Machine Translated by Google

LPIC-2 考试准备

第 6 版,适用于 4.5 版

版权所有 2013-2017 Snow BV

Machine Translated by Google

版权所有 © 2001、2002、2003、2004、2005、2006、2007、2008、2009、2010、2011、2012、2013、2014、2015、2016、2017 Snow BV

Machine Translated by Google

合作者

	标题： LPIC-2 考试准备		
行动	姓名	日期	签名
写的	由许多 Snow BV 同事撰写、更新和审阅。、乔斯·詹森和约斯特·赫尔伯格	2017年	

修订记录

数字	日期	描述	姓名

版权所有 2013-2017 Snow BV

Machine Translated by Google

内容

0 容量规划 (200) 1个
0.1 衡量资源使用情况并排除故障 (200.1)。 . . 1个
0.1.1 目标。 . . . 1个
0.1.2 iostat 。 2个
0.1.3 iotop 。 3个
0.1.4 虚拟机统计。 3个
0.1.5 网络统计。 4个
0.1.6 秒。 5个
0.1.7 iptraf 。 6个
0.1.8 皮秒。 6个
0.1.9 pstree 。 7
0.1.10 瓦特。 7
0.1.11 lsof。 8个
0.1.12 免费。 8个
0.1.13 顶部。 8个
0.1.14 顶层。 9
0.1.15 正常运行时间。 10
0.1.16 特区。 10
0.1.17 将系统症状与可能出现的问题进行匹配/关联。 11
0.1.18 估计吞吐量并识别系统中的瓶颈,包括网络。 11
0.2 预测未来的资源需求 (200.2)。 12
0.2.1 13
0.2.2 监控 IT 基础设施 13
0.2.3 预测未来增长。 13
0.2.4 资源枯竭 13
0.3 问与答。 14

版权所有 2013-2017 Snow BV

Machine Translated by Google

1 Linux 内核 (201) 15
1.1 内核组件 (201.1)。 15
1.1.1 关键知识领域：. 15
1.1.2 条款和实用程序。 15
1.1.3 不同类型的内核映像。 16
1.1.4 内核和补丁的编号方案概述。 16
1.1.5 计划到 2.6.0 内核 16
1.1.6 内核版本控制从内核版本 2.6.0 到 3.0。 16
1.1.7 内核版本控制从 3.0 版到 4.0 版。 17
1.1.8 内核版本控制从 4.0 开始。 17
1.1.9 XZ压缩。 17
1.1.10 什么是内核模块 17
1.2 编译 Linux 内核 (201.2)。 18
1.2.1 重点知识领域 18
1.2.2 条款和实用程序。 18
1.2.3 获取内核源码 19
1.2.4 清理内核 19
1.2.5 创建.config 文件 20
1.2.6 进行配置。 21
1.2.7 制作菜单配置。 21
1.2.8 制作xconfig和gconfig 。 22
1.2.9 制作旧配置。 25
1.2.10 编译内核 25
1.2.10.1打扫卫生。 25
1.2.10.2 制作zImage/bzImage 。 25
1.2.10.3 制作模块。 25
1.2.10.4 制作模块安装。 25
1.2.11 安装新内核 26
1.2.12 初始内存盘（initrd）。 26
1.2.13 手动创建initrd 26
1.2.14 给内核打补丁 28
1.2.15 从生产内核中删除内核补丁 29
1.2.16 知识管理系统。 29
1.2.17 德拉卡特 31
1.3 内核运行时管理和故障排除 (201.3) 。 31
1.3.1 自定义、构建和安装自定义内核和内核模块 32
1.3.2 操作模块 32
1.3.3 配置模块 35
1.3.4 模块依赖文件 36

版权所有 2013-2017 Snow BV

Machine Translated by Google

1.3.5 kmod 与 kerneld 。 36
1.3.6 构建自定义内核 37
1.3.7 /proc 文件系统。 37
1.3.8 /、/boot 和 /lib/modules 的内容。 37
1.3.9 跟踪软件及其系统和库调用的工具和实用程序 38
1.3.10 启动过程。 40
1.3.11 硬件和内核信息 41
1.3.12 开发者。 44
1.4 问与答。 45

2 系统启动（202） 47
2.1 定制系统启动（202.1）。 47
2.1.1 重点知识领域 47
2.1.2 条款和实用程序。 47
2.1.3 使用 mkinitrd 创建 initrd。 48岁
2.1.4 使用 mkinitramfs 创建 initrd。 48岁
2.1.5 设置根设备。 49
2.1.6 Linux 启动过程。 49
2.1.7 初始化进程。 50
2.1.7.1 配置 /etc/inittab 。 51
2.1.7.2 /etc/init.d/rc 脚本。 52
2.1.8 更新-rc.d。 53
2.1.9 使用 systemd 目标。 54
2.1.10 LSB标准 56
2.1.11 bootscript环境及命令 57
2.1.11.1 更改和配置运行级别。 57
2.1.11.2 chkconfig 命令 58
2.2 系统恢复（202.2）。 59
2.2.1 目标。 59
2.2.2 重点知识领域 59
2.2.3 条款和实用程序。 59
2.2.4 GRUB 解释。 60
2.2.4.1 格鲁布 2 60
2.2.4.2 GRUB 配置文件 61
2.2.5 与 GRUB Legacy 的区别。 62
2.2.6 GRUB 旧版。 62
2.2.7 影响正常开机过程。 63
2.2.7.1 选择另一个内核 63
2.2.7.2 引导至单用户模式或特定运行级别。 64

版权所有 2013-2017 Snow BV

Machine Translated by Google

2.2.7.3 切换运行级别。 64
2.2.7.4 向内核传递参数 64
2.2.7.4.1 如果设备不工作。 64
2.2.8 救援启动过程。 64
2.2.8.1 fsck启动失败时 64
2.2.8.2 如果您的根 (/) 文件系统已损坏。 65
2.2.8.2.1 使用发行版的引导媒体 65
2.2.9 UEFI 和 NVMe 引导注意事项 65
2.3 备用引导加载程序 (202.3)。 67
2.3.1 重点知识领域 67
2.3.2 条款和实用程序。 67
2.3.3 星际宝贝。 68
2.3.3.1 /etc/lilo.conf 和 /sbin/lilo 。 68
2.3.4 SYSLINUX、ISOLINUX、PXELINUX:Syslinux 项目。 69
2.3.4.1 系统操作系统。 69
2.3.4.2 SYSLINUX 安装程序选项。 69
2.3.4.3 MSDOS 特定选项。 69
2.3.4.4 仅限 Linux。 69
2.3.4.5 ISOLINUX、EXTLINUX、PXELINUX 的例外情况。 69
2.3.5 Syslinux 引导配置。 70
2.3.6 PXELINUX。 70
2.3.7 了解 PXE 71
2.3.8 PXE 的代理 DHCP 72
2.3.9 DHCP 请求示例。 72
2.3.10 带有 UEFI 的系统。 73
2.3.11 使用 Systemd-boot 启动。 74
2.3.12 使用 Das U-boot 启动。 74
2.4 问与答。 74

3 文件系统和设备 (203) 76
3.1 操作 Linux 文件系统 (203.1) 。 76
3.1.1 重点知识领域 76
3.1.2 条款和实用程序。 76
3.1.3 文件层次结构。 77
3.1.4 文件系统。 77
3.1.5 创建文件系统。 78
3.1.6 挂载和卸载。 78
3.1.7 Systemd 安装单元。 80
3.1.8 交换。 80

版权所有 2013-2017 Snow BV

Machine Translated by Google

3.1.9 UUID。 82
3.1.10同步。 83
3.2 维护 Linux 文件系统 (203.2)。 83
3.2.1 重点知识领域 83
3.2.2 条款和实用程序。 83
3.2.3 磁盘检查。 83
3.2.4 fsck (fsck.*) 。 84
3.2.5 mkfs (mkfs.*) 。 84
3.2.6 tune2fs 85
3.2.7 dumpe2fs 。 85
3.2.8 坏块 86
3.2.9 调试文件。 86
3.2.10 ext4 86
3.2.11 btrfs 87
3.2.12 马克斯瓦普。 90后
3.2.13 xfs_info 。 90后
3.2.14 xfs_check 。 90后
3.2.15 xfs_repair 。 90后
3.2.16 smartmontools:smartd和smartctl 90后
3.2.17 ZFS:zpool 和 zfs 90后
3.3 创建和配置文件系统选项 (203.3)。 92
3.3.1 重点知识领域 92
3.3.2 条款和实用程序。 92
3.3.3 Autofs 和自动挂载器 93
3.3.4 CD-ROM 文件系统。 95
3.3.4.1 为 CD-ROM 创建映像。 95
3.3.4.2 测试 CD 映像。 96
3.3.4.3 将 CD 映像写入 CD。 96
3.3.4.4 制作数据 CD 的副本。 97
3.3.5 加密文件系统。 97
3.4 问与答。 100

4 高级存储设备管理（204） 101
4.1 配置 RAID (204.1) 。 101
4.1.1 重点知识领域 101
4.1.2 条款和实用程序。 101
4.1.3 什么是RAID? 101
4.1.4 RAID 级别。 102
4.1.5 硬件RAID 103

版权所有 2013-2017 Snow BV

Machine Translated by Google

4.1.6 软件RAID 103
4.1.7 在您的 Linux 系统上识别 RAID。 103
4.1.8 配置 RAID（使用 mdadm）。 104
4.1.9 配置 RAID（备选）。 105
4.2 调整存储设备访问（204.2）。 106
4.2.1 关键知识领域 106
4.2.2 条款和实用程序。 106
4.2.3 配置磁盘 107
4.2.3.1 配置 iSCSI。 107
4.2.3.1.1 iSCSI 启动器 107
4.2.3.1.2 iSCSI 目标。 108
4.2.3.1.2.1 添加新目标（在目标主机上）。 108
4.2.3.1.3 世界大战。 109
4.2.3.1.4 伦。 110
4.2.3.1.4.1 配置设备名称持久性（也称为 LUN 持久性）。 110
4.2.3.1.4.2 在没有多路径的情况下实现设备名称持久性。 110
4.2.3.1.4.3 使用多路径实现设备名称持久性。 111
4.2.3.2 物理安装 112
4.2.3.3 使用 tune2fs 。 113
4.2.3.4 使用 hdparm。 113
4.2.3.5 使用 sdparm 。 114
4.2.4 配置内核选项。 114
4.2.4.1 使用 /proc 文件系统。 114
4.2.4.2 使用系统控制。 115
4.3 逻辑卷管理器（204.3）。 115
4.3.1 重点知识领域 115
4.3.2 条款和实用程序。 116
4.3.3 配置逻辑卷管理。 116
4.3.4 修改逻辑卷、卷组和物理卷。 117
4.3.5 LVM 快照。 118
4.3.6 LVM 命令 118
4.3.7 设备映射器。 119
4.3.8 lvm.conf 。 119
4.4 问与答。 120

版权所有 2013-2017 Snow BV

Machine Translated by Google

5 网络配置（205） 121
5.1 基本网络配置（205.1）。 121
5.1.1 重点知识领域 121
5.1.2 条款和实用程序。 121
5.1.3 配置网络接口 122
5.1.4 环回接口 122
5.1.5 以太网接口 123
5.1.6 通过网关路由。 123
5.1.7 ip 命令 123
5.1.8 ARP,地址解析协议 125
5.1.9 无线网络。 125
5.1.9.1 我。 125
5.1.9.2 iwconfig 。 126
5.1.9.3 iwlist 。 127
5.2 高级网络配置和故障排除 (205.2)。 127
5.2.1 重点知识领域 127
5.2.2 条款和实用程序。 127
5.2.3 虚拟专用网。 128
5.2.3.1 什么是 VPN。 128
5.2.3.2 VPN 类型。 128
5.2.3.2.1 IPSEC。 128
5.2.4 故障排除。 132
5.2.4.1 if 配置。 132
5.2.4.2 ping 和 ping6 。 133
5.2.4.3 路线 133
5.2.4.4 跟踪路由 134
5.2.4.5 arp 和 arpwatch 。 135
5.2.4.6 TCP 转储。 135
5.2.4.7 地图。 135
5.2.4.8 线鲨。 136
5.2.4.9 /usr/sbin/lsof 。 136
5.2.4.10 /usr/sbin/ss 。 137
5.2.4.11 /bin/netstat 。 137
5.2.4.12 /usr/bin/nc 。 137
5.2.4.13 /usr/bin/mtr 。 138
5.2.4.14 /sbin/ip 。 138
5.3 解决网络问题 (205.3) 。 138
5.3.1 重点知识领域 138
5.3.2 条款和实用程序。 138

版权所有 2013-2017 Snow BV

Machine Translated by Google

5.3.3 网络故障排除介绍。 139
5.3.4 示例情况 139
5.3.5 名称解析问题。 141
5.3.6 系统初始化错误。 142
5.3.7 安全设置。 142
5.3.8 网络配置。 142
5.3.9 网络管理器。 142
5.4 问与答。 142

6 系统维护 (206) 144
6.1 从源 (206.1) 制作和安装程序。 144
6.1.1 重点知识领域 144
6.1.2 条款和实用程序。 144
6.1.3 源码解包 145
6.1.4 从源代码构建 145
6.1.5 补丁 146
6.2 备份操作（206.2）。 147
6.2.1 重点知识领域 147
6.2.2 条款和实用程序。 147
6.2.3 为什么？ 148
6.2.4 什么？ 148
6.2.5 什么时候？ 148
6.2.6 如何？ 149
6.2.7 在哪里？ 149
6.2.7.1 胶带。 149
6.2.7.2 磁盘 150
6.2.7.3 光学媒体 150
6.2.7.4 远程/网络存储。 150
6.2.8 备份实用程序 150
6.2.8.1 同步。 150
6.2.8.2 焦油 . . 150
6.2.8.3 日 . . 151
6.2.8.4 中央处理器。 151
6.2.9 备份方案 151
6.3 就系统相关问题通知用户 (206.3)。 152
6.3.1 重点知识领域 152
6.3.2 条款和实用程序。 152
6.3.3 /etc/issue、/etc/issue.net 和 /etc/motd 文件 152
6.3.4 墙命令 153

版权所有 2013-2017 Snow BV

Machine Translated by Google

6.3.5 关机指令通讯。 153
6.3.6 systemctl 命令通信，. . . . 153
6.3.6.1 管理服务。 153
6.3.6.1.1 启动和停止服务。 153
6.3.6.1.2 重新启动和重新加载服务。 154
6.3.6.1.3 启用和禁用服务。 154
6.3.6.1.4 检查服务状态。 154
6.3.6.2 系统状态概览。 155
6.3.6.2.1 列出当前单位 155
6.3.6.2.2 列出单元文件 156
6.3.6.3 单元管理。 156
6.3.6.3.1 显示单元文件 156
6.3.6.3.2 显示依赖关系。 . 157
6.3.6.3.3 检查单元属性。 . 157
6.3.6.3.4 屏蔽和取消屏蔽单元 157
6.3.6.4 编辑单元文件 158
6.3.6.5 使用目标调整系统启动（运行级别）。 159
6.3.6.5.1 获取和设置默认目标。 159
6.3.6.5.2 列出可用目标。 . . 159
6.3.6.5.3 隔离目标。 159
6.3.6.5.4 对重要事件使用快捷方式。 160
6.4 问与答。 160

7 域名服务器 (207) 162
7.1 基本 DNS 服务器配置 (207.1) 。 162
7.1.1 关键知识领域 162
7.1.2 条款和实用程序。 163
7.1.3 BIND 中的名称服务器组件。 163
7.1.4 named.conf 文件 164
7.1.4.1 named.conf 的位置。 164
7.1.4.2 仅缓存名称服务器。 164
7.1.4.3 语法。 165
7.1.4.4 选项声明 165
7.1.4.5 日志语句 167
7.1.4.6 预定义区域语句 167
7.1.5 具名名称服务器守护进程 168
7.1.6 rndc 程序。 168
7.1.7 named-checkconf 实用程序。 169
7.1.8 向 named 发送信号。 169

版权所有 2013-2017 Snow BV

Machine Translated by Google

7.1.9 使用启动/停止脚本控制命名。 169
7.1.10 域名管理系统。 169
7.1.10.1 动态域名。 170
7.1.10.2 电源域名系统。 170
7.1.11 dig 和 host 实用程序 170
7.2 创建和维护 DNS 区域 (207.2)。 174
7.2.1 关键知识领域 174
7.2.2 条款和实用程序。 174
7.2.3 区域和反向区域 174
7.2.3.1 db.local 文件 175
7.2.3.2 db.127 文件 175
7.2.3.3 提示文件 176
7.2.3.4 区域文件 176
7.2.3.4.1 $TTL 语句 177
7.2.3.4.2 SOA资源记录 177
7.2.3.4.3 A资源记录 179
7.2.3.4.4 CNAME资源记录 179
7.2.3.4.5 NS资源记录 179
7.2.3.4.6 MX资源记录 179
7.2.3.4.7 MX 域 180
7.2.3.5 反向区域文件 180
7.2.3.5.1 PTR记录 180
7.2.3.6 IPv6 记录 181
7.2.3.6.1 IPv6地址格式 181
7.2.3.6.2 AAAA记录 181
7.2.3.6.3 PTR记录 181
7.2.4 主从服务器。 182
7.2.4.1 配置主站 182
7.2.4.2 配置从站。 182
7.2.4.3 存根名称服务器。 183
7.2.5 创建子域 183
7.2.5.1 授权 DNS 区域 183
7.2.6 检查区域文件的语法错误 184
7.2.7 DNS 实用程序。 184
7.2.7.1 挖掘程序。 185
7.2.7.2 宿主程序。 186
7.2.7.3 nslookup 程序。 186
7.2.7.4 DNS 漫游。 187
7.3 保护 DNS 服务器 (207.3)。 187

版权所有 2013-2017 Snow BV

Machine Translated by Google

7.3.1 关键知识领域 187
7.3.2 条款和实用程序。 188
7.3.3 DNS 安全策略。 188
7.3.4 使信息更难获得。 188
7.3.4.1 隐藏版本号 188
7.3.4.2 限制访问 189
7.3.4.2.1 限制区转移 189
7.3.4.2.2 限制查询。 191
7.3.5 控制请求。 192
7.3.6 限制入侵的影响 192
7.3.6.1 以较少的权限运行 BIND。 192
7.3.6.2 在 chroot 监狱中运行 BIND。 192
7.3.6.2.1 准备一个 chroot 监狱。 193
7.3.6.2.2 运行 BIND chroot 193
7.3.6.2.3 chrooted BIND 的配置。 194
7.3.6.3 结合特殊用户和 chroot 194
7.3.7 保护名称服务器连接 195
7.3.7.1 密码学速成班 195
7.3.7.2 使用 dnssec-keygen 命令 196
7.3.7.3 密钥文件格式 196
7.3.7.4 使用钥匙。 197
7.3.8 dnssec-signzone 。 197
7.3.9 丹麦人。 198
7.3.10 内部 DNS。 200
7.3.10.1 限制协商。 200
7.3.10.2 拆分 DNS:独立的内部主机 200
7.3.10.2.1 在 privdns 上配置主机。 201
7.3.10.2.2 在 liongate 上配置 DNS。 203
7.3.10.2.2.1 备选方案。 203
7.3.10.3 拆分 DNS:一台机器上有两个 DNS 服务器 203
7.3.10.3.1 一台主机上的两个名称服务器 203
7.3.10.3.2 配置内部名称服务器。 204
7.3.10.3.3 配置可见名称服务器。 204
7.3.10.3.3.1 一个问题。 206
7.3.11 TSIG。 206
7.3.11.1 为 BIND 配置 TSIG。 206
7.4 问与答。 207

版权所有 2013-2017 Snow BV

Machine Translated by Google

8 HTTP 服务 (208) 209
8.1 基本 Apache 配置 (208.1)。 209
8.1.1 关键知识领域 209
8.1.2 术语和实用程序 210
8.1.3 安装 Apache 网络服务器。 210
8.1.4 模块化。 211
8.1.5 模块运行时加载 (DSO)。 211
8.1.5.1 APache eXtenSion (APXS) 支持工具 212
8.1.6 监控 Apache 负载和性能。 212
8.1.7 增强 Apache 性能。 212
8.1.8 Apache access_log 文件 213
8.1.9 Apache error_log 文件 213
8.1.10 限制客户端用户访问 214
8.1.11 配置认证模块 217
8.1.12 用户文件 218
8.1.13 组文件 219
8.1.14 配置 mod_perl。 219
8.1.14.1 从源代码构建 Apache 220
8.1.15 配置 mod_php 支持。 220
8.1.16 httpd 二进制文件。 221
8.1.17 配置 Apache 服务器选项。 221
8.1.18 Apache 虚拟主机。 222
8.1.18.1 基于名称的虚拟主机。 222
8.1.18.2 基于 IP 的虚拟主机。 224
8.1.18.2.1 设置多个守护进程 224
8.1.18.2.2 设置单个守护进程 225
8.1.19 自定义文件访问 225
8.2 HTTPS (208.2) 的 Apache 配置。 225
8.2.1 关键知识领域 226
8.2.2 条款和实用程序： 226
8.2.3 Apache2 配置文件 226
8.2.4 加密网络服务器:SSL 。 227
8.2.4.1 公钥加密。 227
8.2.4.2 带有 mod_ssl 的 Apache。 228
8.2.5 目录 /etc/ssl/ 。 229
8.2.6 如何创建 SSL 服务器证书。 230
8.2.7 Apache SSL 指令。 233
8.2.8 为 Apache 创建和安装自签名证书。 234
8.2.9 其他 Apache 指令。 236

版权所有 2013-2017 Snow BV

Machine Translated by Google

8.2.10 SSL 与 Apache 虚拟主机 236
8.2.11 SSL 安全问题 237
8.2.12 禁用不安全的协议和密码。 238
8.3 将 Squid 实现为缓存代理（208.3）。 239
8.3.1 关键知识领域 239
8.3.2 以下是所用文件、术语和实用程序的部分列表： 239
8.3.3 网络缓存。 239
8.3.4 鱿鱼。 239
8.3.5 重定向器 241
8.3.6 认证器 242
8.3.7 访问策略。 243
8.3.8 认证者行为。 244
8.3.9 利用内存使用。 244
8.4 将 Nginx 实施为 Web 服务器和反向代理 (208.4)。 245
8.4.1 关键知识领域 245
8.4.2 条款和实用程序。 245
8.4.3 恩金克斯。 245
8.4.3.1 反向代理。 246
8.4.3.2 基本网络服务器。 247
8.5 问与答。 248

9 文件共享 (209) 250
9.1 SAMBA 服务器配置 (209.1) 。 250
9.1.1 关键知识领域 250
9.1.2 条款和实用程序。 250
9.1.3 什么是Samba？ 251
9.1.4 安装 Samba 组件。 251
9.1.5 Samba 命令 251
9.1.5.1 Samba 核心命令 251
9.1.5.1.1 smb 状态 251
9.1.5.1.2 测试参数。 252
9.1.5.1.3 密码密码。 252
9.1.5.1.4 nmblookup。 252
9.1.5.1.5 smb 客户端 253
9.1.5.1.6 桑巴工具 253
9.1.5.1.7 网 254
9.1.5.2 命令不是 Samba 核心的一部分 255
9.1.5.2.1 挂载 255
9.1.6 Samba 日志记录。 256

版权所有 2013-2017 Snow BV

Machine Translated by Google

9.1.7 账户信息数据库 256
9.1.7.1 密码密码。 256
9.1.7.2 tdbsam 256
9.1.7.3 ldapsam。 256
9.1.8 Samba 配置。 256
9.1.8.1 Samba 配置目录/etc/smb 或/etc/samba。 256
9.1.8.2 smb.conf 。 257
9.1.8.2.1 特别栏目 257
9.1.8.2.2 配置参数。 257
9.1.8.2.2.1 [全球的] 。 257
9.1.8.2.2.2 服务部门 258
9.1.9 安全级别和模式 259
9.1.9.1 用户级安全。 259
9.1.9.2 共享级安全。 259
9.1.10 例子。 260
9.1.10.1 支持示例的基本 [global] 部分。 261
9.1.10.2 示例:向所有人提供“公共”共享。 261
9.1.10.3 示例:使“share1”共享对 alice 可用 262
9.1.10.4 示例:使“share2”共享对经过身份验证的用户可用 263
9.1.10.5 示例:使主目录对其各自的所有者可用。 263
9.1.10.6 示例:将远程用户“alice.jones”映射到 Linux 用户“alice” 264
9.1.10.7 示例:使“windows”上的共享对“sambaclient”上的用户可用 265
9.1.10.8 示例:允许所有人在“sambaserver”上的所有打印机上打印。 266
9.1.10.9 示例:禁止从“sambaclient”在“Printer_1”上打印。 268
9.1.10.10 示例:列出“sambaserver”上的可用服务。 268
9.1.11 设置 nmbd WINS 服务器。 269
9.1.11.1 什么是 WINS 服务器? 269
9.1.11.2 使用 Samba 作为 WINS 服务器。 269
9.1.12 为客户端创建登录脚本 269
9.1.12.1 基于用户名 270
9.1.12.2 基于客户名称 270
9.1.13 将 Samba 配置为域成员 270
9.1.13.1 配置 DNS。 270
9.1.13.2 配置 Kerberos 271
9.1.13.3 配置 Samba。 271
9.1.13.3.1 设置 smb.conf 文件 271
9.1.13.3.2 加入域 272
9.1.13.3.3 配置名称服务开关 (NSS)。 272
9.1.13.3.4 启动服务。 272

版权所有 2013-2017 Snow BV

Machine Translated by Google

9.1.13.3.5 测试 winbind 连接。 272
9.2 配置 NFS 服务器 (209.2)。 272
9.2.1 关键知识领域 273
9.2.2 条款和实用程序。 273
9.2.3 NFS 网络文件系统。 273
9.2.3.1 客户端、服务器或两者? 273
9.2.4 设置 NFS。 273
9.2.4.1 NFS 的要求。 274
9.2.4.2 为 NFS 配置内核。 274
9.2.4.3 端口映射器。 274
9.2.4.3.1 保护端口映射器。 275
9.2.4.3.2 端口映射和 rpcbind。 276
9.2.4.4 一般 NFS 守护进程 276
9.2.4.4.1 nfs-utils 包。 276
9.2.4.5 NFS 服务器软件。 276
9.2.4.5.1 NFS 守护进程 276
9.2.4.5.2 安装守护进程 276
9.2.4.5.3 锁守护进程 276
9.2.4.5.4 状态守护进程 277
9.2.4.6 导出文件系统。 277
9.2.4.6.1 文件 /etc/exports 。 277
9.2.4.6.1.1 导出选项。 278
9.2.4.6.2 exportfs 命令 279
9.2.4.6.2.1 激活导出列表 279
9.2.4.6.2.2 停用导出列表 280
9.2.4.6.3 showmount 命令 280
9.2.4.7 NFS 客户端:软件和配置。 281
9.2.5 测试 NFS。 282
9.2.5.1 showmount --exports命令 283
9.2.5.2 /proc/mounts 文件 9.2.5.3 283
rpcinfo 。 283
9.2.5.3.1 rpcinfo: 探测系统。 . rpcinfo: 发出 283
9.2.5.3.2 空请求。 284
9.2.5.4 nfsstat 命令 284
9.2.6 保护 NFS。 285
9.2.6.1 限制访问 9.2.6.2 防止人 285
为错误 285
9.2.6.3 最佳 NFS 版本。 286
9.2.7 NFS 组件概述。 286
9.2.8 NFS 协议版本。 286
9.2.9 NFSv4。 287
9.3 问与答。 287

版权所有 2013-2017 Snow BV

Machine Translated by Google

10 网络客户端管理（210） 288
10.1 DHCP 配置（210.1）。 288
10.1.1 关键知识领域 288
10.1.2 条款和实用程序。 288
10.1.3 什么是 DHCP？ 289
10.1.4 服务器如何配置？ 289
10.1.4.1 什么是（全局)参数？ 289
10.1.4.2 什么是共享网络声明？ 289
10.1.4.3 什么是子网声明？ 290
10.1.4.4 什么是组声明？ 290
10.1.4.5 什么是主机声明？ 290
10.1.5 一个例子。 290
10.1.5.1 网络架构 291
10.1.5.2 工作站可用的网络服务。 292
10.1.5.2.1 子网独立服务。 292
10.1.5.2.2 子网相关服务。 292
10.1.5.3 构建 DHCP 服务器的配置文件 292
10.1.5.3.1 服务的全局参数。 293
10.1.5.3.2 公司的共享网络和子网 293
10.1.5.4 静态主机 296
10.1.5.5 静态 BOOTP 主机 297
10.1.6 控制 DHCP 服务器的行为。 297
10.1.6.1 租赁 297
10.1.6.2 DHCP 服务器监听的接口 298
10.1.6.3 更改后重新加载 DHCP 服务器。 298
10.1.6.4 记录。 298
10.1.7 DHCP 中继。 298
10.1.7.1 什么是 DHCP 中继？ 298
10.1.8 在 IPv6 网络中分配地址。 299
10.2 PAM 认证（210.2）。 299
10.2.1 关键知识领域 299
10.2.2 条款和实用程序。 299
10.2.3 资源 300
10.2.4 什么是 PAM? 300
10.2.5 它是如何工作的？ 300
10.2.6 模块 301
10.2.6.1 pam_unix 。 301
10.2.6.2 pam_nis 。 302
10.2.6.3 pam_ldap 。 302

版权所有 2013-2017 Snow BV

Machine Translated by Google

10.2.6.4 pam_cracklib。 303
10.2.6.5 pam_limits。 303
10.2.6.6 pam_listfile。 303
10.2.7 固态硬盘。 303
10.3 LDAP 客户端使用 (210.3)。 304
10.3.1 关键知识领域 304
10.3.2 条款和实用程序。 305
10.3.3 LDAP 305
10.3.4 LDAP 搜索。 306
10.3.4.1 LDAP 过滤器 306
10.3.5 ldappasswd。 307
10.3.6 LDAP 添加。 307
10.3.7 ldap 删除。 307
10.3.8 有关 LDAP 的更多信息 308
10.4 配置 OpenLDAP 服务器 (210.4)。 308
10.4.1 关键知识领域 308
10.4.2 条款和实用程序。 308
10.4.3 开放式 LDAP。 309
10.4.3.1 访问控制。 309
10.4.3.2 专有名称。 309
10.4.3.3 slapd 配置。 310
10.4.3.4 LDIF 312
10.4.3.5 目录 313
10.4.3.6 模式和白页。 313
10.4.4 参考资料 314
10.5 问与答。 314

11 电子邮件服务 (211) 315
11.1 使用电子邮件服务器 (211.1)。 315
11.1.1 关键知识领域 315
11.1.2 条款和实用程序。 315
11.1.3 SMTP协议基础知识。 316
11.1.4 发送邮件。 317
11.1.4.1 比照 317
11.1.4.2 域 318
11.1.4.3 特征 318
11.1.4.4 破解 318
11.1.4.5 平方米 318
11.1.4.6 邮件程序 318

版权所有 2013-2017 Snow BV

Machine Translated by Google

11.1.4.7 操作系统类型。 318
11.1.4.8 嘘 318
11.1.4.9 站点配置。 318
11.1.5 重要的 sendmail 文件 318
11.1.6 抗中继。 319
11.1.7 Sendmail 测试选项。 320
11.1.8 Sendmail 和 DNS。 320
11.1.9 sendmail.cf 中的手动条目。 320
11.1.10进出口 320
11.1.11 后缀 320
11.1.11.1 Postfix main.cf 文件格式 321
11.1.11.2 Postfix master.cf 文件格式 323
11.1.11.3 后缀准备工作。 323
11.1.11.3.1 我的起源。 323
11.1.11.3.2 我的目的地。 323
11.1.11.3.3 中继域。 324
11.1.11.3.4 中继主机。 324
11.1.12 记录。 324
11.1.12.1 虚拟域 324
11.1.13 Sendmail仿真层命令 325
11.1.14 /var/spool/mail 。 325
11.1.15 。 325
11.2 管理电子邮件传送 (211.2)。 328
11.2.1 关键知识领域 328
11.2.2 条款和实用程序。 328
11.2.3 邮箱 328
11.2.4 筛分。 328
11.2.4.1 筛选语法。 329
11.2.4.1.1 控制命令 330
11.2.4.1.2 测试命令 330
11.2.4.1.3 动作命令 331
11.2.5 Mbox 和 maildir 存储格式 332
11.2.5.1 Mbox 格式 332
11.2.5.1.1 优点： 332
11.2.5.1.2 缺点： 332
11.2.5.2 邮件目录格式 332
11.2.5.2.1 优点： 332
11.2.5.2.2 缺点： 332
11.2.5.3 procmailrc 的 mbox 和 maildir 之间的配方差异。 333

版权所有 2013-2017 Snow BV

Machine Translated by Google

11.3 管理邮箱访问（211.3）。 333
11.3.1 快递。 333
11.3.2 鸽舍。 333
11.3.2.1 认证 333
11.3.2.2 邮箱位置 334
11.3.2.3 SSL。 334
11.3.2.4 POP3 服务器。 336
11.4 问与答。 337

12 系统安全 (212) 338
12.1 配置路由器 (212.1) 。 338
12.1.1 关键知识领域 338
12.1.2 条款和实用程序。 339
12.1.3 私有网络地址 339
12.1.4 网络地址转换 (NAT)。 340
12.1.5 Linux 防火墙,概述。 340
12.1.5.1 实施。 340
12.1.5.2 Netfilter“钩子” 340
12.1.5.3 表和链。 341
12.1.5.4 FILTER 表 341
12.1.5.5 NAT 表 341
12.1.5.6 MANGLE 表 341
12.1.5.7 连接跟踪:状态防火墙。 341
12.1.5.8 钩子、桌子和链条放在一起。 342
12.1.5.9 添加额外功能。 343
12.1.5.10 iptables 选项。 345
12.1.5.11 iptables 参数。 346
12.1.5.12 iptables 匹配扩展 346
12.1.5.13 带有 IPTABLES 的公司网络。 347
12.1.5.14 (1) 由防火墙发起并发往 Internet 的流量 348
12.1.5.15 (2) 从 Internet 发起并以防火墙为目的地的流量。 348
12.1.5.16 (3) 由防火墙发起并发往内部网络的流量。 348
12.1.5.17 (4) 由内部网络发起并前往防火墙的流量。 348
12.1.5.18 (5) 由内部网络发起并发往互联网的流量 349
12.1.5.19 (6) 由互联网发起并以内部网络为目的地的流量。 349
12.1.5.20 (!) 流量作为启动流量的结果 349
12.1.5.21 所有 iptables 命令放在一起。 349
12.1.6 保存和恢复防火墙规则 351
12.1.7 端口和/或 IP 转发。 352

版权所有 2013-2017 Snow BV

Machine Translated by Google

12.1.8 拒绝服务 (DoS) 攻击 353
12.1.8.1 说明。 353
12.1.8.2 预防。 353
12.1.9 使用 /proc/sys/net/ipv4 (sysctl) 防止简单的 DOS 攻击 353
12.1.10 路由 354
12.1.10.1 何时使用路由 354
12.1.11 管理路由表的工具、命令和实用程序 354
12.1.11.1 路线。 354
12.1.11.2 网络统计。 355
12.1.12 ip6 表。 355
12.2 管理 FTP 服务器 (212.2)。 356
12.2.1 关键知识领域 356
12.2.2 条款和实用程序。 356
12.2.3 FTP连接方式 356
12.2.4 主动模式 356
12.2.5 被动模式 356
12.2.6 启用通过防火墙的连接。 357
12.2.7 vsftpd 。 357
12.2.7.1 匿名上传和下载的最小配置示例。 357
12.2.8 纯 FTPd 358
12.2.8.1 配置。 358
12.2.8.2 重要的命令行选项。 358
12.2.8.3 匿名上传和下载的最小配置示例。 359
12.2.9 其他 FTP 服务器。 359
12.2.9.1 ProFTPd 359
12.3 安全外壳 (SSH) (212.3)。 359
12.3.1 关键知识领域 360
12.3.2 术语和实用程序 360
12.3.3 SSH 客户端和服务器。 360
12.3.4 键及其用途。 360
12.3.4.1 主机密钥。 360
12.3.4.2 用户密钥,公钥和私钥。 361
12.3.5 配置 sshd。 361
12.3.5.1 允许或拒绝 root 登录。 362
12.3.5.2 允许或拒绝非根登录。 362
12.3.5.3 启用或禁用 X 转发。 363
12.3.5.4 无密码认证 363
12.3.6 ssh 代理。 364
12.3.6.1 启用代理转发。 364

版权所有 2013-2017 Snow BV

Machine Translated by Google

12.3.6.2 登录会话 364
12.3.6.3 使用 ssh-agent 启用 X 会话。 364
12.3.7 使用端口映射通过 ssh 隧道化应用程序协议。 364
12.3.7.1 说明。 364
12.3.7.2 示例。 365
12.4 安全任务 (212.4)。 366
12.4.1 关键知识领域： 366
12.4.2 条款和实用程序： 366
12.4.3 数控（网猫）。 366
12.4.3.1 说明。 366
12.4.3.2 示例 netcat。使用 netcat 执行端口扫描 367
12.4.4 fail2ban 命令 367
12.4.4.1 说明。 367
12.4.5 nmap 命令 367
12.4.5.1 说明。 367
12.4.5.2 使用 nmap 命令 368
12.4.6 开放增值服务。 369
12.4.7 Snort IDS（入侵检测系统）。 369
12.4.7.1 Snort规则的基本结构 370
12.4.7.2 Snort 规则头的结构 370
12.4.8 入侵检测和预防系统。 370
12.4.9 跟踪安全警报 371
12.4.9.1 安全警报 371
12.4.9.2 漏洞追踪器。 371
12.4.9.2.1 说明。 371
12.4.9.2.2 Bugtraq 网站。 371
12.4.9.2.3 如何订阅 Bugtraq。 371
12.4.9.3 计算机紧急响应小组。 371
12.4.9.3.1 说明。 371
12.4.9.3.2 网站。 371
12.4.9.3.3 如何订阅 CERT 咨询邮件列表 371
12.4.9.4 中央情报局。 372
12.4.9.4.1 说明。 372
12.4.9.4.2 网站。 372
12.4.9.4.3 订阅邮件列表 372
12.4.9.4.4 取消订阅邮件列表 372
12.4.10 使用 telnet 测试开放邮件中继 372
12.4.10.1 说明。 372
12.4.10.2 开放邮件中继测试。 372

版权所有 2013-2017 Snow BV

Machine Translated by Google

12.5 OpenVPN (212.5)。 373
12.5.1 关键知识领域： 373
12.5.2 条款和实用程序： 373
12.5.3 开放式VPN。 373
12.5.3.1 安装 。 374
12.5.3.2 openvpn 选项。 374
12.5.3.3 配置。 374
12.5.3.3.1 简单的点对点示例。 374
12.6 问与答。 375

13 参考书目 378

LPIC 2 级目标 382

14 指数 384

版权所有 2013-2017 Snow BV

Machine Translated by Google

图列表

2.1 pxelinux.0 嵌入式选项（可选）。 71

版权所有 2013-2017 Snow BV

Machine Translated by Google

表格列表

2.1 运行级别和目标。 56

3.1 Linux 文件层次结构回顾。 77

7.1 主要的 BIND 组件。 163

7.2 控制命名。 168

7.3 /etc/init.d/绑定参数。 170

9.1 NFS 的内核选项。 275

9.2 exportfs 概述。 279

9.3 showmount 概述。 280

9.4 nfsstat 程序的一些选项。 284

9.5 NFS 相关程序和文件概述 286

9.6 NFS 协议版本概述。 286

10.1 前两个八位字节是 21.31 。 292

10.2 全公司服务。 292

10.3 子网相关服务。 293

10.4 LDAP 字段操作员。 306

12.1 每个表的有效链 341

A.1 LPIC 200-206 级目标及其相对权重。 382

A.2 LPIC 级别 207-212 目标及其相对权重。 383

版权所有 2013-2017 Snow BV

Machine Translated by Google

抽象的

读者:本书旨在帮助人们准备 LPIC-2 考试。您将需要至少 2 年的 Unix 实践经验,最好是 Linux。尽管您可以在没有它的情况下参加 LPIC-2 考试,但您应该是 LPIC-1 校友才能获得 LPIC-2 认证附带的头衔和权利。

方法:我们想创建一组文档来帮助我们和其他人通过 LPIC-2 考试。本书包含通过考试所需的所有信息（以及更多信息）。

资料来源:我们的信息来源部分来自互联网上的资料。作者和其他人的实践经验以及作者所做的研究也应归功于此。我们试图在到期时给予信任,但很容易出错。我们抱歉。

警告

尽管在准备本书时采取了一切预防措施,但我们对错误或遗漏不承担任何责任。当您觉得我们没有给您适当的信任或觉得我们可能侵犯了您的权利,或者当您对我们如何改进我们的工作有任何建议时,请立即通过lpic2-editor@snow.nl通知我们,以便我们采取纠正措施。

本书的组织:本书的组织遵循 2017 年 2 月 13 日 LPIC-2 认证修订版 4.5.0 的 Linux Professional Institute 2 级目标。详细目标可通过https://www.lpi.org/获得study-resources/-lpic-2-201-exam-objectives/和https://www.lpi.org/study-resources/lpic-2-202-exam-objectives/ 。

如果有新的目标发布,这本书将很快面世。第4 版目标的书仍然可用,但不再维护,并且会在 4.5 版激活后的一段时间内消失。

作者在本书中使用 DocBook 文档标准。

Machine Translated by Google

前言

本书反映了众多经验丰富的 Linux 专业人士为准备 LPIC-2 考试所做的努力。它是 而且永远是 一项正在进行的工作。作者之前获得了他们的 LPIC-1 级别,并出于各种原因想参加考试:帮助 LPI 进行 Beta 测试,从中学习并帮助其他人通过考试。最后但同样重要的是,为了它的乐趣。

在我看来,Linux 世界最重要的里程碑之一就是证明我们技能的可能性。诚然,编写开源软件或开源文档不需要认证 您的同行当然不会要求证书,但会根据您的工作来评判您。

Linux 不仅仅是软件工程师的好玩具。它一直是一个非常稳定和值得信赖的操作系统 与它的闭源替代品相比更是如此。在闭源供应商可疑的许可政策、安全风险、错误和供应商锁定的推动下,越来越多的 IT 经理选择了 Linux 替代方案。尽管将 Linux 系统的系统管理外包是完全可行的 如今许多主要和次要参与者都支持 Linux,而且 Linux 非常稳定 但许多公司更愿意聘请自己的系统管理员。 las,大多数招聘人员不认识 Linux 管理员,即使该管理员会落在他们身上。这些招聘人员需要一种方法来证明和认可应聘者的技能。求职者还需要一种方法来向自己和招聘人员证明自己的技能。

许多组织提供 Linux 认证。其中一些是商业组织,既进行认证又进行教育。
其中一些确实做得很好 但我真诚地认为认证机构应该是独立的,尤其是独立于教育机构。 Linux Professional Institute 满足这些质量和独立性的先决条件。他们也是我们社区的一部分,因此请支持他们。

本书的初稿由数量有限的人撰写。我们的时间有限:我们在 2001 年 8 月获得了进行 Beta 考试的机会,我们提前一个月就开始了。由您来评判我们的努力,并希望改进我们的工作。对于许多人,我们无意中忘记了应有的荣誉,从心底里:我们感谢你们。

Henk Klöpping,2001 年 9 月

版权所有 2013-2017 Snow BV

Machine Translated by Google

第0章

容量规划 (200)

本主题总权重为 8 分,包含以下目标:

目标 200.1;测量资源使用情况并排除故障（6 分） 考生应该能够测量硬件资源和网络带宽,识别资源问题并排除故障。

目标 200.2;预测未来的资源需求（2 分)考生应该能够监控资源使用情况以预测
未来的资源需求。

衡量资源使用情况并排除故障 (200.1)

目标

考生应该能够测量硬件资源和网络带宽,识别和解决资源问题。

关键知识领域:

- 测量 CPU 使用率
- 测量内存使用情况
- 测量磁盘 I/O
- 测量网络 I/O
- 测量防火墙和路由吞吐量
- 映射客户端带宽使用情况。
- 将系统症状与可能出现的问题进行匹配/关联
- 估计吞吐量并在包括网络在内的系统中识别

以下是使用的文件、条款和实用程序的部分列表:

- iostat
- iotop
- vmstat
- 网络统计
- SS

版权所有 2013-2017 Snow BV

Machine Translated by Google

- iptraf
- pstree, ps
- 瓦
- lsof
- 顶部
- htop
- 正常运行时间
- 特区
- 交换
- 在 I/O 上阻塞的进程
- 积木
- 封锁
- 网络

iostat

注意根

据您的 Linux 版本和分发版本,可能需要安装像 Debian sysstat这样的软件包来安装像iostat、 sar、 mpstat等工具。Debian procps软件包包含诸如free、 uptime、 vmstat、 w、 sysctl等实用程序.

