait: /etc/init.d/rc 6

#当 CTRL-ALT-DEL 按下去了，该做什么？一般都是 shutdown -r now
ca: 12345: ctrlaltdel: /sbin/shutdown -t1 -r now
# Action on special keypress (ALT-UpArrow).
kb: : kbrequest: /bin/echo "Keyboard Request--edit /etc/inittab to let this
work."

系统掉电时将会发生什么呢？
pf: : powerwait: /etc/init.d/genpowerfail start
#pn: : powerfailnow: /etc/init.d/genpowerfail now
po: : powerokwait: /etc/init.d/genpowerfail stop
#pg: : powerokwait: /etc/init.d/genpowerfail stop

# /sbin/getty invocations for the runlevels.
# console 出来了。
# 格式 (Format) 如下 (其中 The "id" field MUST be the same as the last
# characters of the device (after "tty")) 。# 例如在 runlevel=3 时， 会有
六个
#virtual console (tty1~tty6):
#  <id>: <runlevels>: <action>: <Process>
1: 2345: respawn: /sbin/agetty 19200 tty1
2: 23: respawn: /sbin/agetty 19200 tty2
3: 23: respawn: /sbin/agetty 19200 tty3
4: 23: respawn: /sbin/agetty 19200 tty4
5: 23: respawn: /sbin/agetty 19200 tty5
6: 23: respawn: /sbin/agetty 19200 tty6
```

上面的设定，令系统在一运行完Kernel bootstrapping后，就去执行/etc/init.d/boot这个shell script，如果没什么问题，进入缺省运行级（default runlevel），

下面要讲到的是/etc/init.d/boot，首先讲这个script应该做些什么才是我们要的，然后再来讲一下/etc/init.d/boot这个script。

1.3.2 Boot script文件

在内核引导（Kernel bootstrapping）完以后，就要开始做一些很基本的检查、设定，以及做一些准备工作，这就是Boot script文件完成的工作。

1. 概述

在 mount root as read-only前会先做一些工作，先设定这个script的路径（PATH）及umask，挂入Kerneld这个服务（daemon），并执行如下操作：

- mdadd -ar， 把md device运行起来。
- swap on -a，把所有的swap partition打开来用。
- 挂入update(bdflush)这个daemon。

Kerneld是Linux Kernell.3.xx有了modules化后，一个会自动插modules进Kernel的daemon，也会把经一段时间后不曾用到的modules拔出Kernel。

md device是Linux Kernel 1.3.69后新加入的功能，它把两个以上的partition合成一个大的md device之后，直接做出file system或swap space，而且可以“交错地”安排block位置，这就像RAID-0一样，所以不但可以将一堆小的partition合成大的来用，也可以增进速度。

update(bdflush)这个daemon是每隔一段时间(预设值是5秒)就把“dirty blocks”flush回disk中。这个一定要在运行fsck等主要的I/O动作前就先挂入的了。

最基本的准备完毕后，接下来就要fsck了。首先是把“/”（“根”）mount起来，mount成read-only型：

```
* mount -n -o remount, ro /
```

其中-n参数是不把mount的动作写入/etc/mtab中，因为现在是把“/”mount成read-only，根本不能写入。然后开始fsck：

```
* fsck -A -a
```

参数-A是对/etc/fstab中的东西全部check一次，-a的参数是指auto-repair。在检查后如果有东西实在是不能修好，就会执行sulogin，然后reboot。如果正常，那就把‘/’remount成读写（read-write）型：

```
* mount -n -o remount, rw /
```

因为后面还会mount -a，所以这次还是用了-n参数。接下来运行如下操作：

- 运行modules的设定。
- 把/etc及/下的一些文件清除。
- 更新psdatabase。

2. Boot script示例

举例如下：

```
----------[/etc/init.d/boot 部分内容]------------------------
# 调用合适的模块
if [ -x /etc/init.d/modules ]
then
  /etc/init.d/modules
fi
```