---

iostat 命令用于监视系统输入/输出 (I/O) 设备负载。这是通过观察设备的活动时间与其平均传输速率的关系来完成的。不带任何选项的 iostat 命令显示自上次系统重启以来的 iostat 统计信息。带有序列选项的 iostat 命令,使用间隔和计数,首先显示自上次系统重新引导以来的统计信息,除非使用 -y 选项省略,然后是每个指定间隔期间的统计信息。

用法:

```
$ iostat [选项] [间隔 [计数]]
```

例子:

```
$ iostat
Linux 3.2.0-4-686-pae (debian) 05/07/2013 _i686_ (2 CPU)

avg-cpu: %user %nice %system %iowait %steal %idle 1.25 0.20 0.00 94.46
                0.32          3.76

设备:                处理      kB_read/s          kB_wrtn/s          kB_read          kB_wrtn
sda                速度 12.21          214.81              17.38          333479            26980

$ iostat 1 3 Linux
3.2.0-4-686-pae (debian) 05/07/2013 _i686_ (2 CPU)

avg-cpu: %user %nice %system %iowait %steal %idle 24.24 4.97
                1.51          7.49                0.00 61.79

设备:                tps Blk_read/s Blk_wrtn/s Blk_read Blk_wrtn 14.06 249.94 121.22 1337998 648912
sda
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
avg-cpu: %user %nice %system %iowait %steal %idle 13.13 0.60
                    0.00          6.21                    0.00 80.06

设备:                    tps Blk_read/s Blk_wrtn/s Blk_read Blk_wrtn
sda                    2.51       4.01       28.86         40        288

avg-cpu: %user %nice %system %iowait %steal %idle 11.11
                    0.00          5.71          0.00          0.00 83.18

设备:                    tps Blk_read/s Blk_wrtn/s Blk_read Blk_wrtn
sda                    0.30       0.00        3.20          0         32

$ iostat -c Linux
3.2.0-4-686-pae (debian) 05/07/2013 _i686_ (2 CPU)

avg-cpu: %user %nice %system %iowait %steal %idle 9.16 0.05 17.01 1.37 0.00 72.42
```

物联网

iotop 命令类似于 top 命令。它显示 Linux 内核输出的 I/O 使用情况信息,并显示系统上进程或线程当前 I/O 使用情况的表格。

iotop 显示采样期间每个进程/线程读取和写入的 I/O 带宽的列。它还显示线程/进程在换入和等待 I/O 时花费的时间百分比。对于每个进程,显示其 I/O 优先级（类/级别）。另外,界面上方显示采样周期内读写的总I/O带宽。

用法:

```
$ iotop [选项]
```

例子:

```
$ iotop -b |头
总磁盘读取:          0.00 B/秒 |总磁盘写入:0.00 B/s |实际磁盘写入:          213.38 米/秒 0.00
实际磁盘读取:                                                                  米/秒
TID PRIO USER 6 be/4 root 191    磁盘读取磁盘写入交换 0.00 B/s 0.00 B/s 0.00 B/s 213.38 M/s    输入输出    命令
be/3 root 2976 be/4 root 1    0.00 % 0.00 % dd if=/dev/zero of=/tmp/foo 0.00 B/s 0.00
be/4 root 2 be/4 root 3 be/4    B/s 0.00 B/s 0.00 字节/秒 0.00 B/s 0.00 % 99.99 % [kworker/u8:0]
root 5 be/0 root                                          0.00 B/s 0.00 % 94.14 % [jbd2/sda1-8]
                                                          0.00 B/s 0.00 % 0.00 % 初始化 0.00 B/s 0.00 % 0.00 %
                                                          [kthreadd]
                                                          0.00 B/s 0.00 % 0.00 % [ksoftirqd/0]
                                                          0.00 B/s 0.00 % 0.00 % [kworker/0:0H]
```

虚拟机统计

vmstat 命令报告有关进程、内存、分页、块 IO、陷阱和 CPU 使用率的虚拟内存统计信息。

用法:

```
$ vmstat [选项] [延迟 [计数]]
```

例子:

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
$ vmstat 2 2 procs
----------memory---------- ---swap-- -----io---- -system-- ---- -cpu---- rb swpd free buff cache si so bo in cs us sy id wa 0 0 109112 33824
242204 26 196 516 7 11 81 1                                                                                                    双
        0                                                                    0          0603
0       0                0 109152 33824 242204                               0          0          0          2 124 239 0 1 98 0
```

请注意,第一行将始终显示自机器启动以来的平均测量值,因此应忽略。与内存和 I/O 相关的值以千字节（1024 字节)表示。旧的（2.6 之前的)内核可能会将块报告为 512、2048 或 4096 字节。与 CPU 测量相关的值表示为总 CPU 时间的百分比。在解释多 CPU 系统的测量值时,请记住这一点。 - 在这些系统上,vmstat 将 CPU 数平均到输出中。对于显示的每个间隔,所有五个 CPU 字段的总和应为 100%。这与处理器或内核的数量无关。换句话说:vmstat 不能用于显示每个处理器或内核的统计信息（在这种情况下应该使用 mpstat 或 ps）。 vmstat 将接受以秒为单位的延迟和一些计数（重复)作为参数,但进程和内存测量结果将始终保持即时。

第一个进程列“r”列出了当前分配给处理器运行队列的进程数。这些进程正在等待处理器运行时间,也称为 CPU 时间。

第二个进程列“b”列出了当前分配给块队列的进程数。这些进程被列为处于不可中断的睡眠状态,这意味着它们正在等待设备返回输入或输出 (I/O)。

第一个内存列“swpd”列出了以千字节（1024 字节)表示的正在使用的虚拟内存量。虚拟内存由来自磁盘的交换空间组成,这比内存芯片内部分配的物理内存要慢得多。

第二个内存列“free”列出了当前未使用、未缓存和未缓冲的内存量。

第三个内存列“buff”列出了当前分配给缓冲区的内存量。缓冲内存包含原始磁盘块。

第四个内存列“缓存”列出了当前分配给缓存的内存量。缓存内存包含文件。

第五个内存列“inact”列出了非活动内存的数量。这仅使用 -a 选项显示。

第六个内存列“活动”列出了活动内存的数量。这仅使用 -a 选项显示。

第一个交换列“si”列出了从磁盘换入的内存量（每秒）。

第二个交换列“so”列出了换出到磁盘的内存量（每秒）。

第一个 io 列“bi”列出了每秒从块设备接收的块数量。

第二个 io 列“bo”列出了每秒发送到块设备的块数。

第一个系统列“in”列出了每秒的中断数（包括时钟）。

第二个系统列“cs”列出了每秒上下文切换的次数。

cpu 列表示为总 CPU 时间的百分比。

第一个 cpu 列“us”（用户代码)显示了运行非内核代码所花费的时间百分比。

第二个 cpu 列“sy”（系统代码)显示了运行内核代码所花费的时间百分比。

第三个 cpu 列，“id”显示空闲时间的百分比。

第四个 cpu 列，“wa”显示了等待 I/O（输入/输出)所花费时间的百分比。

第五个 cpu 列“st”（窃取时间)显示从虚拟机窃取的时间百分比。这是虚拟机（管理程序或 VirtualBox)分配给运行虚拟机以外的任务的实际 CPU 时间量。

网络统计

netstat 命令显示网络连接、路由表、接口统计、伪装连接和多播成员。结果取决于第一个参数：

- （未给出参数）- 将列出所有已配置地址系列的所有活动套接字。

版权所有 2013-2017 Snow BV

Machine Translated by Google

- --route, -r - 显示内核路由表,输出与 route -e相同（注意:为了使用路由,提升可能需要权限,而 netstat -r可以使用用户权限运行）。
- --groups, -g - 列出 IPv4 和 IPv6 的多播组成员信息
- --interfaces, -i - 列出所有网络接口和某些特定属性
- --statistics, -s - 列出每个协议的统计信息摘要,类似于 SNMP 输出
- --masquerade, -M - 列出 pre-2.4 内核上的伪装连接。在较新的内核上,请改用 cat /proc/net/ip_conntrack。为了让它工作,必须加载 ipt_MASQUERADE 内核模块。这适用于 2.x 和 3.x 内核。

用法:

```
$ netstat [address_family_options] [选项]
```

例子:

```
$ netstat -aln --tcp
活跃的 Internet 连接（服务器和已建立）
Proto Recv-Q Send-Q 本地地址 tcp tcp                    国外地址 0.0.0.0:*          状态
                0          0 0.0.0.0:22                                           听
                0          0 127.0.0.1:25               0.0.0.0:*                 听
TCP             0          0 0.0.0.0:34236              0.0.0.0:*                 听
TCP             0          0 0.0.0.0:389                0.0.0.0:*                 听
TCP             0          0 0.0.0.0:111                0.0.0.0:*                 听
TCP6            0          0:::22                       :::*                      听
TCP6            0          0 :: 1 : 25                  :::*                      听
TCP6            0          0 :::32831                   :::*                      听

$ netstat -al --tcp
活跃的 Internet 连接（服务器和已建立）
Proto Recv-Q Send-Q 本地地址                            国外地址                  状态
tcp             0          0 *:ssh                      *:*                       听
tcp             0          0 本地主机:smtp 0 *:34236    *:*                       听
tcp             0                                       *:*                       听
tcp             0          0 *:ldap 0                   *:*                       听
tcp             0          *:sunrpc 0 [::]:ssh          *:*                       听
tcp6            0                                       [::]:*                    听
tcp6            0          0 本地主机:smtp 0 [::]:32831 [::]:*                    听
tcp6            0                                       [::]:*                    听
```

SS

ss 命令用于显示套接字统计信息。它可以显示 PACKET 套接字、TCP 套接字、UDP 套接字、DCCP 套接字、RAW 套接字、Unix 域套接字等的统计信息。它允许显示类似于 netstat 命令的信息,但它可以显示更多的 TCP 和状态信息。

大多数 Linux 发行版都附带 ss。熟悉此工具有助于增强您对系统套接字中正在发生的事情的理解,并帮助您找到性能问题的可能原因。

用法:

```
$ ss [选项] [过滤器]
```

例子:

```
$ ss -t -a
状态        接收-Q 发送-Q                    本地地址:港口                    对端地址:端口
听          0      5个                       192.168.122.1:域名                    *:*
听          0      128                                  *:ssh                     *:*
听          0      128                       127.0.0.1:ipp                        *:*
听          0      100                       127.0.0.1:smtp                       *:*
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

iptraf

iptraf 是 IP 网络的网络监控实用程序,可用于监控 IP 网络上的负载。它拦截网络上的数据包并给出有关当前 IP 流量的各种信息。

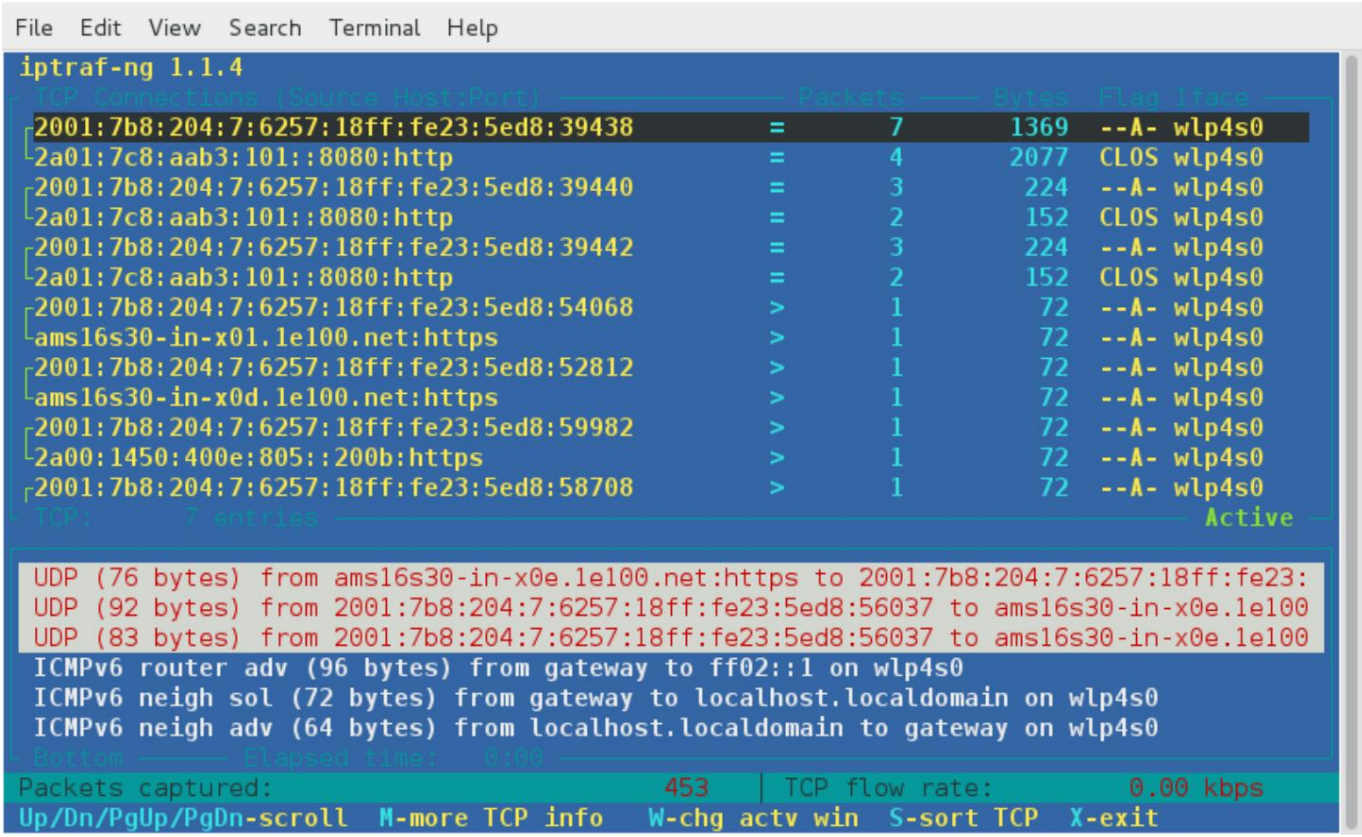
iptraf 收集 TCP 连接数据包和字节计数、接口统计信息和活动指标、TCP/UDP 流量故障以及 LAN 站数据包和字节计数等数据。 IPTraf 功能包括 IP 流量监视器,它显示 TCP 标志信息、数据包和字节计数、ICMP 详细信息、OSPF 数据包类型和超大 IP 数据包警告。

用法:

```
$ iptraf [选项]
```

例子:

```
$ iptraf
```



iptraf 窗口。

附言

用法:

```
$ ps [选项]
```

ps 命令显示当前正在运行的进程的列表。这些是 top 命令显示的相同进程。 GNU 版本的 ps 接受三种不同的选项:

1. UNIX 选项 - 这些选项可以分组并且前面必须有破折号
2. BSD 选项 - 这些可以分组并且必须在没有破折号的情况下使用

版权所有 2013-2017 Snow BV

Machine Translated by Google

3. GNU 长选项 - 这些选项前面有两个破折号

这些选项可能在某种程度上混合在 GNU ps 上,但请记住,根据您正在使用的 Linux 版本,您可能会遇到不太灵活的 ps 变体。 ps 联机帮助页可能长达近 900 行,具体取决于所查询的发行版。由于其通用性,我们鼓励您通读联机帮助页并尝试 ps 必须提供的一些选项。

例子:

```
$ ps EF
PID TTY                 统计时间命令
4417 分/0 4522 分/0       不锈钢        0:00 bashDISPLAY=:0 PWD=/home/user HOME=/home/user SESSI 0:00 \_ ps efSSH_AGENT_PID=4206 GPG_AGENT_INFO=/
                        R+            home/user/

$ ps -ef
唯一标识符          PID PPID C STIME TTY                时间指令
根                  1个     0 0 02:02 ?               00:00:01 初始化 [2] 00:00:00
根                  2个     0 0 02:02 ?               [kthreadd] 00:00:01 [ksoftirqd/0]
根                  3个     2 0 02:02 ?               00:00:01 [kworker/0:0] 00:00:00 [迁移/0] 00
根                  4个     2 0 02:02 ?               :00:00 [看门狗/0]
根                  6个     2 0 02:02 ?
根                  7       2 0 02:02 ?
```

pstree

pstree 命令显示与 ps 和 top 相同的进程,但输出格式为树。该树以 pid 为根（如果省略 pid,则为 init）,如果指定了用户名,该树将以该用户名拥有的所有进程为根。 pstree 提供了一种简单的方法来将进程追溯到其父进程 ID (PPID)。以数字为前缀的方括号之间的输出是分组在一起的相同进程分支,前缀数字表示重复计数。分组的子线程也显示在方括号之间,但进程名称将显示在大括号之间作为补充。输出的最后一行显示给定进程的子进程数。

用法:

```
$ pstree [选项] [pid|用户名]
```

例子:

```
$ pstree 3655 gnome-
terminal---bash---pstree |-bash---man---pager |-gnome-pty-
                                helpe    -3*[{gnome-terminal}]
```

w

■ w 命令显示有关当前登录到机器的用户的信息、他们的进程以及由 uptime 命令提供的相同统计信息。

用法:

```
$ w [选项] [用户]
```

例子:

```
$ w -s
02:52:10 向上 49 分钟,2 个用户,平均负载:0.11、0.10、0.13
用户        电传        从        闲置什么
用户        tty9 点/    :0       49:51 gdm-session-worker [pam/gdm3]
用户        0           :0       0.00sw-s
```

选项 -s 代表“短格式”。

版权所有 2013-2017 Snow BV

Machine Translated by Google

lsof

lsof 命令用于列出有关打开的文件及其相应进程的信息。 lsof 将处理常规文件、目录、块特殊文件、字符特殊文件、执行文本引用、库、流或网络文件。默认情况下,lsof 将显示未格式化的输出,这可能难以阅读但非常适合由其他程序解释。 -F 选项在这里起着重要的作用。 -F 选项用于获取可供 C、Perl 和 awk 等程序使用的输出。阅读联机帮助页以了解详细用法和可能性。

用法:

```
$ lsof [选项] [名称]
```

names在这里充当过滤器,没有选项 lsof 将显示属于所有活动进程的所有打开文件。

例子:

```
$ sudo lsof /var/运行/utmp
命令                    PID 用户 FD 类型设备大小/关闭节点名称 5376 636 /run/utmp
gdm-simpl 4040 root 10u REG 0,14

$ sudo lsof +d /var/日志
命令 PID 用户 FD 类型 设备大小/关闭 节点名称 rsyslogd 2039 root rsyslogd 2039 root rsyslogd 2039 root rsyslogd 2039 root
rsyslogd 2039 root rsyslogd 2039 root rsyslogd 2039 root 1w 寄存器          8,1 44399 262162 /var/log/syslog 8,1 180069 271272 /var/log/messages
                                                          2w 寄存器          8,1 54012 271269 /var/log/auth.log 8,1 232316 271268 /var/log/kern.log 8, 1
                                                          3w 寄存器          447350 271267 /var/log/daemon.log 8,1 8,1 8,1
                                                          6w 寄存器
                                                          7w 寄存器
                                                          8w 寄存器                68368 271271 /var/日志/调试
                                                          9w 寄存器                  7888 271270 /var/log/user.log 31030 262393 /var/log/
Xorg 4041 根目录                                          0r寄存器                Xorg.0.log
```

最后一个示例导致 lsof 搜索目录 /var/log 的所有打开实例以及它包含在其顶层的文件和目录。

自由的

free 命令显示系统物理和虚拟内存总量的当前概览,以及可用内存量、正在使用的内存和内核使用的缓冲区。

名为 shared 的第四列已过时,但现在用于显示用于 tmpfs 的内存（/proc/mem 信息中的 shmem）

用法:

```
$ 免费 [选项]
```

例子:

```
$免费-h
                 全部的      用过的      自由的      共享      缓冲器      缓存
记忆:            502M        489M        13M         50B       44M         290M
-/+ 缓冲区/缓存:             154M        347M
交换:            382M        3.9M        379M
```

最佳

top 命令提供了正在运行的系统的“动态实时视图”。

用法:

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
$ 顶部 [选项]
```

例子：

```
$ 顶部
顶部 - 03:01:24 向上 59 分钟,2 个用户,平均负载:0.15、0.19、0.16 任务:总共 117 个,2 个正在运行,115 个正在睡眠,0 个已停止,%Cpu(s):0.9 us,4.5 sy,0.1
ni , 94.3 id, 0.1 wa, 0.0 hi, 0.1 si, 0.0 st KiB Mem: 514332 total, 497828 used, 16504 free, 63132 buffers KiB Swap: 392188 total, 0 used, 392188
free, 270552 cached

  PID用户          PR NI VIRT RES SHR S %CPU %MEM                    时间+命令
4041根             20 0 106m 31m 9556 R 30.4 6.3 3:05.58 Xorg 20 0 527m 71m 36m S 18.2 14.3 2:04.42 gnome-shell
4262 用户
```

由于其交互模式,操作 top 时最重要的键是帮助键 h 或 ?和退出键 q。以下方案概述了最重要的功能键及其替代方案：

```
钥匙        等效键组合 alt + \ 或 alt + k alt + /
向上
下                                  或 alt + j 或 alt + h
剩下        替代 + <
正确的      替代 + >                或 alt + l（小写 L）
上一页      alt + Up alt +          或 alt + ctrl + k
PgDn        Down 或 alt + ctrl + j alt + Left 或 alt + ctrl + h
家
结尾        alt + 右或 alt + ctrl + l
```

顶层

用法：

```
$ htop [选项]
```

htop 命令类似于 top 命令,但允许您垂直和水平滚动,因此您可以看到系统上运行的所有进程,以及它们的完整命令行。无需输入进程的 PID 即可完成与进程相关的任务（killing、renicing）。

例子：

```
$ htop
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
 1  [|                          2.0%]   Tasks: 138, 445 thr; 1 running
 2  [|                          0.7%]   Load average: 0.46 0.19 0.10
 3  [                           0.0%]   Uptime: 05:07:28
 4  [                           0.0%]
 Mem[||||||||             2.29G/15.5G]
 Swp[                        0K/7.84G]

 USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 root       20   0  771M 48796 10560 S  0.7  0.3  0:00.81 /usr/libexec/fwupd/fwupd
 root       20   0  276M  141M  141M S  0.0  0.9  0:21.70 /usr/lib/systemd/systemd-jour
 root       20   0  683M 15696 11916 S  0.0  0.1  0:17.02 /usr/sbin/NetworkManager --nc
 root       20   0  683M 15696 11916 S  0.0  0.1  0:03.10 /usr/sbin/NetworkManager --nc
 root       20   0  264M  5704  4760 S  0.0  0.0  0:05.93 /usr/sbin/iio-sensor-proxy
 root       20   0 66408  8152  7208 S  0.0  0.1  0:01.20 /usr/sbin/wpa_supplicant -c /
 root       20   0  339M 11880 10564 S  0.0  0.1  0:01.36 /usr/bin/abrt-dump-journal-xc
 root       20   0  4416   652   588 S  0.0  0.0  0:00.83 /sbin/rngd -f
 root       20   0  370M 16888 15720 S  0.0  0.1  0:01.25 /usr/bin/abrt-dump-journal-oc
 root       20   0  771M 48796 10560 S  0.0  0.3  0:00.32 /usr/libexec/fwupd/fwupd
 root       20   0  526M 18980 15376 S  0.0  0.1  0:01.62 /usr/libexec/packagekitd
 root       20   0  209M 10616  7512 S  0.0  0.1  0:01.77 /usr/lib/systemd/systemd --sv
 root       20   0 47796  7364  4836 S  0.0  0.0  0:00.24 /usr/lib/systemd/systemd-udev
 root       16  -4 55552  3436  3068 S  0.0  0.0  0:00.00 /sbin/auditd -n
 root       16  -4 55552  3436  3068 S  0.0  0.0  0:00.02 /sbin/auditd -n
 root       12  -8 84544  1788  1652 S  0.0  0.0  0:00.02 /sbin/audispd
 root       12  -8 84544  1788  1652 S  0.0  0.0  0:00.03 /sbin/audispd
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

htop 窗口。

正常运行时间

uptime 命令显示系统已经运行了多长时间、有多少用户登录、过去 1、5 和 15 分钟的系统平均负载以及当前时间。它支持版本信息的 -V 选项。

用法：

```
$正常运行时间[选项]
```

例子：

```
$ uptime
03:03:12 up 1:00,2 个用户 ,平均负载:0.17、0.18、0.16
```

特区

sar 命令收集、报告或保存系统活动信息。

用法：

```
$ sar [选项] [间隔 [计数]]
```

例子：

```
$ sar
Linux 3.2.0-4-686-pae (debian) 05/07/2013 _i686_ (2 CPU)

02:02:34          重启
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
02:05:01        中央处理器   %用户    %nice %system %iowait 1.06 0.23          %偷     %闲置的
02:15:01        全部         0.15     0.00                                     0.00    98.56
02:25:01        全部         0.98     0.83     3.84     0.04                    0.00    94.31
02:35:01        全部         0.46     0.00     4.84     0.04                    0.00    94.66
02:45:01        全部         0.90     0.00     5.29     0.01                    0.00    93.80
02:55:01        全部         0.66     0.00     4.64     0.03                    0.00    94.67
03:05:02        全部         0.66     0.00     5.57     0.01                    0.00    93.76
平均:           全部         0.64     0.14     4.19     0.06                    0.00    94.98
```

如果没有选项,sar 将输出上面的统计信息。

使用 -d 选项 sar 将输出磁盘统计信息。

```
$ sar -d 06:45:01
                    开发者    tps rd_sec/s wr_sec/s avgrq-sz avgqu-sz 等待 svctm %util 6.89 41.59 2.63 0.95
06:55:01 dev8-0                227.01    59.67                0.02                1.38
07:05:01 dev8-0      2.08      17.73     17.78    17.06       0.00    2.19    0.94    0.20
07:15:01 dev8-0      1.50      12月16日   12.96    16.69       0.00    1.35    0.68    0.10
平均:dev8-0          3.49      85.63     30.14    33.15       0.01    2.36    1.19    0.42
```

-b 选项开关显示与 I/O 和传输速率统计相关的输出:

```
$ 特区-b
06:45:01          交易费      实时传输      wtps 面包/s bwrtn/s 2.38 227.01 59.67
06:55:01          率 6.89     速率 4.52
07:05:01          2.08        0.95          1.13       17.73      17.78
07:15:01          1.50        0.50          1.00       12月16日    12.96
平均:             3.49        1.99          1.50       85.63      30.14
```

与 sar 一起使用的一些最重要的选项是:

- -c 系统调用
- -p 和 -w 分页和交换活动
- -q 运行队列
- -r 随时间释放内存和交换

将系统症状与可能出现的问题进行匹配/关联

为了能够解决某个问题,首先必须能够区分正常的系统行为和异常的系统行为。

在上一节中,解释了一些非常具体的系统实用程序及其使用方法。在本节中,重点在于关联这些测量并能够检测异常,而异常又可能与异常系统行为相关联。与资源相关的问题有一个非常明显的共同点:

它们是一种或多种资源在某些情况下无法满足需求的结果。

这些资源可能与但不限于:CPU、物理或虚拟内存、存储、网络接口和连接,或者这些组件中的一个或多个组件之间的输入/输出。

估计吞吐量并识别系统中的瓶颈,包括网络

要确定某个问题是否与资源缺乏有关,必须首先正确地表述问题本身。然后,必须将这种制定的“偏离”行为与无故障操作系统产生的预期行为进行比较。

版权所有 2013-2017 Snow BV

Machine Translated by Google

如果可能,应该调查来自 sar 或其他工具的历史数据,并将其与 top、vmstat、netstat 和 iostat 等实时工具进行比较。

用户或报告工具报告的问题通常与可用性有关,或者某些资源无法以有序的方式提供或根本不可用。有序时尚在这里可能被解释为“在可接受的时间段内”。这类问题经常被报告,因为它们对用户来说非常明显,会影响他们的用户体验。

此类问题的示例可能是在特定时间段内无法访问或响应的某些文件或 (web) 应用程序。为了充分分析情况,手头有一个基线会很方便,它规定了“预期行为”应该是什么。这个基线应该在一个行为正常的系统上确定,并且应该考虑一定数量的阈值。

如果没有基线,测量本身应该能够帮助确定资源相关问题的根本原因。
如果上述资源之一达到其容量的 100%,则系统异常行为的原因应该很容易解释。然而,找到原因需要更多的努力。然而,前一章中介绍的实用程序在这里也应该能够提供帮助。

例子:

```
$ 顶部 $
vmstat
$ iostat
$ 网络统计
```

识别网络环境中的瓶颈需要几个步骤。最佳实践方法可概述如下:

- 创建网络地图
- 识别时间依赖行为
- 确定问题
- 识别异常行为
- 查明原因

预测未来的资源需求 (200.2)

考生应该能够预测未来的资源需求。

关键知识领域:

- 使用监控和测量工具监控 IT 基础设施的使用情况
- 预测配置的容量断点
- 观察容量使用增长率
- 绘制容量使用趋势图。
- 了解Icanga2、 Nagios、 collectd 、 MRTG和Cacti 等监控解决方案。

以下是使用的文件、术语和实用程序的部分列表:

- 诊断
- 预测增长
- 资源枯竭

版权所有 2013-2017 Snow BV

Machine Translated by Google

使用前两章介绍的工具和知识,应该可以诊断特定组件或进程的资源使用情况。提到的工具之一是sar,它能够记录较长时间的测量结果。能够使用记录的数据进行趋势分析是使用能够记录测量值的工具的优势之一。

监控 IT 基础设施

collectd 是一种可用于监控 IT 基础设施的工具。 collectd 是一个守护进程,它定期收集系统性能统计数据并提供以多种方式存储这些值的机制。它收集有关其运行的系统的统计信息并存储此信息。然后可以使用这些统计数据来查找当前的性能瓶颈（即性能分析)和预测未来的系统负载（即容量规划）。为了性能和可移植性,collectd 是用 C 语言编写的,允许它在没有脚本语言或 cron 守护进程的系统上运行,例如嵌入式系统。请注意,collectd 仅收集数据,要显示收集的数据,需要额外的工具。

Icanga2、 Nagios、 MRTG和Cacti等其他监控工具也可用于测量、收集和显示资源性能统计数据。使用监控工具可以帮助您识别与测量资源相关的可能瓶颈,但也可以帮助您预测未来的增长。

预测未来增长

通过分析和观察测量数据,随着时间的推移,应该可以预测资源需求的统计增长。我们在这里特意说统计增长,因为有很多情况会影响资源需求。例如,对传真机和电话线的需求因电子邮件的引入而受到影响。但基于数字或统计的增长估计也缺乏线性:当由于需求增加而扩张时,扩张通常包含各种服务。对于所有提供的服务,对这些服务的需求通常不会以相同的速度增长。这意味着不应仅仅分析测量数据,还应评估其相关性。

预测未来需求的步骤可以按照以下步骤进行:

- 决定衡量什么。
- 使用适当的工具来衡量和记录相关数据以实现您的目标。
- 分析测量结果,从最大的波动开始。
- 根据分析预测未来需求。

资源枯竭

当资源不能再以有序的方式交付给请求时,它就耗尽了。需求和交付不再对齐,资源的可用性将成为问题。资源耗尽会导致拒绝服务。

除了破坏资源的可用性外,配置为“无法打开”的设备可能会因资源耗尽而受骗。例如,当 ARP 表被淹没时,一些交换机回退到将所有流量转发到所有端口。

大多数时候,可以从收集的测量数据中提取耗尽的单个资源。这就是我们所说的瓶颈:系统中的一个点会缩小吞吐量并减慢下面的一切。清楚地了解这里的大局是很重要的。简单地解决一个特定的瓶颈只会转移问题,如果你增加一个组件的容量,另一个组件一旦达到容量限制就会成为限制因素。

因此,重要的是在分析过程中立即识别尽可能多的瓶颈。

版权所有 2013-2017 Snow BV

Machine Translated by Google

问题和解答

容量规划

1. 哪些工具可以用来显示块设备的输入/输出统计信息?

   iostat、vmstat 和 sar 是一些可用于输出块设备 I/O 的工具。 iostat 命令[2] vmstat 命令[3] sar 命令[10]

2. vmstat 输出的第一列“r”代表什么?

   当前分配给处理器运行队列的进程数。处理器运行队列[4]

3. 为了 /proc/net/ip_conntrack 的存在,必须在 post-2.4 内核上加载哪个内核模块?

   只有加载了 ipt_MASQUERADE 内核模块，/proc/net/ip_conntrack 才会存在。 IP 伪装[5]

4. netstat -r 和 route -e 之间的主要区别是什么,因为它们的输出相同?

   route -e 需要提升的权限,而 netstat -r 可以使用用户权限运行。 netstat --route 或 -r 选项[5]

5. 可以传递给 GNU ps 命令的三种不同类型的选项是什么?

   GNU 长选项、UNIX 和 BSD。 ps 命令[6]

6. 什么命令可以用来显示格式化为树的进程?

   pstree pstree 命令[7]

7. 什么命令可以列出所有打开的文件及其对应的进程?

   lsof 将列出所有打开的文件及其相应的进程。 lsof 命令[8]

8.资源相关的问题是什么导致的?

   资源相关问题是一种或多种资源在某些情况下无法满足需求的结果。资源枯竭[13]

9. 用户报告的问题通常与可用性或资源有关,尤其是他们的应用程序的性能。需要首先确定什么才能确定这些基于性能缺乏的声明是否确实有效?

   需要在行为正常的系统上确定基线。然后可以将该基线用作相同或相似系统的参考。创建基线[12]

10. 在调查可能与网络相关的瓶颈时,首先需要创建什么?

    在对网络问题进行故障排除之前,应创建网络图。识别网络瓶颈[12]

11. 什么命令可以同时显示物理内存和虚拟内存的数量,以及正在使用的内存量
    用过的?

    可以使用 top 和 htop 等几个实用程序,但 free 命令将显示有关正在使用的内存分配的详细信息。 （htop 是 Debian htop 软件包中的交互式进程查看器)自由命令[8]

版权所有 2013-2017 Snow BV

Machine Translated by Google

第1章

Linux 内核 (201)

本主题总权重为9分,包含以下三个目标:

目标 201.1:内核组件（权重:2） 考生应该能够利用特定硬件、硬件驱动程序、系统资源和要求所必需的内核组件。此目标包括实施不同类型的内核映像、了解稳定和长期的内核内核和补丁,以及使用内核模块。

目标 201.2:编译 Linux 内核（权重:3） 考生应该能够正确配置内核以根据需要包含或禁用 Linux 内核的特定功能。此目标包括根据需要编译和重新编译 Linux 内核、更新和记录新内核中的更改、创建 initrd 映像和安装新内核。

目标 201.3:内核运行时管理和故障排除（权重:4） 考生应该能够管理和/或查询 2.6.x、3.x 或 4.x 内核及其可加载模块。考生应该能够识别和纠正常见的启动和运行时问题。考生应该了解使用 udev 进行设备检测和管理。此目标包括对 udev 规则进行故障排除。

内核组件 (201.1)

考生应该能够利用特定硬件、硬件驱动程序、系统资源和要求所必需的内核组件。此目标包括实施不同类型的内核映像、识别稳定和开发内核和补丁,以及使用内核模块。

关键知识领域:

- 内核 2.6.x 文档
- 内核 3.x 文档
- 内核 4.x 文档

条款和实用程序

- /usr/src/linux
- /usr/src/linux/文档
- zImage
- bzImage
- xz 压缩

资源: wikiXZ、 tukaani;

版权所有 2013-2017 Snow BV

Machine Translated by Google

不同类型的内核映像

Linux 内核最初设计为单片内核。整体内核包含所有各种类型的受支持硬件的所有驱动程序,无论您的系统是否使用该硬件。随着支持硬件列表的增加,从未在任何给定系统上使用的代码量也在增加。因此引入了一个允许内核动态加载一些硬件驱动程序的系统。这些可加载的设备驱动程序被命名为“内核模块”。

尽管 Linux 内核可以加载和卸载模块,但它不符合微内核的条件。微内核的设计使得只有尽可能少的代码在管理员模式下运行 这从来都不是 Linux 内核的设计目标。 Linux 内核最好被描述为混合内核:它能够像微内核一样加载和卸载代码,但几乎完全以管理模式运行,就像单片内核一样。

仍然可以将 Linux 内核构建为一个整体内核。但很少这样做,因为更新设备驱动程序需要完全重新编译内核。然而,构建一个整体内核也有其优势:它可能占用空间更小,因为您可以只下载和构建您需要的部分,并且依赖关系更清晰。

当存储在磁盘上时,大多数内核映像都被压缩以节省空间。有两种类型的压缩内核类型:zImage 和 bzImage。

zImage 和 bzImage 文件有不同的布局和加载算法。 zImage 允许的最大内核大小为 512Kb,而 bzImage 没有此限制。因此,bzImage 内核是较大内核的首选图像类型。 zImage 将加载到低内存中,如果需要,bzImage 也可以加载到高内存中。

注意

zImage和bzImage都使用 gzip 压缩。 bzImage中的“bz”指的是“big zImage” 而不是“bzip”压缩算法。

内核和补丁的编号方案概述

多年来,Linux 内核使用的编号方案发生了数次变化:原始方案适用于 2.6.0 版及以下的所有内核,适用于 2.6.0 版至 3.0 版内核的方案,以前的方案适用于 3.0 版内核及更高版本,以及从 4.0 版开始的当前方案。在接下来的部分中,我们将讨论它们中的每一个。

计划到 2.6.0 内核

最初,内核版本号由三部分组成:主版本号、次版本号和补丁级别,全部由句点分隔。

当对内核进行重大更改时,将增加主要版本。

当进行重大更改和添加时,次要版本会增加。偶数编号的次要版本,例如 2.2、2.4,被认为是稳定版本,奇数编号的版本,例如 2.1、2.3、2.5 被认为是开发版本。

它们主要由内核开发人员和那些喜欢冒着不稳定风险的前沿功能的人使用。

内核版本号的最后一部分表示补丁级别。随着代码中的错误得到纠正（和/或添加了功能）,补丁级别增加了。只有当当前内核存在功能或安全问题时,才应将内核升级到更高的补丁级别。

从内核版本 2.6.0 到 3.0 的内核版本控制

2004年,2.6.0发布后,版本系统发生了变化,决定采用基于时间的发布周期。在接下来的七年里,内核保持在 2.6,第三个数字随着每个新版本的发布而增加（每两三个月发生一次）。添加了第四个数字以说明错误和安全修复。这种方案的一个例子是内核 2.6.32.71。不再使用奇偶编号系统。

版权所有 2013-2017 Snow BV

Machine Translated by Google

从 3.0 版到 4.0 版的内核版本控制

2011 年 5 月 29 日,Linus Torvalds 宣布发布内核版本 3.0.0 以纪念 Linux 诞生 20 周年。这再次改变了编号方案。它仍然是一个基于时间的发布系统,但第二个数字将表示版本号,第三个数字将是补丁号。对于测试版本,使用 -rc 名称。按照这个方案,3.2.84 将指代稳定的内核版本。另一方面,3.2-rc4 将指向 3.2 内核的第四个候选发布版。

从 4.0 开始的内核版本控制

2015 年 4 月,内核版本 4.0.0 发布。这次版本控制系统没有改变。在撰写本文时,内核版本 4.9.2 是可通过https://kernel.org 获得的最新稳定版本。然而,4.x 内核确实引入了一些新特性。执行“实时修补”的可能性是更值得注意的可能性之一。实时修补提供了无需重启系统即可安装内核补丁的可能性。这可以通过卸载和加载适当的内核模块来完成。每次发布新的 Linux 内核版本时,都会附有更新日志。这些变更日志文件包含有关此版本与以前版本相比发生了什么变化的详细信息。

XZ压缩

每个 Linux 发行版都带有一个由发行版开发人员配置和编译的内核。大多数 Linux 发行版还提供了通过某种包系统升级内核二进制文件的可能性。然而,也可以使用前面提到的网站kernel.org中的内核源代码为您的系统编译内核。这些内核源代码使用 tar 打包并使用 XZ 压缩方法进行压缩。 XZ 是 LZMA 和 LZMA2 的继承者。最近的 Linux 内核提供了对 XZ 的内置支持。根据所使用的 Linux 发行版,可能需要安装 xz 实用程序或等效程序包来解压缩 xz 压缩文件。在将可用内核之一的内核源下载为 tar.xz 存档后,可以使用以下命令行解压缩这些源文件:

```
$ tar xvf linux-4.10-rc3.tar.xz
```

注意
GNU tar至少需要 1.22 版才能使上述命令生效。

什么是内核模块

Linux 内核模块是由 C 编译器生成的目标文件（.ko 文件）,但没有链接到完整的可执行文件中。内核模块可以加载到内核中以在需要时添加功能。大多数模块与内核一起分发并与内核一起编译。每个内核版本都有自己的一组模块。

模块存储在 /lib/modules/kernel-version 下的目录层次结构中,其中 kernel-version 是 uname -r 报告的字符串或在 /proc/sys/kernel/osrelease 中找到的字符串,例如 2.6.5-15smp。如果安装了多个内核，/lib/modules 下可以使用多个模块层次结构。

/lib/modules/ kernel-version 目录下存在包含特定类型模块的子目录。这种分组对管理员来说很方便,而且还启用了 modprobe 命令中的重要功能。

典型模块类型:

块模块,用于一些特定于块的设备,例如 RAID 控制器或 IDE 磁带驱动器。

cdrom 非标准 CD-ROM 驱动器的设备驱动程序模块。

fs 文件系统的驱动程序,例如 MS-DOS（msdos.ko 模块）。

ipv4 包括与 IP 处理有关的模块化内核功能,例如 IP 伪装。

版权所有 2013-2017 Snow BV

Machine Translated by Google

misc 任何不适合其他子目录之一的东西都在这里结束。请注意,没有模块存储在顶部这棵树。

net 网络接口驱动程序模块。

scsi 包含 SCSI 控制器的驱动程序模块。

video 用于视频适配器的特殊驱动程序模块。

在模块操作命令的上下文中,模块目录也称为标记。

编译 Linux 内核 (201.2)

考生应该能够正确配置内核以根据需要包含或禁用 Linux 内核的特定功能。
此目标包括根据需要编译和重新编译 Linux 内核、更新和记录新内核中的更改、创建 initrd 映像和安装新内核。

关键知识领域

- /usr/src/linux/
- 内核 Makefile
- 内核 2.6.x、3.x 和 4.x make 目标
- 自定义当前内核配置
- 构建新内核和合适的内核模块
- 安装新内核和任何模块
- 确保引导管理器可以找到新内核和相关文件
- 模块配置文件
- 使用DKMS 编译内核模块
- dracut 意识

条款和实用程序

- mkinitrd
- mkinitramfs
- 制作
- make targets (all, config, xconfig, menuconfig, gconfig, oldconfig, mrproper, zImage, bzImage, modules, modules_install, rpm-pkg、binrpm-pkg、deb-pkg)
- gzip
- bzip2
- 模块工具
- /usr/src/linux/.config
- /lib/modules/kernel-version/
- depmod

资源： archDKMS、 debianDKMS;维基知识管理系统； githubDKMS;德拉库特；

版权所有 2013-2017 Snow BV

Machine Translated by Google

获取内核源

几乎所有内核版本的内核源代码都可以在The Linux Kernel Archives 中找到。

Linux Kernel Archive 中的文件名模仿内核的版本编号约定。例如:内核版本 3.0 和 4.0 的文件名格式为 linux-kernel-version.tar.xz 因此,linux-3.18.43.tar.xz 是版本“3.18.43”的内核存档。

3.0 和 4.0 内核使用的版本编号约定是 linux-ABCtar.xz,其中:

- A 表示内核版本。只有当代码和概念发生重大变化时,它才会发生变化。
- B 表示修订。
- C 是补丁号

请参阅有关内核版本控制的段落,以了解有关正在使用和已经使用的各种约定的更多信息。

存储和解压缩内核源代码的常见位置是 /usr/src。只要创建从新源目录到 /usr/src/linux 的符号链接,就可以使用其他位置

内核的源代码以 xz（.xz 扩展名)格式的压缩 tar 存档形式提供。使用 unxz 解压缩存档。生成的 tar 存档可以使用 tar 实用程序解压缩,例如:

```
# unxz linux-3.18.43.tar.xz
# tar xvf linux-3.18.43.tar
```

您还可以使用Joption一步解压缩和解压缩 tar：

```
# tar Jxvf linux-3.18.43.tar.xz
```

有关详细信息,请参阅 tar 和 xz 的联机帮助页。

清理内核

为确保您从干净的状态开始,您应该首先“清理”内核。当您将内核编译成对象时,make 实用程序会跟踪事情并且不会重新编译它认为之前已正确编译的任何代码。然而,在某些情况下,这可能会导致问题,尤其是当您更改内核配置时。因此,如果您重新配置内核,通常会“清理”源目录。

清洁可以在三个层面上进行:

make clean删除大部分生成的文件,但留下足够的文件来构建外部模块。

make mrproper删除当前配置和所有生成的文件。

make distclean删除编辑器备份文件、补丁遗留文件等。

在配置和构建内核之前运行 make mrproper通常是个好主意。

注意

请注意， make mrproper也会删除主配置文件。您可能需要先备份它以备将来参考。

版权所有 2013-2017 Snow BV

Machine Translated by Google

创建一个.config文件

首先,您需要配置内核。配置信息存储在.config 文件中。该文件中有超过 500 个选项,例如文件系统、SCSI 和网络支持。大多数选项允许您选择是将它们直接编译到内核中还是将它们编译为模块。一些选择意味着一组其他选择。例如,当您表示希望包括 SCSI 支持时,附加选项将可用于特定的 SCSI 驱动程序和功能。

一些内核支持选项必须编译为模块,一些只能编译为内核的永久部分,对于某些选项,您将能够选择任何一种可能性。

配置内核的方法有很多种,但无论您使用哪种方法,您选择的结果总是存储在内核配置文件/usr/src/linux/.config 中。它是一个纯文本文件,其中将所有选项列为 shell 变量。

示例 1.1 示例 .config 内容

```
#
# 自动生成的 make config: 不要编辑 # Linux 内核版本: 2.6.28

# 2010 年 2 月 6 日星期六 18:16:23
#
CONFIG_64BIT=y
# CONFIG_X86_32 未设置
CONFIG_X86_64=y
CONFIG_X86=y
CONFIG_ARCH_DEFCONFIG=    arch/x86/configs/x86_64_defconfig
CONFIG_GENERIC_TIME=y
CONFIG_GENERIC_CMOS_UPDATE=y
CONFIG_CLOCKSOURCE_WATCHDOG=y
CONFIG_GENERIC_CLOCKEVENTS=y
CONFIG_GENERIC_CLOCKEVENTS_BROADCAST=y
CONFIG_LOCKDEP_SUPPORT=y
CONFIG_STACKTRACE_SUPPORT=y
CONFIG_HAVE_LATENCYTOP_SUPPORT=y
CONFIG_FAST_CMPXCHG_LOCAL=y
配置_MMU=y
CONFIG_ZONE_DMA=y
CONFIG_GENERIC_ISA_DMA=y
CONFIG_GENERIC_IOMAP=y
CONFIG_GENERIC_BUG=y
CONFIG_GENERIC_HWEIGHT=y
 ...
```

要开始配置,请将当前工作目录更改为源代码树的顶部:

```
# cd /usr/src/linux
```

如前所述,有多种方法可以创建或修改 .config 文件。强烈建议手动编辑此文件。相反,您应该使用带有四个适当目标之一的 make 命令来配置您的内核。

这四个目标是:

- 配置
- 菜单配置
- xconfig|gconfig
- 旧配置

下面将更详细地解释这些目标。

版权所有 2013-2017 Snow BV

Machine Translated by Google

进行配置

运行 make config是最基本的方法。

它有明显的优点和缺点：

- 它不依赖于全屏显示功能。您可以在极慢的链接或非常有限的系统上使用它显示能力。
- 您必须解决所有与内核选项有关的可能问题。系统将按顺序无一例外地呈现它们。只有当您回答了所有问题后，您才可以保存配置文件。鉴于此，有数百种选择要经过，因此这种方法很乏味。因为你不能在各种问题中来回移动，所以如果你犯了错误，你将被迫重做所有的事情。

示例会话如下所示：

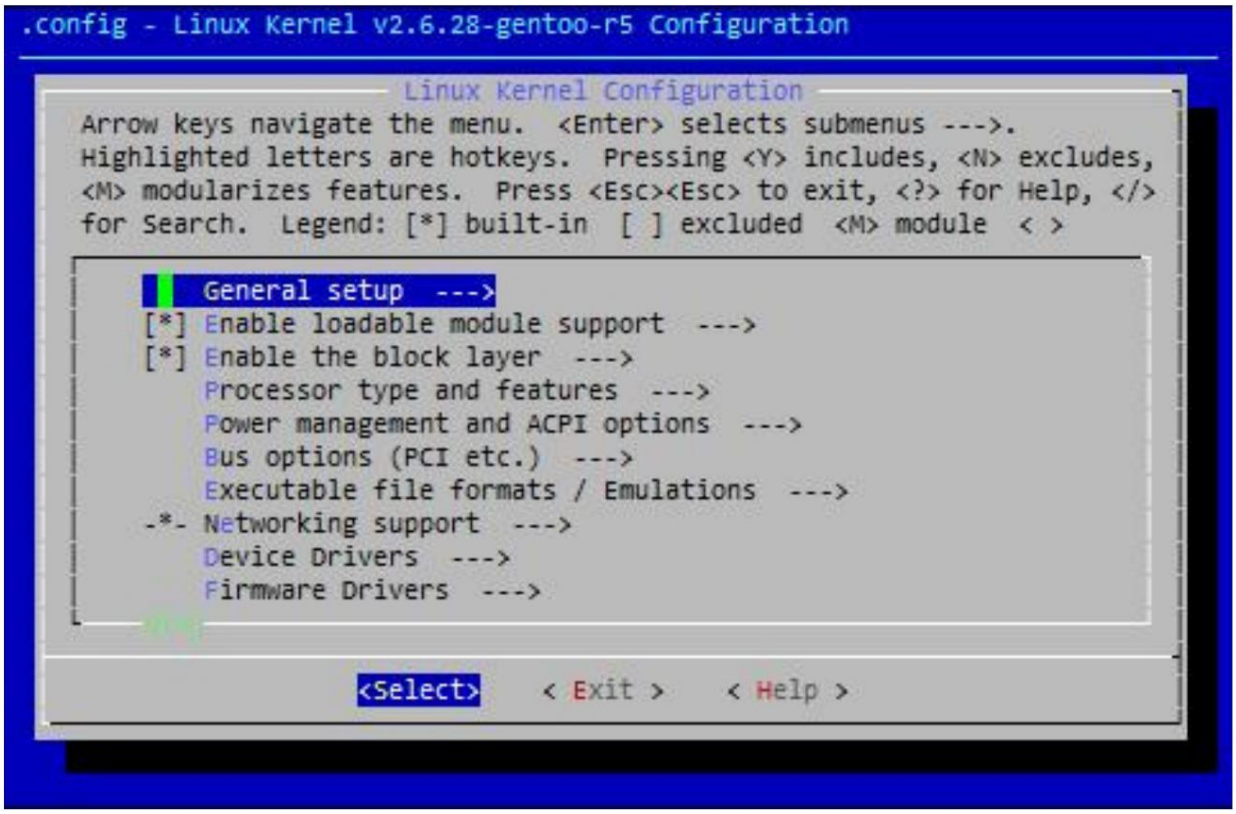
```
# make config HOSTCC
    脚本/basic/fixdep HOSTCC 脚本/basic/docproc
    HOSTCC 脚本/basic/hash HOSTCC 脚本/kconfig/
    conf.o

scripts/kconfig/conf.c: 在函数 conf_askvalue 中:scripts/kconfig/conf.c:104: 警告:忽略
  fgets 的返回值,\ 用属性 warn_unused_result 声明 scripts/kconfig/conf.c: 在函数 conf_choice : scripts/kconfig/
      conf.c:306: 警告:忽略 fgets 的返回值,\ 用属性 warn_unused_result 声明



    HOSTCC 脚本/kconfig/kxgettext.o HOSTCC 脚本/kconfig/
    zconf.tab.o
在 scripts/kconfig/zconf.tab.c:2486 包含的文件中:scripts/kconfig/confdata.c:在函数“conf_write”
中:scripts/kconfig/confdata.c:501:警告:忽略“fwrite”的返回值， \

      使用属性 warn_unused_result scripts/kconfig/confdata.c 声明:在函数
“conf_write_autoconf”中:scripts/kconfig/confdata.c:739: 警告:忽略“fwrite”的返回值,\

      用属性 warn_unused_result 声明
scripts/kconfig/confdata.c:740: 警告:忽略 fwrite 的返回值,\
      用属性 warn_unused_result 声明
在 scripts/kconfig/zconf.tab.c:2487 包含的文件中:scripts/kconfig/expr.c:在函数
“expr_print_file_helper”中:scripts/kconfig/expr.c:1090:警告:忽略“fwrite”的返回值， \

      用属性 warn_unused_result 声明
    HOSTLD 脚本/kconfig/conf 脚本/kconfig/conf
arch/x86/Kconfig
*
* Linux内核配置
*
*
* 一般设置
*
提示开发和/或不完整的代码/驱动程序（实验性)[Y/n/?]
```

制作菜单配置

make menuconfig方法更直观，可以作为 make config 的替代方法。它基于 ncurses 库创建文本模式窗口环境。您可以在选项之间来回切换。这些部分以类似菜单的结构布局，易于导航，您可以随时保存和退出。如果您喜欢较深的配色方案，请使用 make nconfig。

版权所有 2013-2017 Snow BV

Machine Translated by Google



make menuconfig菜单显示。

完成后,使用箭头键选择屏幕底部的退出选项。如果进行了任何更改,系统将提示您是否要保存新配置。您还可以选择使用文件系统中的其他名称和/或位置来保存配置。

注意如

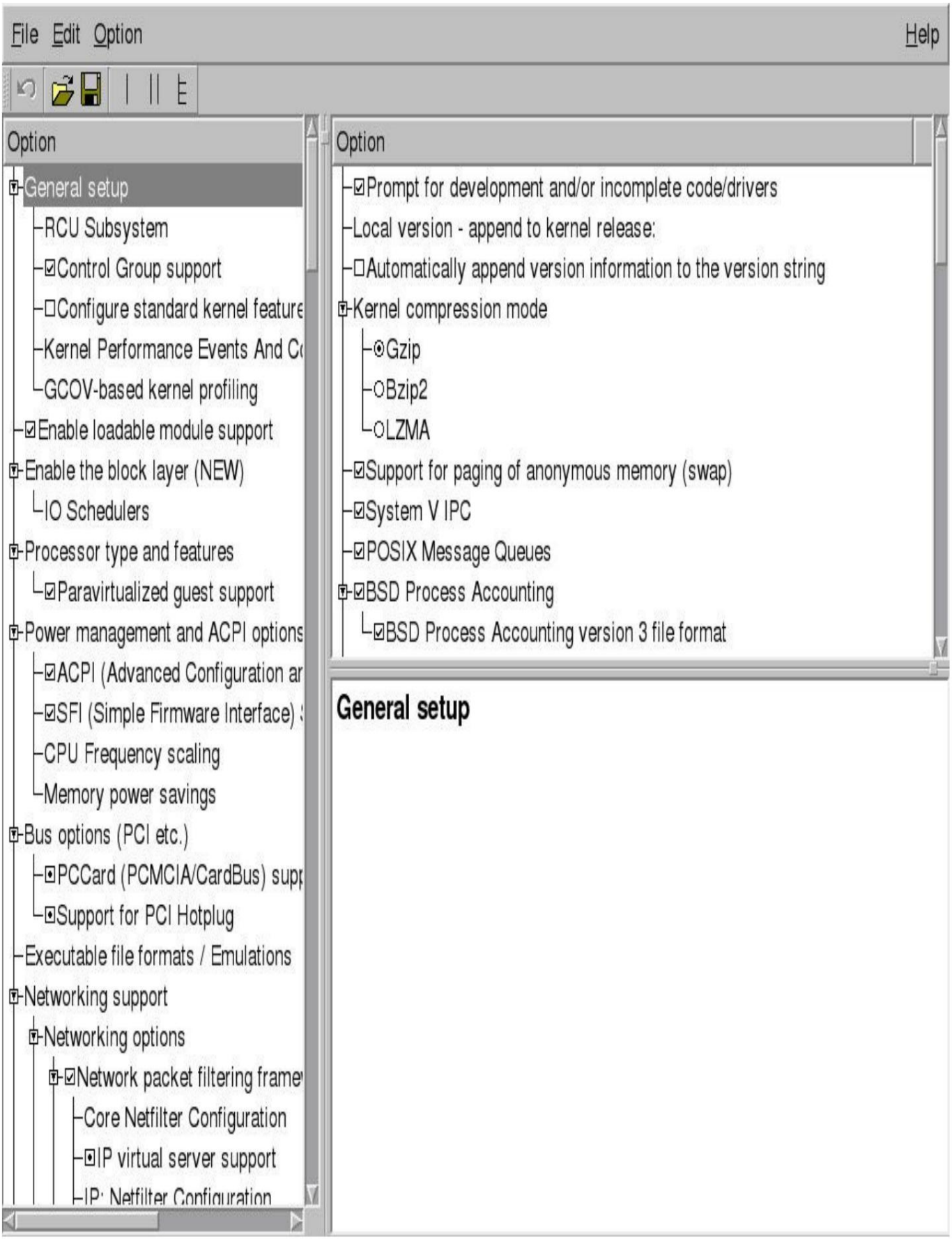
果选择其他名称或位置,则需要将.config文件移动到/usr/src/linux目录以编译内核。

制作xconfig和gconfig

make xconfig命令提供了一个用于配置内核的 GUI 菜单。它需要一个工作的 X 窗口系统和 QT 开发库才能工作。它将提供一个可以使用鼠标进行导航的菜单。使用 make gconfig来使用 Gnome 而不是 QT。这需要 GTK+ 2.x 开发库可用。首先,我们向您展示顶级 make xconfig窗口的外观:

版权所有 2013-2017 Snow BV

Machine Translated by Google

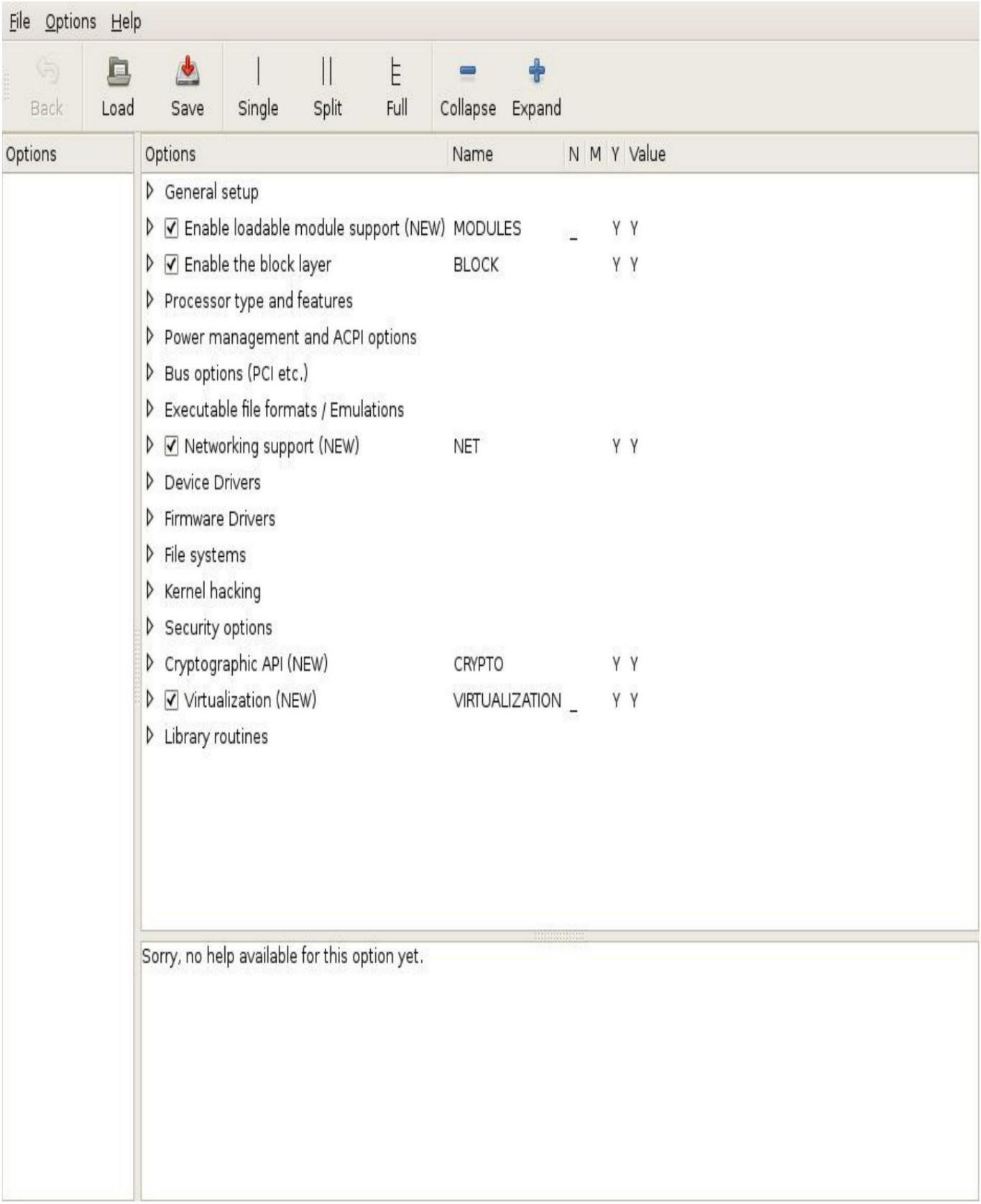


make xconfig顶层窗口。

版权所有 2013-2017 Snow BV

Machine Translated by Google

如前所述,命令 make gconfig的作用完全相同,但使用 GTK 而不是 QT:



make gconfig顶层窗口。

版权所有 2013-2017 Snow BV

Machine Translated by Google

制作旧配置

make oldconfig可用于保留您在早期内核构建期间选择的选项。

确保将作为早期构建结果的 .config 文件复制到 /usr/src/linux/。当您运行 make oldconfig 时,原始的 .config 文件将被移动到 .config.old 并创建一个新的 .config。系统将提示您提供在以前的配置文件中找不到的答案,例如当新内核有新选项时。

笔记

请务必在升级内核源代码之前备份.config ,因为发行版可能包含默认的.config文件,覆盖您的旧文件。

请注

意,make xconfig、 make gconfig和make menuconfig将自动使用旧的.config文件（如果可用)构建一个新文件,尽可能多地保留选项,同时使用默认值添加新选项。

编译内核

使用以下 make 命令序列来构建和安装内核和模块：

1.打扫卫生

2.制作zImage/bzImage

3.制作模块

4.制作模块安装

打扫干净

“clean”参数删除了以前内核构建中可能存在的旧输出文件。其中包括核心文件、系统映射文件等。

制作zImage/bzImage

zImage 和 bzImage 参数都有效地构建了内核。 zImage 与 bzImage [16]中解释了这两者之间的区别。

在编译过程之后,可以在 /usr/src/linux/arch/i386/boot 目录（在 i386 系统上)中找到内核映像。

制作模块

modules 参数构建模块;设备驱动程序和其他配置为模块的项目。

制作模块安装

modules_install 参数安装您刚刚在 /lib/modules/kernel-version 下编译的模块。如果不存在,将创建内核版本目录。

版权所有 2013-2017 Snow BV

Machine Translated by Google

安装新内核

编译新内核后,系统可以配置为引导它。

首先,您需要将新 bzImage 的副本放入引导目录（该目录应位于其自己的引导分区中）。为清楚起见,内核文件的名称应包含内核版本号,例如:vmlinuz-2.6.31:

```
# cp /usr/src/linux/arch/x86_64/boot/bzImage /boot/vmlinuz-2.6.31
```

这也确保您可以在 /boot 目录中拥有多个内核版本,例如,如果由于新内核出现问题而需要启动旧内核。

将内核文件移动到正确位置后,您需要配置引导管理器 (GRUB),以便它能够引导新内核。

有关 GRUB 的更多具体信息,请参阅GRUB。

初始 ram 磁盘(initrd)

假设你的引导盘上有引导加载程序、内核和适当的模块。鉴于内核模块的优点,您决定使用它们。但是,如果您还想将它们用于引导设备驱动程序,就会遇到问题。 GRUB 将加载内核,然后执行它。内核将尝试访问磁盘以获取模块。但是,由于它还没有加载正确的模块,它无法访问该磁盘并挂起。

一个完美的解决方案是构建一个内核,并将所需的磁盘驱动程序硬编码到其中。但是,如果您要维护大量不同的系统,则您要么需要为每种类型的系统提供个性化的配置和内核,要么不得不忍受臃肿的内核。为了避免所有这些问题,内核开发人员提出了一个解决方案:initrd RAM 磁盘。

RAM 磁盘是一块内存,内核将其视为磁盘。它可以像任何其他磁盘一样安装。内核默认支持 RAM 磁盘。 GRUB 和 LILO 也可以处理 RAM 磁盘。您可以指示他们从文件加载 RAM 磁盘,当内核启动时,RAM 磁盘随时可用。此类 RAM 磁盘通常用于保存脚本和模块以帮助引导过程。

按照惯例,保存初始 RAM 磁盘的映像的名称是 initrd。该名称是“初始 ram 磁盘”的缩写。

引导加载程序加载 initrd,它被内核挂载为它的根文件系统。一旦挂载为根文件系统,就可以从中运行程序并从中加载内核模块。在这一步之后,可以从不同的设备安装一个新的根文件系统。先前的根目录（来自 initrd)然后被移动到目录 /initrd 或被卸载。

有多种方法可以创建您自己的 initrd 文件。一种非常方便的方法,主要由 Red Hat（基于)发行版使用,是使用 mkinitrd 脚本。这是一个 shell 脚本,您可能想要检查它以了解它是如何工作的。在基于 Debian 的发行版上,可以使用名为 mkinitramfs 的实用程序来达到相同的目的。您也可以选择手动构建文件,请参阅下面的章节。

手动创建initrd

initrd 文件是包含最小根文件系统文件的压缩档案。这个根文件系统通常包含模块、脚本和一些额外的二进制文件,以允许内核正确地继续其引导。

如前所述,mkinitrd 脚本提供了一种构建 initrd 文件的便捷方式,但并非所有发行版都提供它。如果你想（或必须)手动构建一个,步骤是:创建一个根文件系统,用模块和文件填充它,从中创建一个 tar 或 cpio 存档,最后 gzip 它。

使用哪种类型的存档取决于发行版和内核版本。较旧的内核使用 tar,较新的内核使用 cpio。如果您不确定并且手边有随发行版一起提供的 initrd,您可以使用如下所示的命令序列来检查:

```
$ sudo zcat /boot/initrd-2.6.18-348.6.1.el5.img |文件 - /dev/stdin:ASCII cpio 存档（没有 CRC 的
SVR4）
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

上面的示例显示了使用 cpio 作为其归档工具的 CentOS 5 发行版的输出。

为了能够使用 initrd 映像,内核必须在支持 RAM 磁盘的情况下进行编译,并进行配置以便使用它。无论您在初始 RAM 磁盘上放置什么,它都应该与您将使用的内核和体系结构兼容。例如,您的引导内核应该能够识别映像中使用的文件系统类型,并且您包含的模块应该与引导内核版本相匹配。

下一步是实际创建 RAM 磁盘映像。首先在块设备上创建一个文件系统,然后根据需要将文件复制到该文件系统。适合用于此目的的块设备是:

1. RAM盘（速度快,分配物理内存）
2. 一个loopback设备（稍慢,分配磁盘空间）

在本示例的其余部分,我们将使用 RAM 磁盘方法,因此我们需要确保存在 RAM 磁盘设备节点（可能不止一个）:

```
# ls -la /dev/ram0
brw-rw---- 1 root disk 1, 0 Feb 13 00:18 /dev/ram0
```

笔记

系统默认可用的 RAM 磁盘数量是内核配置中的一个选项:CON FIG_BLK_DEV_RAM_COUNT。

接下来需要创建一个适当大小的空文件系统:

```
# mke2fs -m0 /dev/ram0 300
```

笔记

如果空间很紧,您可能希望使用空间效率更高的文件系统,例如 Minix FS。请记住,启动内核需要对您选择的任何文件系统提供内置支持。

创建文件系统后,您需要将其挂载到适当的目录:

```
# mount -t ext2 /dev/ram0 /mnt
```

现在需要创建控制台设备的存根。这将是 initrd 处于活动状态时将使用的设备节点。

```
# mkdir /mnt/dev # mknod /
mnt/dev/tty1 c 4 1
```

接下来,将您认为需要的所有文件复制到图像中;模块、脚本、二进制文件,没关系。请参阅/、/boot 和 /lib/modules [37]的目录,以获取至少需要的目录和文件的示例。要复制的最重要的文件之一是 /linuxrc。每当内核设置为使用 initrd 映像时,它将搜索文件 /linuxrc 文件并执行它。它可以是脚本或已编译的二进制文件。因此,挂载图像文件后会发生什么完全在您的控制之下。在本例中,我们将使 /linuxrc 成为 /bin/sh 的链接。确保 /linuxrc 被授予执行权限。

```
# ln -s /bin/sh /mnt/linuxrc
```

完成复制文件并确保 /linuxrc 具有正确的属性后,您可以卸载 RAM 磁盘映像:

```
# 卸载 /dev/ram0
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

然后可以将 RAM 磁盘映像复制到文件中：

```
# dd if=/dev/ram0 bs=1k count=300 of=/boot/initrd
```

最后,如果您不再需要 RAM 磁盘并希望回收内存,请释放 RAM 磁盘：

```
# freeramdisk /dev/ram0
```

要测试新创建的 initrd,请向 GRUB 菜单文件添加一个新部分,它引用您刚刚创建的 initrd 映像：

```
title=initrd 测试入口 root (hd0,0) kernel /
vmlinuz-2.6.28

初始化文件/初始化文件
```

如果您已执行上述步骤并使用引导加载程序菜单中的此测试条目重新启动,系统将继续启动。几秒钟后,您应该会发现自己处于命令提示符下,因为 /linuxrc 指的是 /bin/sh,一个 shell。

当然,真正的 initrd 文件将包含一个更复杂的 /linuxrc 引导文件,它加载模块、挂载真正的根文件系统等。

修补内核

注意
本节提供有关不再属于 LPIC-2 目标的主题的信息。它得到维护是因为它仍然包含有效且有价值的信息。

在旧版本的 LPIC-2 目标中,候选人被假定能够正确地修补内核的源代码以添加对新硬件的支持。目标包括能够从已打补丁的内核中删除内核补丁。

关键文件、术语和实用程序包括：

- 内核 Makefile
- 修补
- xz

补丁文件包含一个文件的两个版本之间的差异列表。标准命令 diff 能够生成这样的列表。命令 patch 可用于应用补丁文件的内容,将文件从旧版本更新到新版本。

给内核打补丁非常简单：

1. 将补丁文件放在/usr/src 目录下。
2. 将目录更改为/usr/src。
3. 使用unxz解压补丁文件
4. 使用补丁实用程序将补丁文件应用于内核源代码：

   ```
   # patch -p1 <补丁文件
   ```

5. 检查故障。
6. 构建内核。

版权所有 2013-2017 Snow BV

Machine Translated by Google

如果补丁实用程序无法应用补丁的一部分,它会将那部分放入拒绝文件中。拒绝文件的名称是输出文件的名称加上 .rej 后缀,或者如果添加 .rej 会生成太长的文件名,则为 #。如果仅添加 # 会导致文件名太长,则文件名的最后一个字符将替换为 #。

补丁实用程序的常用选项:

-pnumber, --strip=number从补丁文件中找到的每个文件名中去除包含数字前导斜杠的最小前缀。一个或多个相邻斜线的序列被算作一个斜线。这控制了如何处理在补丁文件中找到的文件名,以防您将文件保存在与发出补丁的人不同的目录中。例如,假设补丁文件中的文件名为 /u/howard/src/blurfl/blurfl.c,那么使用 -p0会给出未修改的整个文件名,而使用 -p1会给出 u/howard/src/blurfl/blurfl 。C。

-s, --silent, --quiet静默工作（抑制输出）,除非发生错误。

-E, --remove-empty-files删除应用补丁后为空的输出文件。通常这个选项是不必要的,因为 patch 可以检查文件头上的时间戳来确定文件在打补丁后是否应该存在。但是,如果输入不是上下文差异或者 patch 符合 POSIX 规范,除非给出此选项,否则 patch 不会删除空的修补文件。当 patch 删除文件时,它还会尝试删除任何空的祖先目录。

-R, --reverse假设这个补丁是用颠倒的旧文件和新文件创建的,所以你基本上是将补丁应用到已经包含补丁文件中修改的文件。该补丁将尝试在应用之前交换每个大块,并且拒绝将以交换格式出现。 -R选项不适用于 ed diff 脚本,因为信息太少无法重建反向操作。如果补丁的第一个大块失败,补丁会反转大块以查看是否可以那样应用。如果可以,系统会询问您是否要设置-R选项。如果不能,补丁将继续正常应用。

注意如

果它是一个正常的差异并且如果第一个命令是追加（即它应该是删除）,则此方法无法检测到反向补丁,因为追加总是成功的。这是因为空上下文在任何地方都匹配。幸运的是,大多数补丁添加或更改行而不是删除它们,因此大多数反向正常差异以删除开始,删除失败,触发启发式。

有关详细信息,请参阅 diff 命令和 patch 命令的联机帮助页。

从生产内核中删除内核补丁

通过从生产内核源代码树中删除内核补丁并编译新内核,可以将内核补丁从生产内核中删除。在上一个主题中,我们了解到要从文件中删除补丁,您需要再次应用它,或者使用-R参数运行补丁:

```
# patch -p1<patch-2.6.28 补丁文件 linux/
Documentation/Configure.help 检测到反向（或以前应用的)补丁 !假设-R? [n] 是
```

知识管理系统

$ DKMS（动态内核模块支持)由 Dell 于 2003 年开发。DKMS 的创建是为了解决由内核和内核模块之间的依赖关系引起的软件问题。作为运行（除其他外）Linux 操作系统的计算机系统供应商,戴尔为客户提供软件支持。当客户升级 Linux 内核时,内核模块也必须升级。每当 Dell 发布更新版本的内核模块以支持硬件时,这些模块都必须与正在使用的 Linux 内核相匹配。

DKMS 是一个框架,能够为系统上可用的每个内核版本自动编译和/或安装内核模块。 DKMS 通过将内核模块文件或源代码与实际的内核源代码树分开来实现此功能。这样,内核和内核模块都可以相互独立地升级。主要 Linux 发行版提供

版权所有 2013-2017 Snow BV

Machine Translated by Google

DKMS 框架通过他们的包系统。当运行 DKMS 的系统上的包管理器软件升级内核时,一个挂钩将负责决定是否需要为新内核编译和/或安装任何内核模块。相反,新的内核模块可以由 DKMS 编译和/或安装,而不需要内核版本。

DKMS 确实有一些要求才能正常运行。软件依赖项取决于所使用的 Linux 发行版。但是所有发行版的一个统一要求是内核头文件的必要性。可以使用以下包管理器命令安装正在运行的内核版本的标头:

在基于 Red Hat 的 Linux 发行版上:

```
$ yum 安装内核开发
```

在基于 Debian 的 Linux 发行版上:

```
$ apt-get 安装 linux-headers-$(uname -r)
```

由于 DKMS 在主要 Linux 发行版中的采用,许多通过包管理器可用的内核模块(也)可作为 DKMS 模块使用。在基于 Debian 的 Linux 系统上,这些 DKMS 内核模块可以通过它们的命名约定来识别。这些文件的包名称以 -dpkg 结尾。例如:oss4-dkms。安装这些软件包后,内核模块源文件与 dkms.conf 配置文件一起放在相应的 /usr/src/module-version 目录中。每当内核或内核模块更改触发 DKMS 系统时,将检查 /etc/dkms/framework.conf 中 source_tree 变量指定的目录是否存在包含 dkms.conf 文件的子目录。这些 dkms.conf 文件的内容决定接下来会发生什么。以下示例来自基于 Debian 的 Linux 系统,应该澄清这个解释:

```
$ sudo apt-get install flashcache-dkms $ ls /usr/src/
flashcache-3.1.1+git20140801 linux-headers-3.16.0-4-amd64 linux-
headers-3.16.0-4-common ←    linux-kbuild -3.16

$ ls /usr/src/flashcache-3.1.1+git20140801/
dkms配置文件                闪存缓存.h                flashcache_ioctl.h flashcache_procfs.c                ←
        flashcache_subr.c
flashcache_conf.c flashcache_ioctl.c flashcache_main.c flashcache_reclaim.c 生成文件 $ cat /usr/src/flashcache-3.1.1+git20140801/dkms.conf BUILT_MODULE_NAME=    flashcache


DEST_MODULE_LOCATION=    /内核/驱动程序/块
PACKAGE_NAME=    闪存缓存
PACKAGE_VERSION=    3.1.1+git20140801
自动安装=“是”
REMAKE_INITRD=    是
MAKE=    COMMIT_REV=3.1.1+git20140801 KERNEL_TREE=$kernel_source_dir 制作模块
```

BUILT_MODULE_NAME 确定编译模块的名称。如果 DKMS 模块包包含多个模块,则该指令是强制性的。 DEST_MODULE_LOCATION 确定编译模块的位置。该指令的值应始终以“/kernel”开头,然后重定向到 /lib/modules/kernelversion/kernel。

此值是强制性的,但以下使用特定于发行版目录的 Linux 发行版除外:Fedora Core 6 及更高版本、RHEL 5 及更高版本、Novell SuSE Linux ES 10 及更高版本以及 Ubuntu。 PACKAGE_NAME 确定与整个模块包关联的名称。该指令是强制性的。 PACKAGE_VERSION 确定与整个模块包关联的版本并且是必需的。 AUTOINSTALL 是一个布尔值,用于确定 dkms_autoinstaller 服务是否会尝试为系统引导到的每个内核安装此模块。 REMAKE_INITRD 确定安装此模块后是否应再次生成 initrd 映像。该指令的值默认为“no”。配置值时,要知道第一个字符之后的所有字符都将被丢弃。第一个字符只有在它是“y”或“Y”时才会被解释。 MAKE 是将其值存储到数组中的众多指令之一。 MAKE 值决定构建选项。未定义时,DKMS 将尝试使用通用 MAKE 命令构建模块。

在基于 Debian 的 Linux 发行版上,目录 /usr/share/doc/dkms/examples 包含示例配置文件。
在基于 Red Hat 的 Linux 系统上,可以在 /usr/share/doc/dkms 目录中找到示例文件。

在 DKMS 构建模块后,它是可扩展框架的一部分。 DKMS 提供了多个命令以在模块上发出。涵盖所有这些命令超出了本书的范围。但除了 man dkms

dkms 命令

版权所有 2013-2017 Snow BV

Machine Translated by Google

应该很熟悉。 dkms 命令可以在源代码树中添加或删除模块。一旦成为源代码树的一部分,就可以构建模块。构建之后,可以使用安装选项将模块安装到它构建的内核上。卸载恢复这个过程。状态选项将有关已添加模块的信息打印到标准输出。

上面的示例使用了包管理器中的一个模块。 DKMS 还能够接受包含二进制模块、模块源或两者的存档格式。以这种方式将模块添加到 DKMS 时,重要的是存档还包含有效的 dkms.conf 文件。使用 dkms mktarball 命令,可以基于从当前系统中提取的模块来创建这样的存档。然后可以使用 dkms ldtarball 命令将该 tarball 存档导入到源代码树中。

德拉库特

就像 DKMS 可以配置为事件驱动工具一样,dracut 可以以类似的方式运行。 dracut 无需编译内核模块,而是可以在需要时负责生成新的 initramfs 映像。

内核运行时管理和故障排除 (201.3)

考生应该能够管理和/或查询 2.6.x、3.x 或 4.x 内核及其可加载模块。考生应该能够识别和纠正常见的启动和运行时问题。考生应该了解使用 udev 进行设备检测和管理。此目标包括对 udev 规则进行故障排除。

关键知识领域:

- 使用命令行实用程序获取有关当前运行的内核和内核模块的信息。
- 手动加载和卸载内核模块。
- 确定何时可以卸载模块。
- 确定模块接受哪些参数。
- 配置系统以使用文件名以外的名称加载模块。
- /proc 文件系统
- /, /boot 的内容 , 和 /lib/modules
- 用于分析有关可用硬件信息的工具和实用程序
- udev 规则

以下是使用的文件、术语和实用程序的部分列表:

- /lib/modules/kernel-version/modules.dep
- /etc 中的模块配置文件
- /proc/sys/内核/
- /sbin/depmod
- /sbin/rmmod
- /sbin/modinfo
- /bin/dmesg
- /sbin/lspci
- /usr/bin/lsdev

版权所有 2013-2017 Snow BV

Machine Translated by Google

- /sbin/lsmod
- /sbin/modprobe
- /sbin/insmod
- /bin/uname
- /usr/bin/lsusb
- /etc/sysctl.conf, /etc/sysctl.d/
- /sbin/sysctl
- udevmonitor
- udevadm 监视器
- /etc/udev

自定义、构建和安装自定义内核和内核模块

在什么是内核模块段落中,您了解了内核模块是什么。有各种与这些模块关联的实用程序和配置文件。本段将解释如何使用这些来管理正在运行的内核及其模块。

操作模块

lsmod

对于每个加载的内核模块,显示其名称、大小、使用次数和其他引用模块的列表。此命令产生与 /proc/modules 中可用的信息相同的信息。例如,在特定笔记本电脑上,命令 /sbin/lsmod 报告:

```
模块                    serial_cb 1120 1 tulip_cb
31968 2 cb_enabler 2512 4 [serial_cb tulip_cb] ds
使用的大小

                        6384 2 [cb_enabler] 22384 2
i82365
pcmcia_core 50080 0 [cb_enabler ds i82365]
```

格式为名称、大小、使用次数、引用模块列表。如果模块通过“can_unload”例程控制自己的卸载,则 lsmod 显示的用户计数始终为 -1,与实际使用计数无关。

insmod

将模块插入正在运行的内核中。模块自动定位并插入。您必须以超级用户身份登录才能插入模块。通常建议改用 modprobe,因为 insmod 只返回非常一般的错误代码,而 modprobe 对错误更具体。此外,您需要将完整的文件名传递给 insmod,而 modprobe 将仅使用模块名称。

,

常用选项:

-s 将结果写入系统日志而不是终端。

-v 设置详细模式。

版权所有 2013-2017 Snow BV

Machine Translated by Google

示例 1.2 insmod 假设内

核编译时带有对特定 scsi 卡的模块化支持。要验证此特定模块是否存在,在本例中为 sym53c8xx.ko,请在/lib/modules/kernel-version/kernel/drivers/scsi/ 目录中查找文件 sym53c8xx.ko:

```
# insmod sym53c8xx.ko # echo $?

0
```

模块可以依赖于其他模块。在此示例中,之前已加载所有先决条件模块,因此该特定模块已成功加载。但是,insmod 不知道先决条件,它只是加载您告诉它加载的任何内容。
如果缺少先决条件,这可能会出错:

```
# insmod snd-ens1371.ko insmod: error
inserting    snd-ens1371.ko   : -1 Unknown symbol in module # echo $? 1个
```

该消息表明模块需要当前内核中缺少的函数或变量。这通常是由于没有将必备模块加载或编译到内核中造成的。

rmmod

除非一个模块正在使用或被另一个模块引用,否则该模块将从运行的内核中删除。您必须以超级用户身份登录才能删除模块。

例 1.3 rmmod

```
# rmmod snd_ac97_codec
错误:snd_ens1371 正在使用模块 snd_ac97_codec # echo $?

1个
```

在此示例中,模块无法卸载,因为它正在使用中,在本例中由 snd_ens1371 模块使用。解决办法是先把snd_ens1371模块去掉:

```
# rmmod snd_ens1371 # rmmod
snd_ac97_codec # echo $? 0
```

发出命令 lsmod 以验证模块确实已从正在运行的内核中删除。

调制探头

与 insmod 一样,modprobe 用于插入模块。但是,modprobe 能够加载单个模块、模块及其先决条件,或存储在特定目录中的所有模块。 modprobe 命令与 -r 选项结合使用时也可以删除模块。它的错误消息也比 insmod 更具体。

可以使用可选的symbol=value参数插入模块,例如irq=5或dma=3 。这些参数可以在命令行中指定,也可以在模块配置文件中指定,后面会解释。如果模块依赖于其他模块,则这些模块将首先加载。 modprobe 命令通过读取模块目录层次结构顶部的文件 modules.dep 来确定模块之间的先决条件关系,例如 /lib/modules/2.6。 31/modules.dep。您必须以超级用户身份登录才能插入模块。

常用选项:

-a 加载所有模块。当与 -t 标签一起使用时,“all”仅限于标签目录中的模块。此操作通过对单一类型硬件(例如网络适配器)的连续模块插入尝试来探测硬件。这可能是必要的,例如,探测不止一种网络接口。

-c 显示完整的模块配置,包括 /etc/modules.conf(或 /etc/conf.modules,取决于您的模块实用程序版本)中的默认值和指令。-c 选项不与任何其他选项一起使用。

版权所有 2013-2017 Snow BV

Machine Translated by Google

-l 列出模块。与 -t 标签一起使用时,仅列出目录标签中的模块。

-r 删除模块,类似于 rmmod。可以指定多个模块。

-s 在 syslog 中而不是在终端上显示结果。

-t tag尝试加载在目录标签中找到的多个模块,直到一个模块成功或标签中的所有模块都用完。此操作通过对单一类型硬件（例如网络适配器)的连续模块插入尝试来探测硬件。

-v设置详细模式。

示例 1.4 modprobe 使用

modprobe 加载声音模块。

```
# modprobe snd_ens1371 # echo $?

0
# lsmod
模块                    使用尺寸
snd_ens1371            20704 0
snd_rawmidi            22448 1 snd_ens1371
snd_seq_device          7436 1 snd_rawmidi
snd_ac97_codec        112280 1 snd_ens1371
ac97_总线                1992 1 snd_ac97_codec
snd_pcm                72016 2 snd_ens1371,snd_ac97_codec
snd_timer              21256 1 snd_pcm 62392 6
snd                    snd_ens1371,snd_rawmidi,snd_seq_device,snd_ac97_codec,←。
        snd_pcm,snd_timer
配乐                    7952 1 秒
snd_page_alloc          9528 1 snd_pcm
```

snd_ens1371 的所有先决条件模块都已加载。要删除 snd_ac97_codec 和 snd_ens1371 模块,请使用 modprobe -r snd_ac97_codec。

模块配置在文件 /etc/modules.conf 中处理。如果此文件不存在,modprobe 实用程序将尝试读取 /etc/conf.modules。后一个文件是历史名称,已弃用。它应该替换为 /etc/modules.conf。

注意在

某些系统上,此文件称为/etc/modprobe.conf或者在名为 /etc/modprobe.d 的目录中有配置文件。

有关模块配置的更多信息,请参阅配置模块部分。

修改信息

从模块目标文件中显示有关模块的信息。有些模块根本不包含任何信息,有些只有一行简短的描述,有些则有相当详细的描述信息。

手册页中的选项:

-a, --author 显示模块的作者。

-d, --description 显示模块的描述。

-n, --filename 显示模块的文件名。

-fformat_string, --format format_string让我们指定一个任意的格式字符串,它可以从包含模块信息的 module_file 中的 ELF 部分中提取值。替换由一个百分号和后跟花括号中的标签名称组成。始终支持 %{filename} 的标记名称,即使模块没有 modinfo 部分。

版权所有 2013-2017 Snow BV

Machine Translated by Google

-p, --parameters显示模块可能支持的类型化参数。

-h, --help显示一个小的使用屏幕。

-V, --version显示 modinfo 的版本。

示例 1.5 修改信息

可以从 snd-ens1371 模块检索哪些信息：

```
# modinfo snd_ens1371 文件名：/lib/
modules/2.6.31/kernel/sound/pci/snd-ens1371.ko Ensoniq/Creative AudioPCI ES1371+
描述： 许可证：
                    GPL
作者：              Jaroslav Kysela <perex@perex.cz>,托马斯·赛勒 <sailer@ife.ee.ethz.ch> *sd*bc*sc*i* snd-pcm,snd,snd-rawmidi,snd-ac97-codec 2.6.31-
别名：              gentoo-r10 SMP mod_unload modversions 索引:Ensoniq AudioPCI 声卡的索引值。 （整数数组)id:Ensoniq AudioPCI 声卡的 ID 字符串。 （charp
别名：              数组)启用:启用 Ensoniq AudioPCI 声卡。 （bool 数组)spdif:S/PDIF 输出（-1 = 无,0 = 自动,1 = 强制）。 （int 数组)lineio:Line In 到 Rear Out（0 =
别名：              自动,1 = 强制）。 （整数数组）
取决于：
vermagic：
参数：
参数：
参数：
参数：
参数：
```

配置模块

有时您可能需要控制模块使用的资源分配,例如硬件中断或直接内存访问 (DMA) 通道。其他情况可能需要特殊程序来准备模块插入或移除,或在模块插入或移除之后进行清理。这种对模块的特殊控制在文件 /etc/modules.conf 中配置。

IN/E T C/M MODULE SC ON F的常用指令：

keep keep 指令,当在任何路径指令之前找到时,会导致保留默认路径并将其添加到任何路径
指定的。

depfile=full_path 该指令覆盖模块依赖文件的默认位置,modules.dep 它将
在下一节中描述。

path=path该指令指定一个额外的目录来搜索模块。

options modulename module-specific-options 可以使用 modules.conf 中的选项配置行或在 modprobe 命令行上指定模块的选项。命令行覆盖文件中的配置。
modulename 是不带 .ko 扩展名的单个模块文件的名称。特定于模块的选项指定为名称=值对,其中名称由模块理解并使用 modinfo -p 报告。

alias aliasname 结果 别名可用于将通用名称与特定模块相关联,例如：

```
别名/dev/ppp ppp_generic 别名 char-major-108
ppp_generic 别名 tty-ldisc-3 ppp_async 别名 tty-ldisc-14
ppp_synctty 别名 ppp-compress-21 bsd_comp 别名 ppp-
compress-24 ppp_deflate 别名 ppp-compress-26 ppp_deflate
```

预安装模块命令 该指令导致在插入模块之前执行指定的 shell 命令。例如,PCMCIA 服务需要在安装 pcmcia_core 模块之前启动：

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
预安装 pcmcia_core /etc/init.d/pcmcia start
```

install module command 该指令允许特定的 shell 命令覆盖默认的模块插入命令。

安装后模块命令 该指令导致在插入模块后执行指定的 shell 命令
乌尔。

pre-remove module command 该指令导致在删除模块之前执行指定的 shell 命令。

删除模块命令 该指令允许特定的 shell 命令覆盖默认的模块删除命令。

post-remove module command 该指令导致在删除模块后执行指定的 shell 命令。

有关模块配置文件的更多详细信息,请参阅 man modules.conf。

空行和以 # 开头的行在 modules.conf 中被忽略。

模块依赖文件

命令 modprobe 可以确定模块依赖关系并自动安装必备模块。为此,mod probe 扫描/lib/modules/kernel-version/modules.dep的第一列以查找要安装的模块。 modules.dep 中的行采用以下形式:

```
module_name.(k)o: dependency1 dependency2 ...
```

在下面找到散热和处理器模块（属于 ACPI 层的一部分)的示例:

```
/lib/modules/2.6.31/kernel/drivers/acpi/thermal.ko:\
        /lib/modules/2.6.31/kernel/drivers/thermal/thermal_sys.ko
/lib/modules/2.6.31/kernel/drivers/acpi/processor.ko:\
        /lib/modules/2.6.31/kernel/drivers/thermal/thermal_sys.ko
```

在这种情况下,处理器和散热模块都依赖于 thermal_sys 模块。

系统上可用的所有模块都应列在 modules.dep 文件中。它们以完整路径和文件名列出,因此包括它们的 .(k)o 扩展名。那些不依赖于其他模块的模块在没有依赖项的情况下列出。
列出的依赖项首先由 modprobe 插入内核,成功插入后,可以加载主题模块本身。

modules.dep 文件必须保持最新以确保 modprobe 的正确操作。如果模块依赖关系在没有对 modules.dep 进行相应修改的情况下发生变化,则 modprobe 将失败,因为依赖关系将被遗漏。因此,每次系统启动时都需要（重新)创建 modules.dep。大多数发行版将通过在系统启动时自动执行 depmod -a 来执行此操作。

depmod -a 过程在模块依赖性发生任何变化后也是必需的。

/lib/modules/kernel-version/modules.dep文件包含模块依赖项列表。它由 depmod 命令生成,该命令读取/lib/modules/kernel-version下的每个模块以确定模块需要哪些符号以及导出哪些符号。默认情况下,该列表写入文件 modules.dep。 modprobe 命令依次使用该文件来确定模块加载到内核或从内核卸载的顺序。

kmod 与 kerneld

kmod 和 kerneld 都提供内核模块的动态加载。当内核第一次需要时加载模块。有关模块的描述,请参阅什么是内核模块[17]。

kerneld 是一个守护进程,kmod 是内核本身中的一个线程。与 kerneld 的通信是通过 System V IPC 完成的。 kmod 直接从内核运行,不使用 System V IPC,因此使其成为可选模块。

从 Linux 内核 2.2.x 开始,kmod 取代了 kerneld。

版权所有 2013-2017 Snow BV

Machine Translated by Google

kerneld 和 kmod 都有助于内核模块的动态加载。两者都使用 modprobe 来管理模块的依赖性和动态加载。

使用 modprobe 或 insmod 手动加载模块是可能的,而不需要 kmod 或 kerneld。在这两种情况下,都必须设置内核选项 CONFIG_MODULES 以启用模块的使用。

要启用 kmod,必须在启用内核选项 CONFIG_KMOD 的情况下编译内核。因为 kmod 是作为内核模块实现的,所以内核选项 CONFIG_MODULES 也需要启用。

构建自定义内核

下面是关于如何配置和编译内核以满足自定义需求的总结。您从配置开始,通常使用 make 目标之一进行配置（make config、 xconfig、 menuconfig、 gconfig、 oldconfig）,请参阅有关创建 .config 文件的部分。

一旦配置了内核,就可以使用命令 make zImage/bzImage 来构建它。要构建模块,请使用 make modules,然后使用 make modules_install。要一步编译内核映像和可加载模块,您可以使用命令 make all。配置和编译内核后,可以使用 make install 安装。安装内核意味着将以下文件安装到 /boot 目录中：

- System.map-2.6.x
- config-2.6.x
- vmlinuz-2.6.x

depmod 命令构建一个 modules.dep 文件,modprobe 需要它来确定（新)构建模块的依赖关系。

除了在本地机器上构建和安装内核之外,您还可以创建包,以允许您将新构建的内核分发到其他机器。常见的包格式都有,每种都有自己的make参数：

- make rpm-pkg:构建源代码和二进制 RPM 包。
- make binrpm-pkg:构建（仅)二进制内核 RPM 包。
- make deb-pkg:将内核包构建为 deb(ian) 包。

/proc 文件系统

/proc 文件系统是一个伪文件系统,这意味着它的内容直接由内核提供。它包含反映正在运行的系统状态的目录和其他文件。一些命令读取 /proc 文件系统以获取有关系统状态的信息。您可以提取统计信息、硬件信息、网络和主机参数以及内存和性能信息。 /proc 文件系统还允许您通过将值写入文件来在运行时修改某些参数。

/、 /boot和/lib/modules的内容

Debian 系统上的这些目录的内容作为示例显示。

/的内容：

```
debian-601a:/$ ls -a
.          开机回家          lib32          媒体处理 selinux tmp .ure vmlinuz mnt
.. dev initrd.img lib64 bin 等 lib          根服务器          u8 usr
                             失物招领处     系统              u9 变量
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

最重要的目录是/bin,其中包含通用的系统范围命令，/var,这是将不断变化的数据（日志等)的顶级目录，/srv,其中包含与系统提供的服务相关的数据（例如网络服务器）), /tmp 用于保存临时文件，/mnt 用于临时挂载 USB 磁盘、磁带机和其他包含文件系统的设备，/lib、/lib32 和 /lib64 用于保存模块和其他库，/boot 用于保存文件在启动期间使用，/sbin 下我们可以找到系统管理员的特殊命令，/opt 用于第三方包,以及 /etc 包含系统的配置文件。

/usr 目录包含模仿根目录部分内容的影子层次结构。因此,您将找到 /usr/bin 目录、/usr/sbin 目录、/usr/lib 目录和 /usr/etc 目录。 “二级层次结构”旨在允许用户覆盖或增强系统,例如允许他们将本地系统管理员命令放在 /usr/sbin 中,将本地系统配置数据放在 /usr/etc 中。一个特例是 /usr/local 及其子目录,其中包含另一个模仿根目录部分内容的影子层次结构,这次包含非常具体的（本地系统)命令和配置数据。因此,您会在 /bin、/usr/bin 和 /usr/local/bin 中找到命令：/bin 中的命令将成为分发的一部分，/usr/bin 中的命令可能是非强制性命令或由系统管理员稍后添加,最后 /usr/local/bin 中的命令是特定于本地系统的命令。

注意

lost+found目录由文件系统本身使用。

/引导的内容：

```
debian-601a:/boot$ ls -a
.                    debian.bmp                    sarge.bmp
..                   debianlilo.bmp grub           sid.bmp 文件
coffee.bmp                         System.map-2.6.32-5-amd64
config-2.6.32-5-amd64 initrd.img-2.6.32-5-amd64 vmlinuz-2.6.32-5-amd64
```

最重要的是 vmlinuz、System.map、initrd.img 和配置文件。 grub 目录也位于此处。还有一些位图图像放在这里。

/lib/模块的内容：

```
debian-601a:/lib/modules$ ls -a
.      .. 2.6.32-5-amd64
```

```
debian-601a:/lib/modules/2.6.32-5-amd64$ ls -a
.          核心          modules.dep          模块.order          模块.symbols.bin
..         模块别名      modules.dep.bin modules.softdep 源码
build modules.alias.bin modules.devname modules.symbols 更新
```

另请参阅有关FHS 标准的部分。

跟踪软件及其系统和库调用的工具和实用程序

跟踪

strace - 跟踪系统调用和信号

```
概要
        strace [-CdffhiqrtttTvxx] [-sstrsize] [-uusername]]   [-eexpr] ...                    [-ofile] [-acolumn]   [ -ppid ] [ 命令 [
        ...                                                   [-Evar=val] ... [-Evar] ...
        参数...]

        strace -c [-eexpr] ...   [-开销]   [-分类]   [ 命令 [ 参数 ... ]   ]
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

strace 是一种有用的诊断、指导和调试工具。系统管理员、诊断人员和故障排除人员会发现它对于解决源代码不易获得的程序的问题具有无可估量的价值，因为它们不需要重新编译来跟踪它们。学生、黑客和过于好奇的人会发现，即使是通过跟踪普通程序，也可以了解有关系统及其系统调用的大量信息。程序员会发现，由于系统调用和信号是发生在用户/内核接口上的事件，因此仔细检查此边界对于错误隔离、完整性检查和尝试捕获竞争条件非常有用。

在最简单的情况下，strace 会运行指定的命令，直到它退出。它拦截并记录进程调用的系统调用和进程接收的信号。每个系统调用的名称、它的参数和它的返回值都打印在标准错误上或打印到用-o选项指定的文件中。

默认情况下，strace 报告系统调用的名称、它的参数和标准错误的返回值。

例子：

```
debian-601a:~$ strace cat /dev/null execve(    /bin/cat    , [    cat    ,
    /dev/null    ], [/* 34 vars */]) = 0 brk(0) 访问(    /etc/ld.so.nohwcap    , F_OK) mmap(NULL, 8192, PROT_READ|
PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fcac8fb4000 访问(    /etc/ld.so.preload    , R_OK)打开
（“/etc/ld.so.cache”，O_RDONLY)fstat（3，{st_mode=S_IFREG|0644，st_size=59695，...}）= 0 访问（没有那个文件或目录）