```
#删除/etc/matb*, /etc/rmtab, /etc/nologin and /fastboot 文件
rm -f /etc/mtab* /etc/nologin /fastboot /etc/rmtab

# 更新/etc/psdatabase 文件，以下两种方法均可。
psupdate 2> /dev/null 或 ps -U 2> /dev/null
```

接下来就是把所有的本地分区（local partitions）都mount起来。如下：

```
* mount -avt nonfs
```

参数-t nonfs有什么用处呢？因为我们还没开始设定network。如果有一些swap file是在mount -a后才出现的，这时就要再运行一次swapon：

```
 * swapon -a 2>/dev/null
```

执行上述命令后才可以把swap file开来用。

然后设定网络（就是去调用一个独立的script，如果这个script不存在，我们就无法设定network)及主机名称，然后再mount -a -t nfs来加挂别人export出来的fs。

我们看看下面的例子：

```
if [ -x /etc/init.d/network ]
then
   /etc/init.d/network
fi
# 然后设定 hostname
# 如果没有/etc/HOSTNAME，使用缺省值。
if [ ! -r /etc/HOSTNAME ];  then
   echo "xxxx.xxxx.xxxx.xxxx" > /etc/HOSTNAME
fi
cat /etc/HOSTNAME | cut -f1 -d .> /etc/hostname
hostname --file /etc/hostname
# TCP/IP配置好了，接下来mountNFS文件系统至/etc/fstab中。
echo "Mounting remote file systems..."
mount -a -t nfs
```

这时才把所有的文件系统(含nfs)都mount起来了，所以现在立刻要做的事，就是更新/etc/ld.so.cache这个文件，设定system clock，然后清除/tmp、/var/run及/var/lock下的大部分垃圾，如下：

```
/sbin/ldconfig
clock -s
```

清除 /tmp，/var/run，/var/lock下的垃圾，/tmp，/var/run及/var/lock这些目录下的垃圾都清空了，这时才去执行/etc/rc.boot/下的所有script(其中run-parts是一个工具程序，它会把给定的参数[目录]下所有的scripts都运行一次)：

```
run-parts /etc/rc.boot
```

如果没有run-parts这个工具，自己可以试着用shell script写一个，或是写在这个/etc/init.d/bootscript内。

然后修改/dev/ttyXX的属性，如下：

```
chmod 666 /dev/tty[pqrstuvwxyzabcde]*
chown root.tty /dev/tty[pqrstuvwxyzabcde]*
```

最后看看还有什么工作想在这儿先处理，可以一并在此写入，或是写个script放到/etc/rc.boot/下也是一样的。例如把powerd运行起来，建立/etc/motd，建立/etc/issue.net，建立一些links。

以上讨论了boot文件应该完成的工作，下面是一个/etc/init.d/boot例子：

```
PATH="/sbin: /bin: /usr/sbin: /usr/bin"
umask 022
echo
echo "Running /etc/init.d/boot..."
echo
# 激活Kerneld进程
if [ -x /sbin/Kerneld ];  then
  /sbin/Kerneld
fi
# 把 md 运行起来
if [ -s /etc/mdtab -a -f /sbin/mdadd ]
then
  mdadd -ar
fi
echo "Activating swap..."
swapon -a 2>/dev/null
# 确保dbflush调用主要的I/O以前就已经运行起来了。
# 以下就是一个这方面的例子。
update &
# 检查所有文件系统的完整性（非快速启动情况下）。
if [ ! -f /fastboot ]
then
# 在执行fsck前，确保启动是Ensure that root is quiescent and read-only
#before fsck'ing.
   mount -n -o remount, ro /
   if [ $? = 0 ]
   then
     echo "Checking file systems..."
     fsck -A -a
  #如果失败，则转入单用户模式。
  #
  # 注意："失败"由一个 2 或大于 2 的返回码表示。返回码为 1 则表示文件系统错误已被#修正，启动继续进行。
```