                                                                                                    = 0xc25000
                                                                                                    = -1 ENOENT（没有那个文件或目录）= 3

mmap(NULL, 59695, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fcac8fa5000 close(3) access(    /etc/ld.so.nohwcap    , F_OK)
打开(    /lib/libc.so.6    , O_RDONLY)读（3，“\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0    \355\ 1\0\0\0\0\0    ...,
832) = 832 fstat(3, {st_mode=S_IFREG|0755, st_size=1432968, ...}) = 0 mmap(NULL, 3962088, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0 ←。
                                                                                                    = 0
                                                                                                    = -1 ENOENT（没有那个文件或目录）= 3

          x7fcac8a38000
mprotect(0x7fcac8b90000, 2093056, PROT_NONE) = 0 mmap(0x7fcac8d8f000, 20480,
PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, ←。
            0x157000) = 0x7fcac8d8f000
mmap(0x7fcac8d94000, 18472, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, ←
          -1, 0) = 0x7fcac8d94000 关闭(3) mmap(NULL,
4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|                                                           = 0
MAP_ANONYMOUS, -1, 0) = 0x7fcac8fa4000 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1 )= 0x7fcac8fa3000 mmap（null,4096，
prot_read | prot_write,map_private | map_anonymous，-1,0）= 0x7fcac8fa2000 Arch_prctl（Arch_set_set_fs,0x7fcac8fa3700） = 0 munmap(0x7fcac8fa5000, 59695)
brk(0)
                                                                                                    = 0
                                                                                                    = 0xc25000
brk(0xc46000) open(    /                                                                            = 0xc46000
usr/lib/locale/locale-archive    , O_RDONLY) = 3 fstat(3, {st_mode=S_IFREG|0644,
st_size=1527584, ...}) = 0
mmap(NULL, 1527584, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fcac8e2d000 close(3) fstat(1, {st_mode=S_IFCHR|0620,
st_rdev=makedev(136, 1), ...}) = 0 打开(    /dev/null    , O_RDONLY) fstat(3, {st_mode=S_IFCHR|0666, st_rdev=makedev(1,
3), ...}) = 0 读取(3,        , 32768) 关闭(3) 关闭( 1) 关闭(2) 退出组(0)
                                                                                                    = 3

                                                                                                    = 0
                                                                                                    = 0
                                                                                                    = 0
                                                                                                    = 0
                                                                                                    =?
```

strace 的另一个非常有用的特性是它能够使用-p标志连接到正在运行的进程。结合-f标志，您可以连接到一个似乎出现故障的正在运行的守护进程，并了解它实际在做什么。

字符串

版权所有 2013-2017 Snow BV

Machine Translated by Google

strings - 打印文件中可打印字符的字符串。字符串对于读取非文本文件很有用。

```
概要
        字符串 [-afovV] [-min-len]
                [-n min-len] [--bytes=min-len] [-t radix] [--radix=radix] [-e
                encoding] [--encoding=encoding] [-] [--all] [- -打印文件名]


                [-T bfdname] [--target=bfdname] [--help] [--version] 文件...
```

对于每个给定的文件,GNU 字符串打印至少 4 个字符长的可打印字符序列（或下面选项给出的数字）,然后是不可打印的字符。默认情况下,它只打印目标文件的初始化和加载部分的字符串;对于其他类型的文件,它会打印整个文件中的字符串。

strings 主要用于确定非文本文件的内容,例如可执行文件。通常用于检查可执行文件使用的环境变量和配置文件的名称。

跟踪

ltrace - 库调用跟踪器

```
概要
        ltrace [-CfhiLrStttV] [-a column] [-A maxelts] [-D level] [-e expr] [-l file name] [-n nr] [-o filename] [-p pid] ... [ -s strsize] [-u 用户名] [-X extern] [-x extern] [--debug=level]
        [--demangle] [--help] [--indent=nr] [--library=文件名] [--output=文件名] [--version] [命令 [arg ...]]
                                ...             [--对齐=列]
```

类似于strace,拦截并记录系统调用,ltrace拦截并记录被执行进程调用的动态库调用以及该进程接收到的信号。要跟踪的程序不需要为此重新编译,因此您可以在手边没有源代码的二进制文件上使用它。默认情况下,ltrace 将启动您在命令行中指定的程序并跟踪它直到程序退出。

例子:

```
debian-601a:~$ ltrace cat /dev/null
__libc_start_main(0x401ad0, 2, 0x7fff5f357f38, 0x409110, 0x409100 <未完成...> getpagesize() strrchr(  cat  ,  /  ) setlocale(6,     ) bindtextdomain(  coreutils  ,  /
usr/share/locale  “)文本域（“coreutils”）                                                                                                  = 4096
                                                                                                                                           = 空
                                                                                                                                           =  en_US.utf8  =  /
                                                                                                                                           usr/share/locale
                                                                                                                                           = “核心工具”
__cxa_atexit(0x4043d0, 0, 0, 0x736c6974756572, 0x7f4069c04ea8) getenv(  POSIXLY_CORRECT  )                                                 = 0
__fxstat(1, 1, 0x7fff5f357d80) 打开(  /dev/null  , 0, 02) __fxstat(0x75f3, 02) __fxstat(0x75f3, 02)3                                       = 空
                                                                                                                                           = 0
                                                                                                                                           = 3
                                                                                                                                           = 0
malloc(36863) 读取(3,                                                                                                                      = 0x014c4030
      , 32768) 免费(0x014c4030)                                                                                                            = 0
关闭(3) 退出(0 <未完成...>                                                                                                                 = <无效>
__fpending(0x7f4069c03780, 0,                                                                                                              = 0
0x7f4069c04330, 0x7f4069c04330,
0x7f4069c04e40) = 0fclose(9c04e40) = 0f4069c04e40) = 0fclose(0x7f4069c04e40) 0x7f4069c03860, 0, 0x7f4069c04df0, 0, 0x7f4069e13700) = 0
fclose(0x7f4069c03860) = 0 +++ 退出（状态 0）  +++                                                                                          = 0
```

启动过程

使用 initrd 时,系统会经历以下步骤:

版权所有 2013-2017 Snow BV

Machine Translated by Google

1. 引导加载程序加载内核和初始 RAM 磁盘。
2. 内核将 initrd 转换为“普通”RAM 磁盘并释放 initrd 映像使用的内存。
3. initrd 映像以 root 身份以读写方式挂载
4. linuxrc 被执行（这可以是任何有效的可执行文件,包括 shell 脚本;它以 uid 0 运行并且可以做任何事情初始化可以做）
5. linuxrc 终止后,挂载“真正的”根文件系统
6. 如果目录 /initrd 存在,则将 initrd 映像移动到那里,否则,卸载 initrd 映像
7. 通常的启动顺序（例如调用 /sbin/init)在根文件系统上执行

由于将 initrd 从 / 移动到 /initrd 不需要卸载它,因此使用 RAMdisk 上的文件的进程可能会继续运行。所有已挂载的文件系统在移动期间将保持挂载状态。但是,如果 /initrd 不存在,则不会进行移动,在这种情况下,任何使用来自 initrd 映像的文件的正在运行的进程都将阻止它被卸载。它将保留在内存中,直到通过继续引导过程强制不存在。

此外,即使可以进行移动并且挂载在 initrd 下的文件系统仍然可以访问，/proc/mounts 中的条目也不会更新。还要记住,如果目录 /initrd 不存在,则无法卸载 RAM 磁盘映像。在剩余的引导过程中,该映像将被强制退出,因此安装在其中的任何文件系统也将消失并且无法重新安装。因此,强烈建议在从 initrd 文件系统切换到普通根文件系统（包括 /proc 文件系统)之前卸载所有文件系统。

可以回收用于 initrd 映像的内存。为此,必须在卸载 /initrd 后直接使用命令 freeramdisk。

将 initrd 支持包含到内核中会向引导命令行添加选项：

initrd=此选项加载指定为初始 RAM 磁盘的文件。

noinitrd此选项会保留 initrd 数据,但不会将其转换为 RAM 磁盘,而是挂载普通的根文件系统。可以从 /dev/initrd 读取 initrd 数据。如果通过 /dev/initrd 读取,数据可以具有任何结构,因此它不一定是文件系统映像。该选项主要用于调试

目的。

注意/

dev/initrd是只读的,只能使用一次。一旦最后一个进程关闭它,所有内存都将被释放并且/dev/initrd不能再被访问。

---

root=/dev/ram initrd 以 root 身份挂载,随后启动 /linuxrc。如果没有 /linuxrc,将执行正常的引导程序。如果不使用此参数,initrd 将被移动或卸载,但在这种情况下,根文件系统将继续是 RAM 磁盘。这样做的好处是它允许使用压缩文件系统并且速度稍快。

硬件和内核信息

名字

uname 打印系统信息,例如主机的机器类型、网络主机名、操作系统版本、操作系统名称、操作系统版本和处理器类型。参数是：

-a, --all按以下顺序打印所有信息,除非 -p 和 -i 未知,否则省略。

-s, --kernel-name打印内核名称

-n, --nodename打印网络节点主机名

版权所有 2013-2017 Snow BV

Machine Translated by Google

-r, --kernel-release打印内核版本

-v, --kernel-version打印内核构建版本、日期和时间

-m, --machine打印机器硬件架构名称

-p, --processor打印处理器类型的名称

-i, --hardware-platform打印硬件平台的名称

-o, --operating-system打印操作系统的名称

例子：

```
debian-601a:~$ uname -a Linux
debian-601a 2.6.32-5-amd64 #1 SMP Mon Mar 7 21:35:22 UTC 2011 x86_64 GNU/Linux
```

/proc/sys/kernel/ 是 /proc 伪文件系统中的一个目录。它包含允许您调整和监控 Linux 内核的文件。请小心，因为修改某些文件可能会导致系统挂起或功能失调。由于这些参数高度依赖于内核版本，因此建议在实际进行调整之前阅读文档和源代码。另请参阅有关 /proc 文件系统的部分。

有些文件是无害的，可以安全地用于获取信息，例如显示正在运行的内核的版本：

```
debian-601a:~$ cat /proc/sys/kernel/osrelease
```

一些文件可用于在内核中设置信息。例如，以下将告诉内核不要在 panic 时循环，而是在 20 秒后自动重启：

```
debian-601a:~$ echo 20 > /proc/sys/kernel/panic
```

lspci

使用 lspci，您可以显示有关系统中所有 PCI 总线以及连接到它们的所有设备的信息。请记住，您需要拥有 Linux 内核 2.1.82 或更新版本。对于较旧的内核，直接硬件访问仅适用于 root。
为了使 lspci 的输出更详细，可以添加一个或多个 -v 参数（最多 3 个）。在许多操作系统上，对 PCI 配置空间的某些部分的访问仅限于 root。所以普通用户可用的 lspci 功能是有限的。

在 Virtualbox 中运行 Debian 的系统上的 lspci 输出示例：

```
debian-601a:~$ lspci 00:00.0 主机桥：
Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA 桥：Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE 接口：Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
00:02.0 VGA 兼容控制器：InnoTek Systemberatung GmbH VirtualBox Graphics ←。
        适配器
00:03.0 以太网控制器：英特尔公司 82540EM 千兆以太网控制器（rev ←。
        02)
00:04.0 系统外围设备：InnoTek Systemberatung GmbH VirtualBox Guest Service 00:05.0 多媒体音频控制器：Intel Corporation 82801AA AC   97 音
频控制器（←
        修订版 01)
00:06.0 USB 控制器：Apple Computer Inc. KeyLargo/Intrepid USB 00:07.0 桥接器：Intel Corporation 82371AB/EB/MB
PIIX4 ACPI (rev 08) 00:0d.0 SATA 控制器：Intel Corporation 82801HBM/HEM (ICH8M/ICH8M- E) SATA AHCI ←

        控制器（修订版 02）
```

lsusb

lsusb 类似于 lspci，但检查 USB 总线和设备。要使用 lsusb，需要一个支持 /proc/bus/usb 接口的 Linux 内核（Linux 2.3.15 或更新版本）。 lsusb -v 将提供详细信息。

在运行 Ubuntu 的笔记本电脑上生成的 lsusb 输出示例：

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
ubuntu:/var/log$ lsusb Bus 007 Device
003: ID 03f0:0324 Hewlett-Packard SK-2885 keyboard Bus 007 Device 002: ID 045e:0040 Microsoft Corp. Wheel Mouse Optical
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 根集线器

总线 006 设备 001:ID 1d6b:0001 Linux Foundation 1.1 根集线器
总线 005 设备 001:ID 1d6b:0001 Linux Foundation 1.1 根集线器
总线 004 设备 002:ID 147e:2016 Upek Biometric Touchchip/Touchstrip 指纹传感器总线 004 设备 001:ID 1d6b:0001 Linux Foundation 1.1 根集线器

总线 003 设备 001:ID 1d6b:0001 Linux Foundation 1.1 根集线器
总线 002 设备 001:ID 1d6b:0002 Linux Foundation 2.0 根集线器
Bus 001 Device 002: ID 04f2:b018 Chicony Electronics Co., Ltd 2M UVC Webcam Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 根集线器
```

lsdev

lsdev 显示有关已安装硬件的信息。它从中断、DMA 文件和来自 /proc 目录的中断中收集信息。 lsdev 给出了关于哪个设备正在使用什么 I/O 地址和 IRQ/DMA 通道的快速概览。正在使用的文件：

```
/proc/中断
```

```
/proc/ioports
```

```
/过程/DMA
```

基于 Debian 的系统上的 lsdev 输出示例：

```
debian-601a:/var/log$ lsdev
设备                                    DMA IRQ I/O 端口
---------------------------------------
0000:00:01.1                                    0170-0177 01f0-01f7 0376-0376 03f6-03f6 d000-d00f
0000:00:03.0                                    d010-d017
0000:00:04.0                                    d020-d03f
0000:00:05.0                                    d100-d1ff d200-d23f
0000:00:0d.0                                    d240-d247 d250-d257 d260-d26f
82801AA-ICH                                 5
ACPI                                            4000-4003 4004-4005 4008-400b 4020-4021
ahci                                                d240-d247 d250-d257 d260-d26f
ata_piix级联                            14 15       0170-0177 01f0-01f7 0376-0376 03f6-03f6 d000-d00f
                                        4       2
DMA                                             0080-008f
DMA1                                            0000-001f
DMA2                                            00c0-00df
e1000                                               d010-d017
eth1                                        10
处理                                            00f0-00ff
器i8042                                 1 12
英特尔                                              d100-d1ff d200-d23f
键盘                                            0060-0060 0064-0064
ohci_hcd:usb1                               11
PCI                                             0cf8-0cff
pic1                                            0020-0021
pic2                                            00a0-00a1
rtc0                                        8       0070-0071
rtc_cmos 定时                                   0070-0071
器                                          0
计时器0                                         0040-0043
定时器1                                         0050-0053
vbox客人                                    9
显卡+                                           03c0-03df
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

注意在

某些系统上缺少lsdev ,请改用procinfo 。

系统控制

sysctl 用于在运行时修改内核参数。可用的参数列在 /proc/sys/ 下。配置文件通常可以在 /etc/sysctl.conf 中找到。重要的是要知道在使用 sysctl 之后加载的模块可能会覆盖其设置。您可以通过仅在加载所有模块后运行 sysctl 来防止这种情况。

dmesg

使用 dmesg 命令,您可以将内核消息写入标准输出。或者使用 dmesg > /var/log/-boot.messages 将它们写入文件。这在解决系统问题或只想获取有关系统硬件的一些信息时会很有帮助。 dmesg 的输出通常可以在 /var/log/dmesg 中找到。将 dmesg 与 grep 结合使用可查找特定信息。

udev

考生应该了解使用 udev 进行设备检测和管理。此目标包括对 udev 规则进行故障排除

关键知识领域:

- udev 规则
- 内核接口

udev 旨在通过将设备处理从系统空间转移到用户空间来使 Linux 设备处理更加灵活和安全。

udev 由一个用户空间守护进程 (udevd) 组成,它从内核接收“uevents”。用户空间守护进程和内核之间的通信是通过 sysfs 伪文件系统完成的。当内核检测到硬件的添加或移除时,例如当您插入相机或 USB 磁盘时,内核会发送上述“uevents”。基于一组匹配这些 uevent 的规则,内核提供了一个动态设备目录,其中仅包含实际存在的设备的设备文件,并且可以启动脚本等来执行各种操作。因此,例如,可以插入一个照相机,它会被自动检测并正确安装在正确的位置。之后,可能会启动一个脚本,将所有文件复制到本地磁盘到一个正确命名的子目录中,该子目录将使用适当的权限创建。

/etc/udev/

/etc/udev/ 目录包含 udev 实用程序的配置和规则文件。 udev.conf 是 udev 的主要配置文件。例如,这里可以更改日志记录优先级。

开发规则

udev 规则是从位于默认规则目录中的文件中读取的。 /lib/udev/rules.d/。覆盖这些默认规则的自定义规则在 /etc/udev/rules.d/ 目录中指定。

当设备被初始化或移除时,内核发送一个“uevent”。这些uevents包含子系统（如net、sub）、动作和属性（如mac、vendor)等信息。 udev 监听这些事件,将 uevent 信息匹配到指定的规则,并做出相应的响应。

udev 规则由多个以逗号分隔的键值对组成。每个键值对代表一个匹配键,它与 uevent 提供的信息匹配,或者一个分配键,它将值、名称和操作分配给由 udev 维护的设备节点:

```
SUBSYSTEM==net, ACTION==ADD, DRIVERS=?*, ATTR{address}==00:21:86:9e:c2:c4,
ATTR{type}==1, KERNEL=eth*, NAME==eth0
```

上面指定的规则将为 MAC 地址为 00:21:86:9e:c2:c4 的网卡添加设备 /dev/eth0

如图所示,键值对还指定了一个运算符。根据使用的运算符,采取不同的操作。有效运算符是:

版权所有 2013-2017 Snow BV

Machine Translated by Google

== 比较是否相等 != 比较是否相
等 = 为键分配一个值。代表列表的
键被重置,并且只分配这个单个值 += 最后将值添加到键;不允许任何以后的更改,可用于防止任何以后的规则更改

指定规则时,注意不要创建冲突的规则（例如,不要将两个不同的网卡指向相同的设备名称）。此外,将设备名称更改为其他名称可能会导致用户空间软件不兼容。你有力量,明智地使用它。

udevmonitor

udevmonitor 会将 udev 和内核 uevents 打印到标准输出。您可以使用它通过比较内核 uevent 和 udev 事件的时间戳来分析事件时序。通常 udevmonitor 是到 udevadm 的符号链接。在某些发行版中,此符号链接不再存在。要访问这些系统上的监视器,请使用命令 udevadm monitor。

问题和解答

内核

1.现代内核使用了哪种技术来减少磁盘占用?

压缩内核映像以减少磁盘空间。内核映像的类型[16]

2. 使用内核模块的目的是什么?

为了只在需要的时候使用资源,内核模块可以按需加载到内核中。此外,无需重新启动内核即可更改和重新加载模块。什么是内核模块[17]

3. 为什么运行命令 make mrproper 通常是个好主意?

因为它可以防止后续配置和/或内核构建出现问题。使用 mrproper [19]

4. 运行 make config 而不是 make menuconfig、make xconfig 或 make 最重要的优点是什么
配置?

该命令不依赖于任何全屏显示功能,因此可用于慢速链接和显示功能有限的系统。为什么要使用 make config? [21]

5. make modules运行后需要做什么?

make modules_install 在成功构建模块（设备驱动程序和其他配置为模块的项目)后使用。这会将它们安装在 /lib/modules/ kernel-version 下。如何安装模块? [25]

6. 对于哪些发行版,您应该使用 mkinitramfs 来创建 initrd 映像?

debian（基于)发行版使用 mkinitramfs。使用 mkinitramfs [48]创建点燃图像

7. 可以通过什么方式从生产内核中删除补丁?

再次应用补丁或使用 -R 参数运行命令补丁。从生产内核中删除内核补丁[29]

8. 为什么推荐使用 modprobe 而不是使用 insmod 来插入模块?

Modprobe 对错误更具体。它也只与模块名称一起工作,并且可以处理模块之间的先决条件关系。使用 modprobe [32]

9. 通过什么方式可以判断运行内核的版本?

您可以使用 cat /proc/sys/kernel/osrelease 或使用 uname -r 进行检查。获取正在运行的内核的版本[42]

10.哪个命令生成文件/lib/modules/kernel-version/modules.depfile（模块de列表
悬而未决）?

该文件由 depmod 命令生成。获取正在运行的内核的版本[42]

版权所有 2013-2017 Snow BV

Machine Translated by Google

11.现代内核使用了哪种技术来减少内存占用?

按需加载模块用于减少内核的内存占用。内核映像的类型[16]

12. udev 是出于什么原因设计的?

它旨在通过将设备处理从内核转移到用户空间来使 Linux 设备处理更加灵活。 udev 设备处理[44]

版权所有 2013-2017 Snow BV

Machine Translated by Google

第2章

系统启动 (202)

本题目总权重为9分,包含以下3个目标:

目标 202.1 自定义 SysV-init 系统启动（3 分） 考生应该能够查询和修改系统服务在不同运行级别的行为。需要对 init 结构和引导过程有透彻的了解。此目标包括与运行级别的交互。

目标 202.2 系统恢复（4 分） 考生应该能够在引导过程和恢复模式期间正确操作 Linux 系统。此目标包括使用 init 实用程序和与 init 相关的内核选项。

考生应该能够确定加载错误的原因和引导加载程序的使用。 GRUB 版本 2 和 GRUB Legacy 是感兴趣的引导加载程序。

目标 202.3 备用引导加载程序（2 分） 考生应了解其他引导加载程序及其主要功能。

自定义系统启动 (202.1)

考生应该能够在各种目标/运行级别查询和修改系统服务的行为。需要对 systemd、SysV Init 和 Linux 引导过程有透彻的了解。此目标包括与 systemd 目标和 SysV init 运行级别进行交互。

关键知识领域

·系统

· SysV 初始化

· Linux 标准基本规范 (LSB)

条款和实用程序

· /usr/lib/systemd/

· /etc/系统/

· /运行/系统/

· 系统控制

· systemd-delta

· /etc/inittab

版权所有 2013-2017 Snow BV

Machine Translated by Google

- /etc/init.d/
- /etc/rc.d/
- 检查配置
- 更新-rc.d
- 初始化和 telinit

使用 mkinitrd创建initrd

注意

mkinitrd 在上一节中进行了讨论,该节还讨论了如何手动创建此类映像。

为了限制内核的大小,通常使用初始 ramdisk (initrd) 映像来预加载访问根文件系统所需的任何模块。

mkinitrd 是一个特定于基于 RPM 的发行版（例如 Red Hat、SuSE 等)的工具。该工具自动执行创建 initrd 文件的过程,从而确保正确遵循相对复杂的过程。

在大多数较大的 Linux 发行版中,initrd 映像包含几乎所有必需的内核模块;很少会直接编译进内核。这使得可以通过 RPM 包对内核及其模块进行简单的修复和补丁部署:单个模块的更新不需要重新编译或替换整个内核,而只需要单个模块,或者最坏的情况是几个依赖模块.因为这些模块包含在 initrd 文件中,所以每次（手动)重新编译内核或安装内核（模块)补丁时都需要重新生成该文件。如果使用许多发行版提供的标准工具,生成新的 initrd 映像非常简单:

```
# mkinitrd initrd-image 内核版本
```

mkinitrd 的有用选项包括:

--version此选项显示 mkinitrd 实用程序的版本号。

-f通过指定此开关,该实用程序将覆盖任何具有相同名称的现有图像文件。

--builtin=这会导致 mkinitrd 假定指定的模块已编译到内核中。它不会寻找模块,如果不存在也不会显示任何错误。

--omit-lvm-modules, --omit-raid-modules, --omit-scsi-modules使用这些选项可以防止分别包含 LVM、RAID 或 SCSI 模块,即使它们存在,或者该实用程序通常会根据 /etc/fstab 和/或 /etc/raidtab 的内容包含它们。

使用 mkinitramfs创建initrd

由于对基于 RPM 的发行版使用的工具 (mkinitrd) 不满意,一些 Debian 开发人员编写了另一个实用程序来生成 initrd 文件。这个工具叫做 mkinitramfs。
mkinitramfs 工具是一个 shell 脚本,它生成一个 gzip 压缩的 cpio 图像。
它被设计成比 mkinitrd 简单得多（在代码和使用上）。该脚本由大约 380 行代码组成。

mkinitramfs 的配置是通过一个配置文件完成的:initramfs.conf。该文件通常位于 /etc/initramfs-tools/initramfs.conf 中。这个配置文件来源于脚本 它包含标准的 bash 声明。注释以# 为前缀。变量指定为:

```
变量=值
```

可以与 mkinitramfs 一起使用的选项包括:

-d confdir此选项设置备用配置目录。

版权所有 2013-2017 Snow BV

Machine Translated by Google

-k “保留”用于创建映像的临时目录。

-o outfile将生成的图像写入 outfile。

-r root覆盖 initramfs.conf 文件中的ROOT设置。

注意

在 Debian（基于)的发行版上,您应该始终使用mkinitramfs ,因为mkinitrd在更新的内核中已损坏。

设置根设备

设置根设备是众多内核设置之一。内核设置源自或可以被覆盖：

- 源代码中设置的默认值，
- 默认值由 rdev 命令设置，
- 在引导时传递给内核的值,例如root=/dev/xyz
- GRUB 配置文件中指定的值。

最容易使用的是块设备,如硬盘、SAN 存储、CD 或 DVD。您甚至可以安装 NFS 根磁盘,这需要使用 initrd 并设置nfs_root_name和nfs_root_addrs引导选项。您可以在 initrd 环境中将根设备设置或更改为几乎任何东西。为此,请确保 /proc 文件系统已由 initrd 映像中的脚本挂载。 /proc 中提供以下文件：

```
/proc/sys/kernel/real-root-dev /proc/sys/kernel/
nfs-root-name /proc/sys/kernel/nfs-root-addrs
```

real-root-dev是根文件系统设备的节点号。可以通过向其中写入新数字轻松更改它：

```
# echo 0x301>/proc/sys/kernel/real-root-dev
```

这会将真正的根更改为 /dev/hda1 上的文件系统。如果您希望使用 NFS 挂载的根目录,则必须使用适当的值设置文件 nfs-root-name 和 nfs-root-addrs，并且设备号应设置为 0xff：

```
# echo /var/nfsroot >/proc/sys/kernel/nfs-root-name # echo
193.8.232.2:193.8.232.7::255.255.255.0:idefix \
    >/proc/sys/kernel/nfs-root-addrs # echo 255>/proc/
sys/kernel/real-root-dev
```

注意如

果根设备设置为 RAM 磁盘,则根文件系统不会移动到/initrd,但通过在初始 RAM 磁盘上启动 init 来简单地继续引导过程。

Linux 启动过程

启动过程中有七个可区分的阶段：

1. 内核加载器加载、设置和执行
2. 注册设置

版权所有 2013-2017 Snow BV

Machine Translated by Google

3.内核解压

4.内核和内存初始化

5.内核设置

6.启用剩余的CPU

7.init进程创建

引导过程在Gustavo Duarte 的“内核引导过程”中有详细描述

内核在引导过程中的最后一步 [*] 尝试按顺序执行这些命令,直到一个成功:

1. /sbin/初始化
2. /etc/初始化
3. /bin/初始化
4. /bin/sh

如果这些都不成功,内核就会崩溃。

初始化进程_

init 是所有进程的父进程,它读取文件 /etc/inittab 并根据其内容创建进程。它通常做的一件事是生成 gettys 以允许用户登录。它还定义了“运行级别”。

“运行级别”是系统的软件配置,它只允许一组选定的进程存在。

INIT 可以处于以下八个运行级别之一

运行级别 0（保留） 运行级别 0 用于停止系统。

runlevel 1 (reserved) Runlevel 1 用于让系统进入单用户模式。

运行级别 2-5 运行级别 2、3、4 和 5 是多用户运行级别。

runlevel 6 Runlevel 6 用于重启系统。

运行级别 7-9 运行级别 7、8 和 9 可以根据需要使用。大多数 Unix(/Linux) 变体不使用这些运行级别。例如,在典型的 Debian Linux 系统上,我们稍后将讨论的 /etc/rc<runlevel>.d 目录对于这些运行级别不可用,尽管那是完全合法的。

运行级别 s 或 S 运行级别 s 和 S 在内部是相同的。它使系统进入“单用户模式”。 /etc/rcS.d 目录中的脚本在系统启动时执行。虽然运行级别 S 并不意味着由用户激活,但它可以。

运行级别 A、B 和 C 运行级别 A、B 和 C 是所谓的“按需”运行级别。例如,如果当前运行级别为“2”,并且执行了 init A 命令,则执行运行级别“A”内启动或停止进程的脚本,但实际运行级别仍为“2”。

[*] 如内核函数 kernel_init() 中所定义,以前称为 init_post()。资料来源： d6b212380 ... （git.kernel.org）

版权所有 2013-2017 Snow BV

Machine Translated by Google

配置/etc/inittab

如前所述,init 读取文件 /etc/inittab 以确定它应该做什么。此文件中的条目具有以下格式：

id:runlevels:action:process

下面包括一个示例 /etc/inittab 文件。

```
# 默认运行级别。
编号:2:初始化默认值:

# 启动时系统配置/初始化脚本。
# 除了以紧急 (-b) 模式启动时,它会首先运行。 si::sysinit:/etc/init.d/rcS


# 在单用户模式下做什么。
~~:S:wait:/sbin/sulogin

# /etc/init.d 在更改运行级别时执行 S 和 K 脚本。

#
# 运行级别 0 停止。
# 运行级别 1 是单用户。
# 运行级别 2-5 是多用户的。
# 运行级别 6 是重启。

l0:0:等待:/etc/init.d/rc 0
l1:1:等待:/etc/init.d/rc 1
l2:2:等待:/etc/init.d/rc 2
l3:3:等待:/etc/init.d/rc 3 l4:4:等待:/etc/init.d/rc
4 l5:5:等待:/etc/init.d/rc 5 l6:6:等待:/etc/init.d/
rc 6 # 平时达不到,紧急情况下掉线。 z6:6:重生:/
sbin/sulogin



# /sbin/getty 调用运行级别。 # # “id”字段必须与设备的最后 # 个字符相同（在“tty”
之后）。 # # 格式:# <id>:<runlevels>:<action>:<process> 1:2345:respawn:/sbin/
getty 38400 tty1 2:23:respawn:/sbin/getty 38400 tty2
```

/ET C/I NITTAB中条目的描述

id id 字段唯一标识文件 /etc/inittab 中的条目,长度可以为 1-4 个字符。但是对于gettys等登录进程,id字段应该包含相应tty的后缀,否则登录计费可能无法工作。

runlevels 该字段包含应对其执行指定操作的运行级别。

动作“动作”字段值

respawn 进程将在它终止时重新启动，（例如 getty）。 wait 当进入指定的运行级别时，

进程将启动一次,init 将等待其终止。 once 当进入指定的运行级别时,进程将执行一次。 boot 进程将在系统启动时执行。 runlevels

字段被忽略。

版权所有 2013-2017 Snow BV

Machine Translated by Google

bootwait 该进程将在系统启动期间执行,而 init 等待其终止（例如 /etc/rc）。 runlevels 字段被忽略。 off 这绝对没有任何作用。 ondemand 标有按需运行级别的进程将在指定的按需运行级别运行时执行

叫。但是,不会发生运行级别更改（按需运行级别为“a”、“b”和“c”）。

initdefault initdefault 条目指定系统启动后应进入的运行级别。如果不存在,init 将在控制台上请求一个运行级别。过程字段被忽略。在上面的示例中,系统将在启动后进入运行级别 2。

sysinit 进程会在系统启动时执行。它将在任何引导或引导等待条目之前执行。 runlevels 字段被忽略。 powerwait 该进程将在电源掉电时执行。 init 通常通过与连接到计算机的 UPS 对话的进程来了解这一点。 init 将在继续之前等待进程完成。

powerfail 与 powerwait 一样,只是 init 不等待进程完成。 powerokwait 一旦通知 init 电源已恢复,

该过程将立即执行。 powerfailnow 当 init 被告知外部 UPS 的电池几乎耗尽并且电源出现故障时,将执行此过程（前提是外

部 UPS 和监控过程能够检测到这种情况）。 ctrlaltdel 进程会在init收到SIGINT信号时执行。这意味着有人在系统控制台上按下了 CTRL-ALT-DEL 组合键。通常,人们想要执行某种关机以进入单用户级别或重新启动机器。

kbdrequest 当 init 从键盘处理程序接收到在控制台键盘上按下特殊组合键的信号时,该进程将被执行。基本上你想将一些键盘组合映射到“KeyboardSignal”动作。例如,要为此目的映射 Alt-Uparrow,请在您的键映射文件中使用以下内容:alt keycode 103 =KeyboardSignal。

process 此字段指定应执行的进程。如果进程字段以“+”开头,init 将不会执行 utmp 和 wtmp会计。一些 gettys 坚持自己做家务。

/etc/init.d/rc脚本_

对于每个运行级别 0-6，/etc/inittab 中都有一个执行 /etc/init.d/rc 的条目?在哪里 ”？”是 0-6,正如您在前面示例中的以下行中所见：

```
l2:2:等待:/etc/init.d/rc 2
```

因此,实际发生的是 /etc/init.d/rc 以运行级别作为参数被调用。

目录 /etc 包含几个特定于运行级别的目录,这些目录又包含指向 /etc/init.d/ 中脚本的特定于运行级别的符号链接。这些目录是：

```
$ ls -d /etc/rc* /etc/rc.boot /
etc/rc1.d /etc/rc3.d /etc/rc5.d /etc/rcS.d /etc/rc0.d
                                /etc/rc2.d /etc/rc4.d /etc/rc6.d
```

如您所见,还有一个 /etc/rc.boot 目录。该目录已过时,已被目录 /etc/rcS.d 取代。在启动时,首先扫描目录 /etc/rcS.d,然后为了向后兼容,扫描 /etc/rc.boot。

符号链接的名称以“S”或“K”开头。让我们检查一下 /etc/rc2.d 目录：

```
$ ls /etc/rc2.d K20gpm
S10ipchains S12kerneld S11pcmcia S20logoutd S20ssh                    S89cron
S20lpd S10sysklogd S14ppp S11klogd S20inetd          S20xfs           S91阿帕奇
                                S20makedev S22ntpdate S99gdm
                                S20mysql             S89atd           S99登录
```

如果符号链接的名称以“K”开头,则以“stop”作为参数调用脚本以停止进程。 K20gpm 就是这种情况,因此命令变为 K20gpm 停止。让我们找出调用的程序或脚本：

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
$ ls -l /etc/rc2.d/K20gpm lrwxrwxrwx 1 root
root 13 Mar 23 2001 /etc/rc2.d/K20gpm -> ../init.d/gpm
```

因此,K20gpm 停止导致 /etc/init.d/gpm 停止。让我们通过检查脚本的一部分来了解“stop”参数会发生什么：

```
#!/bin/sh#

# 启动鼠标事件服务器
...
以 $1 开头）

        gpm_start
        ;;
停止）
       gpm_stop
        ;;
强制重新加载|重新启动)gpm_stop sleep
        3 gpm_start



        ;;
*)
        echo 用法：/etc/init.d/gpm {start|stop|restart|force-reload}
        出口 1
esac
```

在 case..esac 中,检查第一个参数 $1,如果其值为“stop”,则执行 gpm_stop。

另一方面,如果符号链接的名称以“S”开头,则以“start”作为参数调用脚本以启动该过程。

脚本按照文件名的词法排序顺序执行。

假设我们有一个守护进程 SomeDaemon,一个附带的脚本 /etc/init.d/SDscript 并且我们希望 SomeDaemon 在系统处于运行级别 2 时运行,而不是在系统处于运行级别 3 时运行。

如你所知,这意味着我们需要一个符号链接,以“S”开头,用于运行级别 2 和一个符号链接,以“K”开头,用于运行级别 3。我们还确定守护进程 SomeDaemon 是在 S19someotherdaemon 之后启动,这涉及 S20 和 K80,因为启动/停止是对称的,即最先启动的最后停止。这是通过以下一组命令完成的：

```
# cd /etc/rc2.d # ln -s ../
init.d/SDscript S20SomeDaemon # cd /etc/rc3.d

# ln -s ../init.d/SDscript K80SomeDaemon
```

如果您希望手动启动、重新启动或停止进程,最好使用 /etc/init.d/ 中的适当脚本,例如 /etc/init.d/gpm restart 来启动进程的重新启动。

更新-rc.d

笔记

本节仅适用于 Debian（基于)发行版

Debian 衍生的 Linux 发行版使用 update-rc.d 命令来安装和删除上一节中提到的 init 脚本链接。

如果你在 /etc/init.d/ 中有一个名为“foobar”的启动脚本,并且想将它添加到默认运行级别,你可以使用：

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
# update-rc.d foobar defaults 为 /etc/init.d/foobar
添加系统启动 ...
   /etc/rc0.d/K20foobar -> ../init.d/foobar /etc/rc1.d/K20foobar -> ../
   init.d/foobar /etc/rc6.d/K20foobar -> ../init .d/foobar /etc/rc2.d/
   S20foobar -> ../init.d/foobar /etc/rc3.d/S20foobar -> ../init.d/foobar /
   etc/rc4.d/S20foobar -> ../init.d/foobar /etc/rc5.d/S20foobar -> ../
   init.d/foobar
```

update-rc.d 将在 rc0.d、rc1.d 和 rc6.d 中创建 K（停止)链接,在 rc2.d、rc3.d、rc4.d 和 rc5.d 中创建 S（开始)链接。

如果您不希望已安装的软件包自动启动,请使用 update-rc.d 删除启动链接,例如禁用启动时启动 dovecot：

```
# update-rc.d -f dovecot remove 删除 /etc/init.d/
dovecot 的所有系统启动链接 ...
   /etc/rc2.d/S24dovecot /etc/rc3.d/
   S24dovecot /etc/rc4.d/S24dovecot /
   etc/rc5.d/S24dovecot
```

如果rc 脚本仍然存在,则需要-f （强制)选项。如果您安装更新的 dovecot 包,链接将被恢复。
为了防止这种情况在启动运行级别目录中创建“停止”链接：

```
# update-rc.d -f dovecot 停止 24 2 3 4 5 。
为 /etc/init.d/dovecot 添加系统启动 ... /etc/rc2.d/K24dovecot -> ../init.d/dovecot /etc/
   rc3.d/K24dovecot -> ../init.d/ dovecot /etc/rc4.d/K24dovecot -> ../init.d/dovecot /
   etc/rc5.d/K24dovecot -> ../init.d/dovecot
```

注意不

要忘记尾随。（点）。

使用系统目标

systemd 使用目标来定义系统状态,而不是预定义的运行级别。这些目标由目标单位表示。
目标单元以 .target 文件扩展名结尾,它们的唯一目的是通过依赖链将其他 systemd 单元组合在一起。这也意味着,与 init 运行级别相比,多个目标可以同时处于活动状态。

例如,graphical.target 单元作为 GNOME 显示管理器启动服务,但也依赖于 multi-user.target（这是非图形系统状态）,而 multi-user.target 又依赖于 basic.target。

在我们继续管理和使用目标之前,您应该知道哪些目录用于存储默认目标文件以及如何覆盖它们。

与所有单元一样,默认目标文件存储在 /usr/lib/systemd 中,此目录中的文件由您安装的软件的供应商创建,除非安装脚本,否则不应更改这些文件。您可以存储自定义目标和覆盖的目录是 /etc/systemd,在此目录中写入的所有内容都优先于 /usr/lib/systemd 中的文件。

有多种方法可以覆盖或附加属性到单元文件。您可以在 /etc/systemd/system 中创建一个具有相同名称的全新文件,例如 ssh.server。这将覆盖完整的单元定义。如果您只想附加或更改某些属性,您可以在 /etc/systemd/system 中创建一个新目录,并在其名称后附加 .d,例如 sshd.server.d。在此目录中,您可以创建扩展名为 .conf 的文件,您可以在其中放置您喜欢附加或覆盖的属性。这两种方式也可以通过使用 systemctl 命令来完成,这将在第 206.3 章中进一步描述。

版权所有 2013-2017 Snow BV

Machine Translated by Google

还有第三个位置,您可以在其中放置文件来覆盖您的单元定义,这是 /run/systemd 目录。此目录中的定义优先于 /usr/lib/systemd 中的文件,但不优先于 /etc/systemd 中的文件。
/run/systemd 中的覆盖只会在系统重新启动之前使用,因为 /run/systemd 中的所有文件都将在系统重新启动时被删除。

要获得所有覆盖活动的概述,您可以使用 systemd-delta 命令。此命令可以返回以下类型:

masked 被屏蔽的单位,不能启动的单位。

equivalent 内容相同的覆盖文件。

将符号链接重定向到其他单元文件。

overridden 覆盖的单元文件。

extended 使用 .conf 文件的扩展单元文件。

您还可以使用 --type= 标志的 -t 按上述类型进行过滤,这些会列出上述类型。如果您还想查看未更改的文件,您可以将未更改的文件添加为一种类型。如果您不希望 systemd-delta 显示被覆盖文件的差异,您可以使用的其他选项是 --diff=false,如果您不希望输出通过管道传输到寻呼机,则可以使用 --no-pager。

现在我们知道了单元文件的存储位置以及我们如何覆盖它们,我们可以看看更改系统状态。

为了获取和更改(默认)系统状态,我们使用 systemctl 命令。可以在第 206.3 章中找到对该命令的可能性的完整解释。

如果我们想获得默认目标,我们可以运行以下命令:

```
$ systemctl 获取默认值
```

这将输出在启动时使用的当前默认目标。要获取所有当前加载的目标单元的列表,您可以运行以下命令L

```
$ systemctl list-units --type=目标
```

这将为您提供所有当前活动目标的列表。要获取所有可用的目标单位,您可以运行以下命令:

```
$ systemctl 列表单元文件 --type=target
```

如果要更改默认目标,可以使用 systemctl set-default 命令。例如,要将默认目标设置为 multi-user.target,您可以运行以下命令:

```
$ systemctl set-default 多用户.target
```

此命令将创建一个符号链接 /etc/systemd/system/default.target,它链接到 /usr/lib/sys temd/system 中的目标文件。

init 运行级别和系统目标之间的比较:

您还可以在运行时更改系统状态,这可以使用 systemctl isolate 命令来完成。这将停止所有未为所选目标定义的服务,并启动所有为目标定义的服务。

例如,要进入救援模式,您可以运行以下命令:

```
$ systemctl 隔离 rescue.target
```

这将停止除为救援目标定义的服务之外的所有服务。请注意,此命令不会通知任何登录用户该操作。

还有一些快捷方式可用于更改系统状态,这些快捷方式有一个额外的好处,即它们会通知登录用户该操作。例如,另一种使系统进入救援模式的方法如下:

```
$ systemctl 救援
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

运行级别	目标	描述 关闭并关闭
0	runlevel0.target, poweroff.target	系统。
1个	runlevel1.target, rescue.target	设置救援外壳。
2个	runlevel2.target, multi-user.target	设置非图形多用户系统。
3个	runlevel3.target, 多用户.target	设置非图形多用户系统。
4个	runlevel4.target, 多用户.target	设置非图形多用户系统。
5	runlevel5.target、graphical.target	设置一个图形化的多用户系统。
6	runlevel6.target、reboot.target	关闭并重新启动系统。

表 2.1:运行级别和目标

这将首先通知登录用户该操作,然后停止目标中未定义的所有服务。如果您不希望用户收到通知,您可以添加 --no-wall 标志。

如果您的系统坏到无法使用救援模式,也可以使用紧急模式。这可以通过使用以下命令启动:

```
$ systemctl 紧急情况
```

如果你想在启动时启动某些服务,你必须启用它们。您可以使用 systemctl enable 命令启用服务。例如,要启用 sshd,您可以运行以下命令:

```
$ systemctl 启用 sshd.service
```

要再次禁用该服务,您可以使用 systemctl disable。例如:

```
$ systemctl 禁用 sshd.service
```

服务的单元文件定义了服务运行的系统状态。这是通过在 [Install] 部分下设置 WantedBy= 选项来配置的。如果将其设置为 multi-user.target,那么它将在非图形和图形模式下运行。

LSB 标准

Linux Standard Base (LSB) 为应用程序定义了一个接口,这些应用程序是为符合 LSB 的实现而编译和打包的。因此,在 LSB 兼容环境中编译的程序将在支持 LSB 标准的任何发行版上运行。 LSB 兼容程序可以依赖某些标准库的可用性。该标准还包括一系列强制性实用程序和脚本,它们定义了适合安装 LSB 兼容二进制文件的环境。

该规范包括处理器架构特定信息。这意味着 LSB 是一系列规范,而不是单个规范。换句话说:如果您的 LSB 兼容二进制文件是为基于 Intel 的系统编译的,它不会在基于 Alpha 的 LSB 兼容系统上运行,但会在任何基于 Intel 的 LSB 兼容系统上安装和运行。

因此,LSB 规范由通用部分和特定于体系结构的部分组成; “LSB-generic”或“generic LSB”和“LSB-arch”或“archLSB”。

LSB 标准列出了哪些通用库应该可用,例如 libdl、libcrypt、libpthread 等等,并提供了处理器特定库的列表,例如 libc 和 libm。该标准还列出了这些库的搜索路径、它们的名称和格式 (ELF)。另一部分处理动态链接的实现方式。对于每个标准库,都给出了一个函数列表,并列出了数据定义和附带的头文件。

LSB 定义了 130 多个命令的列表,这些命令应该在 LSB 兼容系统上可用,以及它们的调用约定和行为。一些示例是 cp、tar、kill 和 gzip,以及运行时语言 perl 和 python。

LSB 兼容系统在系统初始化期间的预期行为是 LSB 规范的一部分。 cron 系统的定义也是如此,初始化脚本的操作、功能和位置也是如此。任何符合 LSB 的初始化脚本都应该

版权所有 2013-2017 Snow BV

Machine Translated by Google

能够处理以下选项:启动、停止、重新启动、强制重新加载和状态。重新加载和尝试重新启动选项是可选的。该标准还列出了运行级别的定义以及用户名和组名及其对应的 UID/GID 的列表。

尽管可以在不使用包管理器的情况下安装 LSB 兼容程序（通过应用仅包含 LSB 兼容命令的脚本）,LSB 规范包含对软件包及其命名约定的描述。

注意

LSB 采用 Red Hat Package Manager 标准。基于 Debian 的 LSB 兼容发行版可以使用alien命令读取 RPM 包。

LSB 标准经常引用其他众所周知的标准,例如 ISO/IEC 9945-2009（基于便携式操作系统,非常类似于 Unix）。任何符合 LSB 的实现都需要提供文件系统层次结构标准 (FHS)中指定的文件系统层次结构的强制性部分,以及一些 LSB 特定要求。另请参阅有关FHS 标准的部分。

引导脚本环境和命令

最初,Linux 仅包含一组有限的服务,并且具有非常简单的引导环境。随着 Linux 的老化和发行版中服务数量的增加,启动脚本的数量也相应增加。一段时间后,出现了一套标准。 Init 脚本通常会包含一些其他脚本,这些脚本包含启动、停止和验证进程的功能。

LSB 标准列出了许多应该可用于运行级别脚本的函数。这些函数应该列在目录 /lib/lsb/init-functions 中的文件中,并且需要实现（至少)以下函数:

1. start_daemon [-f] [-n nicelevel] [-p pidfile] 路径名 [args...]

   将指定的程序作为守护进程运行。 start_daemon 函数将检查程序是否已经在运行。
   如果是这样,它将不会启动守护程序的另一个副本,除非给出 -f 选项。 -n 选项指定一个很好的级别。

2. killproc [-ppidfile] 路径名 [信号]

   将停止指定的程序,首先尝试使用指定的信号终止它。如果失败,将发送 SIGTERM 信号。如果程序已终止,如果终止的进程尚未完成,则应删除 pidfile

   所以。

3. pidofproc [-p pidfile] 路径名返回一个或多个特定

   守护程序的进程标识符,如路径名所指定。多个进程标识符由一个空格分隔。

在某些情况下,这些功能作为独立命令提供,脚本只是确保正确设置这些脚本的路径。通常还包括一些日志记录功能和显示状态行的功能。

更改和配置运行级别

在正在运行的机器上更改运行级别需要将在当前运行级别中运行的服务与需要在新运行级别中运行的服务进行比较。随后,很可能需要停止某些进程而需要启动其他进程。

回想一下,运行级别“X”的初始化脚本被分组在目录 /etc/rc.d/rcX.d 中（或者,在较新的（基于 LSB 的)系统中,在 /etc/init.d/rcX.d 中）。文件名决定了脚本的调用方式:如果名称以“K”开头,则脚本将以停止选项运行,如果名称以“S”开头,则脚本将以启动选项运行。运行级别更改期间的正常过程是先停止多余的进程,然后启动新进程。

实际的初始化脚本位于 /etc/init.d 中。您在 rcX.d 目录中找到的文件是链接到这些文件的符号链接。在许多情况下,启动脚本和停止脚本是指向同一脚本的符号链接。这意味着这样的初始化脚本应该至少能够处理启动和停止选项。

版权所有 2013-2017 Snow BV

Machine Translated by Google

例如，/etc/init.d/rc3.d 中名为 S06syslog 的符号链接可能指向脚本 /etc/init.d/syslog,就像在 /etc/init.d/rc2.d 中找到的符号链接一样,命名为 K17syslog。

服务停止或启动的顺序可能非常重要。有些服务可能会同时启动,有些则需要按严格的顺序启动。例如,您的网络需要在启动 httpd 之前启动。顺序由符号链接的名称决定。命名约定规定初始化脚本的名称（在 rcN.d 目录中找到的脚本)包括两个数字,紧跟在首字母之后。它们按字母顺序执行。

在早期,系统管理员手工创建这些链接。后来大多数 Linux 发行商决定提供 Linux 命令/脚本,允许管理员在某些运行级别禁用或启用某些脚本,并检查哪些系统（命令)将在哪个运行级别启动。这些命令通常将管理上述两个链接,并将以脚本以正确顺序运行的方式命名这些链接。

检查配置命令

另一个管理启动（初始化)脚本正确链接的工具是 chckconfig。在某些系统（例如 SuSE/Novell)上,它充当 insserv 的前端并使用 LSB 标准化注释块来维护其管理。在旧系统上,它维护自己的特殊注释部分,该部分具有更简单且不太灵活的语法。这个较旧的语法由两行组成,其中一行是服务的描述,它以关键字 description: 开头。另一行以关键字 chkconfig: 开头,并列出启动服务的运行级别和优先级（这决定了在更改运行级别时脚本将以什么顺序运行）。例如：

```
# foo 守护进程的初始化脚本#

# 描述:食物,foo 守护进程# chkconfig: 2345 55 25 #


#
```

这表示 foo 守护进程将在运行级别 2、3、4 和 5 中启动,在启动期间运行的初始化脚本队列中的优先级为 55,如果守护进程需要停止,则在运行的初始化脚本队列中的优先级为 25 .

chkconfig 实用程序可用于列出哪些服务将在哪些运行级别启动,将服务添加到运行级别或从运行级别删除服务,以及将整个服务添加到启动脚本或从启动脚本中删除。

笔记

我们在这里提供了一些示例,但请注意:周围有各种版本的chkconfig 。请先阅读您的发行版上chkconfig命令的手册页。

---

chkconfig 不会立即自动禁用或启用服务,而只是更改符号链接。如果 cron 守护进程正在运行并且您在运行级别 2 的基于 Red Hat 的系统上,则命令

```
# chkconfig --levels 2345 crond 关闭
```

会改变管理,但不会立即停止 cron 守护进程。还要注意,在 Red Hat 系统上,可以指定多个运行级别,就像我们在前面的示例中所做的那样。在 Novell/SuSE 系统上,您可以使用：

```
# chkconfig 食物 2345
```

并更改它,使其仅在运行级别 1 中运行,只需使用

```
# 检查配置食物 1
```

```
# 检查配置 --list
```

将列出服务的当前状态和它们处于活动状态的运行级别。例如,以下两行可能是输出的一部分：

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
xdm                    0:关 1:关 2:关 3:关 4:关 5:开                    6:关闭
xfs                    0:关闭 1:关闭 2:关闭 3:关闭 4:关闭 5:关闭 6:关闭
```

它们表明 xfs 服务没有在任何运行级别启动,只有在切换到运行级别 5 时才会启动 xdm 服务。

要添加一个新服务,比方说 foo 守护进程,我们创建一个新的初始化脚本并以服务命名它,在这种情况下我们可能会使用 food。该脚本连续放入 /etc/init.d 目录,之后我们需要在该脚本中插入正确的标头（旧的 chkconfig 标头,或较新的 LSB 兼容标头)然后运行

```
# chkconfig --添加食物
```

要从所有运行级别中删除 foo 服务,您可以键入：

```
# chkconfig --del 食物
```

请注意,食物脚本将保留在 /etc/init.d/ 目录中。

系统恢复（202.2）

目标

考生应该能够在引导过程和恢复模式期间正确操作 Linux 系统。此目标包括使用 init 实用程序和与 init 相关的内核选项。考生应该能够确定加载错误的原因和引导加载程序的使用。 GRUB 版本 2 和 GRUB Legacy 是感兴趣的引导加载程序。

关键知识领域

- GRUB 版本 2 和 Legacy
- 蛴螬壳
- 引导加载程序启动并移交给内核
- 内核加载
- 硬件初始化和设置
- 守护程序/服务初始化和设置
- 了解硬盘或可移动设备上不同的引导加载程序安装位置
- 覆盖标准引导加载程序选项并使用引导加载程序外壳
- 对 UEFI 的认识
- UEFI 和 NVMe 引导

条款和实用程序

- 山
- fsck
- inittab、telinit 和带有 SysV init 的 init
- /boot/ 和 /boot/grub/ 的内容

版权所有 2013-2017 Snow BV

Machine Translated by Google

- GRUB
- grub 安装
- initrd、initramfs
- 主引导记录

$资源： archNVMe,$ LPIC2sybex2nd;维基UEFI;架构UEFI； tomsUEFI； UEFI;卢特尔； Aplus901902；

GRUB 解释

GRUB（GRand Unified Bootloader 的缩写)加载操作系统内核并将执行控制权转移给它。

GRUB 有两个主要版本。当前版本称为 GRUB,但实际上是 GRUB 2。GRUB 是在 2011 年左右开发的。旧版本是在 1999 年开发的,现在称为 GRUB Legacy。 GRUB Legacy 仍在使用,但其开发已被冻结。除非另有说明,否则 GRUB 从这里开始暗示 GRUB 2。

蛴螬 2

GRUB 是一个模块化的引导加载程序,支持从 PC UEFI、PC BIOS 和其他平台引导。其模块化设计的优势在于,随着新的文件系统和/或存储解决方案被添加到内核中,引导支持可以很容易地以单独的模块添加到 GRUB 2 中。

此类模块添加的引导支持示例包括文件系统支持模块（如 ext4、NTFS、btrf 和 zfs）,以及 LVM 和软件 RAID 设备。

GRUB 能够引导许多操作系统,包括免费的和专有的。 GRUB 直接支持开放操作系统,如 FreeBSD、NetBSD、OpenBSD 和 Linux。使用 GRUB 的链加载功能支持专有内核（例如 DOS、Windows 和 OS/2）。链式加载意味着 GRUB 将用于引导系统,然后加载并运行专有系统引导加载程序,然后引导操作系统。

GRUB 引导过程具有菜单界面和命令行界面 (CLI)。调用的 CLI 称为 GRUB shell,它允许您执行命令以选择根设备（root 命令）,从中加载内核（linux 命令）,如果有必要,加载一些额外的内核模块（insmod)并随后引导内核（引导命令）。菜单界面提供了一种快速选择所需运行时环境的方法。启动时,两个界面都可用。启动时会显示菜单,用户只需选择其中一个菜单项即可。在没有用户交互的情况下,系统将在经过预定义的时间值后启动默认条目。

或者,用户可以在启动前按 e 编辑当前条目,或按 c 进入 CLI。一些 Linux 发行版在引导期间隐藏 GRUB 屏幕。在 BIOS/UEFI 初始化后立即按下 SHIFT 键将取消隐藏 GRUB 屏幕。

调用 GRUB shell 后,用户可以键入以下列表中的命令。命令列表可能会有所不同,具体取决于系统上存在哪些模块。帮助命令将生成可用命令列表。

o acpi: o	加载 ACPI 表
badram:	过滤掉 RAM 的坏区域
o 黑名单：	打印阻止列表
引导：	启动你的操作系统
猫：	显示一个文件的内容
o 链式装载机：	链式加载另一个引导加载程序
o cmp：	比较两个文件
o 配置文件： o cpuid：	加载配置文件 检查 CPU 特性
o 校验码：	计算 CRC32 校验和
o 日期：	显示或设置当前日期和时间
驱动图：	将一个驱动器映射到另一个
o 回声：	显示一行文字
o 导出： o 错误：	导出环境变量 什么都不做,不成功
o 获取文本：	翻译一个字符串

版权所有 2013-2017 Snow BV

Machine Translated by Google

o gptsync:o 暂停:	根据 GPT 条目填充 MBR
	关闭计算机
o 帮助:	显示帮助信息
o initrd:	加载 Linux initrd
o initrd16:	加载 Linux initrd（16 位模式）
o insmod:	插入模块
o 密钥状态: o	检查键修饰符状态
linux:	加载 Linux 内核
Ø Linux16:	加载 Linux 内核（16 位模式）
Ø list_env:	列出环境块中的变量
o 加载环境:	从环境块加载变量
o 环回: o ls:	从文件系统映像制作设备
	列出设备或文件
正常:	进入正常模式
o normal_exit:o parttool:	退出正常模式
o 密码:o password_pbkdf2:	修改分区表项
o 播放:o pxe_unload:o	设置明文密码
阅读:	设置哈希密码
	演奏曲子
	卸载 PXE 环境
	读取用户输入
重启:	重新启动计算机
o save_env: o 搜索:	将变量保存到环境块
	按文件、标签或 UUID 搜索设备
o 发送密钥: o	模拟击键
设置:	设置环境变量
o 是的:	什么都不做,成功
o 取消设置:	取消设置环境变量
o 上层内存:	设置上位内存大小

GRUB 使用它自己的语法来描述硬盘。设备名称需要用括号括起来,例如

```
(fd0)
```

表示软盘,和

```
(高清0,1)
```

表示第一个硬盘上的第一个分区。请注意,虽然磁盘编号从零开始,但分区编号从一开始,因此最后一个示例引用第一个磁盘和第一个分区。

GRUB 使用计算机 BIOS 找出可用的硬盘驱动器。但它不能总是弄清楚 Linux 设备文件名和 BIOS 驱动器之间的关系。可以创建特殊文件 /boot/grub/device.map 来映射这些,例如:

```
(fd0) /dev/fd0 (hd0) /
dev/hda
```

请注意,当您使用软件 RAID-1（镜像)时,您需要在两个磁盘上设置 GRUB。启动时,系统还不能使用软件 RAID 系统,因此只能从一个磁盘启动。如果只在第一个磁盘上设置 GRUB,那么该磁盘就会损坏,系统将无法启动。

GRUB 配置文件

GRUB 2 的配置文件是 /boot/grub/grub.cfg。 GRUB 配置文件是用类似 shell 的脚本语言编写的,带有条件语句和函数。

不建议直接修改grub.cfg;每当使用发行版的包管理器添加、更新或删除内核时,或者当用户运行 update-grub 脚本时,配置文件都会更新。 update-grub 是 grub-mkconfig 的包装器,指定 grub.cfg 作为其输出文件。 grub-mkconfig 的行为由目录 /etc/grub.d 中的文件和 /etc/default/grub 文件中的关键字控制。

版权所有 2013-2017 Snow BV

Machine Translated by Google

示例关键字:要启动的默认菜单项 (GRUB_DEFAULT) 或显示菜单后启动默认菜单项的超时秒数 (GRUB_TIMEOUT)。

/etc/grub.d/30_os_prober 脚本会自动检测操作系统,包括 Windows 等外国操作系统。用户可以修改自定义文件（默认为 40_custom)以创建自定义条目。

GRUB 2 菜单项以 menuentry 关键字开头。菜单项的标题可以在菜单项行的引号内找到。 menuentry 行以左大括号 ({) 结尾。菜单项以右花括号 (}) 结尾。

一个非常简单的例子:

```
menuentry    Linux 3.3.10    {
<...>
}
```

与 GRUB Legacy 的区别

乍一看,这两个版本差别不大。但是,有一些明显的区别:

- GRUB 配置文件现在称为/boot/grub/menu.list,而基于Red Hat 的发行版更喜欢/boot/grub/grub.conf 文件名。除了名称略有不同外,配置文件的语法也有所不同。 grub.cfg 文件现在在 grub-install 期间生成,不应手动编辑。

- 核心GRUB 引擎更小且平台依赖性更低。现在在单独的模块中提供对许多不同文件系统和平台的支持。因此,所使用的平台和文件系统决定了引导顺序期间加载的模块。相比之下,GRUB Legacy 具有固定的引导顺序,关键组件被硬编码,因此灵活性较低。

- GRUB 2 中的分区编号从 1 开始,而不是 0。磁盘仍然从 0 开始编号。这可能有点令人困惑。
- GRUB 2 内核规范是通过 linux 命令完成的,而在 GRUB Legacy 中,我们使用 kernel 命令代替。
- 可以使用set root 代替root 命令来选择root 设备。也可以从搜索中设置根设备
  可以通过磁盘标签或 UUID 查找设备的命令。
- GRUB 2 使用insmod 加载模块。在 GRUB Legacy 中,模块使用 module 或 modulenounzip 加载。

GRUB 遗产

菜单项的 GRUB Legacy 定义存储在 /boot/grub/menu.lst 中。在某些系统上,您可能会在 /etc 或 /boot/grub 目录中找到一个链接。因为 GRUB 直接访问该文件,所以 grub.conf 将立[29] 该文件中的任何更改
即影响引导加载程序。

在具有 Legacy 引导加载程序的系统上,GRUB shell 可用于安装和模拟它。此 shell 模拟引导加载程序,可用于安装引导加载程序。它还可以方便地检查您当前的设置并对其进行修改。要启动它（以 root 身份）,只需键入 grub。在以下示例中,我们显示帮助屏幕:

```
# grub grub>
帮助黑名单文件
                                                                                开机
目录文件                                                                        chainloader [--force] FILE 配置文件 FILE displayapm
颜色 NORMAL [HIGHLIGHT] device DRIVE DEVICE                                     find FILENAME

displaymem
geometry DRIVE [CYLINDER HEAD SECTOR [ halt [--no-apm] help [--all] [PATTERN ...] initrd FILE [ARG ...]
makeactive                                                                      隐藏分区
                                                                                内核 [--no-mem-option] [--type=TYPE] 映射 TO_DRIVE FROM_DRIVE
```

[29] 不要与 GRUB 2 grub.cfg 配置文件混淆。

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
md5crypt                                                    模块 FILE [ARG ...] 寻呼机 [FLAG] parttype
modulenounzip FILE [ARG ...] partnew PART TYPE              PART TYPE 重启
START LEN quit root [DEVICE [HDBIAS]]

                                                            rootnoverify [DEVICE [HDBIAS]] serial [--unit=UNIT] [--
port=PORT] [--setkey [TO_KEY FROM_KEY] setup [--prefix=DIR] [--stage2=STAGE2_ terminal [--dumb] [--超时=秒数] [--

测试模式                                                    取消隐藏分区
上层内存 KBYTES                                             vbe 探头 [模式]

蛴螬>_
```

注意

请注意,grub shell 不可用于 GRUB 2。相反,您可以安装 Grub 仿真器grub-emu。

其他 GRUB Legacy 命令包括 blocklist 命令,可用于找出文件存储在哪个磁盘块上,或 geometry 命令,可用于找出磁盘几何结构。您可以使用 partnew 命令创建新的（主)分区,使用 initrd 命令加载 initrd 映像,等等。 GRUB 文档中描述了所有选项。 GRUB 是 GNU 软件库的一部分,因此使用信息系统进行记录。在大多数系统上,也有一个有限的手册页可用。

初始引导过程,在引导时,BIOS 访问硬盘的初始扇区,即所谓的 MBR（主引导记录）,加载在内存中找到的数据并将执行转移到它。如果使用 GRUB,MBR 包含 GRUB 第一阶段的副本,它会尝试加载阶段 2。

为了能够加载阶段 2,GRUB 需要访问代码来处理文件系统。有许多文件系统类型,处理它们的代码将无法容纳在 512 字节的 MBR 中,更不用说因为 MBR 还包含分区表。
因此,处理文件系统的 GRUB 部分存储在所谓的 DOS 兼容区域中。该区域由 MBR 所在的同一柱面（柱面 0)上的扇区组成。在过去,当使用 CHS（柱面/磁头/扇区)规范寻址磁盘时,MBR 通常会加载 DOS。 DOS 要求它的映像在同一个柱面上。

因此,按照传统,磁盘上的第一个柱面是保留的,GRUB 使用这个空间来存储文件系统代码。
该部分称为第 1.5 阶段。 Stage 1.5 通常称为 core.img;它是由安装程序根据 grub 在引导期间需要支持的文件系统从几个文件构建的。

第 2 阶段包含大部分引导逻辑。它向最终用户提供一个菜单和一个额外的命令提示符,用户可以在其中手动指定引导参数。 GRUB 通常配置为在超时后自动加载特定内核。
一旦最终用户做出选择,GRUB 就会将选定的内核加载到内存中并将控制权传递给内核。在此阶段,如果操作系统需要,GRUB 可以使用链加载将引导过程的控制权传递给另一个加载程序。

在 Linux 中,grub-install 命令用于将阶段 1 安装到 MBR 或分区内。

影响正常启动过程

常规引导过程是（重新)引导系统时通常发生的过程。此过程可能会受到 GRUB 提示符的影响。可以影响的内容将在以下部分讨论,但首先我们必须激活提示。

选择另一个内核

如果您刚刚编译了一个新内核并且在使用新内核时遇到困难,那么您很可能想要恢复到旧内核。

对于 GRUB,一旦您看到引导屏幕,请使用光标键选择您要引导的内核,然后按 Enter 键引导它。

版权所有 2013-2017 Snow BV

Machine Translated by Google

引导至单用户模式或特定运行级别

例如,如果您安装了无法正常运行的图形环境,这将很有用。您要么根本看不到任何东西,要么系统没有达到有限状态,因为它一直试图一遍又一遍地启动 X。

引导到单用户模式或图形环境未运行的另一个运行级别将使您能够访问系统,以便您可以更正问题。

要在 GRUB 中引导到单用户模式,请将光标指向您要引导的内核条目并按 e。然后在 GRUB Legacy 中选择以“linux”(对于 GRUB 2)或“kernel”开头的行。转到行尾,添加“single”。之后,按 Enter 退出编辑模式,然后按 CTRL-x (GRUB 2),或 GRUB Legacy 中的 b 退出编辑器并启动该条目。

切换运行级别

在 Linux 中可以切换到与当前活动级别不同的运行级别。这是通过 telinit 命令完成的。
它的语法很简单： telinit [OPTION] RUNLEVEL 其中 RUNLEVEL 是运行级别的编号。

telinit 支持的唯一选项是-e KEY=VALUE。它用于指定要与 RUNLEVEL 和 PREVLEVEL 一起包含在事件中的附加环境变量。通常你不会使用这个选项。

你会发现你使用 telinit 主要是为了切换到单用户模式（运行级别 1），例如能够卸载文件系统并 fsck 它。在这种情况下,您可以使用：

```
# telinit 1
```

请注意,大多数系统上的 telinit 是指向 init 命令的符号链接。

使用命令 /sbin/init q 强制 init 重新加载 /etc/inittab。

向内核传递参数

如果设备不工作

一个可能的原因可能是内核中的设备驱动程序必须被告知使用另一个 irq 和/或另一个 I/O 端口。这仅适用于已将对设备的支持编译到内核中的情况,而不适用于使用可加载模块的情况。

作为一个例子,让我们假设我们有一个系统有两个相同的以太网卡,支持被编译到内核中。默认情况下只会检测到一张卡,因此我们需要告诉内核中的驱动程序检测两张卡。假设第一张卡成为 eth0,地址为 0x300,irq 为 5,第二张卡成为 eth1,irq 为 11,地址为 0x340。对于 GRUB,您可以像引导进入单用户模式一样添加附加项,将关键字“single”替换为您需要传递的参数。

对于上面的例子,传递给内核的关键字是：

```
以太=5,0x300,eth0 以太=11,0x340,eth1
```

救援启动过程

当 fsck 启动但失败时

在启动期间检查文件系统。在 Debian 系统上,这是由 /etc/rcS.d/S30check.fs 完成的。根据 /etc/fstab 的内容检查所有文件系统。

如果命令 fsck 返回大于 1 的退出状态,则该命令失败。退出状态是以下一种或多种情况的结果：

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
0          - 没有错误
1          - 更正了文件系统错误
2          - 系统应该重新启动
4          - 文件系统错误未更正
8          - 操作错误
16 - 用法或语法错误
128 - 共享库错误
```

如果命令失败,您将收到一条消息:

```
fsck 失败。请手动修复

 “CONTROL-D”将从这个 shell 中退出并且
继续系统启动。
```

如果你不按 Ctrl-D 但输入 root 密码,你会得到一个 shell,实际上 /sbin/sulogin 被启动,如果 root 文件系统以只读方式挂载,你应该能够运行 fsck 并解决问题.

或者（请参阅下一节)您可以从引导媒体引导。

如果您的根 (/) 文件系统已损坏

使用发行版的引导媒体

许多发行版都附带一张或多张 CD 或可以放在 USB 记忆棒上的引导映像。其中一张 CD 通常包含一个“救援”选项,用于在核心中引导 Linux。这使您可以修复问题。

请记住在 BIOS 中将启动顺序设置为先从 CD-ROM 或 U 盘启动,然后再从硬盘启动。如果是 U 盘,可能还需要在 bios 中启用“USB Legacy Support”。

救援模式需要特定于分发。但它应该允许您打开具有 root 权限的 shell。您可以在已卸载的损坏文件系统上运行 fsck。

假设您的根分区是 /dev/sda2。然后,您可以通过键入 fsck -y /dev/sda2 对根文件系统运行文件系统检查。 “-y”标志可防止 fsck 询问您必须回答的问题（这可能导致大量输入)并导致 fsck 使用“是”作为所有问题的答案。

虽然救援映像的根 (/) 文件系统完全位于 RAM 中,但您可以将文件系统从硬盘挂载到 RAM 中现有的挂载点,例如 /target。或者,您可以先创建一个目录,然后在其中挂载一个硬盘分区。

更正错误后,不要忘记在重新引导系统之前卸载已挂载的文件系统,否则在引导期间您将收到一条消息,指出一个或多个文件系统尚未完全卸载,并且 fsck 将尝试再次修复它。

UEFI 和 NVMe 引导注意事项

几十年来,系统 BIOS（基本输入输出系统)负责引导过程中的硬件和软件初始化。早期的 BIOS 版本需要手动配置主板上的物理跳线。后来的版本用软件菜单取代了手动跳线例程,能够提供一个界面来配置最基本的计算机设置。这听起来很方便,但随着计算机系统的不断发展,即使是最复杂的 BIOS 软件也被证明有其局限性。为了克服这些限制,英特尔于 1998 年开发了 EFI（可扩展固件接口)系统作为 BIOS 的替代品。 EFI 系统直到 2005 年左右被 UEFI 论坛采纳后才流行起来。然后该标准从 EFI（重新)命名为 UEFI（通用可扩展固件接口）。

UEFI 有时也称为 (U)EFI。 Linux 内核 3.15 及更新版本应该能够利用 UEFI 的优势。

您可能会问这些优势是什么?要回答这个问题,我们必须首先查看 BIOS 限制。引导操作系统时,BIOS 系统的局限性之一很明显。传统上,可以将 BIOS 配置为按特定顺序使用一个或多个引导设备。引导设备可以是光驱、硬盘、便携式 USB 卷或网络接口卡。 BIOS 执行 POST（开机自检)后,将检查每个配置的引导设备是否存在引导加载程序。将加载检测到的第一个引导加载程序。如果是硬盘驱动器,BIOS 需要引导加载程序

版权所有 2013-2017 Snow BV

Machine Translated by Google

位于扇区 0 或 MBR（主引导记录）。由于 MBR 只允许存储少量数据（446 字节），因此 MBR 通常包含指向磁盘上另一段代码的指令。这种两阶段方法称为链加载。这另一段代码可以包含一个启动管理器。引导管理器能够加载位于存储卷上不同位置的操作系统。引导代码的第一阶段和第二阶段都必须存储在硬盘驱动器可用存储空间的第一个兆字节内。

UEFI 使用不同的方法。 UEFI 不局限于某个特定驱动器的 MBR 内容，而是从 ESP 分区读取引导数据。 ESP 代表 EFI 系统分区。 ESP 是指定的引导分区。文件系统通常是 FAT 类型，它可以容纳任何大小的引导加载程序，甚至多个引导加载程序。在 Linux 系统上，ESP 通常挂载为 /boot/efi。该挂载点下方将是一个目录结构，该目录结构取决于特定的操作系统。位于这些目录中的引导文件内容可能因所使用的操作系统而异。使用 UEFI，UEFI 软件充当迷你引导加载程序，在预定义

```
# cd /boot/efi/
# ls -a
.        ..        电喷
# cd EFI
#ls
开机软呢帽
# ls -l 启动
总计 1332
-rw-r--r-- 1 根 root 1293304 2016 年 5 月 17 日 BOOTX64.EFI -rw-r--r-- 1 根 root 66072 2016 年 5 月 17 日
fallback.efi # ls -l fedora/

总计 3852
-rw-r--r-- 1根根                                    104 May 17 2016 BOOT.CSV 4096 Sep 28 22:17
drwxr-xr-x 2根根                                 fw -rwxr-xr-x 1 root root 70864 Sep 28 22:17
fwupx64.efi -rw-r--r-- 1 root root 1276192 2016 年 5 月 17 日 MokManager。 efi -rw-r--r-- 1 root root 1293304
2016 年 5 月 17 日 shim.efi -rw-r--r-- 1 root root 1287000 2016 年 5 月 17 日 shim-fedora.efi
```

在上面的示例中，每个以 .efi 结尾的文件都可以向 UEFI 系统添加功能。因此，基于 BIOS 的系统依赖硬盘元数据来启动系统，而基于 UEFI 的系统能够读取硬盘 ESP 部分内的文件。 UEFI 提供对传统 BIOS 功能的向后兼容性，同时为现代计算机提供更高级的功能。使用 BIOS 软件的计算机在处理当今的 8TB 硬盘驱动器时遇到了麻烦。基于 UEFI 的计算机能够使用 GPT 磁盘布局，突破其 BIOS 对应项的 2TB 分区限制。 UEFI 软件带有对 IPv4 和 IPv6 的网络支持。支持 TCP 和 UDP，并且支持使用 TFTP 甚至 HTTP 启动远程启动媒体。通过 HTTP 引导确实需要 UEFI 2.5 或更新版本。 2.5 版于 2016 年 1 月发布。

LPIC-2 考生应了解在现代计算机上可以在 UEFI 和 Legacy BIOS 启动模式之间切换的可能性。尽管 UEFI 可能具有优势，但也应满足一些要求。 .efi 引导文件应位于特定路径下。启用安全启动时，必须对启动代码进行数字签名。否则，系统可能会遇到启动问题。在对现代 Linux 计算机上的启动问题进行故障排除时，请尝试区分 MBR 和 GPT 磁盘布局。使用 UEFI 启动模式时，确认使用的 Linux 发行版也可以处理 UEFI 启动。

启用安全启动后，确认满足所需条件。如有疑问，请切换回 UEFI 界面中的“Legacy BIOS”或等效项。从 USB 启动时，可能需要在 UEFI 界面中为大容量存储设备启用“传统 USB”设置。

NVMe 在上一章中，提到了 8TB 硬盘驱动器作为最近计算机存储发展的结果。这些传统的 SATA（串行高级技术附件）硬盘驱动器具有活动部件，并使用称为 AHCI（高级主机配置接口）的协议进行控制。近年来，SSD（固态硬盘）硬盘越来越受欢迎。

这些驱动器的优点之一是没有移动部件。这使得 SSD 硬盘不仅比机械硬盘更节能，而且速度更快。由于 SSD 驱动器必须与现有计算机兼容，因此它们与机械硬盘驱动器使用的相同 SATA 连接器连接。他们也使用相同的 AHCI 协议。该协议最初是为机械硬盘设计的。 AHCI 使用 1 个队列和 32 个命令来控制硬盘驱动器。这对新一代 SSD 硬盘驱动器构成了瓶颈。为了解决这个瓶颈，一种称为 NVMe（非易失性存储器高速）的新技术已经开发出来。 NVMe 允许 SSD 硬盘驱动器连接到 NVMe 控制器，该控制器连接到主板上的 PCI-E 总线。然后使用 NVMHCI（非易失性内存主机）控制 SSD 硬盘

版权所有 2013-2017 Snow BV

Machine Translated by Google

配置接口)协议。现在可以使用 65.000 个队列来控制 SSD,而不是 1 个队列一次包含 32 个命令,每个队列最多包含 65.000 个命令。这是可能的,因为 PCI-E 总线比 SATA 总线快得多。
与 PCI-E 连接的 AHCI 硬盘驱动器相比,使用 NVMe 的最新一代高速 SSD 硬盘驱动器的吞吐速度最高可提高 7 倍。

正如连接到 Linux 计算机的传统硬盘驱动器由 /dev/hda* 或 /dev/sda* 引用表示,NVMe 硬盘驱动器在 Linux 文件系统树中由 /dev/nvme* 表示。使用这些硬盘驱动器时,请注意磁盘符号从 0 开始,但磁盘上的命名空间和分区从 1 开始。因此,系统的第一个 NVMe 硬盘驱动器上第一个命名空间的第一个分区由 /dev/ 表示nvme0n1p1。更多关于在 204.2 引导的 UEFI 和 NVMe

备用引导加载程序 (202.3)

考生应了解其他引导加载程序及其主要功能。

关键知识领域

- SYSLINUX、ISOLINUX、PXELINUX
- 了解 BIOS 和 UEFI 的 PXE
- 了解 systemd-boot 和 U-Boot
- 引导 UEFI 系统
- 系统启动
- U-启动

条款和实用程序

- 操作系统
- 外部操作系统
- isolinux.bin
- isolinux.cfg
- isohdpfx.bin
- efiboot.img
- pxelinux.0
- pxelinux.cfg/
- uefi/shim.efi
- uefi/grubx64.efi
- UEFI
- 系统启动
- U-boot

版权所有 2013-2017 Snow BV

Machine Translated by Google

利洛

lilo 引导程序由两个阶段组成。在两阶段引导期间,LILO 通过将单词“LILO”中的连续字母打印到 BIOS 控制台来指示其进度,每个阶段的开始和结束各有一个字母。

任一阶段的错误都会导致只打印部分单词,可选地后跟数字 BIOS 错误代码或单个字符。例如,LIL- 或 LIL?。

在交互模式下（/etc/lilo.conf 中的 prompt 关键字)LILO 向用户提供最多 16 个条目的选择。如果没有指定 timeout=nnn（几十秒后的 nnn），bootloader 将无限期地等待。如果超时到期,或者没有包含任何提示,LILO 继续加载列出的第一个图像。这可以用 default=name 关键字否决。

/etc/lilo.conf 和 /sbin/lilo

除非是现代引导加载程序,否则 LILO 的两个阶段实际上都不知道 /etc/lilo.conf 的内容。相反,该文件仅用作 lilo 安装程序的规范,通常位于 /sbin/lilo。如果没有其他选项，/sbin/lilo 会将引导加载程序安装到 MBR 中,并处理 /etc/lilo.conf 的内容,创建指定要存储到 /boot/map 中的任何文件的映射。

关于 /etc/lilo.conf 的详细描述,请在线参考 lilo.conf(5) 手册页。

示例 /etc/lilo.conf（来自旧的 debian 发行版）

```
# lilo.conf # # 全局选
项:boot=/dev/hda 提
示timeout=150



lba32
compact
vga=normal root=/
dev/hda1 read-only menu-
title=  John  s Computer                                    “
#

# 可引导内核映像:image=/boot/zImage-1.5.99

        标签=尝试图像=/
boot/zImage-1.0.9
        标签=1.0.9 图片=/
tamu/vmlinuz 标签=tamu

        initrd=initramdisk.img root=/dev/hdb2
        vga=ask

#
# 其他操作系统:other=/dev/hda3

        标签=dos
        启动为=0x80                 # 必须是 C:
其他=/dev/hdb1 标签=Win98

        启动为=0x80                 # 必须是 C:
其他=/dev/hdb5 标签=os2

        loader=os2_d table=E:
        # os2 视为 E:
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

SYSLINUX、ISOLINUX、PXELINUX:Syslinux 项目

SYSLINUX 是一个 linux 引导加载程序,旨在从 MS-DOS/Windows FAT 文件系统运行。它仅限于 Intel/AMD 硬件。

随着时间的推移,Syslinux 项目(http://syslinux.org)扩展到包括对从 CD-ROM (ISOLINUX)、linux 文件系统 (EXTLINUX) 和 PXE (PXELINUX) 本地启动的支持。

此摘要处理 LPIC-2 的特定目标。可以在位于http://www.syslinux.org/-wiki/index.php/SYSLINUX的 syslinux wiki 中找到完整的描述

系统Linux

SYSLINUX 是一个 Intel Linux 引导加载程序,能够从基于 Windows/MS-DOS FAT 的文件系统引导。它可以使用同名命令从 Windows、MS-DOS 或 linux 安装。

syslinux 系统由实际的引导加载程序及其安装程序组成。引导加载程序本身是用汇编程序编写的 32 位本机二进制文件。

syslinux 安装程序命令有不同的版本:适用于 Windows 的 syslinux.exe,适用于 linux 的 syslinux。甚至还有一个用于基于 DOS 的系统的 syslinux.com。

SYSLINUX 安装程序选项

- 抵消	-t	文件系统在设备上的偏移量
--目录--安装	-d	安装目标目录
	-一世	在当前引导扇区上安装
--update --	-U	更新以前的安装
sectors=# --	-S	强制每个磁道的扇区数
heads=# --stupid	-H	强制头数
--raid	-s	缓慢、安全和愚蠢的模式
	-r	启动失败时回退到下一个设备
--一次=...		开机执行一次命令
--清除一次	-O	清除 boot-once 命令
--reset-adv		重置辅助数据
--菜单保存=	-M	设置标签以在下次启动时选择为默认值
- 力量	-F	忽视注意事项

MSDOS 特定选项

-m	MBR:将可引导的 MBR 扇区安装到驱动器的开头。
-一个	活动:标记使用活动的分区（=可启动）

仅限 Linux

-t	(-o 在旧系统上)指定文件中文件系统映像的字节偏移量。它必须与磁盘映像文件一起使用。

ISOLINUX、EXTLINUX、PXELINUX 的例外情况

ISOLINUX 安装程序 当 syslinux 期望目标设备编写引导加载程序时,isolinux 安装程序从目录结构生成 ISO 映像。该目录必须包含一个子目录 isolinux,该子目录又必须包含实际的 isolinux.bin 引导加载程序。

EXTLINUX 安装程序 extlinux 安装程序需要一个已安装的文件系统来安装引导加载程序。

PXELINUX 在第2.3.6 节中进行了处理。

版权所有 2013-2017 Snow BV

Machine Translated by Google

Syslinux 引导配置

这些实用程序安装的引导加载程序将在以下三个目录中查找 syslinux.cfg 文件：/boot/syslinux、/syslinux、/（root）。

ISOLINUX 引导加载程序将首先查找 /boot/isolinux 和 /isolinux。 EXTLINUX 引导加载程序首先查找 /boot/extlinux 和 /extlinux。

找到配置文件的目录将是引导过程中其他路径名的默认目录。

CONFIG 关键字将使用新的配置文件重新启动引导过程。如果提供了两个路径名，则第二个参数会覆盖默认目录。

引导过程将在 syslinux.cfg 文件中查找带有“LABEL linux”的行。找到后，它将使用任何后续关键字来指导引导过程。 （默认标签短语可用于覆盖“linux”标签。）

引导配置中的典型关键字：

KERNEL image KERNEL 关键字指定一个图像文件。这不一定是实际的内核映像，但可以是下一阶段引导程序的名称。 SYSLINUX ea 依靠文件扩展名来决定文件格式。

.0 与 PXELINUX 一起用于 PXE NBP（网络引导程序），默认情况下 pxelinux.0。 .bin与ISOLINUX一起使用，指CD引导扇区，

.bs 或 .bss 指的是（修补过的)DOS 引导扇区
[SYSLINUX]。 .com、.cbt 和 .c32 是 COMBOOT 映像（DOS、非 DOS、32 位）。对于版本 5.00 和更高版本，c32 从 COMBOOT 更改为 ELF 二进制格式。
.img 是一个 ISOLINUX 磁盘映像。

任何其他文件扩展名（或根本没有)表示 linux 内核映像。

也可以通过使用 KERNEL 关键字别名之一来强制文件类型：LINUX、BOOT、BSS、PXE、FDIMAGE、COMBOOT 或 COM32。

APPEND string APPEND 关键字指定附加到内核命令行的引导参数字符串。
只会应用最后一个 APPEND 行。

SYSAPPEND SYSAPPEND 关键字需要一个被解释为位图的数字参数。此位图中的每一位
将向内核命令行添加一个特定的自动生成的字符串。

示例 2.1 SYSAPPEND 示例 1:添加一个带有网络
信息的字符串:ip=client:bootserver:gw:netmask 2:添加 BOOTIF= ...,通过其 mac 地址识别活动网 络接口。 4:添加字符串SYSUUID=... 8: 添加 CPU=... 高阶位（0x00010 到 0x10000)控制来自 DMI/SMBIOS 的附加字符串（如果可用）。完整列表可以在 Syslinux 维基中找到： http://www.syslinux.org/wiki/index.php/SYSLINUX”

INITRD 文件名 INITRD 关键字等同于 APPEND initrd=filename。

TIMEOUT num 对于交互式使用，TIMEOUT 的参数表示 SYSLINUX 应该等待控制台或串行端口上的输入的秒数。

PXELINUX 操作系统

PXELINUX 引导加载程序用作 PXE 网络引导的第二阶段。 PXE 网络引导机制在第2.3.7 节中进一步解释。

PXELINUX 需要一个标准的 TFTP 服务器，其 /tftpboot 目录包含 pxelinux.0 syslinux 引导程序和 ldlinux.c32 库模块。

此外，目录 /tftpboot/pxelinux.cfg 必须存在以获取其他配置详细信息。

PXE TFTP 引导服务器可以为许多不同的客户端提供服务，并且需要一种方法来为不同（类别)的客户端维护不同的配置文件。可以通过多种不同的方式来指定配置文件的名称。

版权所有 2013-2017 Snow BV

Machine Translated by Google

组合 DHCP 选项 209 和 210 选项 209 (pxelinux.config-file) 指定配置文件的文件名。

选项 210 (pxelinux.pathprefix) 指定 TFTP 服务器命名空间上的搜索路径（目录前缀）（以文件系统的特定于操作系统的分隔符结尾）。

硬编码在 pxelinux.0 映像中。 pxelinux-options 命令可用于对选项进行硬编码,如图所示
当然2.1。

```
      （域名服务器），（域名），（下一个服务器），
15    （配置文件），（路径前缀），（重启时间）。
54
209
210
211
```

图 2.1:pxelinux.0 嵌入式选项（可选）

选项可以指定为“before-options”,其中 DHCP 具有优先权,或者指定为“after-options”,它覆盖 DHCP。

源自 UUID、MAC 地址或 IP 地址 如果未指定配置文件,则文件名源自变量列表
能力。将使用 TFTP 服务器上实际存在的列表中的第一个文件。
变量列表是:

· 客户端的UUID,小写。 · 客户端的MAC 地

址,小写十六进制,字节之间用破折号（“-”)分隔。 · 客户端ipv4 地址的大写十六进制表示的最长可能前缀。每次字符串

不匹配时,PXELINUX 都会删除字符串中的最后一个字符并重试,只要结果至少包含一个字符即可。

· 作为最后的手段,PXELINUX 将尝试检索名为“default”的文件。

了解 PXE

PXE 是 Intel 制定的规范,用于增强原有的网络启动协议:BOOTP、TFTP 和 DHCP。

BOOTP、RARP 和 TFTP 由 IETF 创建,使系统能够从服务器自动检索其网络配置和初始引导加载程序。

最初的 BOOTP 标准仅限于客户端和服务器可以交换信息的一些固定字段。客户端可以在 chaddr 中提供其硬件地址,并请求特定类型的文件,并将接收其 ip 地址作为 yiaddr 和服务器名 sname。结合服务器 IP 地址字段 (siaddr) 和网关 IP 地址字段,以及返回的引导文件名 (file),这将告诉引导客户端使用 TFTP 从何处检索其引导映像。

BOOTP字段

CIADDR 4		客户端IP地址
亚地址 4 亚地址 4		你的IP地址 服务器的IP地址
giaddr 4 chaddr 16		网关 IP 地址 客户端硬件地址
名字 64		可选的服务器主机名,空终止字符串。引导文件名,空终止字符串； n=64 在原始的 BOOTP 中,以 4 字节的
文件	128	DHCP“幻数”开始。
鬻	n	

随着时间的推移,网络和 IT 基础设施变得越来越复杂,要求也越来越苛刻。为了允许客户端提供更多关于他们自己的信息并检索定制信息,BOOTP 收到了 BOOTP 供应商信息扩展 [RFC 1048],它又通过新协议 DHCP 得到增强。 DHCP 使用许多标准选项扩展 BOOTP,定义不同类型的消息。某些 DHCP 选项可能与标准 BOOTP 字段重叠,在这种情况下应包含相同的值。

版权所有 2013-2017 Snow BV

Machine Translated by Google

注意

DHCP 消息是一个 BOOTP 数据包（请求或响应），在 BOOTP“供应商信息字段”中有一个特殊的 4 字节值（DHCP 魔术饼干）。接下来是 DHCP 选项，由单字节选项类型、长度字段和选项内容的长度字节组成。

此规则有两个例外：填充(0)和选项结束(255)的长度仅为一个字节，并且缺少长度字段。

最后，Intel 引入了 PXE，以进一步增强 BOOTP/DHCP 协议，试图标准化客户端识别自己的方式。这允许引导客户端和服务器在决定正确的参数和开始运行所需的引导程序之前，将需要交换的数据包数量降至最低。

PXE 引导请求以 DHCP Discover 消息开始，包括至少五个选项，其中三个是特定于 PXE 的：

```
(53) DHCP 消息类型（DHCP 发现），
(55) 参数请求列表，
(93) 客户端系统架构，
(94) 客户端网络设备接口，
(97) 基于 UUID/GUID 的客户端标识符
```

PXE 规范中定义了选项 93、94 和 97。此外，选项 55，即参数请求列表，必须*还*请求选项 128 到 135，即使不需要服务器对它们提供响应。此列表和上面列出的三个选项用于将客户端标识为 PXE 感知。

PXE 的代理 DHCP

并非每个 DHCP 服务器（尤其是嵌入在网络设备中的服务器)都能够处理 PXE 请求。

PXE 规范允许 PXE 感知的 DHCP 服务器与简单的 DHCP 服务器共存，其中默认的 DHCP 服务器提供基本的网络详细信息。然后，PXE 感知服务器可以为实际的 TFTP 引导过程提供更多详细信息。
这称为代理 DHCP。

甚至可以在同一台服务器上分离 DHCP 服务，其中代理 DHCP 服务期望监听 UDP 端口 4011。

示例 DHCP 请求

请参阅下面的示例 DHCP 发现消息，包括对标准网络详细信息的请求，例如 (1) 子网掩码，(3)
路由器、(6) 名称服务器、(12) 主机名、(15) 域名等。

示例 2.2 DHCP 发现消息

```
选项:(53) DHCP 消息类型
        长度:1
        DHCP:发现 (1)
选项:(57) 最大 DHCP 消息大小
        长度:2
        最大 DHCP 消息大小:1464
选项:(55) 参数请求列表
        长度:35
        参数请求列表项：(1)子网掩码
        参数请求列表项：(2)时间偏移
        参数请求列表项：(3)路由器
        参数请求列表项：(4)时间服务器
        参数请求列表项：(5)名称服务器
        参数请求列表项：(6)域名服务器
        参数请求列表项：(12)主机名
        参数请求列表项:(13) 引导文件大小
        参数请求列表项：(15)域名
        参数请求列表项：(17)根路径
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
        参数请求列表项：(18)扩展路径
        参数请求列表项:(22) 最大数据报重组大小
        参数请求列表项:(23) 默认 IP 生存时间
        参数请求列表项：(28)广播地址
        参数请求列表项：(40)网络信息服务域
        参数请求列表项：(41)网络信息服务服务器
        参数请求列表项:(42) 网络时间协议服务器
        参数请求列表项目:(43) 供应商特定信息
        参数请求列表项：(50)请求的IP地址
        参数请求列表项：(51)IP地址租用时间
        参数请求列表项:(54) DHCP 服务器标识符
        参数请求列表项：(58)更新时间值
        参数请求列表项:(59) 重新绑定时间值
        参数请求列表项:(60) 供应商类标识符
        参数请求列表项:(66) TFTP 服务器名称
        参数请求列表项:(67) Bootfile 名称
        参数请求列表项:(97) UUID/GUID-based Client Identifier
        参数请求列表项:(128) DOCSIS 全安全服务器 IP [TODO]
        参数请求列表项:(129) PXE - 未定义（供应商特定）
        参数请求列表项:(130) PXE - 未定义（供应商特定）
        参数请求列表项:(131) PXE - 未定义（供应商特定）
        参数请求列表项:(132) PXE - 未定义（供应商特定）
        参数请求列表项:(133) PXE - 未定义（供应商特定）
        参数请求列表项:(134) PXE - 未定义（供应商特定）
        参数请求列表项:(135) PXE - 未定义（供应商特定）
选项:(97) 基于 UUID/GUID 的客户端标识符

长度:17
        客户端标识符 (UUID):00000000-0000-0000-0000-44123456789a
选项:(94) 客户端网络设备接口
        长度:3
        主要版本:3
        次要版本:16
选项:(93) 客户端系统架构
        长度:2
        客户端系统架构:EFI BC (7)
选项:(60) 供应商类别标识符
        长度:32
        供应商类标识符:PXEClient:Arch:00007:UNDI:003016
选项:(255) 结束
        选项结束:255
```

带 UEFI 的系统

UEFI 是一种称为统一可扩展固件接口 (UEFI) 安全启动的协议。这将成为老化 BIOS 系统的现代替代品,并有助于确保启动时恶意软件不会被注入系统。

BIOS 替代品 UEFI 需要为操作系统安装数字密钥才能通过 UEFI 固件检查才能启动。
例如,Red Hat、Ubuntu 和 Suse 等主流 Linux 发行版已经购买了这些密钥,因此它们在安全启动系统上没有问题。

如果没有这个数字密钥,您通常仍然应该能够在安全引导系统上使用 Linux。您可以先在 BIOS 中禁用以下功能：

```
快速启动/快速启动

英特尔智能响应技术 (ISRT)

FastStartUp（如果您有 Windows 8）。
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

如果出现安全启动或签名错误,则需要禁用安全启动。如果您的系统运行的是 Windows 7,您可以通过输入进入 BIOS 设置和禁用安全启动所需的键盘键来进入 BIOS。如果系统附带 Windows 8,您将需要启动到 Windows 并选择执行高级启动。这应该允许您进入 BIOS 并禁用安全启动。不是:有时 BIOS 能够在 EFI 或传统模式下运行。如果您的系统允许这样做,那么安装 Linux 应该没有任何问题

使用 Systemd-boot 启动

Systemd 带有 Systemd-boot。这是为与 EFI 系统一起使用而设计的。它只能启动 EFI 可执行文件,例如 Linux 内核 EFISTUB、UEFI Shell、GRUB 和 Windows 引导管理器。 Systemd-boot 使用 bootctl 命令进行管理。 systemd-boot 需要 EFI 系统分区 (ESP),最好安装在 /boot 上。 ESP 必须包含 EFI 二进制文件。
如需更多信息和示例,请访问https://www.freedesktop.org/software/systemd/man/index.html

使用 Das U-boot 启动

Das U-boot "通用引导加载程序"是一个开源的、针对嵌入式设备的主要引导加载程序。它用于打包引导内核代码的指令。它支持许多计算机架构,包括 68k、ARM、AVR32、Blackfin、MicroBlaze、MIPS、Nios、SuperH、PPC 和 x86。如果有大小限制,U-Boot 可以分成几个阶段。 U-Boot 需要关于内存地址在哪里复制内核、ramdisk 等数据的明确命令,这与其他自动选择内存位置的引导加载程序相反。由于 U-Boot 命令是低级的,启动内核需要多个步骤。这使得 U-Boot 非常灵活。 U-Boot 可以从板载存储、网络甚至串行端口启动。

问题和解答

系统启动

1. 命名所有四个用户模式进程,内核将尝试执行其中一个作为引导过程的最后一步。

   在引导过程的最后一步,内核将尝试执行以下程序之一作为第一个用户模式进程：/sbin/init、/etc/init、/bin/init 和/bin/sh。第一个用户态进程[50]

2. 可以发现所有进程 init 的父进程运行在八个运行级别之一。运行级别 2、3、4、5?

   这些运行级别是多用户运行级别。多用户运行级别[50]

3、文件/etc/inittab中的action字段包含字符串wait是什么意思?

   当进入指定的运行级别时,同一行第四个字段中指定的进程将仅启动一次,init 将等待其终止。文件 /etc/inittab [51]

4.目录/etc/rc*包含什么类型的文件?

   这些目录包含指向 /etc/init.d 中以 S 或 K 开头的脚本的符号链接,它们将使用启动或停止参数调用脚本。 /etc/init.d/rc 脚本[52]

5. Linux Standard Base (LSB) 标准的目的是什么?

   它确保在 LSB 兼容环境中编译的程序将在支持 LSB 标准（在特定处理器架构内)的任何发行版上运行。 LSB 标准[56]

6. 什么是链式加载?

   链式加载意味着 GRUB 将用于引导系统,然后加载并运行专有系统引导加载程序,然后引导操作系统。 GRUB 链式加载[60]

7. grub-install 命令的用途是什么?

   grub-install 命令用于将阶段 1 安装到 MBR 或分区内。命令 grub-install [63]

版权所有 2013-2017 Snow BV

Machine Translated by Google

8. 哪个命令可以让你切换到不同的运行级别?

   这是通过 telinit 命令完成的。命令 telinit [64]

9. 引导期间文件系统检查失败。你读到类似“fsck 失败。请手动修复”,然后是“CONTROL D 将从该 shell 退出并继续系统启动”。你如何进行?

   如果你不按 CTRL-D 而是输入 root 密码,你会得到一个 shell,实际上 /sbin/sulogin 已经启动,如果 root 文件系统以只读方式挂载,你应该能够运行 fsck 并解决问题.当 fsck 启动失败时[64]

版权所有 2013-2017 Snow BV

Machine Translated by Google

第3章

文件系统和设备 (203)

本主题权重为 9 分,包含以下三个目标:

目标 203.1;操作 Linux 文件系统（4 分） 考生应该能够正确配置和导航标准 Linux 文件系统。此目标包括配置和安装各种文件系统类型。

目标 203.2;维护 Linux 文件系统（3 分） 考生应该能够使用系统实用程序正确维护 Linux 文件系统。该目标包括操作标准文件系统和监控 SMART 设备。

目标 203.3;创建和配置文件系统选项（2 分） 考生应该能够使用 AutoFS 配置自动挂载文件系统。此目标包括为网络和设备文件系统配置自动挂载。还包括为 CD-ROM 等设备创建文件系统和加密文件系统的基本特征知识。

操作 Linux 文件系统 (203.1)

考生应该能够正确配置和导航标准 Linux 文件系统。此目标包括配置和安装各种文件系统类型。

资源： LinuxRef07 , Wirzenius98 。

关键知识领域

- fstab 配置的概念
- 用于处理 SWAP 分区和文件的工具和实用程序
- 使用 UUID 识别和挂载文件系统
- 了解 systemd 挂载单元

条款和实用程序

- /etc/fstab
- /etc/mtab
- /proc/mounts
- 安装和卸载
- 同步

版权所有 2013-2017 Snow BV

Machine Translated by Google

- 交换
- 交换
- blkid

文件层次结构

从历史上看,某些文件和实用程序的位置并不总是标准的（或固定的）。这导致了 Linux 不同“发行版”之间的开发和升级问题。 Linux 目录结构（或文件层次结构)基于现有的 UNIX 风格,但随着它的发展,出现了某些不一致之处。这些通常是一些小事,例如某些配置文件的位置（或布置）,但这导致了在主机之间移植软件的困难。

为了平衡这些差异,开发了文件标准。到目前为止,这是一个不断发展的过程,导致 Linux 文件层次结构的模型相当静态。这个文件系统层次结构在文件系统层次结构标准中被标准化。当前版本是 2.3。有关 FHS 的更多信息和文档可以在文件系统层次结构标准主页上找到。另请参阅有关FHS 标准的部分。

Linux 文件层次结构的顶层称为根（或 / ）。根目录通常包含几个其他目录。在讨论根文件系统内容的部分中已经给出了概述。回顾一下：

bin/	所需的启动时二进制文件
boot/	操作系统加载程序和内核映像的引导配置文件
dev/etc/	设备文件
home/	系统配置文件和脚本 用户主目录 主要操作系统共
lib/	享库和内核模块 “恢复”文件的存储目录 CD-ROM、
	闪存盘和软盘等可移动媒体的挂载点 系统管理员需要的文件
lost+found/media/	系统临时挂载点保留用于安装大型附加应用程序软件包
	Linux 系统使用的“虚拟”文件系统,用于存储有关内核、进程和当前资源使用情况的信息 Linux（非标准)根用户主目录。
蒙特/	
选择/	
过程/	
根/	替代位置是 / 目录本身 系统管理二进制文件和工具 临时文件
sbin/	的位置 可共享的只读数据,包含例如用户命令、C 程序头文件
tmp/usr/	和非重要系统二进制文件 变量数据,通常是特定于机器的。包
	括邮件和新闻、管理和日志数据的假脱机目录
变量/	

表 3.1:Linux 文件层次结构回顾

通常,根目录不应包含任何其他文件　一个可能的例外是用于各种目的的挂载点。

文件系统

文件系统由操作系统用来跟踪磁盘或分区上的文件的方法和数据结构组成;也就是说,文件在磁盘上的组织方式。该词还用于指代用于存储文件或文件系统类型的分区或磁盘。因此,有人可能会说“我有两个文件系统”,意思是其中一个有两个存储文件的分区,或者有人可能会说“我正在使用 XFS 文件系统”,意思是文件系统的类型。

版权所有 2013-2017 Snow BV

Machine Translated by Google

重要磁盘或
分区与其包含的文件系统之间的区别很重要。一些程序（包括创建文件系统的程序)直接在磁盘或分区的原始扇区上运行;如果文件系统已经存在,它将被破坏或严重损坏。大多数程序在文件系统上运行,因此不会在不包含文件系统（或包含错误类型的文件系统)的分区上运行。

在分区或磁盘可以用作文件系统之前,需要对其进行初始化,并且需要将簿记数据结构写入磁盘。这个过程称为制作文件系统。

大多数 UNIX 文件系统类型都具有相似的一般结构,尽管具体细节差异很大。核心概念是超级块、索引节点、数据块、目录块和间接块。超级块包含有关整个文件系统的信息,例如它的大小（此处的确切信息取决于文件系统）。 inode 包含文件的所有信息,除了文件名。名称与索引节点号一起存储在目录中。目录条目由文件名和代表该文件的索引节点号组成。 inode 包含若干个数据块的编号,这些数据块用于存储文件中的数据。然而,inode 中只有几个数据块编号的空间,如果需要更多空间,则动态分配更多空间用于指向数据块的指针。这些动态分配的块是间接块;顾名思义,为了找到数据块,必须先在间接块中找到它的编号。

创建文件系统

在挂载（或使用)分区之前,必须首先在其上安装文件系统　对于 ext2,这是创建索引节点和数据块的过程。

这个过程相当于初始化分区。在 Linux 下,创建文件系统的命令称为 mkfs。

该命令以下列方式发出：

```
mkfs [-c] [-t fstype] 文件系统 [块]
```

例如

```
mkfs -t ext2 /dev/fd0 # 在软盘上创建一个 ext2 文件系统
```

在哪里：

-c强制检查坏块

-t fstype指定文件系统类型。对于大多数文件系统类型,都有一个简写形式,例如:mkfs -t ext2也可以称为 mke2fs 或 mkfs.ext2,而 mkfs -t vfat或 mkfs -t msdos也可以称为 mkfs.vfat、mkfs.msdos 或 mkdosfs

filesystem是与分区或设备关联的设备文件,或者是安装文件系统的目录
（这用于擦除旧文件系统并创建新文件系统）

注意
在具有现有文件系统的设备上创建文件系统将导致旧文件系统上的所有数据被擦除。

挂载和卸载

Linux 将所有文件系统呈现为一个目录树。因此,要添加一个带有文件系统的新设备,它的文件系统需要成为该目录树的一部分。完成此操作的方法是将新文件系统附加到现有（最好是空)目录下,该目录是现有目录树的一部分 - “挂载”点。

要将新文件系统附加到目录层次结构,您必须挂载其关联的设备文件。首先你需要创建挂载点;将附加设备的目录。由于目录也是文件系统的一部分,因此存在挂载点

版权所有 2013-2017 Snow BV

Machine Translated by Google

在先前安装的设备上。它应该是空的。如果不是,目录中的文件在设备安装到目录时将不可见,但会在设备断开连接（或卸载)后重新出现。这种默默无闻的安全性有时用于向不经意的旁观者隐藏信息。

要挂载设备,请使用 mount 命令：

```
mount [选项] device_file mount_point
```

对于某些设备,mount 将检测设备上存在的文件系统类型,但更常见的是以以下形式使用 mount：

```
mount [选项] -t file_system_type device_file mount_point
```

通常,只有 root 用户可以使用 mount 命令　主要是因为设备文件归 root 所有。例如,要将第二个 (IDE) 硬盘驱动器上的第一个分区挂载到 /usr 目录之外并假设它包含 ext2 文件系统,您将输入以下命令：

```
mount -t ext2 /dev/hdb1 /usr
```

安装的常见设备是软盘驱动器。软盘通常包含 FAT,也称为 msdos,文件系统（但不总是）,并使用以下命令挂载：

```
mount -t msdos /dev/fd0 /mnt
```

请注意,软盘安装在 /mnt 目录下。这是因为 /mnt 目录通常是临时挂载设备的地方。

要查看您当前安装了哪些设备,只需键入命令 mount。一些示例输出：

```
/dev/hda3 on / type ext2 (rw) /dev/hda1 on /dos
type msdos (rw) none on /proc type proc (rw) /dev/
cdrom on /cdrom type iso9660 (ro) /dev/fd0 on / mnt 类
型 msdos (rw)
```

每行显示挂载了哪个设备文件、挂载位置、每个分区的文件系统类型以及挂载方式（ro = 只读， rw = 读/写）。注意第三行的奇怪条目　proc 文件系统。这是一个特殊的“虚拟”文件系统,Linux 系统使用它来存储有关内核、进程和当前资源使用情况的信息。它实际上是系统内存的一部分　换句话说,内核预留了一块内存区域,用于存储有关系统的信息。这个相同的区域被挂载到文件系统上,以便用户程序可以访问这些信息。

proc 文件系统中的信息也可用于通过发出以下命令查看安装了哪些文件系统：

```
$ cat /proc/mounts /dev/root /
ext2 rw 0 0 proc /proc proc rw 0 0 /dev/
hda1 /dos msdos rw 0 0 /dev/cdrom /
cdrom iso9660 ro 0 0 /dev/fd0 /mnt msdos rw
0 0
```

/etc/mtab 和 /proc/mounts 之间的区别在于 /etc/mtab 是由 mount 保存的用户空间管理,而 /proc/mounts 是由内核保存的信息。后者反映了用户空间中的信息。由于这些不同的实现， /proc/mounts 中的信息始终是最新的,而 /etc/mtab 中的信息可能会变得不一致。

要释放设备并将其与文件系统断开连接,可以使用 umount 命令。它以以下形式发布：

```
卸载设备文件
```

或者

```
卸载挂载点
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

例如,要释放软盘,您可以发出命令:

```
卸载/dev/fd0
```

或者

```
卸载/mnt
```

同样,您必须是 root 用户或有权执行此操作的用户。您不能卸载用户正在使用的设备/安装点（例如,用户的当前工作目录在安装点内)或进程正在使用。您也不能卸载设备/安装点,而这些设备/安装点又安装有设备。

系统需要在启动时挂载设备。在真正的 UNIX 方式中,有一个文件管理引导时挂载设备的行为。在 Linux 中,这个文件是 /etc/fstab。文件中的行使用以下格式:

```
device_file mount_point file_system_type mount_options [n] [n]
```

前三个字段是不言自明的;第四个字段mount_options定义设备将如何挂载（这包括访问模式ro / rw执行权限和其他信息的信息） - 有关此的信息可以在挂载手册页中找到（请注意,该字段通常包含单词“默认值”）。第五和第六个字段分别由系统实用程序 dump 和 fsck 使用 - 有关详细信息,请参阅手册。

还有一个名为 /etc/mtab 的文件。它以 fstab 形式列出当前挂载的分区。

系统安装单元

采用 systemd 初始化系统的 Linux 发行版有一种附加的文件系统挂载方式。可以使用挂载单元文件配置文件系统,而不是使用 fstab 文件进行持久挂载。这个挂载单元文件包含 systemd 持久挂载文件系统的配置细节。

systemd 挂载单元文件具有特定的命名约定。文件名是指挂载的绝对目录,文件扩展名为.mount。对于文件名,它所代表的安装路径的第一个和最后一个正斜杠 (/) 被删除,其余斜杠转换为破折号 (-)。因此,例如,如果文件系统挂载到挂载点 /home/user/data/ 挂载单元文件必须命名为 home-user-data.mount

在安装文件中定义了三个必需的部分:[Unit]、[Mount] 和 [Install]。挂载单元文件 /etc/systemd/system/home-user-data.mount 的示例:

```
[单元]
描述=用户数据

[山]
什么=/dev/sda2
其中=/home/user/data
类型=ext4
选项=默认值

[安装]
WantedBy=多用户.target
```

要测试配置,请使用命令 systemctl daemon-reload 重新加载 systemctl 守护进程,然后使用命令 systemctl start 手动启动挂载单元文件,后跟挂载单元文件。在我们的示例中,它是 systemctl start home-user-data.mount。接下来,您可以通过从 mount 获取概述来检查文件系统是否已正确安装。

如果一切按预期工作,通过使用命令 systemctl enable home-user-data.mount 启用挂载单元文件,使文件系统挂载持久化。

交换

Linux 中的交换空间是一个分区或文件,用于在 RAM 变满时将 RAM 的非活动页面的内容移动到。
Linux 可以使用文件系统中的普通文件或单独的分区作为交换空间。交换分区更快,但更容易

版权所有 2013-2017 Snow BV

Machine Translated by Google

更改交换文件的大小（无需重新分区整个硬盘,并可能从头开始安装所有内容）。
当你知道你需要多少交换空间时,你应该使用一个交换分区,但如果你有疑问,你可以先使用一个交换文件,然后使用系统一段时间,这样你就可以感觉到有多少交换空间你需要,然后当你对它的大小有信心时制作一个交换分区。建议使用单独的分区,因为这排除了文件系统碎片的可能性,这会降低性能。另外,通过使用单独的交换分区,可以保证交换区域位于磁盘最快的位置。在当前的 HDD 上,这是在盘片的开头（外缘,第一个圆柱体）。

可以同时使用多个交换分区和/或交换文件。这意味着如果您只是偶尔需要不寻常数量的交换空间,您可以在这种时候设置一个额外的交换文件,而不是一直分配全部数量。

命令 mkswap 用于初始化交换分区或交换文件。分区或文件需要存在才能被初始化。交换分区是使用磁盘分区工具（如 fdisk)创建的,交换文件可以使用以下命令创建：

```
dd if=/dev/zero of=swapfile bs=1024 count=65535
```

创建分区或文件时,可以使用以下方式对其进行初始化：

```
mkswap {设备|文件}
```

初始化的交换空间与 swapon 一起使用。此命令告诉内核可以使用交换空间。交换空间的路径作为参数给出,因此要开始交换临时交换文件,可以使用以下命令：

```
swapon /交换文件
```

或者,当使用交换分区时：

```
swapon /dev/hda8
```

可以通过在文件 /etc/fstab 中列出它们来自动使用交换空间：

```
/dev/hda8 无交换 sw 0 0 /swapfile 无交换 sw 0 0
```

启动脚本将运行命令 swapon -a,这将开始交换 /etc/fstab 中列出的所有交换空间。
因此,swapon 命令通常只在需要额外交换时才使用。您可以使用 free 监视交换空间的使用情况。它将报告已使用的交换空间总量：

```
$ free total
used free shared buffers 缓存
内存:127148 122588 4560 50 1584 69352
-/+ 缓冲区/缓存:51652 75496
交换:130748 57716 73032
```

输出的第一行 (Mem:) 显示物理内存。 total 列没有显示内核使用的物理内存,它是在引导过程中加载到 RAM 内存中的。使用的列显示使用的内存量（第二行不计算缓冲区）。空闲列显示完全未使用的内存。共享列显示了 tmpfs 使用的内存量（/proc/meminfo 中的 shmem）;缓冲区列显示磁盘缓冲区缓存的当前大小。

最后一行 (Swap:) 显示了类似的交换空间信息。如果此行全为零,则交换空间未激活。

可以通过在文件 /proc/meminfo 上使用 cat 来显示格式略有不同的相同信息：

```
$ cat /proc/内存信息
缓存的已用空闲共享缓冲区总数
内存:130199552 125177856 5021696 0 1622016 89280512
交换:133885952 59101184 74784768
内存总数:127148 kB
无内存:4904 kB
内存共享:0 kB
缓冲区:1584 kB
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
缓存:69120 kB
交换缓存:18068 kB
活动:80240 kB
非活动:31080 kB
总计:0 kB
高自由度:0 kB
总计:127148 kB
低免费:4904 kB
SwapTotal:130748 kB
无交换:73032 kB
```

要禁用设备或交换文件,请使用 swapoff 命令:

```
# swapoff /dev/sda3
```

UUID

UUID 一词代表通用唯一标识符。它是一个 128 位的数字,基本上可以用来识别任何东西。
可以使用适当的软件生成此类 UUID。有 5 个不同版本的 UUID,它们都使用(伪)随机元素、当前系统时间和一些主要是唯一的硬件 ID,例如 MAC 地址。从理论上讲,UUID 不是唯一的可能性非常非常小,但在实践中这被认为是不可能的。

在 Linux 上,对 UUID 的支持是在 e2fsprogs 包中开始的。对于文件系统,UUID 用于表示特定的文件系统。例如,您可以使用 /etc/fstab 中的 UUID 来表示要挂载的分区。

通常,UUID 表示为 32 个十六进制数字,按 8、4、4、4 和 12 位数字的序列分组,并用连字符分隔。
下面是带有 UUID 说明符的 fstab 条目:

```
UUID=652b786e-b87f-49d2-af23-8087ced0c828 / ext4 错误=remount-ro,noatime 0 1
```

您可能想知道在 fstab 中使用 UUID,因为设备名称工作正常。当磁盘移动到不同的连接器或计算机、计算机上安装了多个操作系统,或者在保持文件系统完好无损的同时设备名称可能发生变化的其他情况下,UUID 会派上用场。只要文件系统不变,UUID 就会保持不变

相同的。

注意“只要文件系统不变”。这意味着,当您重新格式化分区时,UUID 将会改变。
例如,当您使用 mke2fs 重新格式化分区 /dev/sda3 时,UUID 将被更改。所以,如果你在 /etc/fstab 中使用 UUID,你也必须调整它们。

如果您想知道特定分区的 UUID,请使用 blkid /path/to/partition:

```
# blkid /dev/sda5 /dev/sda5:
UUID=  24df5f2a-a23f-4130-ae45-90e1016031bc   TYPE=  swap
```

注意可

以创建一个新的文件系统并仍然使它具有与以前相同的 UUID,至少对于“ext”类型的文件系统而言。

```
# tune2fs /dev/sda5 -U 24df5f2a-a23f-4130-ae45-90e1016031bc
```

注意在

大多数 Linux 发行版上,您可以使用命令uuidgen生成自己的 UUID 。

版权所有 2013-2017 Snow BV

Machine Translated by Google

同步

为了提高 Linux 文件系统的性能,许多操作都在存储在 RAM 中的文件系统缓冲区中完成。要将这些缓冲区中包含的数据实际刷新到磁盘,使用 sync 命令。

sync 在重新启动或停止系统时的适当时刻自动调用。您很少需要自己使用该命令。例如,sync 可用于强制将数据同步到 USB 设备,然后再将其从系统中删除。 sync 没有任何影响操作的选项,因此需要时,只需在命令行执行“sync”即可。

维护 Linux 文件系统 (203.2)

考生应该能够使用系统实用程序正确维护 Linux 文件系统。该目标包括操作标准文件系统和监控 SMART 设备。

资源： Bandel97； Btrfs； e2fsck、badblocks、dumpe2fs、debugfs 和 tune2fs 的手册页。

关键知识领域

- 用于操作 ext2、ext3 和 ext4 的工具和实用程序。
- 执行Btrfs 操作的工具和实用程序,包括子卷和快照。
- 操作xfs 的工具和实用程序。
- ZFS 的意识。

条款和实用程序

- fsck (fsck.*)
- mkfs (mkfs.*)
- mkswap
- tune2fs
- dumpe2fs
- 调试文件
- btrfs
- btrfs 转换
- xfs_info、xfs_check、xfs_repair、xfs_dump 和 xfs_restore
- smartd 和 smartctl

磁盘检查

良好的磁盘维护需要定期磁盘检查。最好的工具是 fsck,应该至少每月运行一次。默认检查通常会在 20 次系统重新启动后运行,但如果您的系统一次持续数周或数月,您将需要不时地强制进行检查。最好的办法是执行例行的系统备份并时常检查丢失+找到的目录。

可以使用 tune2fs 更改系统重启时的检查频率。此实用程序还可用于更改安装计数,这将防止系统在第 20 次重新启动时必须检查所有文件系统（这可能需要很长时间）。

dumpe2fs 实用程序将提供有关在超级块中找到的硬盘操作参数的重要信息,而 badblocks 将执行表面检查。最后,可以使用 debugfs 完成删除磁盘上损坏区域的外科手术。

版权所有 2013-2017 Snow BV

Machine Translated by Google

fsck (fsck.*)

fsck 是一个用于检查和修复 Linux 文件系统的实用程序。实际上,fsck 只是Linux 下可用的各种文件系统检查器(fsck.fstype)的前端。

fsck 在系统启动时自动调用。如果文件系统被标记为“不干净”,或者达到最大挂载计数或检查之间的时间超过,则检查文件系统。要更改最大安装计数或检查之间的时间,请使用 tune2fs。

fsck 的常用选项包括:

-s序列化 fsck 操作。如果您正在检查多个文件系统并且检查器处于交互式状态,这是一个好主意
模式。

-A遍历 /etc/fstab 文件并尝试在一次运行中检查所有文件系统。此选项通常用于 /etc/rc 系统初始化文件,而不是用于检查单个文件系统的多个命令。

首先检查根文件系统。之后,将按照 /etc/fstab 文件中的fs_passno （第六个)字段指定的顺序检查文件系统。 fs_passno值为 0 的文件系统将被跳过并且根本不会被检查。如果有多个文件系统具有相同的密码,fsck 将尝试并行检查它们,尽管它会避免在同一物理磁盘上运行多个文件系统检查。

-R当使用 -A 标志检查所有文件系统时,跳过根文件系统（以防它已经以读写方式挂载）。

fsck 不理解的选项被传递给特定于文件系统的检查器。这些 arguments(=options) 不能带参数,因为 fsck 无法正确猜测哪些参数带选项,哪些不带参数。 --后面的选项和参数被视为要传递给特定于文件系统的检查器的特定于文件系统的选项。

ext2 文件系统的文件系统检查器称为 fsck.e2fs 或 e2fsck。经常使用的选项包括:

-a此选项与 -p 选项的作用相同。它只是为了向后兼容而提供的;建议人们尽可能使用 -p 选项。

-c此选项使 e2fsck 运行 badblocks(8) 程序以查找文件系统上的任何坏块,然后通过将它们添加到坏块索引节点来将它们标记为坏块。

-C此选项使 e2fsck 将完成信息写入指定的文件描述符,以便可以监视文件系统检查的进度。此选项通常由运行 e2fsck 的程序使用。如果指定的文件描述符为 0,则 e2fsck 将在处理其业务时打印一个完成栏。这要求 e2fsck 在视频控制台或终端上运行。

-f即使文件系统看起来很干净也强制检查。

-n以只读方式打开文件系统,并假定对所有问题的回答都是“否”。允许以非交互方式使用 e2fsck。
（注意:如果除了 -n 选项之外还指定了 -c、-l 或 -L 选项,那么文件系统将以读写方式打开,以允许更新坏块列表。但是,没有其他将对文件系统进行更改。）

-p毫无疑问地自动修复（“整理”)文件系统。

-y假设对所有问题的回答都是“是”;允许以非交互方式使用 e2fsck。

mkfs (mkfs.*)

mk2fs 命令用于创建 Linux 文件系统。它可以通过指定-t 文件系统或通过提供 mkfs.filesystem 命令来创建不同类型的文件系统。实际上 mkfs 是几个 mkfs.fstype 命令的前端。
请阅读 mkfs 手册页和部分以获取更多信息。创建文件系统

版权所有 2013-2017 Snow BV

Machine Translated by Google

tune2fs

tune2fs 用于“调整”文件系统。这主要用于设置文件系统检查选项,例如最大挂载计数和文件系统检查之间的时间。

也可以将安装计数设置为特定值。这可用于“错开”不同文件系统的安装计数,从而确保在重新启动时不会同时检查所有文件系统。

因此,对于包含 5 个分区并且大约每月启动一次的系统,您可以执行以下操作来错开安装计数:

```
tune2fs -c 5 -C 0
分区 1 tune2fs
-c 5 -C 1
分区2 tune2fs
-c 5 -C 2
分区3
tune2fs -c 5 -C 3
分区 4 tune2fs
-c 5 -C 4
分区5
```

最大安装数为 20,但对于不经常重新启动的系统,建议使用较低的值。

经常使用的选项包括:

-c max-mount-counts调整两次文件系统检查之间的最大安装计数。如果 max-mount-counts 是 0 那么文件系统被挂载的次数将被 e2fsck(8) 和内核忽略。交错强制检查文件系统的安装计数将避免在使用日志文件系统时同时检查所有文件系统。

您应该认真考虑完全禁用挂载计数相关检查的后果。损坏的磁盘驱动器、电缆、内存和内核错误都可能损坏文件系统,而不会将文件系统标记为脏或错误。如果你在你的文件系统上使用日志记录,你的文件系统将永远不会被标记为脏,所以它通常不会被检查。内核检测到的文件系统错误仍会在下次重新启动时强制执行 fsck,但此时可能已经太晚无法防止数据丢失。

-C mount-count设置文件系统被挂载的次数。可以与 -c 结合使用来强制下次重启时在文件系统上执行 fsck。

-i interval-between-checks[d|m|w]调整两次文件系统检查之间的最长时间。没有后缀或 d 导致天,m 是月,w 是周。零值将禁用时间相关检查。

强烈建议启用 -c（挂载计数相关)或 -i（时间相关)检查以强制定期对文件系统进行完整的 e2fsck(8) 检查。如果不这样做,可能会由于磁盘、电缆或内存损坏而导致文件系统损坏,或者内核错误会被忽视,直到它们导致数据丢失或损坏。

-m reserved-blocks-percentage设置保留文件系统块的百分比。

-r reserved-blocks-count设置保留的文件系统块数。

dumpe2fs

dumpe2fs 打印设备上存在的文件系统的超级块和块组信息。

-b打印在文件系统中保留为坏块的块。

-h只显示超级块信息而不显示任何块组描述符的详细信息。

版权所有 2013-2017 Snow BV

Machine Translated by Google

坏块

badblocks 是一个 Linux 实用程序,用于检查磁盘驱动器上损坏的扇区。它标记了这些扇区,以便将来不会使用它们,从而不会导致数据损坏。它是 e2fsprogs 项目的一部分。

强烈建议不要直接运行 badblocks,而是通过e2fsck 或 mke2fs 中的-c选项调用它。
一个常用的选项是:

-o output-file将坏块列表写入输出文件。

调试文件

使用 debugfs,您可以通过直接磁盘写入来修改磁盘。由于此实用程序非常强大,您通常希望以只读方式调用它,直到您准备好实际进行更改并将它们写入磁盘。要以只读模式调用 debugfs,请不要使用任何选项。要以读写模式打开,请添加 -w 选项。您可能还想在命令行中包含您希望使用的设备,如 /dev/hda1 或 /dev/sda1 等。一旦它被调用,您应该会看到 debugfs 提示符。

当分区的超级块损坏时,可以指定不同的超级块使用:

```
debugfs -b 1024 -s 8193 /dev/hda1
```

这意味着将使用块 8193 处的超级块,块大小为 1024。请注意,当您要使用不同的超级块时,必须指定块大小。有关块大小和备份超级块的信息可以在以下位置找到:

```
dumpe2fs /dev/hda1
```

调用 debugfs 后要尝试的第一个命令是 params,用于显示模式（只读或读写)和当前文件系统。如果你在没有打开文件系统的情况下运行这个命令,它几乎肯定会转储核心并退出。当检查一个以上的文件系统时,另外两个命令,open 和 close,可能会很有趣。 Close 不带任何参数,并且足够恰当地关闭当前打开的文件系统。 Open 将设备名称作为参数。要查看超级块的磁盘统计信息,命令 stats 将按组显示信息。命令 testb 检查块是否正在使用中。

这可用于测试 badblocks 命令标记为“坏”的块中是否有任何数据丢失。要获取块的文件名,首先使用 icheck 命令获取 inode,然后使用 ncheck 获取文件名。处理坏块的最佳方法是将块标记为“坏”并从备份中恢复文件。

要获得所有命令的完整列表,请参阅 debugfs 的手册页或键入 ?、lr 或 list_requests。

分机4

Ext4 是最常用的 Linux 文件系统 Ext3 的演变。在许多方面,Ext4 比 Ext3 比 Ext2 更深入地改进了 Ext3。 Ext3 主要是为 Ext2 添加日志功能,但 Ext4 修改了文件系统的重要数据结构,例如用于存储文件数据的数据结构。结果是文件系统具有改进的设计、更好的性能、可靠性和功能。因此,将 ext3 转换为 ext4 并不像将 ext2 转换为 ext3 那样直接和容易。

要从头开始创建 ext4 分区,请使用:

```
mkfs.ext4 /dev/sdxY
```

> 注意提
> 示:有关更多选项,请参阅 mkfs.ext4 手册页;编辑/etc/mke2fs.conf以查看/配置默认选项。

请注意,默认情况下,mkfs.ext4 使用相当低的每 inode 字节比率来计算要创建的 inode 的固定数量。

> 笔记
> 注意:特别是对于当代 HDD（750 GB+）,这通常会导致 inode 数量太大,因此可能浪费很多 GB。比率可以直接通过 -i 选项设置； 6291456 中的一个为 2 TB 分区生成了 476928 个 inode。

对于其余的 ext4,可以使用可用于 ext2/ext3 类型文件系统的所有相同工具来操作,例如 badblocks、dumpe2fs、e2fsck 和 tune2fs。

版权所有 2013-2017 Snow BV

Machine Translated by Google

btrfs

Btrfs（缩写:BTree 文件系统)是 Linux 的一种新的写时复制 (CoW) 文件系统,旨在实现高级功能,同时专注于容错、修复和易于管理。 Btrfs 由 Oracle、Red Hat、Fujitsu、Intel、SUSE、STRATO 和许多其他公司联合开发,已根据 GPL 获得许可并开放供任何人贡献。 Btrfs 具有存储设备的几个特性。它旨在使文件系统容错,并便于在错误发生时进行检测和修复。它使用校验和来确保数据和元数据的有效性,并维护可用于备份或修复的文件系统快照。 btrfs 使用的核心数据结构是 B 树 因此得名。

笔记

Btrfs 仍在大力开发中,但正在尽一切努力保持文件系统稳定和快速。由于开发速度,您应该尽可能运行最新的内核（来自 kernel.org 的最新版本内核,或最新的 -rc 内核。

截至 2013 年初,Btrfs 已包含在默认内核中,其工具 (btrfs-progs) 是默认安装的一部分。 GRUB 2、mkinitcpio 和 Syslinux 支持 Btrfs,无需额外配置。

目前可用的主要 Btrfs 功能包括：

- 基于范围的文件存储
- 2^64 字节 == 16 EiB 最大文件大小
- 节省空间的小文件打包
- 节省空间的索引目录
- 动态 inode 分配
- 可写快照、只读快照
- 子卷（独立的内部文件系统根目录）
- 数据和元数据校验和 (crc32c)
- 压缩（zlib 和 LZO）
- 集成多设备支持
- 文件条带化、文件镜像、文件条带化+镜像、带单奇偶校验和双奇偶校验实施的条带化
- SSD（闪存存储)意识（TRIM/Discard 用于报告空闲块以供重用)和优化（例如,避免不必要的寻道优化,在集群中发送写入,即使它们来自不相关的文件。这会导致更大的写入操作和更快的写入吞吐量）
- 高效的增量备份
- 用于查找和修复具有冗余副本的文件错误的后台清理过程
- 在线文件系统碎片整理
- 离线文件系统检查
- 转换现有的 ext3/4 文件系统
- 种子设备。创建一个（只读)文件系统,充当其他 Btrfs 文件系统的模板。原始文件系统和设备作为新文件系统的只读起点包含在内。使用写时复制,所有修改都存储在不同的设备上;原来是不变的。
- 子卷感知配额支持
- 发送/接收子卷变化

版权所有 2013-2017 Snow BV

Machine Translated by Google

· 高效的增量文件系统镜像

最显着（独特)的 btrfs 功能是：

RAID 功能 Btrfs 文件系统提供对集成 RAID 功能的支持,并且可以在许多设备之上创建。创建文件系统后可以添加更多设备。默认情况下,元数据将跨两个设备进行镜像,数据将跨所有存在的设备条带化。如果只有一台设备存在,元数据将在该设备上复制。

Btrfs 可以在线添加和删除设备,并在创建文件系统后在 RAID 级别之间自由转换。 Btrfs 支持 raid0、raid1、raid10、raid5 和 raid6,它还可以在单个 spindle 上复制元数据。读入块时,将验证校验和。如果有任何错误,Btrfs 会尝试从备用副本读取,如果备用副本成功,将修复损坏的副本。
mkfs.btrfs 将在命令行上接受多个设备。它具有控制数据 (-d) 和元数据 (-m) 的 raid 配置的选项。有效的选择是 raid0、raid1、raid10 和 single。选项 -m single 表示不复制元数据,这在使用硬件 raid 时可能是需要的。这里有一些关于创建文件系统的例子。

```
# 创建跨四个驱动器的文件系统（元数据镜像,线性数据分配） mkfs.btrfs /dev/sdb /dev/sdc /dev/sdd /dev/sde

# 条带化数据而不镜像 mkfs.btrfs -d raid0 /dev/sdb /dev/sdc

# 对数据和元数据使用 raid10 mkfs.btrfs -m raid10 -d raid10 /dev/sdb /
dev/sdc /dev/sdd /dev/sde

# 不要在单个驱动器上复制元数据（单个 SSD 上的默认设置） mkfs.btrfs -m single /dev/sdb
```

创建文件系统后,它可以像任何其他文件系统一样挂载。要查看文件系统使用的设备,您可以使用以下命令：

```
butterfs 文件系统显示标签:无 uuid：
c67b1e23-887b-4cb9-b037-5958f6c0a333
    总设备 2 FS 字节使用 383.00KiB devid 1 大小 4.00 GiB 使用 847.12MiB 路
    径 /dev/sdb devid 2 大小 4.00 GiB 使用 837.12MiB 路径 /dev/sdc

Btrfs v4.8.5
```

快照 Btrfs 的快照易于使用和理解。快照将显示为快照目录下的普通目录,您可以像在任何目录中一样进入它并在其中四处走动。

默认情况下,所有快照在 Btrfs 中都是可写的,但您可以选择创建只读快照。如果您只是为备份拍摄快照,然后在备份完成后将其删除,则只读快照非常有用。
可写快照很方便,因为您可以在执行系统更新之前对文件系统进行快照等操作;如果更新破坏了您的系统,您可以重新启动到快照并像使用普通文件系统一样使用它。当您创建新的 Btrfs 文件系统时,根目录是一个子卷。只能对子卷进行快照,因为子卷代表完全不同的文件系统树的根,您只能对文件系统树进行快照。

想到这一点的最简单方法是为 /home 创建一个子卷,这样您就可以彼此独立地快照 / 和 /home。因此,您可以运行以下命令来创建子卷：

```
btrfs 子卷创建 /home
```

然后在以后的某个时刻,当您需要快照 /home 进行备份时,您只需运行：

```
btrfs 子卷快照 /home/ /home-snap
```

完成备份后,您可以使用以下命令删除快照

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
btrfs 子卷删除 /home-snap/
```

取消链接快照树的艰苦工作是在后台完成的,因此您可能会注意到 I/O 发生在一个看似空闲的盒子上;这只是 Btrfs 清理旧快照。如果你有很多快照或者不记得你创建了哪些目录作为子卷,你可以运行命令:

```
# btrfs 子卷列表 /mnt/btrfs-test/
ID 267 顶级 5 路径回家
ID 268 顶级 5 路径 snap-home
ID 270 顶级 5 路径家/约瑟夫
```

这不会区分快照和普通子卷,因此您可能应该一致地命名您的快照,以便稍后您可以分辨哪个是哪个。

子卷 btrfs 中的子卷与 LVM 逻辑卷或 ZFS 子卷不同。使用 LVM,逻辑卷本身就是一个块设备; Btrfs 不是这种情况。 btrfs 子卷不是块设备,不能将其视为一个。

相反,可以将 btrfs 子卷视为 POSIX 文件命名空间。这个命名空间可以通过文件系统的顶级子卷访问,或者它可以自己挂载。因此,给定这样的文件系统结构:

```
顶层
        --- dir_a                                   * 只是一个普通的目录
                        ---
    |                       p
    |                   ---
                            q
        --- subvol_z                                * 子卷
                ---
                    r
                ---
                    秒
```

可以挂载文件系统的根目录,在挂载点可以看到完整的文件系统结构;或者,可以挂载子卷（使用挂载选项 subvol=subvol_z）,并且只有文件 r 和 s 在挂载点可见。

btrfs 文件系统有一个默认的子卷,它最初被设置为顶级子卷。如果没有将 subvol 或 subvolid 选项传递给挂载,则它是默认挂载的子卷。使用 btrfs subvolume default 更改默认子卷将使文件系统的顶层不可访问,除非使用 subvolid=0 挂载选项。

节省空间的索引目录 在 Btrfs 中,目录和文件在磁盘上看起来是一样的,这与 UNIX 的处理方式一致。 ext 文件系统变体在创建文件系统时必须预先分配它们的 inode 空间,因此您在创建文件系统后可以创建的文件数量受到限制。

使用 Btrfs,我们会在您创建新文件时向 B 树添加几个项目,这只会限制您在文件系统中拥有的元数据空间量。如果您曾经在 ext 文件系统的一个目录中创建了数千个文件,然后删除了这些文件,您可能已经注意到,在该目录上执行 ls 所花费的时间比您预期的要长得多,因为可能只有几个文件目录中的文件。

您甚至可能不得不运行此命令:

```
e2fsck -D /dev/sda1
```

重新优化您的分机目录。这是由于目录索引在 ext 中的存储方式存在缺陷:它们无法收缩。因此,一旦您添加了数千个文件并且内部目录索引树增长到很大,它就不会随着您删除文件而缩小。 Btrfs 不是这种情况。

在 Btrfs 中,我们在文件系统 B 树中的目录 inode 旁边存储一个文件索引。 B-tree 会根据需要增长和收缩,所以如果你在一个目录中创建了十亿个文件然后删除所有这些文件,那么 ls 将只花费你刚创建目录时的时间。

Btrfs 还为每个文件创建了一个基于文件名的索引。这很方便,因为我们不必通过包含目录的文件索引来搜索匹配项,我们只需散列文件名并在 B 树中搜索此散列值即可。此项存储在文件的 inode 项旁边,因此查找名称通常会在包含您需要的所有重要信息的同一块中读取。同样,这限制了完成基本任务所需的 I/O 量。

版权所有 2013-2017 Snow BV

Machine Translated by Google

交换

mkswap 在设备或文件中设置一个 Linux 交换区。 （创建交换区后,您需要调用 swapon 命令开始使用它。通常交换区列在 /etc/fstab 中,以便在启动时通过某些启动脚本中的 swapon -a命令使用它们.) 请参见交换文件使用[80]

xfs_info

xfs_info 显示 XFS 文件系统的文件系统几何结构。 xfs_info 等同于使用 -n 选项调用 xfs_growfs。

xfs_check

xfs_check 检查 XFS 文件系统是否一致。只有当有理由相信文件系统存在一致性问题时才需要它。由于 XFS 是一个日志文件系统,它允许它保持文件系统的一致性,所以几乎不需要运行 xfs_check。

xfs_repair

xfs_repair 修复损坏或损坏的 XFS 文件系统。 xfs_repair 将尝试查找与指定块设备关联的原始设备,并将使用该原始设备。无论如何,必须卸载要修复的文件系统,否则,生成的文件系统可能会变得不一致或损坏。

smartmontools:smartd 和 smartctl

可以使用两个实用程序 smartctl 和 smartd（安装 smartmontools 软件包时可用)使用自我监控、分析和报告技术系统 (SMART) 来监控和控制存储系统。 SMART 内置于大多数现代 ATA 和 SCSI 硬盘和固态驱动器中。 SMART的目的是监控硬盘的可靠性和预测硬盘故障,并进行不同类型的硬盘自检。

smartdis 一个守护进程,它将尝试在 ATA 设备上启用 SMARTmonitoring 并每 30 分钟轮询一次这些设备和 SCSI 设备（可配置） ,通过 SYSLOG 接口记录 SMARTerrors 和 SMARTAttributes 的更改。 smartd 还可以配置为在检测到问题时发送电子邮件警告。根据问题的类型,您可能需要在磁盘上运行自检、备份磁盘、更换磁盘或使用制造商的实用程序来强制重新分配坏的或不可读的磁盘扇区。

smartd 可以在启动时使用配置文件 /usr/local/etc/smartd.conf 进行配置。当 USR1 信号发送到 smartd 时,它会立即检查磁盘的状态,然后每 30 分钟返回一次轮询磁盘。请查阅 smartd 的手册页以了解特定的配置选项。

smartctl 实用程序控制 SMART 系统。它可用于扫描设备并打印有关它们的信息,在特定磁盘上启用或禁用 SMART,配置检测到（即将发生的)错误时要执行的操作。有关详细信息,请参阅 smartctl 手册页。

ZFS:zpool 和 zfs

ZFS 目前归 Oracle Corporation 所有,是 Sun Microsystems 开发的下一代文件系统,旨在实现近乎无限的可扩展性并且摆脱传统的设计范式。目前只有 Ubuntu 为 Linux 上的 ZFS 提供内核集成。
其他 Linux 发行版可以借助 Fuse 通过用户空间使用 ZFS。

目前可用的主要 ZFS 功能包括:

- 高容错性和数据完整性
- 由于 128 位设计,近乎无限的存储空间

版权所有 2013-2017 Snow BV

Machine Translated by Google

- 具有 RAID 功能的混合卷/文件系统管理
- 数据压缩/重复数据删除
- 数据快照
- 卷配置 (zvols)
- 能够使用不同的设备进行缓存和日志记录
- 能够使用委派管理权限给非特权用户

高容错和数据完整性设计使得它不需要像 fsck 这样的命令可用。 ZFS 在事务中工作,其中 ueberblock 仅在一切完成后才更新。以前的 uberblocks (128) 的副本以循环方式保存。所谓的 vdev 标签,用于标识 zfs 池中使用的磁盘,也有多个副本:2 个在磁盘开头,2 个在结尾。定期清理池可以揭示数据完整性问题,如果一个池配备了多个设备,可以即时修复它。

ZFS 允许额外的块校验和算法,并且可以存储这些块的多个副本。这可以根据 ZFS 文件系统进行配置。

ZFS 池可以被视为逻辑卷管理器,可以由单个或多个磁盘组成。池可以附加单独的缓存和日志记录设备,以便将读取/写入卸载到更快的设备。在一个池中拥有多个磁盘可以进行即时数据恢复。

一个典型的简单 ZFS 池:

```
$ sudo zpool 列表 -v
NAME SIZE ALLOC FREE EXPANDSZ FRAG                    CAP DEDUP HEALTH ALTROOT
坦克 1.98G 448K 1.98G                  -      0%      0% 1.00x 在线 -
  sdb 1.98G 448K 1.98G                 -      0%      0%
```

池的状态:

```
$ sudo zpool 状态 -v 池:坦克

  状态:在线
    扫描:无请求配置:


    姓名              状态          读写校验和
    坦克              在线的          0     0     0
      安全数据库      在线的          0     0     0
    错误:没有已知的数据错误
```

ZFS 池的历史概述:

```
$ sudo zpool history "tank"的历史记录:
2017-01-11.11:32:50 zpool create -f -m /
tank tank /dev/sdb 2017-01-11.12:37:44 zpool scrub tank
```

ZFS 池用作一个或多个 ZFS 文件系统的后备。默认情况下,池本身有一个以池命名的文件系统。 ZFS 文件系统大小灵活,允许自定义各种属性,如压缩、大小预留、配额、块副本等。可以使用 zfs get all 和可选的 zfs 文件系统、快照或

体积。

典型的 ZFS 文件系统列表

```
$ sudo zfs 列表
使用的名称可用参考安装点
坦克 242K 1.92G                19K/坦克
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

创建 ZFS 文件系统“文档”并使用压缩

```
$ sudo zfs create -o compression=on tank/documents $ sudo zfs list tank/文件

姓名                    使用过的 AVAIL REFER MOUNTPOINT
坦克/文件                19K 1.92G        19K /坦克/文件
```

或者

```
$ sudo zfs 创建坦克/文档 $ sudo zfs set compression=on tank/
documents $ sudo zfs list tank/documents

姓名                    使用过的 AVAIL REFER MOUNTPOINT
坦克/文件                19K 1.92G        19K /坦克/文件
```

通过 /tank/documents 中的一些数据可以看到压缩率

```
$ sudo zfs get compressratio tank/documents
姓名                    财产                价值来源
坦克/文件压缩比 1.68x -
```

创建 /tank/documents 的备份可以立即完成,并且在 /tank/documents 的内容发生变化之前不占用空间。可以通过该 ZFS 文件系统的 .zfs/snapshot 目录访问快照的内容。

```
$ sudo zfs snap tank/documents@backup $ sudo zfs list -t 快照

姓名                    使用过的 AVAIL REFER MOUNTPOINT
坦克/文件@备份           0          - 45.0M -
```

创建和配置文件系统选项 (203.3)

考生应该能够使用 AutoFS 配置自动挂载文件系统。此目标包括为网络和设备文件系统配置自动挂载。还包括为 CD-ROM 等设备创建文件系统。

资源： Don99;尼尔森98;特鲁姆普00。

关键知识领域

- autofs 配置文件
- UDF 和 ISO9660 工具和实用程序
- 了解 CD-ROM 文件系统（UDF、ISO9660、HFS）
- 了解 CD-ROM 文件系统扩展（Joliet、Rock Ridge、El Torito）
- 加密文件系统的基本特征知识

条款和实用程序

- /etc/auto.master
- /etc/auto.[目录]
- mkisofs
- 日
- mke2fs

版权所有 2013-2017 Snow BV

Machine Translated by Google

Autofs 和自动挂载器

自动挂载是文件系统挂载（和卸载)由守护进程自动完成的过程。如果文件系统未挂载并且用户尝试访问它,它将自动（重新)挂载。这在网络环境（特别是当并非所有机器都始终在线时)和可移动设备（如软盘和 CD-ROM)中很有用。

自动挂载的 linux 实现 autofs 由一个内核组件和一个名为 automount 的守护进程组成。 Autofs 在需要时使用 automount 挂载本地和远程文件系统（通过 NFS）,并在不使用时（超时后)卸载它们。您的 /etc/init.d/autofs 脚本首先查看 /etc/auto.master：

```
# 示例 /etc/auto.master 文件 /var/autofs/floppy /etc/
auto.floppy --timeout=2 /var/autofs/cdrom /etc/auto.cdrom --timeout=6
```

该文件包含由三个空格分隔的字段的行。第一个字段列出了将在其中完成挂载的目录。
第二个字段列出了文件名,我们在其中放置了要安装的设备的配置数据,即“补充”文件。最后一个字段指定选项。在我们的示例中,我们指定了超时。超时期限过后,automount 守护程序将卸载补充文件中指定的设备。

补充文件中的配置数据可能包含多行并跨越多个设备。可以自由选择补充文件的文件名和路径。补充文件中的每一行包含三个字段：

```
# 示例 /etc/auto.floppy 文件 floppy -user,fstype=auto :/
dev/fd0
```

第一个值是“伪”目录。如果设备已挂载,将出现一个该名称的目录,您可以进入该目录以浏览已挂载的文件系统。第二个值包含挂载选项。第三个值是“伪”目录所连接的设备（例如/dev/fd0,软盘驱动器）。

重新加载自动挂载程序时重新读取配置文件

```
/etc/init.d/autofs 重新加载
```

请注意,如果挂载的目录（例如：/home)繁忙,autofs 不会重新加载或重新启动。 auto.master 中的每个条目都启动它自己的守护进程。

“伪”目录包含在 /etc/auto.master 中定义的目录中。当用户尝试访问这个“伪”目录时,他们将被重新路由到您指定的设备。例如,如果您指定一个补充文件来挂载 /dev/fd0 到 /var/autofs/floppy/floppy,命令 ls /var/autofs/floppy/floppy将列出软盘的内容。但是,如果您执行命令 ls /var/autofs/floppy,即使目录 /var/autofs/floppy/floppy 应该存在,您也看不到任何东西。那是因为 /var/autofs/floppy/floppy 还不存在。只有当您直接尝试访问该目录时,自动挂载程序才会挂载它。

每个设备都应该有自己的补充文件。因此,例如,软盘驱动器的配置数据和光驱的配置数据不应合并到同一个补充文件中。 /etc/auto.master 文件中的每个定义都会生成它自己的自动挂载守护进程。如果您有多个设备在同一个自动挂载守护进程的控制下运行　这是合法的　但其中一个设备出现故障,则该守护进程可能会挂起或挂起,并且它控制的其他设备也可能无法正确挂载。因此,最好让每个设备都受其自己的自动挂载守护程序的控制,因此 /etc/auto.master 文件中的每个条目的每个补充文件应该只有一个设备。

使用 systemd 自动挂载 要使用 systemd 启动自动挂载,应在 /etc/systemd/system 中创建一个单元文件。单元文件应该以挂载点命名。在这个例子中,文件被命名为:mnt.mount,因为挂载点是/mnt。

```
# cat /etc/systemd/systemd/mnt.mount [单位]

Description=我的新文件系统

 [山]
什么=/dev/sdb1
其中=/mnt
类型=ext4
选项=
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
[安装]
WantedBy=多用户.target
```

创建单元文件后,应使用以下命令将其激活:

```
# systemctl 启动 mnt.mount
```

验证激活的挂载点:

```
# 挂载 | grep sdb1 /dev/sdb1 on /
mnt type ext4 (rw,relatime,data=ordered)
```

要在启动时激活自动挂载,请使用以下命令:

```
# systemctl 启用 mnt.mount
```

启动重新启动并验证分区是否已安装。有关使用 systemd 挂载的更多信息,请参见 systemd.mount(5) 的手册页。

带 systemd 的 Autofs 要启用带 systemd 的 autofs,应在 /etc/systemd/system 中创建一个扩展名为 .automount 的单元文件。该文件中的信息由 systemd 控制和监督。 Automount 单元应该以它们控制的方向命名。在此示例中,自动挂载单元配置文件的名称为:mnt.automount。

```
# cat /etc/systemd/system/mnt.automount [单位]

Description=我的新自动挂载文件系统。

[自动安装]
其中=/mnt

[安装]
WantedBy=多用户.target
```

激活 autofs 自动挂载单元。

```
# systemctl status mnt.automount · mnt.automount - 我
的新自动挂载文件系统。
已加载:已加载（/etc/systemd/system/mnt.automount;已禁用;供应商预设:已禁用）
   活动:不活动（死）
      其中：/mnt

# systemctl enable mnt.automount 创建了从 /etc/systemd/
system/multi-user.target.wants/mnt.automount 到 /etc/ 的符号链接 ←。
      systemd/系统/mnt.automount。

# systemctl 启动 mnt.automount

# systemctl status mnt.automount · mnt.automount - 我
的新自动挂载文件系统。
已加载:已加载（/etc/systemd/system/mnt.automount;已启用;供应商预设:已禁用）
   活动:活动（等待)自星期四 2016-12-29 14:01:57 AST； 1分2秒前
      其中：/mnt

Dec 29 14:01:57 snow systemd[1]:设置自动挂载我的新自动挂载文件系统..
```

启动后,挂载输出显示使用 autofs 启用的挂载点。

```
# mount |grep mnt systemd-1
on /mnt type autofs (rw,relatime,fd=25,pgrp=1,timeout=0,minproto=5,maxproto=5, ←。
      直接的）
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

命令 lsblk 显示该磁盘未安装。

```
#lsblk
名称 MAJ:MIN RM SIZE RO 类型 MOUNTPOINT
sda                 8:0                 0 30G 0盘
&#x251c;&#x2500;sda1 8:1 &#x251c;&#x2500;sda2                0 28.8G 0 部分 / 0 1K 0 部分 0 1.3G
8:2 &#x2514;&#x2500;sda5 8:5 8:16 0                          0 部分 [SWAP]

安全数据库                                          1G 0盘
&#x2514;&#x2500;sdb1 8:17 0 1023M 0 部分
```

让我们做一个 /mnt 目录的文件列表。

```
#ls/mnt
失物招领
```

然后再次执行命令lsblk 验证分区sdb1 是否挂载到/mnt 上。

```
#lsblk
名称 MAJ:MIN RM SIZE RO 类型 MOUNTPOINT
sda                 8:0                 0 30G 0盘
&#x251c;&#x2500;sda1 8:1 &#x251c;&#x2500;sda2                0 28.8G 0 份 / 1K 0 份 0 1.3G 0 份
8:2 &#x2514;&#x2500;sda5 8:5 sdb                             0          [SWAP]

                    8:16 0                          1G 0盘
&#x2514;&#x2500;sdb1 8:17 0 1023M 0 部分/mnt
```

mount 命令显示相同。

```
# 挂载 | grep /mnt systemd-1 on /mnt
type autofs (rw,relatime,fd=25,pgrp=1,timeout=0,minproto=5,maxproto=5, ←
        直接的)
/dev/sdb1 on /mnt type ext4 (rw,relatime,data=ordered)
```

Autofs 与 systemd 的结合现在可以正常工作了。有关更多配置选项,请参阅 systemd.automount(5) 的联机帮助页。

光盘文件系统

为 CD-ROM 创建映像

在硬盘分区上创建文件系统的常用实用程序会将一个空文件系统写入其中,然后由用户在需要时挂载并填充文件。可写 CD 只能写入一次,因此如果我们向其写入一个空文件系统,它将被格式化并永远保持完全空白。对于可重写媒体也是如此,因为您还不能更改任意扇区;您必须擦除整个磁盘。创建文件系统的工具称为 mkisofs。示例用法如下所示:

```
$ mkisofs -r -o cd_image private_collection/
```

选项 -r 将 CD 上所有文件的权限设置为公共可读并启用 Rock Ridge 扩展。你可能想要使用这个选项,除非你真的知道你在做什么（提示:没有 -r 挂载点获得 private_collection 的权限！）。

mkisofs 将尝试将所有文件名映射到 DOS 使用的 8.3 格式,以确保尽可能高的兼容性。在命名冲突的情况下（不同的文件具有相同的 8.3 名称）,文件名中使用数字,并且通过 STDERR（通常是屏幕)打印有关所选文件名的信息。不要惊慌:在 Linux 下,您永远不会看到这些奇怪的 8.3 文件名,因为 Linux 使用包含原始文件信息（权限、文件名等)的 Rock Ridge 扩展名。

如果您想生成对 Windows 更友好的 CD-ROM,请使用选项 -J（MS Joliet 扩展)或使用 mkhybrid。您还可以使用 mkhybrid 来创建 HFS CD-ROM 阅读手册页以获取有关各种选项的详细信息。另一个扩展是 El Torito,用于创建可引导 CD-ROM 文件系统。

版权所有 2013-2017 Snow BV

Machine Translated by Google

除了由 mkisofs 创建的 ISO9660 文件系统外,还有 UDF（通用磁盘格式)文件系统。光存储技术协会对 UDF 文件系统进行了标准化,以形成适用于所有（只读和可重写）光介质的通用文件系统。它旨在取代 ISO9660 文件系统。

创建和维护 UDF 文件系统的工具有：

mkudffs 创建一个新的 UDF 文件系统。可用于硬盘和 CD-R(W)。

udffsck 用于检查 UDF 文件系统的完整性和纠正错误。

wrudf 此命令用于维护 UDF 文件系统。它提供了一个交互式 shell,可以对现有的 UDF 进行操作
文件系统：cp、rm、mkdir、rmdir、ls、....等。

cdrwtool cdrwtool 提供管理 CD-RW 驱动器的工具。这包括格式化新磁盘、设置读取和
写入速度等

这些工具是 UDFtools 包的一部分。

mkisofs的输出没有直接发送到writer设备的原因：

- mkisofs 对驱动 CD 刻录机一无所知；
- 您可能希望在刻录之前测试映像；
- 在慢速机器上它不可靠。

还可以考虑创建一个额外的分区并将图像写入该分区而不是文件。这是可能的,但有一些缺点。如果由于拼写错误而写入错误的分区,则可能会丢失整个 Linux 系统。此外,这是对磁盘空间的浪费,因为 CD 映像是临时数据,可以在写入 CD 后删除。但是,使用原始分区可以节省您删除 650MB 大小文件的时间。

测试 CD 镜像

Linux 能够像挂载磁盘分区一样挂载文件。此功能可用于检查 CD 映像的目录布局和文件访问权限是否符合您的要求。尽管如今介质很便宜,但写入过程仍然很耗时,您可能至少想通过快速测试来节省一些时间。

要将上面创建的文件 cd_image 挂载到目录 /cdrom 上,请输入命令

```
$ mount -t iso9660 -o ro,loop=/dev/loop0 cd_image /cdrom
```

现在您可以检查 /cdrom 下的文件 - 它们看起来与真正的 CD 上完全一样。要卸载 CD 映像,只需说 umount /cdrom。

将 CD 映像写入 CD

命令 cdrecord 用于将图像写入 SCSI CD 刻录机。非 SCSI 写入器需要兼容驱动程序,这使它们看起来就像是真正的 SCSI 设备。

CD 刻录机希望获得源源不断的数据流。因此,将映像写入 CD 的过程不能中断,否则 CD 可能会损坏。很容易无意中中断数据流,例如删除一个非常大的文件。假设您删除了一个旧的 650 Mb CD 映像 - 内核必须更新 650,000 个硬盘块的信息（假设您的文件系统的块大小为 1 Kb）。这需要一些时间,并且会使磁盘活动减慢足够长的时间,以使数据流暂停几秒钟。然而,阅读邮件、浏览网页,甚至编译内核一般不会影响现代机器上的写入过程。

请注意,当它受到干扰时,任何刻录机都不能重新定位其激光并继续在 CD 上的原始位置。因此,任何强烈的振动或其他机械冲击都可能会损坏您正在刻录的 CD。

您需要使用 cdrecord -scanbus 找到 SCSI-BUS、-ID 和 -LUN 编号,并使用这些来写入 CD：

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
#SCSI_BUS=0#
# SCSI_ID=6 # 取自 cdrecord -scanbus
# SCSI_LUN=0 # # cdrecord
-v speed=2 dev=$SCSI_BUS,$SCSI_ID,$SCSI_LUN \ -data cd_image


# 同上,但更短:# cdrecord -v speed=2 dev=0,6,0 -data
cd_image
```

为了更好的可读性,写入器的坐标存储在三个名称实际有意义的环境变量中:SCSI_BUS、SCSI_ID、SCSI_LUN。

如果您使用 cdrecord 覆盖 CD-RW,则必须添加选项 blank=... 以擦除旧内容。请阅读手册页以了解有关清除 CD-RW 的各种方法的更多信息。

如果机器足够快,您可以将 mkisofs 的输出直接输入到 cdrecord 中:

```
# IMG_SIZE=    mkisofs -R -q -print-size private_collection/ 2>&1\ | sed -e    s/.* = //


# echo $IMG_SIZE

# [    0$IMG_SIZE    -ne 0 ] && mkisofs -r\ private_collection/ \ | cdrecord
speed=2 dev=0,6,0 tsize=${IMG_SIZE}s -data -


# 不要忘记 s                                --^ ^-- 从标准输入读取数据
```

第一个命令是空运行以确定图像的大小（您需要来自 cdrecord 发行版的 mkisofs 才能工作）。您需要指定将在最终运行中使用的所有参数（例如 -J 或 -hfs）。如果您的编写器不需要知道要写入的图像的大小,您可以将此试运行保留下来。打印尺寸必须作为 tsize 参数传递给 cdrecord（它存储在环境变量 IMG_SIZE 中）。第二个命令是一系列 mkisofs 和 cdrecord,通过管道耦合。

制作数据 CD 的副本

制作数据 CD 的 1:1 副本是可能的。但您应该知道,阅读原件时的任何错误（由于灰尘或划痕)都会导致副本有缺陷。请注意,这两种方法在音频 CD 上都将失败!

第一种情况:您有一个 CD 刻录机和一个单独的 CD-ROM 驱动器。通过发出命令

```
# cdrecord -v dev=0,6,0 speed=2 -isosize /dev/scd0
```

您从附加为 /dev/scd0 的 CD-ROM 驱动器读取数据流,并将其直接写入 CD 刻录机。

第二种情况:您没有单独的 CD-ROM 驱动器。在这种情况下,您必须先使用 CD-writer 读出 CD-ROM:

```
# dd if=/dev/scd0 of=cdimage
```

此命令从设备 /dev/scd0 读取 CD-ROM 的内容并将其写入文件 cdimage。此文件的内容等同于 mkisofs 生成的内容,因此您可以按照本文档前面所述进行操作（即以文件 cdimage 作为 cdrecord 的输入）。

加密文件系统

Linux 具有本机文件系统加密支持。您可以选择多种对称加密算法来加密您的文件系统,即:Twofish、高级加密标准 (AES) 也称为 Rijndael、数据加密标准 (DES) 等。 Twofish 和 Rijndael 一样也是 AES 候选者。由于性能原因,Rijndael 被选为新的

版权所有 2013-2017 Snow BV

Machine Translated by Google

AES 标准。 Twofish 支持 128 位块大小和高达 256 位的密钥。 DES 是 AES 标准的前身,由于密钥尺寸小,现在被认为是不安全的。使用当前的计算能力,可以在相对较短的时间内暴力破解 56 位（+8 奇偶校验位)DES 密钥。 AES 使用 128 位的块大小和 128、192 或 256 位的密钥大小。多年来,128 位密钥大小就足够了,但随着量子计算机的引入,美国国家安全局发布了使用 256 位密钥的最高机密数据分类指南。英特尔于 2010 年推出了高级加密标准新指令 (AES-NI) 集。这种新的指令集完全在硬件中执行加密和解密,有助于降低侧信道攻击的风险并大大提高 AES 性能。要检查您的 CPU 是否支持 AES-NI 指令集,请使用以下命令:grep aes /proc/cpuinfo。

您可以使用以下命令检查 AES-NI 内核支持:sort -u /proc/crypto | grep 模块并使用命令加载驱动程序:modprobe aesni-intel 作为 root。

Device Mapper 从 linux 2.6 开始,可以使用 devicemapper,这是一个通用的 linux 框架,可以将一个块设备映射到另一个。 Devicemapper 用于软件 RAID 和 LVM。它用作虚拟块设备上的文件系统和要写入硬盘的加密数据之间的过滤器。这使文件系统能够呈现自己的解密状态,而读取和写入的所有内容都将在磁盘上加密。在 /dev/mapper 中创建了一个虚拟块设备,它可以用作任何其他块设备。在映射到另一个块设备之前,进出它的所有数据都会进入加密或解密过滤器。

Device Mapper crypt (dm-crypt) Device mapper crypt 通过使用内核 API 为块设备的透明加密提供了一种通用方法,并且可以与 RAID 或 LVM 卷结合使用。当用户创建一个新的块设备时,他可以指定几个选项:对称密码、加密模式、密钥和 iv 生成模式。 Dm-crypt 不像 LUKS 那样在标头中存储任何信息。加密磁盘后,它将与具有随机数据的磁盘无法区分。

这意味着在无法证明加密数据存在的意义上,加密文件的存在是可否认的。用户应该跟踪 dm-crypt 设置中使用的选项,否则可能会导致数据丢失,因为没有可用的元数据。只能使用一个加密密钥来加密和解密块设备,不能使用主密钥。一旦加密块设备的密码丢失,就不可能恢复数据。 Dm-crypt 只应由高级用户使用。普通用户应该使用 LUKS 进行磁盘加密。

Linux 统一密钥设置 (LUKS) LUKS 是所有通用 Linux 发行版上的标准实用程序。它为不同类型的卷（普通 dm-crypt 卷、LUKS 卷等)提供磁盘加密。它提供了发行版之间的兼容性,并提供了用户密码的安全管理。它还提供加密选项的存储,包括分区标头中的主密钥,从而实现无缝传输或数据迁移。 LUKS 还为每个 LUKS 分区提供多达 8 个不同的密钥,以启用密钥托管（按含义使用密钥）。通过在不同的 Linux 发行版上创建加密分区,默认设置可能会有所不同,但这些设置足以保护卷上的数据。 LUKS 是普通用户首选的数据保护方法。

dm-crypt 示例 这是设置加密文件系统的示例。所有相关模块都应在启动时加载:

```
# echo aes >> /etc/modules # echo
dm_mod >> /etc/modules # echo dm_crypt >> /
etc/modules # modprobe -a aes dm_mod dm_crypt
```

创建设备映射器块设备并为其使用（例如)hda3。使用以下方法选择密码:cryptsetup -y create crypt /dev/hda3 映射设备:

```
# echo    crypt /dev/hda3 none none    >> /etc/crypttab # echo    /dev/mapper/
crypt /crypt reiserfs defaults 0 1    >> /etc/fstab
```

制作一个文件系统:

```
# mkfs.reiserfs /dev/mapper/crypt
```

现在挂载你的加密文件系统。系统将提示您输入使用 cryptsetup 选择的密码。您将被要求在每次启动时提供它:

```
# mkdir /crypt # 挂载 /
crypt
```

LUKS 示例 这是在 linux 环境中创建 512 MiB 加密 LUKS 容器的示例。

```
# dd if=/dev/urandom of=/root/encrypted bs=1M count=512
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

使用 LUKS 格式化稀疏文件并提供密码：

```
# cryptsetup -y luksFormat 加密

警告！
========
这将不可撤销地覆盖加密的数据。

你确定吗？ （键入大写是）:是 输入密码： 验证密码：
```

打开容器,提供名称并输入密码：

```
# cryptsetup luksOpen 加密加密
输入加密密码：
```

在未加密的容器上创建一个文件系统：

```
mkfs.ext4 /dev/mapper/encrypted mke2fs 1.43.3
（2016 年 9 月 4 日）
创建具有 522240 个 1k 块和 130560 个 inode 的文件系统文件系统 UUID:9fb28cc0-5a3e-4e0b-b3cc-3cbd6cf3c8f4 存储在
块上的超级块备份:8193、24577、40961、57345、73729、204801、2211459、4



分配组表:完成
写入 inode 表:完成
创建日志（8192 块）:完成
编写超级块和文件系统记帐信息:完成
```

挂载加密容器使用：

```
# 挂载 /dev/mapper/encrypted /mnt
```

加密容器作为文件系统：

```
# df -h
文件系统udev                已用大小 已用 使用百分比 安装于
                            10M      0 10M 0% /dev 792M 26M 767M
tmpfs /                     4% /run 29G 16G 11G 60% / 2.0G 400K 2.0G 1% /
dev/sda1                    dev/shm 0 5.0M 0% /run/lock 0 2.0G 0% /sys/fs/
tmpfs tmpfs                 cgroup 396M 16K 396M 1% /运行/用户/133 396M 20K 396M
tmpfs tmpfs                 5.0M     1% /运行/用户/0
tmpfs /dev/                 2.0G
mapper/
encrypted
486M 2.3M 455M 1% /mnt
```

/mnt 的内容与任何 ext4 文件系统一样：

```
#ls/mnt
失物招领
```

使用后卸载加密容器：

```
# 卸载 /mnt
```

关闭加密容器：

```
# cryptsetup luksClose /dev/mapper/encrypted
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

问题和解答

文件系统和设备

1. 为什么要开发Linux File Hierarchy?

   某些文件和实用程序的位置未标准化导致在不同 Linux 发行版之间进行开发和升级时出现问题。 Linux 文件层次结构[77]

2. UNIX 文件系统结构的 inode 部分是否包含文件名?

   不,inode 包含文件的所有信息,除了它的名字。 UNIX 文件系统结构[78]

3. 擦除安装在 /mnt 的 ext2 文件系统中所有数据的最快方法是什么?

   命令 mkfs -t ext2 /mnt 将通过创建一个新的空文件系统来擦除 /mnt 文件系统中的所有数据。当然,部分数据可能仍然可以恢复。擦除文件系统中的所有数据[78]

4. 伪目录 /proc 中的哪个文件包含与命令 free 基本相同的交换信息?

   文件 /proc/meminfo 包含类似的信息。交换信息[81]

5. 在重新引导系统之前,您会键入什么以确保在引导期间不检查 ext 文件系统?

   使用例如 tune2fs -c 5 -C 0 /dev/hda1 以及所有其他设备,以推迟 5 次重新安装的文件系统检查。使用 tune2fs [85]

6. 为了以只读模式运行 debugfs,你会使用什么开关?

   默认情况下,Debugfs 以只读模式运行,因此不需要任何开关。使用 debug2fs [86]

7、自动挂载的主要目的是什么?

   它避免了在许多情况下必须使用 mount 命令的必要性。挂载和卸载是在访问要完成挂载的目录时自动完成的。这在网络环境和可移动设备（例如 USB 连接磁盘和 CD-ROM)中很有用。 Autofs 和自动挂载器[93]

8. 如果将软驱和光驱合并到同一个补充文件（autofs)中会怎样?

   每个文件将只有一个为其运行的自动挂载程序,因此如果一个条目失败,另一个也将不起作用。每个条目一个补充文件[93]

9. 哪两个命令用于创建可挂载的CD-ROM?

   mkisofs 用于创建 ISO9660 映像,cdrecord 用于将该映像写入 CD-ROM。 CD-ROM 文件系统[95]

版权所有 2013-2017 Snow BV

Machine Translated by Google

第 4 章

高级存储设备管理 (204)

本主题的权重为 8 分,包含以下目标:

目标 204.1;配置 RAID（3 分） 考生应该能够配置和实施软件 RAID。此目标包括使用和配置 RAID 0、1 和 5。

目标 204.2； Adjusting Storage Device Access（2分)考生应该能够配置内核选项来支持各种驱动器。该目标包括用于查看和修改硬盘设置的软件工具。

目标 204.3;逻辑卷管理器（3 分） 考生应该能够创建和删除逻辑卷以及物理卷。此目标包括快照和调整逻辑卷的大小。

配置 RAID (204.1)

考生应该能够配置和实施软件 RAID。此目标包括使用和配置 RAID 0、1 和 5。

资源： LinuxRef01;罗宾斯01;酒吧00; mdadm 和 mdadm.conf 的联机帮助页。

关键知识领域

- 软件 raid 配置文件和实用程序

条款和实用程序

- mdadm
- mdadm.conf
- fdisk
- /proc/mdstat

什么是RAID?

RAID 代表“廉价磁盘冗余阵列” 1。

[1] 许多人坚持认为“我”代表“独立”

版权所有 2013-2017 Snow BV

Machine Translated by Google

RAID 背后的基本思想是将多个小型、廉价的磁盘驱动器组合成一个阵列,从而产生超过一个大型且昂贵的驱动器的性能。该驱动器阵列在计算机上将显示为单个逻辑存储单元或驱动器。

本章中使用的一些概念是决定在特定情况下使用哪个级别的 RAID 的重要因素。
奇偶校验是一个用于向存储添加冗余的概念。奇偶校验位是添加的二进制数字,以确保给定的一组位中值为 1 的位数始终为偶数或奇数。通过巧妙地使用 RAID 的部分容量来存储奇偶校验位,单个磁盘故障可以发生而不会丢失数据,因为可以使用奇偶校验位重新计算丢失的位。

I/O 事务是进出 RAID 设备或其成员的数据移动。每个 I/O 事务由一个或多个数据块组成。单个磁盘每秒可以处理最大随机数的事务,因为该机制在读取或写入数据之前有一个寻道时间。根据 RAID 的配置,对 RAID 的单个 I/O 事务可以触发对其成员的多个 I/O 事务。这会影响 RAID 设备在最大 I/O 事务数和数据传输速率方面的性能。数据传输率是单个磁盘或 RAID 设备每秒可以处理的数据量。该值通常因读写操作、随机或顺序访问等而异。

RAID 是一种将信息分布在多个磁盘上的方法,使用磁盘条带化（RAID 级别 0）、磁盘镜像（RAID 级别 1)和带分布式奇偶校验的条带化（RAID 级别 5)等技术,以实现冗余、降低延迟和/或用于读取和/或写入磁盘的更高带宽,并最大限度地提高硬盘崩溃的可恢复性。

RAID 的基本概念是数据可以以一致的方式分布在阵列中的每个驱动器上。为此,必须首先将数据分成大小一致的块（通常大小为 32K 或 64K,但也可以使用不同的大小）。然后将每个块依次写入每个驱动器。当要读取数据时,过程相反,给人一种多个驱动器实际上是一个大驱动器的错觉。使用 RAID 的主要原因包括：

- 提高传输速度
- 提高每秒事务数
- 增加单块设备容量
- 从单个磁盘故障中恢复的效率更高

RAID 级别

有许多不同的方法来配置 RAID 子系统　一些最大化性能,其他最大化可用性,而其他提供两者的混合。对于 LPIC-2 考试,以下内容是相关的：

- RAID-0（条带化）
- RAID-1（镜像）
- RAID-4/5
- 线性模式

级别 0 RAID 级别 0,通常称为“条带化”,是一种面向性能的条带化数据映射技术。这意味着写入阵列的数据被分解成条带并写入阵列的成员磁盘。这允许以低固有成本实现高 I/O 性能,但不提供冗余。阵列的存储容量等于成员盘容量的总和。因此,RAID 0 主要用于需要高性能且能够容忍较低可靠性的应用程序。

级别 1 RAID 级别 1,或“镜像”,使用时间比任何其他形式的 RAID 都长。级别 1 通过将相同的数据写入阵列的每个成员磁盘来提供冗余,在每个磁盘上留下一个镜像副本。由于其简单性和高水平的数据可用性,镜像仍然很受欢迎。级别 1 与两个或多个磁盘一起运行,这些磁盘在读取时可以使用并行访问来实现高数据传输率,但更常见的是独立运行以提供高 I/O 事务率。级别 1 提供非常好的数据可靠性并提高读取密集型应用程序的性能,但成本相对较高。阵列容量等于最小成员盘的容量。

级别 4 RAID 级别 4 使用集中在单个磁盘驱动器上的奇偶校验来保护数据。它更适合事务 I/O 而不是大文件传输。因为专用奇偶校验磁盘代表了一个固有的瓶颈,所以很少使用 4 级

版权所有 2013-2017 Snow BV

Machine Translated by Google

回写缓存等伴随技术。阵列容量等于成员盘的容量减去一个成员盘的容量。

5 级 最常见的 RAID 类型是 5 级 RAID。通过在阵列的部分或所有成员磁盘驱动器之间分配奇偶校验,RAID 级别 5 消除了级别 4 中固有的写入瓶颈。唯一的瓶颈是奇偶校验计算过程。
因为现代 CPU 和软件 RAID 的广泛使用不再是真正的问题。与级别 4 一样,结果是不对称的性能,读取性能大大优于写入。级别 5 通常与回写缓存一起使用以减少不对称性。阵列的容量等于所有成员盘的总容量减去一个成员盘的容量。在单个成员磁盘发生故障时,可以根据分布式奇偶校验计算后续读取,从而不会丢失任何数据。
RAID 5 至少需要三个磁盘。

线性RAID

线性 RAID 是一种简单的驱动器分组,用于创建更大的虚拟驱动器。在线性 RAID 中,块从一个成员驱动器按顺序分配,仅当第一个驱动器完全填满时才转到下一个驱动器。与“条带化”的不同之处在于单进程应用程序没有性能提升,大多数情况下所有内容都写入同一个磁盘。磁盘（分区)可以有不同的大小,而“条带化”要求它们的大小大致相同。如果在线性 RAID 设置中有大量最常用的磁盘,则多个进程可能会在读取期间受益,因为每个进程可能会访问不同的驱动器。线性 RAID 也不提供冗余,实际上降低了可靠性　如果任何一个成员驱动器发生故障,则整个阵列都无法使用。

容量是所有成员盘的总和。

RAID 可以用硬件或软件实现;这两种情况都在下面解释。

硬件RAID

基于硬件的系统独立于主机管理 RAID 子系统,并且每个 RAID 阵列仅向主机提供一个磁盘。

典型的硬件 RAID 设备可能连接到 SCSI 控制器并将 RAID 阵列呈现为单个 SCSI 驱动器。外部 RAID 系统将所有 RAID 处理智能转移到位于外部磁盘子系统中的控制器中。

RAID 控制器也以卡的形式出现,就像操作系统的 SCSI 控制器一样,但它们自己处理所有实际的驱动器通信。在这些情况下,您可以像插入 SCSI 控制器一样将驱动器插入 RAID 控制器,然后将它们添加到 RAID 控制器的配置中,而操作系统永远不知道其中的区别。

软件RAID

软件 RAID 在内核磁盘（块设备)代码中实现各种 RAID 级别。它还提供最便宜的解决方案:不需要昂贵的磁盘控制器卡或热插拔机箱,软件 RAID 可与更便宜的 SATA 磁盘和 SCSI 磁盘一起使用。与当今的快速 CPU 相比,软件 RAID 性能可以优于硬件 RAID。

软件 RAID 允许您显着提高 Linux 磁盘 I/O 性能和可靠性,而无需购买昂贵的硬件 RAID 控制器或机箱。 Linux 内核中的 MD 驱动程序是完全独立于硬件的 RAID 解决方案的示例。基于软件的阵列的性能取决于服务器 CPU 性能和负载。此外,软件 RAID 解决方案的实施和设置会显着影响性能。

软件 RAID 背后的概念很简单　它允许您将两个（对于 RAID5 至少三个)或更多块设备（通常是磁盘分区)组合成一个 RAID 设备。因此,如果您有三个空分区（例如:hda3、hdb3 和 hdc3）,您可以使用软件 RAID 组合这些分区并将它们作为单个 RAID 设备 /dev/md0 进行寻址。 /dev/md0 然后可以格式化为包含文件系统并像任何其他分区一样使用。

识别 Linux 系统上的 RAID

在 Linux 系统上检测硬件 raid 可能很棘手,并且没有一种可靠的方法可以做到这一点。由于硬件 RAID 试图将自己呈现给操作系统作为单个块设备,因此在查询系统时它通常显示为单个 SCSI 磁盘。
通常需要特殊的供应商软件或对设备的物理访问才能充分检测和识别硬件 RAID 设备。软件 raid 可以通过块设备的名称 (/dev/mdn) 及其主编号 9 轻松识别。

版权所有 2013-2017 Snow BV

Machine Translated by Google

配置 RAID（使用 mdadm）

使用 mdadm（多设备管理)配置软件 RAID 只需要将 md 驱动程序配置到内核中,或作为内核模块加载。可选的 mdadm.conf 文件可用于指导 mdadm 简化常见任务,例如定义多个阵列,以及定义它们使用的设备。 mdadm.conf 有许多可能的选项,如本文档后面所述,但一般来说,该文件会详细说明阵列和设备。应该注意的是,虽然不是必需的,但 mdadm.conf 大大减轻了管理员在启动时“记住”所需阵列配置的负担。 mdadm 用于（如其首字母缩写词所示)配置和管理多个设备。多个是这里的关键。为了让 RAID 为逻辑磁盘提供任何类型的冗余,阵列中显然必须至少有两个物理块设备（RAID5 为三个)以建立冗余,因此保护。由于 mdadm 管理多个设备,因此它的应用不仅限于 RAID 实现,还可以用于建立多路径。 mdadm 有多种模式,如下所列

组装“重建”一个预先存在的阵列。通常在将阵列迁移到新主机时使用,但更常在系统启动时使用 Build 不创建阵列超级块,因此不会破坏任何预先存在的数据。尝试记录时可能有用 Create 使用预先存在的块设备从头开始创建数组,并激活该数组。

增长 用于修改现有阵列,例如添加或删除设备。预计在 de Misc 期间将扩展功能,用于执行各种松散捆绑的内务处理任务。可用于为 exi 生成初始 mdadm.conf 文件

Linux：/etc/mdadm.conf 和 /etc/rc.d/rc.local（或等效文件）。 Linux 内核提供了一个特殊的驱动程序 /dev/md0 来访问作为逻辑 RAID 单元的单独磁盘分区。 RAID下的这些分区其实不需要是不同的磁盘,但是为了排除风险最好还是使用不同的磁盘。 mdadm 也可用于建立多路径,也可以在文件系统设备上建立（因为多路径是在块级建立的）。然而,与在同一物理磁盘上的多个文件系统上建立 RAID 一样,同一物理磁盘上的多路径仅提供冗余的模糊幻觉,它的使用可能仅限于测试目的或纯粹出于好奇。

按照以下步骤在 Linux 中设置软件 RAID：

1.配置RAID驱动

2.初始化RAID驱动器

3. 使用 /proc/mdstat 检查复制

4.重启后自动激活RAID

5. 将文件系统挂载到 RAID 驱动器上

现在详细描述这些步骤中的每一个：

初始化要在 RAID 设置中使用的分区 使用任何磁盘分区工具创建分区。

配置驱动程序 当 mdadm 创建阵列时,将自动为每个独立阵列创建驱动程序文件,并将遵循约定 /dev/md[n] 用于每个增量。它也可以使用 mknod 手动创建并构建块设备文件,主设备号为 9（md 设备驱动程序位于 /proc/devices）。

初始化 RAID 驱动器

以下是创建和格式化 RAID 驱动器(mdadm) 所需的命令序列示例：

1. 使用 fdisk 为自动 RAID 检测准备分区

   为了让一个分区被自动识别为 RAID 集的一部分,它必须首先将其分区类型设置为“fd”。
   这可以通过使用 fdisk 命令菜单并使用 t 选项更改设置来实现。可以使用 l 菜单选项列出可用的设置。描述可能因实现而异,但应该清楚地表明类型是 Linux auto raid。设置后,必须使用 w 选项将设置保存回分区表。

   在工作实践中,包含选定分区以用于 RAID 组的物理磁盘可能还包含不打算包含在 RAID 组中的本地文件系统分区。为了让 fdisk 写回更改后的分区表,物理磁盘上的所有分区都不能被使用。出于这个原因,RAID 构建规划应考虑可能不允许操作真正在线执行的因素（即需要停机）。

版权所有 2013-2017 Snow BV

Machine Translated by Google

2. 使用 mdadm 创建数组 raidset

首次创建阵列时,我们需要确定将用于形成 RAIDset 的分区,并使用 fdisk -l 验证是否已设置 fd 分区类型。一旦实现了这一点,就可以按如下方式创建数组。mdadm -C /dev/md0 -l raid5 -n 3 /dev/partition1 /dev/partition2 /dev/partition3 这将创建并激活一个名为 md0 的 raid5 阵列,包含三个设备,名为 /dev/partition1 /dev/partition2和 /dev/partiton3。创建并运行后,可以通过 cat ing /proc/mdstat 检查阵列的状态。

3. 在新创建的 raidset 上创建文件系统

然后可以使用 /dev/md0 设备文件将新创建的 RAID 集作为单个设备进行寻址,并正常格式化和安装。例如:mkfs.ext3 /dev/md0。

4. 使用 mdadm 命令创建 /etc/mdadm.conf。

创建 /etc/mdadm.conf 文件非常简单,只需要使用 --scan、--verbose 和 --detail 选项调用 mdadm,并重定向其标准输出。因此,它也可以用作一种方便的工具,通过简单地区分当前和存储的输出来确定阵列上的任何变化。创建如下:

mdadm --detail --scan --verbose > /etc/mdadm.conf

5.创建挂载点并编辑/etc/fstab

必须注意将 RAID 集中使用的任何回收分区从 /etc/fstab 中删除（如果它们仍然存在）。

6.使用mount挂载文件系统

如果阵列已在线创建,并且尚未执行重启,则需要手动挂载文件系统,可以使用 mount 命令手动挂载,也可以简单地使用 mount -a

使用 /proc/mdstat 检查复制 /proc/mdstat 文件显示内核 RAID/md 驱动程序的状态。

重启后自动激活 RAID

在启动文件之一（例如 /etc/rc.d/rc.sysinit 或 /etc/init.d/rcS)中运行 mdadm --assemble当使用 -s 选项（扫描)调用 mdadm --assemble 时,指示mdadm 使用 /etc/mdadm.conf（如果存在）,否则回退到 /proc/mdstat 以获取缺少的信息。例如,典型的系统启动条目可能是 mdadm --assemble -s。

将文件系统挂载到 RAID 驱动器 编辑 /etc/fstab 以自动将文件系统挂载到 RAID 驱动器上。 /etc/fstab 中的条目与包含文件系统的普通块设备相同。

手动 RAID 激活和安装 为每个要手动启动的 RAID 块设备运行 mdadm --assemble。
After starting the RAID device you can mount any filesystem present on the device using mount with the appropriate options.

配置 RAID（备选）

它可以通过 /etc/raidtab 配置并通过 mkraid、raidstart 和 raidstop 实用程序控制,而不是通过 mdadm 管理 RAID。

这是一个示例 /etc/raidtab:

```
#
# 示例 /etc/raidtab 配置文件 #

突袭开发 /dev/md0
    突袭级                          0
    nr-raid-磁盘                    2个
    持久超级块 0 块大小 8


    设备                            /dev/hda1
    磁盘                            0
    设备                            /dev/hdb1
    磁盘                            1个
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
突袭开发 /dev/md1
    突袭级                                  5个
    nr-raid-磁盘                            3个
    nr-spare-disks persistent-              1个
    superblock 1 parity-algorithm 左对称


    设备                                    /开发/sda1
    磁盘                                    0
    设备                                    /开发/sdb1
    磁盘                                    1个
    设备                                    /开发/sdc1 2
    磁盘
    设备                                    /开发/sdd1 0
    备用磁盘
```

mkraid、raidstart 和 raidstop 实用程序都使用此文件作为输入，并分别配置、激活（启动)和取消配置（停止)RAID 设备。所有这些工具的工作原理都相似。如果指定了 -a（或 --all），则对配置文件中提到的所有 RAID 设备执行指定的操作。否则，必须在命令行中指定一个或多个 RAID 设备。

有关详细信息，请参阅 raidtab(8)、mdstart(8) 和 raidstart、raidstop(8) 的手册页。

调整存储设备访问权限 (204.2)

考生应该能够配置内核选项以支持各种驱动器。此目标包括用于查看和修改硬盘设置（包括 iSCSI 设备)的软件工具。

资源： LinuxRef02;罗宾斯01.2； LinuxRef03； USBRef01； LinuxRef04； LinuxRef05;威尔00;卢特尔;各种命令的手册页。

关键知识领域

- 为 IDE 设备（包括 ATAPI 和 SATA)配置 DMA 的工具和实用程序
- 用于操作或分析系统资源（例如中断)的工具和实用程序
- 了解 sdparm 命令及其用途
- iSCSI 的工具和实用程序
- SSD 和 NVMe 配置以及 SAN 感知

条款和实用程序

- 硬盘参数
- sdparm
- tune2fs
- 系统控制
- iscsiadm、scsi_id、iscsid 和 iscsid.conf
- /dev/hd* 和 /dev/sd*
- WWID、WWN、LUN 编号

版权所有 2013-2017 Snow BV

Machine Translated by Google

配置磁盘

配置 iSCSI

iSCSI 是 Internet 小型计算机系统接口的缩写。 iSCSI 只是众所周知的标准化 SCSI 协议的网络实现。 SCSI 定义了用于访问各种设备的模型、布线、电信号和命令。 iSCSI 仅使用模型和命令。 SCSI（以及因此:iSCSI)可用于访问各种设备,虽然主要用于磁盘和磁带机,但也已用于访问 USB 存储、打印机和平板扫描仪,仅举几例。

iSCSI 设置类似于 SCSI 设置,因为它由客户端和服务器组成。介于两者之间的是基于 IP 的网络,可以是 LAN、WAN 或 Internet。客户端像往常一样发出低级 SCSI 命令（例如读、写、查找或擦除）。命令可以由客户端加密,然后封装并通过网络发送。服务器接收并可能解密命令并执行命令。与需要特殊用途布线的传统光纤通道不同,iSCSI 可以使用现有网络基础设施进行长距离运行。

客户被称为“发起者”。这些命令称为“CDB”（命令描述符块）,服务器存储设备称为“目标”。

释义：

iSCSI 目标
iSCSI 启动器

导出的存储实体是目标,导入的实体是发起者。

iSCSI 启动器

在充当 iSCSI 服务器（目标)的客户端（启动器)的 Linux 主机上,您需要安装客户端软件。您还需要编辑配置文件,以便发现所需的目标。在 Red Hat Linux 及其衍生产品上,配置文件是 /etc/iscsi/iscsid.conf。编辑文件,使其指向正确的服务器（目标)并包含一组有效的凭据 - 请参阅本段底部的示例文件。

配置完成后,使用以下命令确保 iSCSI 服务运行：

```
/etc/init.d/iscsi 状态
```

如果需要,使用以下命令启动 iSCSI 服务：

```
/etc/init.d/iscsi 开始
```

使用 iscsiadm 命令发现目标。在我们的示例显示问号的地方,替换为目标的 IP 地址：

```
iscsiadm -m discovery -t sendtargets -p ????.????.????.????
```

成功发现目标后,您可以登录到已发现的卷。这可以使用 iscsiadm 工具完成,该工具也可用于注销卷。添加新发现的卷的快速方法是重新启动 iSCSI 服务：

```
/etc/init.d/iscsi 重启
```

小费
将 iscsi 挂载点添加到fstab文件时,请使用_netdev选项：

```
/dev/sdb1 /mnt/iscsi ext3(4) _netdev 0 0
```

_netdev选项确保在尝试安装设备之前网络已启动。

示例 /etc/iscsi/iscsid.config：

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
################ # iSNS
设置 ################ #
iSNS 服务器地址 isns.address
= ***.** *.***.***

isns.port = 3260

# *************
# CHAP 设置 # *************


# 要启用 CHAP 身份验证设置 node.session.auth.authmethod
# 到第一章。默认值为无。
node.session.auth.authmethod = CHAP

# 为发起者设置 CHAP 用户名和密码 # 由目标进行身份验证,取消注释以下行:
node.session.auth.username = ********

node.session.auth.password = ********

# 为到目标的发现会话启用 CHAP 身份验证 # 将 discovery.sendtargets.auth.authmethod 设置为 CHAP。默认值为无。发
现.sendtargets.auth.authmethod = CHAP


# To set a discovery session CHAP username and password for the initiator # authentication by the target(s) 取消注释以下行:
discovery.sendtargets.auth.username = ******** discovery.sendtargets.auth.password = ********
```

iSCSI 目标

有多种方法可以设置目标。 SCSI 目标框架 (STGT/TGT) 是 linux 2.6.38 之前的标准。
当前标准是 LIO 目标。

另一个著名的实现是 iSCSI Enterprise Target (IET) 及其后继者 SCSI Target Subsystem (SCST)。
IET 也是内核包含的候选者,但 LIO 是最终选择。 LIO（linux-iscsi.org 的缩写)现在是 Linux 中用于共享数据存储的标准开源 SCSI 目标。它支持所有流行的存储结构,包括光纤通道 (QLogic)、FCoE、iEEE 1394、iSCSI、iSER (Mellanox InfiniBand)、SRP (Mellanox InfiniBand)、USB、vHost 等。

添加新目标（在目标主机上）

要添加新的 iSCSI 目标,请编辑 /etc/tgt/targets.conf 配置文件。该文件包含许多已被注释掉的不同配置选项的示例。基本目标可以定义为:

```
<target iqn.2008-09.com.example:server.target1> backing-store /srv/images/
        iscsi-share.img direct-store /dev/sdd </target>
```

此示例定义了一个具有两个 LUN 的目标。 LUN 使用 backing-store 或 direct-store 指令进行描述,其中 backing-store 指的是文件或块设备,而 direct-store 指的是本地 SCSI 设备。设备参数（例如序列号和供应商名称)将传递到新的 iSCSI LUN。

目标服务被恰当地命名为 tgtd。要启动它运行:

```
# 服务 tgtd 启动
```

要停止 tgtd 服务,请运行:

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
# 服务 tgtd 停止
```

如果有打开的连接,请使用:

```
# service tgtd force-stop 警告 使用此命令将
终止所有目标阵列。
```

世界大战

任何连接的 SCSI 设备都被分配了一个唯一的设备名称。设备名称以/dev/sd 开头,例如/dev/sdd、/dev/sda。此外,每个 SCSI 设备都有一个唯一的全球标识符 (WWID)。标识符可以通过发出查询命令从设备本身获得。在 Linux 系统上,您可以在 /dev/disk/by-id/ 目录中找到 WWID,您可以在其中找到名称包含 WWID 并指向分配的设备名称的符号链接。

有两种类型的 WWID,称为“页面 0x83”和“页面 0x80”,任何 SCSI 设备都有其中之一。

例如,具有页面 0x83 标识符的设备将具有:

```
scsi-3600508b400105e210000900000490000 -> ../../sda
```

或者,具有页面 0x80 标识符的设备将具有:

```
scsi-SSEAGATE_ST373453LW_3HW1RHM6 -> ../../sda
```

Red Hat Enterprise Linux 自动维护从基于 WWID 的设备名称到该系统上当前 /dev/sd 名称的正确映射。应用程序可以使用 /dev/disk/by-id/ 名称来引用磁盘上的数据,即使设备路径发生变化,甚至在从不同系统访问设备时也是如此。这也允许检测和访问可插入设备,比如照相机。如果已插入,系统将发送查询命令并创建正确的符号链接,从而允许您以相同的名称访问摄像机。

为设备提供不止一条路径是可能且可行的,例如提供故障转移或回退选项和/或提高性能。这样的“路径”可能被实现为额外的网络路径(最好使用他们自己的网卡)或通过光纤(最好使用他们自己的 HBA)等。如果从系统到设备有多个路径,一个特殊的内核模块和相关的守护进程和其他软件将使用 WWID 来检测路径。 /dev/mapper/wwid 中的单个“伪设备”将由他们创建,例如 /dev/mapper/3600508b400105df70000e00000ac0000,这将提供对设备的访问权限,无论使用的是哪个路径,只要至少其中之一可以使用(“处于活动状态”)。

命令 multipath -l显示到非持久标识符的映射:Host:Channel:Target:LUN、/dev/sd 名称和 major:minor 编号:

```
3600508b400105df70000e00000ac0000 dm-2 供应商,产品 [size=20G][features=1
 queue_if_no_path][hwhandler=0][rw] \_ round-robin 0 [prio=0][active] \_ 5:0:1:1 sdc 8:32 [active]
[undef] \_ 6:0:1:1 sdg 8:96 [active][undef] \_ round-robin 0 [prio=0][enabled] \_ 5:0:0: 1 sdb 8:16
[活动][undef] \_ 6:0:0:1 sdf 8:80 [活动][undef]
```

Device-mapper-multipath 自动维护每个基于 WWID 的设备名称与其在系统上对应的 /dev/sd 名称的正确映射。这些名称在交叉路径更改时是持久的,并且在从不同系统访问设备时它们是一致的。使用 user_friendly_names 功能(device-mapper-multipath 的)时,WWID 映射到 /dev/mapper/mpathn 形式的名称。默认情况下,此映射保存在文件 /etc/multipath/bindings 中。只要文件得到维护,这些 mpath 名称就会一直存在。

笔记

重要 如果您使用 user_friendly_names,则需要额外的步骤才能在集群中获得一致的名称。请参阅《使用 DM 多路径配置和管理》一书中的“集群中一致的多路径设备名称”部分。

版权所有 2013-2017 Snow BV

Machine Translated by Google

伦

另一种寻址 SCSI 设备的方法是使用其 LUN,即逻辑单元号。 LUN 是一个三位数 (0..7),表示目标内的设备。任何目标都应该至少有一个带有 LUN 0 的设备,它可以像任何设备一样使用,但也能够提供有关设备中其他 LUN 的信息（如果有的话）。如果目标中只有一个设备,则该设备具有 LUN 0,因为任何目标都应具有 LUN 0。

假设您购买了包含三个磁盘的 SCSI 设备（全部通过同一根 SCSI 电缆连接）。要查明目标（或磁带机等)中有多少磁盘,您可以向 LUN 0 发送查询。具有 LUN 0 的设备旨在能够告诉您配置并可能报告它知道其他设备。为解决这些问题,客户端使用 LUN 0 指定的 LUN。

因为 SCSI 磁盘通常由它们的 LUN 引用,所以这个习惯一直存在,现在这个术语也用于指代 SAN 中的逻辑磁盘。但是 SAN 中的 LUN 数量可能比旧配置中使用的不起眼的 7 个 LUN 多得多。此外,这样的 SAN-LUN 很可能由一系列磁盘组成,这些磁盘可能被配置为 RAID,可能细分为卷等 - 对于客户端而言,它只是一个带有 LUN 的磁盘。

配置设备名称持久性（也称为 LUN 持久性）

Linux 设备名称可能会在引导时更改。这可能导致混淆和数据损坏。因此,有许多技术允许设备的持久名称。本节介绍如何在具有和不具有多路径的情况下在来宾和主机上实现设备名称持久性。持久性只是意味着无论设备的访问路径如何,您都将拥有相同的设备名称。因此,无论您将其插入何处,您的相机或磁盘始终会被识别并获得相同的设备名称。

在没有多路径的情况下实现设备名称持久化

如果您的系统没有使用多路径,您可以使用 udev 来实现 LUN 持久性。 udev 是一个设备管理器,它监听与内核的接口。如果插入新设备或正在分离设备,udev 将从内核接收信号,并使用一组规则执行操作,例如为设备分配一个额外的设备名称。

在您的系统中实施 LUN 持久性之前,您首先需要从您要控制的设备获取正确的 WWID。这可以通过使用 scsi_id 程序来完成。在您需要设备的 WWID 之后,您将需要编写一个规则,以便 udev 可以为其分配一个永久名称。

必须首先克服一个小障碍。默认情况下,scsi_id 程序不会列出任何设备,只会列出它被告知要查看的设备（“白名单”设备）。要将设备列入白名单,您需要将其 WWID 添加到 scsi_id 配置文件,即 /etc/scsi_id.conf。这很安全,但如果您想发现新设备的 WWID,那就麻烦了。因此,您可以指定 -g 选项以允许检测尚未列出的任何磁盘。如果您认为这应该是默认行为,您可以考虑编辑 /etc/scsi_id.conf。如果有一行说：

```
选项=-b
```

删除它或注释掉它。然后添加一行内容：

```
选项=-g
```

如果已配置,下次程序使用 scsi_id 命令时,它将添加 -g 选项（以及您可能在配置文件中指定的任何其他选项）,从而允许从新设备中提取 WWID。 -g 应该在配置文件中默认使用,或者在任何使用 scsi_id 命令的 udev 规则中手动设置,否则您新插入的设备将不会被识别。

然后您将需要找出您的新设备由内核分配的设备文件。这通常是在插入新设备后使用命令 dmesg 完成的。通常,您会看到类似于以下内容的行：

```
usb 1-5:使用 ehci_hcd 和地址 25 的新高速 USB 设备 usb 1-5:从 1 个选项中选择的配置 #1 scsi7:USB 大容量存储设备的
SCSI 仿真 usb-storage:在 25 找到的设备 usb-storage:等待设备在扫描前安顿下来
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
    供应商:USB                    型号:闪存盘                    修订:8.07
    类型:直接访问                                                ANSI SCSI 修订版:04
SCSI 设备 sdc:7886848 512 字节 hdwr 扇区 (4038 MB)
```

如您所见,我们插入了一个分配有设备名称“sdc”的 4Gb 闪存盘。该设备现在在 /sys 伪文件系统中作为 /sys/ block/sdc 可用,也作为设备文件 /dev/sdc 可用。

要确定设备 WWID,请使用 scsi_id 命令:

```
# scsi_id -g -u -s /块/sdc 3200049454505080f
```

如您所见,我们强行使用了 -g。 -s 选项指定您要使用相对于 sysfs 文件系统 (/sys) 根目录的设备名称。较新版本的 scsi_id（例如 RHEL6 中的版本)不再允许使用 sysfs 名称,因此应该使用设备名称:

```
/sbin/scsi_id -g -u -d /dev/sdc 3200049454505080f
```

输出中的一长串字符是 WWID。当您向系统添加新设备时,WWID 不会更改。获取每个设备的 WWID,以便为设备创建规则。要创建新的设备规则,请编辑 /etc/udev/rules.d 目录中的 20-names.rules 文件。设备命名规则遵循以下格式:

```
KERNEL=  sd<?>  , BUS=  scsi  , PROGRAM=  <scsi_id -options..>  , RESULT=  <WWID>  , NAME=  < ←。
        设备名称>
```

所以,当你插入一个新设备时,内核会向 udev 发出信号,它发现了一个新的 SCSCI 设备。程序将被执行并且结果应该匹配。如果是这样,名称将用于设备文件。

一个例子是:

```
KERNEL=  sd*  , BUS=  scsi  , PROGRAM=  /sbin/scsi_id -g -u -d /dev/$parent  , RESULT ←。
        =  3200049454505080f  , NAME=  bookone
```

或者在 RHEL6 系统上:

```
KERNEL=  sd*  , BUS=  scsi  , PROGRAM=  /sbin/scsi_id -g -u -s /block/$parent  , RESULT ←  =  3200049454505080f  , NAME=  bookone
```

当内核分配一个名称与 KERNEL 模式匹配的新设备时,将执行 PROGRAM,并在找到 RESULT 时创建 NAME。请注意我们包含的 -g 选项,因此我们可以实际看到该设备,尽管它没有列入白名单。

udevd 守护进程通常通过执行本地启动脚本来启动,通常由 init 进程完成并由 /etc/inittab 文件中的一行启动。在旧系统上,您将确保此行位于 /etc/rc.local 中:

```
/sbin/start_udev
```

在某些系统上,例如 RHEL6,该行应该在 /etc/rc.sysinit 中。

使用多路径实现设备名称持久化

您可以使用多路径驱动程序和软件来实现设备名称持久性。当然,通常只有在您的设备有多个路径时才使用它。但是您实际上可以定义一个只有一个活动路径的多路径,它会像任何其他磁盘一样工作。只需定义一个别名,该名称将在整个引导过程中保持不变。

多路径软件由一个守护进程组成,它保护通往各种设备的路径。它在失败时执行外部命令 multipath。多路径命令会在重新配置路径后向守护进程发出信号。

要在多路径环境中实现 LUN 持久性,您可以为多路径设备定义一个别名。在多路径守护进程的配置文件 /etc/multipath.conf 中编辑设备别名:

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
多路径{
        多路径 { wwid
                                3600a0b80001327510000015427b625e
                别名            备份盘
        }
}
```

这定义了一个具有永久设备文件 /dev/mpath/backupdisk 的设备。这将授予对具有 WWID 600a0b80001327510000015427b625e 的设备（无论其名称如何)的访问权限。别名在重新启动后保持不变。

物理安装

要在系统中安装新磁盘,您需要物理安装硬盘驱动器并在计算机的 BIOS 中对其进行配置。
在大多数情况下,Linux 会自动检测新驱动器。你可以输入 dmesg | more 找出驱动器的名称和设备文件。例如:系统中的第二个 IDE 驱动器将命名为 /dev/hdb。我们假设新驱动器是 /dev/hdb。

在较旧的内核上,IDE 设备名称匹配模式 /dev/hd[ad],其中 a 用于主 IDE 设备,b 是主从 IDE 设备,c 是辅助主设备,d 是辅助从 IDE 设备。对于 PATA/SATA 驱动器,使用 Linux SCSI 驱动程序,因为 PATA/SATA 架构类似于 SCSCI。因此较新的设备具有与模式 /dev/sd* 匹配的设备文件。

安装磁盘并被内核识别后,您可以选择（重新)分区新磁盘。作为shell 类型中的root：

```
fdisk /dev/hdb
```

这将带您进入提示

```
命令（m 寻求帮助）：
```

在提示符下,键入p以显示现有分区。如果您有需要删除的分区,请键入d,然后在提示符下键入要删除的分区的编号。接下来,键入n以创建新分区。键入1 （假设这将是新驱动器上的第一个分区） ,然后按Enter。然后系统将提示您选择要在驱动器上启动的气缸。接下来,系统将要求您提供分区的结束柱面。重复这些步骤,直到创建所有分区。

要将干净的文件系统放在新分区驱动器的第一个分区上,请键入：

```
mkfs /dev/hdb1
```

对所有分区重复此步骤。您可以使用-t参数来定义要构建的文件系统类型,例如 ext2、ext3、xfs、minix 等（创建文件系统）。

您现在需要为驱动器上的每个新分区确定挂载点。我们假设/new。类型：

```
mkdir /新
```

为您的驱动器创建安装点。现在您需要编辑 /etc/fstab。您需要在文件末尾创建一个类似这样的条目：

```
/dev/hdb1                /新的          分机2          默认值                1个      1个
```

现代 ssd（固态硬盘)可能会从一些优化中获益。与带盘片的标准硬盘相比,SSD 的速度非常快,但在多次重写块后,它们更有可能出现坏块。 SSD 可以使用 TRIM,您可以有效地丢弃这些块。 noatime 或 relatime 也可用于减少写入磁盘的数量。这将告诉文件系统不要跟踪上次访问时间,而只跟踪上次修改时间。在 /etc/fstab 中它看起来像：

```
/dev/hdb1                /新的          分机4          丢弃,noatime                1个      1个
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

NVM Express (NVMe) 是一种用于访问通过 PCI Express 总线连接的 SSD 的规范。 Linux NVMe 驱动程序包含在内核版本 3.3 及更高版本中。 NVMe 设备可以在 /dev/nvme* 下找到。 NVMe 设备是一个 ssd 系列,不应被丢弃。在使用 ext4 和 LVM 的设置中通常会禁用丢弃,但其他文件系统可能需要明确禁用丢弃。

```
/开发/nvme0n1p1/新 ext4 默认值 0 0
```

将 /tmp 分区移动到内存 (tmpfs) 是减少磁盘写入的另一种方法。如果您的系统有足够的内存,建议这样做。将以下条目添加到 /etc/fstab:

```
没有任何 /tmp/ tmpfs 大小=10% 0 0
```

在为每个分区创建类似的条目后,写入文件。

```
挂载-a
```

会将分区挂载到您在 /etc/fstab 中指定的目录中。

使用 tune2fs

当内核与 ext2 或 ext3 文件系统交互时,可以使用 tune2fs 命令来配置交互。此命令值得注意的选项是:

-e error_behaviour 该选项定义内核在应用该命令的文件系统出现错误时应如何反应。可能的行为是:

continue 忽略文件系统上的错误。向试图访问该部分的应用程序报告读取或写入错误的文件系统。

remount-ro 将文件系统重新挂载为只读。这可以保护已经在光盘上的数据。尝试写入的应用程序到这个文件系统将导致错误。恐慌 这会导致

内核恐慌并且系统将停止。

-m reserved_block_percentage 由于碎片,完整的文件系统性能非常差。为了防止普通用户填满整个文件系统,将保留一定比例的块。普通用户不能使用此保留部分。只有 root 用户可以使用文件系统的这一部分。

预留块的默认百分比为 5%。随着磁盘变得越来越大,预留 5% 将导致大量保留块。对于大型磁盘,1% 的保留块对于大多数文件系统来说应该足够了。

-O [^]mount_option 设置文件系统的挂载选项。命令行选项或 /etc/fstab 中的选项优先。 ^ 符号清除指定的选项。

内核 2.4.20 及更高版本支持此选项。

-s [0|1] 启用或禁用稀疏超级块功能。在大型文件系统上启用此功能可以节省空间,因为更少使用超级块。

启用或禁用此功能后,必须通过运行 e2fsck 命令使文件系统再次有效。

使用 hdparm

hdparm 命令用于设置/获取 SATA/IDE 设备参数。它为 Linux SATA/PATA/SAS“libata”子系统和较旧的 IDE 驱动程序子系统支持的各种内核接口提供命令行接口。较新（自 2008 年起）
USB 磁盘盒现在支持“SAT”（SCSI-ATA 命令转换）,因此它们也可以与 hdparm 一起使用。

此命令的语法是:

```
hdparm [选项] [设备]
```

经常使用的选项是:

版权所有 2013-2017 Snow BV

Machine Translated by Google

-a 获取或设置文件系统预读的扇区计数。

-d [0|1] 获取或设置 using_dma 标志。

-g 显示驱动器几何。

-i 显示标识信息。

-r [0|1] 获取 (1) 或设置 (0) 只读标志。

-t 执行设备读取以进行基准测试。

-T 执行设备缓存读取以进行基准测试。

-v 显示除-i 之外的所有设置。

有关详细信息,另请参阅 man hdparm。

使用 sdparm

sdparm 命令用于访问 SCSI 模式页面、读取 VPD（重要产品数据)页面和发送简单的 SCSI 命令。

该实用程序获取并可以更改 SCSI 设备模式页面。还可以显示包括重要产品数据页面在内的查询数据。与启动和停止媒体相关的命令;装载和卸载介质;和其他管家功能也可以由该实用程序发布。

配置内核选项

在1.1节及以后的章节中,描述了配置和调试内核的过程。可以通过修补内核源代码来配置其他内核选项。通常内核的可调参数列在源代码树的各个头文件中。这里没有适用的黄金法则 您需要阅读内核文档,甚至可能需要浏览内核代码以找到您需要的信息。此外,可以使用 /proc 文件系统或使用 sysctl 命令来配置正在运行的内核。

使用/proc文件系统

当前内核还支持动态设置内核参数。设置内核参数最简单的方法是修改 /proc 文件系统。您可以使用 echo 命令设置参数,格式为:

```
# 回显值 > /proc/kernel/parameter
```

例如激活 IP 转发:

```
# echo 1 >/proc/sys/net/ipv4/ip_forward
```

更改会立即生效,但会在重新启动期间丢失。

/proc/ 文件系统也可以查询。要查看系统使用了哪些中断,例如您可以发出

```
# cat /proc/中断
```

它生成的输出可能如下所示:

```
                CPU0
0:16313284                      XT-PIC定时器
1:          334558              XT-PIC键盘
2:                0             XT-PIC级联
7:          26565               XT-PIC 3c589_cs
8:                2             XT-PIC 实时时钟
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
  9:          0   XT-PIC OPL3-SA (MPU401)
 10:          4   XT-PIC MSS 音频编解码器
 11:          0   XT-PIC USB-UHCI
 12:     714357   XT-PIC PS/2 鼠标
 13:          1   XT-PIC 处理器
 14:     123824   XT-PIC ide0
 15:          7   XT-PIC ide1
纳米:          0
```

在多核系统上,您会看到多个 CPU 列,例如四核系统的 CPU0..CPU3。默认情况下,中断会传送到核心 0。这会造成性能瓶颈,尤其是在网络存储系统上。当 CPU0 被 IRQ 淹没时,系统会感觉迟钝,即使其他内核几乎无事可做。较新的内核尝试对中断进行负载平衡。但是您可以覆盖默认值并选择哪个内核将处理哪个中断。该过程在下一段中列出。

/proc/interrupts 文件中列出的每个中断在 /proc/irq/ 树中都有一个子目录。所以,中断12对应目录/proc/irq/12。在该目录中,文件 smp_affinity 包含描述哪个内核处理中断的数据。该文件包含一个位掩码,每个内核一位。通过设置适当的位,APIC 芯片会将中断传送到相应的内核,例如 2 选择 CPU1,4 选择 CPU2、8-3、16-4.. 等。要设置例如 CPU5 接收所有中断 #13,请执行:

```
# echo 32 >/proc/irq/13/smp_affinity
```

通过仔细选择将哪些中断发送给哪个处理器,您可以显着影响繁忙系统的性能。

使用 sysctl

将sysctl命令放入启动脚本（eg /etc/sysctl.d/99-sysctl.conf)即可实现内核调优自动化。sysctl命令用于在运行时修改内核参数。该命令可修改的参数都列在 /proc/sys 目录树中。Linux 下的 sysctl 支持需要 Procfs。sysctl 可用于读取和写入内核参数。

sysctl 命令的常用选项是:

-a, -A显示当前可用的所有值。

-e使用此选项忽略有关未知键的错误。

-n使用此选项可在打印值时禁用键名的打印。

-p从指定文件加载 sysctl 设置,如果没有指定文件,则加载 /etc/sysctl.conf。指定-作为文件名表示读取数据从标准输入。

-w使用此选项更改 sysctl 设置。

逻辑卷管理器 (204.3)

考生应该能够创建和删除逻辑卷、卷组和物理卷。此目标包括快照和调整逻辑卷的大小。

资源: Hubert00;科利根00;各种命令的手册页。

关键知识领域

- LVM 套件中的工具
- 调整大小、重命名、创建和删除逻辑卷、卷组和物理卷
- 创建和维护快照
- 激活卷组

版权所有 2013-2017 Snow BV

Machine Translated by Google

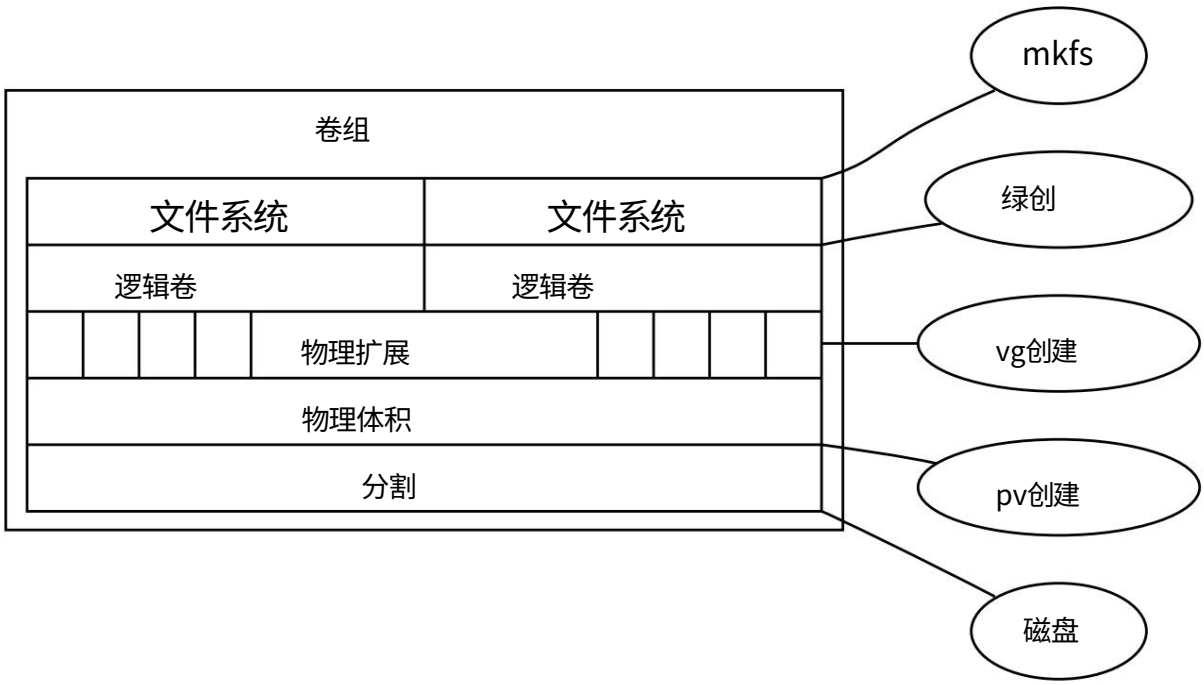
条款和实用程序

- /sbin/pv*
- /sbin/vg*
- /sbin/lv*
- 山
- /开发/映射器
- lvm.conf

配置逻辑卷管理

lvm 是 Linux 的逻辑卷管理器。它使您能够将多个物理卷（硬盘等)连接到所谓的卷组,形成一个存储池,很像一个虚拟磁盘。支持 IDE 和 SCSI 磁盘以及多设备 (MD)。

下图勾勒了lvm使用的概念和术语。右侧显示了可用于创建和/或操作左侧绘制的图层的命令名称。



物理媒体/分区硬盘,或分区,例如/dev/hda、/dev/hda6或/dev/sda。您应该将磁盘或分区的分区类型设置为0x8e,即“Linux LVM”。分区是使用 fdisk 完成的。请注意,您的 fdisk 版本可能还不知道这种类型,因此它将被列为“未知”。您可以将块设备上的任意连续数量的块转换为物理卷：

物理卷 (PV) 一种添加了一些管理数据的物理介质。命令 pvcreate 可用于将管理添加到物理介质上。命令 vgcreate 用于创建一个由一个或多个 PV 组成的卷组。已分组在卷组中的 PV 包含 Physical Extents：

物理范围 (PE) 物理范围是磁盘空间块,通常有几兆字节大小。使用命令 lvcreate
您可以将 PE 分配给逻辑卷：

逻辑卷 (LV) 逻辑卷。在逻辑卷上,我们可以使用命令 mkfs 来获取文件系统：

版权所有 2013-2017 Snow BV

Machine Translated by Google

文件系统 ext2、ReiserFS、NWFS、XFS、JFX、NTFS 等。对于 linux 内核,常规文件系统和普通文件系统没有区别
分区和一个逻辑卷。一个简单的挂载就足以使用您的逻辑卷。

LVM 命令集的一些典型用法示例如下。最初,您需要将用于创建逻辑卷的分区的分区类型设置为 0x8e。假设我们有分区 /dev/hda4 和 /dev/hda5,并且它们被设置为正确的分区类型。要在两个分区上创建物理卷（即设置卷组描述符）,您键入（作为超级用户）：

```
# pvcreate /dev/hda4 /dev/hda5
```

现在我们已经创建了两个物理卷。接下来,我们将创建一个卷组。卷组需要有一个名称（我们选择 volume01）。要使用我们之前定义的物理卷创建我们的卷组,请键入：

```
# vgcreate volume01 /dev/hda5 /dev/hda4
```

前面的命令行是最基本的形式（有关可配置参数列表,请参阅手册页）。这将创建一个物理范围数组,默认情况下它们的大小为 4 Mb。使用这些范围我们可以创建一个或多个逻辑卷,例如：

```
# lvcreate -L 100M volume01
```

·· 这将创建一个逻辑卷,其默认名称由 lvcreate 选择,并以字符串 lvol 后跟一个数字开头 让我们假设 lvol0。将使用卷组 volume01 创建逻辑卷。该卷的设备文件名称将是 /dev/volume01/lvol0。接下来,我们可以像往常一样使用 mkfs 在卷组上创建文件系统,例如 xfs 文件系统：

```
# mkfs -t xfs /dev/volume01/lvol0
```

生成的文件系统可以照常挂载：

```
# 挂载 /dev/volume01/lvol0 /mnt
```

修改逻辑卷、卷组和物理卷

可以修改逻辑卷,以便为位于该逻辑卷之上的文件系统创建更多空间。假设卷组中有足够的空间,可以使用以下命令增加逻辑卷的大小：

```
# lvextend -L +50M /dev/volume01/lvol0
```

增加逻辑卷后,逻辑卷之上的文件系统仍然具有相同的大小。要使用逻辑卷的额外空间,需要调整文件系统的大小。这可以通过以下命令完成：

```
# xfs_growfs /dev/volume01/lvol0
```

对于 ext2/ext3/ext4 文件系统,使用命令 resize2fs。请注意,为了调整文件系统的大小,不得挂载它。

在前面的示例中,假定卷组中有足够的可用空间。如果不是这种情况,可以用类似的方式向卷组添加额外的磁盘空间。为此,请使用以下命令：

```
# vgextend volume01 /dev/hda6
```

第一个设备 hda6 必须使用以下命令转换为物理卷：

```
# pvcreate /dev/hda6
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

LVM 快照

LVM 最好的特性之一是可以拍摄卷快照。快照是卷的虚拟副本,可以轻松备份。 LVM 快照使用一种称为“写时复制”的策略。这意味着快照逻辑卷只保存原始逻辑卷中在原始逻辑卷中发生变化的数据块。为此,逻辑卷管理器首先读取原始数据块上的（未更改的)数据块,然后将数据块写入快照。在具有许多更改的文件系统（例如数据库)上,这可能会导致性能问题。

lvcreate 命令中的-s 选项指定新创建的逻辑卷是一个快照。

```
# lvcreate -L 50M -s -n snapshot0 /dev/volume01/lvol0
```

这将创建一个逻辑卷 /dev/volume01/snapshot0,然后可以将其用于备份目的等。
快照的好处是数据是一致的,即数据在备份过程中不会改变。

备份完成后,必须删除快照,否则前面提到的性能问题将开始发挥作用。要删除快照,请使用：

```
# lvremove /dev/volume01/snapshot0
```

LVM 命令

本节概述了大多数 lvm 相关命令。手册页更详细地描述了这些命令。

pvchange 更改物理卷的属性

pvck 检查物理卷元数据

pvcreate 初始化磁盘或分区以供 LVM 使用

pvdisplay 显示物理卷的属性

pvmove 移动物理范围

pvremove 删除一个物理卷

pvs 报告有关物理卷的信息

pvscan 扫描所有磁盘以查找物理卷

lvchange 更改逻辑卷的属性

lvcreate 在现有卷组中创建逻辑卷

lvdisplay 显示逻辑卷的属性

lvextend 扩展逻辑卷的大小

lvmdiskscan 扫描 LVM2 可见的所有设备

lvreduce 减少逻辑卷的大小

lvremove 删除一个逻辑卷

lvrename 重命名一个逻辑卷

lvresize 调整一个逻辑卷的大小

lvs 报告逻辑卷信息

lvscan 扫描（所有)磁盘以查找逻辑卷

vgcfgbackup 备份卷组描述符区域

版权所有 2013-2017 Snow BV

Machine Translated by Google

vgchange 更改卷组的属性

vgck 检查卷组元数据

vgconvert 转换卷组元数据

vgcreate 创建卷组

vgdisplay 显示卷组的属性

vgextend 将物理卷添加到卷组

vgexport 使卷组对系统未知

vgimport 使导出的卷组为系统所知

vgmerge 合并两个卷组

vgmknodes 重新创建卷组目录和逻辑卷特殊文件

vgreduce 减少卷组

vgremove 删除卷组

vgrename 重命名卷组

vgs 报告有关卷组的信息

vgscan 扫描所有磁盘以查找卷组并重建缓存

vgsplit 将卷组一分为二

设备映射器

设备映射器是一个内核驱动程序,它为卷管理提供了一个框架。它提供了一种创建映射设备的通用方法,这些映射设备可以用作逻辑卷。它并不特别了解卷组或元数据格式。

LVM 逻辑卷是使用设备映射器激活的。每个逻辑卷都被转换为一个映射设备。每个段都转换为描述设备的映射表中的一行。设备映射器支持多种映射目标,包括线性映射、条带映射和错误映射。

配置文件

在系统启动时加载 lvm.conf 配置。默认目录是 /etc/lvm。但是可以用环境变量 LVM_SYSTEM_DIR 来设置。在 lvm.conf 文件中,您可以指定要加载的其他配置。要显示当前设置,您可以执行 lvm dumpconfig 命令。

vgscan 命令扫描带有 lvm 元数据的块设备。这些物理卷存储在 lvm 缓存中。此命令使用 lvm.conf 文件来确定。例如,当您使用多路径设备时,元数据在所有磁盘上的不同路径上都是相同的。当你运行 pvs 时,这会导致很多重复。此外,当您要引导多路径设备时,它可以引导单路径。为此,您可以在 lvm.conf 的设备部分使用过滤器选项,跳过其他设备,只查找多路径设备。过滤器选项使用正则表达式来接受或拒绝块设备,如下例所示。添加所有发现的设备:

```
过滤器 = [ a/.*/ ]
```

过滤以删除 cdrom 设备:

```
过滤器 = [ r|/dev/cdrom| ]
```

仅过滤多路径设备上的设备映射器名称:

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
filter = [ a|/dev/disk/by-id/dm-uuid-.*-mpath-.*| , r|.*| ]
```

此过滤器仅在第一个 IDE 硬盘上添加分区 8,并删除所有其他块设备。

```
过滤器 = [ a|^/dev/hda8$| , r/.*/ ]
```

lvm.conf 的设备部分中其他有用和常用的选项是： preferred_names 最早的模式用于任何输出,如 pvs。所以在我们想要的多路径设备示例中。

```
preferred_names = [ ^/dev/mpath/ , ^/dev/mapper/mpath , ^/dev/[hs]d ]
```

types 如果您从本地设备启动而其余设备是多路径设备,则需要更改 types 选项,在本示例的 filter 选项旁边。我告诉 LVM 可接受的块设备（设备映射器)和允许的分区 (253)

```
类型 = [“设备映射器”,253]
```

对于其他带有选项的部分,请检查:man lvm.conf

问题和解答

高级存储设备管理

1、什么是RAID?

RAID 是一种将信息分布在多个磁盘上的方法,它使用技术来实现冗余、更低的延迟和/或更高的带宽以读取和/或写入磁盘,并最大限度地提高硬盘崩溃的可恢复性。
什么是RAID? [102]

2. 使用线性 RAID 是否真的增加了冗余级别?

不,事实上它降低了可靠性 如果任何一个成员驱动器发生故障,则整个阵列都无法使用。线性RAID

3. 如果 mdadm.conf 存在,您将如何在重启后自动激活 RAID?

在其中一个启动文件中运行 mdadm --assemble -s。重启后自动激活 RAID

4. 使用逻辑卷管理 (LVM) 的目的是什么?

它使您能够将多个物理卷（硬盘等)连接到一个所谓的卷组,形成一个存储池。
配置逻辑卷管理[116]

5. 什么是逻辑卷 (LV)?

逻辑卷是基于从卷组 (VG) 中获取的多个物理范围 (PE) 创建的,在卷组之上可以构建文件系统。逻辑卷 (LV) [116]

版权所有 2013-2017 Snow BV

Machine Translated by Google

第 5 章

网络配置 (205)

本主题的权重为 11 分,包含以下目标:

目标 205.1;基本网络配置（3 分） 考生应该能够配置网络设备以连接到本地、有线或无线以及广域网。此目标包括能够在单个网络内的各个子网之间进行通信。

目标 205.2;高级网络配置和故障排除（4 分） 考生应该能够配置网络设备以实施各种网络身份验证方案。此目标包括配置多宿主网络设备、配置虚拟专用网络以及解决网络和通信问题。

目标 205.3;排除网络问题（4 分） 考生应该能够识别和纠正常见的网络设置问题,包括基本配置文件和命令的位置知识。

基本网络配置 (205.1)

考生应该能够配置网络设备,使其能够连接到本地、有线或无线以及广域网。此目标包括能够在单个网络内的各个子网之间进行通信。

资源： Dawson00、 Drake00。

关键知识领域

- 配置和操作以太网网络接口的实用程序
- 配置无线网络

条款和实用程序

- /sbin/路线
- /sbin/ifconfig
- /sbin/ip
- /usr/sbin/arp
- /sbin/iw
- /sbin/iwconfig
- /sbin/iwlist

版权所有 2013-2017 Snow BV

Machine Translated by Google

配置网络接口

现代内核支持大多数网络设备。但是您需要配置您的设备以适应您的网络。
他们将需要一个 IP 地址（IPv4、IPv6 或两者），可能必须知道一些网关/路由器以允许访问其他网络，并且需要设置默认路由。这些任务通常在您每次引导系统时从网络初始化脚本中执行。此过程的基本工具是 ifconfig（其中“if”代表接口)和路由。

ifconfig 命令仍然被广泛使用。它配置一个接口并使其可供内核网络层访问。可以设置 IP 地址、子掩码、广播地址和各种其他参数。该工具还可用于激活和取消激活界面，也称为“启动”和“关闭”界面。活动接口将通过该接口发送和接收 IP 数据报。调用它的最简单方法是：

```
# ifconfig 接口 ip 地址
```

此命令将ip-address分配给接口并激活它。所有其他参数都设置为默认值。例如，默认网络掩码源自 IP 地址的网络类别，例如 B 类地址的 255.255.0.0。

route 允许您从内核路由表中添加或删除路由。它可以被调用为：

```
# 路由 {add|del} [-net|-host] 目标 [if]
```

add 和 del 参数确定是添加还是删除到目标的路由。 -net 和 -host 参数告诉路由命令目标是网络还是主机（如果不指定则假定为主机）。 if 参数指定了接口并且是可选的，它允许您指定路由应该指向哪个网络接口 如果您不提供此信息，Linux 内核会做出合理的猜测。

环回接口

TCP/IP 实现包括一个虚拟网络接口，可用于模拟同一主机上两个进程之间的网络流量。环回接口不连接到任何真实网络，它完全在操作系统的网络软件中实现。发送到环回 IP 地址（通常使用地址 127.0.0.1)的流量被简单地传递回网络软件堆栈，就好像它是从另一台设备接收到的一样。使用的 IPv6 地址是 ::1，通常使用名称“localhost”作为主机名。它是引导期间要激活的第一个界面：

```
#ifconfig lo 127.0.0.1
```

有时，您会看到使用虚拟主机名 localhost 而不是 IP 地址。这需要正确配置 /etc/hosts 文件：

```
# localhost 127.0.0.1 localhost 的示例 /etc/hosts 条目
```

要查看接口的配置，只需调用 ifconfig 并将接口名称作为唯一参数：

```
$ ifconfig lo Link
罗                encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 UP LOOPBACK RUNNING
MTU:3924 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 错误： 0 掉落:0 超限:0 承运人:0 碰撞：
0
```

此示例显示默认情况下已为环回接口分配网络掩码 255.0.0.0。 127.0.0.1 是 A 类地址。

这些步骤足以在独立主机上使用网络应用程序。将这些行添加到您的网络初始化脚本并通过重新启动您的机器确保其在启动时执行后，您可以测试环回接口。例如，telnet localhost 应该与您的主机建立一个 telnet 连接，给您一个login:提示符。

环回接口在开发过程中经常被用作测试平台，但还有其他应用。例如，所有基于 RPC 的应用程序都使用环回接口在启动时向端口映射程序守护程序注册自己。这些应用程序包括 NIS 和 NFS。因此，无论您的机器是否连接到网络，都应始终配置环回接口。

---

[1] 这通常包含在您的发行版中。

版权所有 2013-2017 Snow BV

Machine Translated by Google

以太网接口

配置以太网接口与环回接口几乎相同 在使用子网划分时只需要多几个参数。

假设我们将 IP 网络（最初是 B 类网络)划分为 C 类子网。为了让界面识别这一点,ifconfig 调用将如下所示：

```
# ifconfig eth0 172.16.1.2 网络掩码 255.255.255.0
```

此命令为 eth0 接口分配 IP 地址 172.16.1.2。如果我们省略了网络掩码,ifconfig 将从 IP 网络类别中推断出网络掩码,这将导致不正确的网络掩码 255.255.0.0。现在快速检查显示：

```
#ifconfig eth0
eth0        链路封装 10Mps 以太网 HWaddr
            00:00:C0:90:B3:42 inet 地址 172.16.1.2 Bcast 172.16.1.255 掩码
            255.255.255.0 UP BROADCAST RUNNING MTU 1500 Metric 1 RX packets 0 errors 0 dropped 0 overrun 0 TX packets 0 errors
            0 dropped 0 overrun 0
```

您可以看到 ifconfig 自动将广播地址（Bcast 字段)设置为常用值,即设置了所有主机位的主机网络号。此外,最大传输单元（内核将为该接口生成的 IP 数据报的最大大小)已设置为以太网数据包的最大大小:1,500 字节。默认值通常是您将使用的值,但如果需要,可以覆盖所有这些值。

通过网关路由

如果您的主机位于单个以太网上,则不需要路由。然而,网络经常通过网关相互连接。这些网关可以简单地连接两个或多个以太网,但也可以提供到外部世界的链接,例如 Internet。为了使用网关,您必须向网络层提供额外的路由信息。

想象一下通过这样一个网关连接的两个以太网,即主机 romeo。假设已经配置了 romeo,我们只需要在路由表中添加一个条目,告诉内核可以通过 romeo 访问其他网络上的所有主机。路由的适当调用如下所示； gw 关键字告诉它下一个参数表示网关：

```
# route add -net 172.16.0.0 网络掩码 255.255.255.0 gw romeo
```

当然,您希望与之通信的其他网络上的任何主机都必须具有我们网络的路由条目。否则只能向对方网络发送数据,而对方网络的主机无法回复。

此示例仅描述了在两个隔离的以太网之间交换数据包的网关。现在假设 romeo 也有一个到 Internet 的连接（比如,通过一个额外的 PPP 链接）。在这种情况下,我们希望将发往任何目标网络的数据报交给 romeo。这可以通过将其设为默认网关来实现：

```
# 路由添加默认 gw romeo
```

经常被误解的是只能配置一个默认网关。您可能有多个网关,但只有一个可以是默认的。

网络名default是0.0.0.0的简写,表示默认路由。默认路线匹配每个目的地,如果更具体的路线不可用,将使用默认路线。

ip 命令

近年来,许多人提倡使用较新的命令/sbin/ip。它也可以用来显示或操作路由和网络设备,也可以用来配置或显示策略路由和隧道。但是,如果更方便的话,也可以使用旧工具 ifconfig 和 route。 ip 的一个用例是以比 ifconfig 更简洁的方式显示网络接口上使用的 IP 地址：

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
# ip addr show 1: lo:
<LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0. 0.1/8 范围主机 lo inet6 ::1/128 范围主机 valid_lft forever preferred_lft forever



2: p8p1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
        链接/以太 00:90:f5:b6:91:d1 brd ff:ff:ff:ff:ff:ff inet 192.168.123.181/24 brd 192.168.123.255 范围全
        局 p8p1 inet6 fe80::290:f5ff:feb6:91d1 /64 范围链接 valid_lft forever preferred_lft forever


3: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc mq state DOWN qlen 1000
        链接/以太 88:53:2e:02:df:14 brd ff:ff:ff:ff:ff:ff
4: vboxnet0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop 状态 DOWN qlen 1000
        链接/以太 0a:00:27:00:00:00 brd ff:ff:ff:ff:ff:ff
```

与 ifconfig 相比

```
# ifconfig -a lo Link
encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0

                     inet6 地址：::1/128 范围:主机
                     向上环回运行 MTU:16436 指标:1
                     RX 数据包:48 错误:0 丢弃:0 溢出:0 帧:0 TX 数据包:48 错误:0 丢弃:0 溢出:0 载体:0 冲突:0 txqueuelen:0
                     RX 字节:4304 (4.2 KiB) TX 字节:4304 (4.2 KB)



p8p1                 链路封装:以太网 HWaddr 00:90:F5:B6:91:D1
                     inet地址:192.168.123.181广播:192.168.123.255掩码:255.255.255.0
                     inet6 地址:fe80::290:f5ff:feb6:91d1/64 范围:链接
                     向上广播运行多播 MTU:1500 指标:1
                     RX 数据包:6653 错误:0 丢弃:0 溢出:0 帧:0 TX 数据包:7609 错误:0 丢弃:0 溢出:0 载体:0 冲突:0 txqueuelen：
                     1000 RX 字节:3843849 (3.6 MiB) TX 字节:938833 (916.8 KiB)


                     中断:53

vboxnet0 链路封装:以太网 HWaddr 0A:00:27:00:00:00
                     广播多播 MTU:1500 公制:1
                     RX 数据包:0 错误:0 丢弃:0 溢出:0 帧:0 TX 数据包:0 错误:0 丢弃:0 溢出:0 载体:0 冲突:0 txqueuelen：
                     1000 RX 字节:0 (0.0 b) TX 字节:0 (0.0 磅)


无线局域网0          链路封装:以太网 HWaddr 88:53:2E:02:DF:14
                     广播多播 MTU:1500 公制:1
                     RX 数据包:2 错误:0 丢弃:0 溢出:0 帧:0 TX 数据包:11 错误:0 丢弃:0 溢出:0 载体:0 冲突:0 txqueuelen：
                     1000 RX 字节:330 (330.0 b) TX 字节:1962 (1.9 KB)
```

在配置网络接口 (p8p1) 时使用 ip 替代 ifconfig 的示例：

```
# ifconfig p8p1 192.168.123.15 网络掩码 255.255.255.0 广播 192.168.123.255
```

可以替换为：

```
        ip 地址添加 192.168.123.15/24 广播 192.168.123.255 dev p8p1
```

ip 也可以用作 route 命令的替代方法：

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
# ip 路由添加 192.168.123.254/24 dev p8p1
```

```
# ip route show
192.168.123.0/24 dev p8p1 proto kernel scope link src 192.168.123.181 metric 1 default via 192.168.123.254 dev p8p1 proto static
```

在这种情况下,route 命令的输出将是:

```
# 路线
目的地          网关                基因掩码          标志公制参考                    使用界面
192.168.123.0 *                     255.255.255.0 你                 1个    0        0 p8p1 0
默认            192.168.123.254 0.0.0.0             优格    0         0              p8p1
```

作为另一个示例,要通过 eth0 添加到网络 192.168.1.0 的静态路由,请使用:

```
# ip 路由添加 192.168.1.0/24 dev eth0
```

有关更多信息,请阅读 ip 的手册页。

ARP,地址解析协议

在 ISO 网络模型中有七层。 Internet 协议是第 3 层协议,NIC 是第 2 层设备。
在本地网络（第 2 层)中,设备通过 MAC（媒体访问控制)地址相互了解。在 IP 网络（第 3 层)中,设备通过其 IP 地址相互了解。

为了允许在第 3 层和从第 3 层之间传输数据,IP 通信需要一个协议来映射第 2 层和第 3 层。该协议称为 ARP 地址解析协议。 ARP 在 IP 地址和配置 IP 地址的 MAC 地址之间创建映射。

当支持 IP 的设备想要通信时,发起设备的内核将 IP 数据包交给网络接口驱动程序软件,并请求将 IP 数据包传递给接收方。在本地以太网上通信的唯一方法是通过 MAC 地址,IP 地址在那里没有用。为了找到具有正确 IP 地址的接收方的 MAC 地址,发起方接口的网络驱动程序将发送 ARP 请求。 ARP 请求是一个广播:它被发送到本地网络上的任何计算机。具有请求的 IP 地址的计算机现在将使用其 MAC 地址进行回复。发送方然后拥有传输数据包所需的所有信息。此外,MAC 和 IP 地址存储在本地缓存中以供将来参考。

命令 arp 可用于显示 ARP 缓存。例子:

```
#arp
地址                HWtype HWaddress ether          标志面具          我面对
10.9.8.126          00:19:bb:2e:df:73 C                               无线局域网0
```

缓存可以手动操作,例如,如果主机关闭,您可能希望删除它的 arp 条目。通常您不需要费心,因为缓存不会过于持久。有关 arp 命令的更多信息,请参见手册页。

注意此

外,还存在反向 ARP 协议 (RARP)。该协议用于允许以太网（第 2 层)设备配置其 IP 地址。 ARP:广播IP,接收MAC。 RARP:广播MAC,接收IP。

无线网络

我

iw 用于配置无线设备。它仅支持 nl80211 (netlink) 标准。因此,如果我没有看到您的设备,这可能就是原因。您应该使用 iwconfig（来自 wireless_tools 包)和 iwlist 来配置无线设备。
这些都使用 WEXT 标准。 wireless_tools 已弃用,但仍得到广泛支持。

句法:

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
# iw 命令 # iw [选项]
[对象] 命令
```

一些常见的选项：

dev 这是一个对象，无线设备的名称 op 应该跟在这个选项之后。使用命令 iw dev 你可以看到你的设备的名称。

link 这是一个命令，用于获取无线设备的链接状态。

scan 这是一个命令，用于扫描网络中可用的接入点。

connect 这是一个让你连接到接入点（essid)的命令，你可以在它后面指定一个通道和/或你的密码。

set 这是一个让您设置不同接口/模式的命令。例如 ibss，如果您想将操作模式设置为 Ad-Hoc。或者设置接口的省电状态。

例子：

```
# iw dev wlan0 link # iw dev
wlan0 scan # iw dev wlan0
connect  Access Point  # iw dev wlan0 connect  Access
Point  2432 # iw dev wlan0 connect  Access Point  0:  Your Key  #
iw dev wlan0 set type ibss # iw dev wlan0 ibss 加入“接入点” 2432 # iw dev wlan0 ibss
离开


# iw dev wlan0 设置 power_save on
```

iwconfig

iwconfig 类似于 ifconfig，但专用于无线接口。它用于设置特定于无线操作的网络接口参数（例如:频率）。 iwconfig 也可用于显示这些参数和无线统计信息。

所有参数和统计数据都取决于设备。每个驱动程序将仅提供其中的一部分，具体取决于硬件支持，并且值的范围可能会发生变化。有关详细信息，请参阅每个设备的手册页。

句法：

```
# iwconfig [接口] # iwconfig 接口 [选
项]
```

一些常见的选项：

essid 设置 ESSID（或网络名称 - 在某些产品中也可能称为域 ID）。对于某些卡，您可以使用 off 或 any（和 on 重新启用它)来禁用 ESSID 检查（ESSID 混杂）。如果您网络的 ESSID 是特殊关键字之一（off、on 或任何），您应该使用 -- 来转义它。

mode 设置设备的工作模式，这取决于网络拓扑。模式可以是Ad-Hoc（网络只由一个cell组成，没有Access Point），Managed（节点连接到一个由多个Access Point组成的网络，具有漫游功能），Master（节点是同步master)或充当接入点）、中继器（该节点在其他无线节点之间转发数据包）、次要（该节点充当备份主设备/中继器）、监控器（该节点不与任何小区相关联并被动地监视无线网络上的所有数据包频率)或自动。

例子：

```
# iwconfig wlan0 essid any # iwconfig wlan0
essid  Access Point  # iwconfig wlan0 essid --  any  #
iwconfig wlan0 mode Ad-Hoc
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

iwlist

iwlist 用于扫描可用的无线网络并显示有关它们的附加信息。语法如下：

```
# iwlist [接口] 命令
```

iwlist 可以显示范围内其他无线节点的 ESSID、频率/信道信息、比特率、加密类型、电源管理信息。显示哪些信息取决于硬件。

一些有用的选项是：

scan[ning] 返回 ad-hoc 网络和接入点的列表。根据卡的类型，显示更多信息，即 ESSID，信号强度，频率。扫描只能由 root 完成。当非 root 用户发出扫描命令时，将返回上次扫描的结果（如果可用）。这也可以通过最后添加选项来实现。此外，选项 essid 可用于扫描特定的 ESSID。根据驱动程序的不同，可能会有更多选项可用。

keys/enc[ryption] 列出支持的加密密钥大小并列出设备中设置的所有加密密钥。

高级网络配置和故障排除 (205.2)

考生应该能够配置网络设备以实施各种网络身份验证方案。该目标包括配置多宿主网络设备、配置虚拟专用网络以及解决网络和通信问题。

资源： Bird01、 Drake00、

关键知识领域

- 操作路由表的实用程序
- 配置和操作以太网网络接口的实用程序
- 用于分析网络设备状态的实用程序
- 用于监控和分析 TCP/IP 流量的实用程序

条款和实用程序

- /sbin/路线
- /sbin/ifconfig
- /bin/netstat
- /斌/平
- /bin/ping6
- /usr/sbin/traceroute
- /usr/sbin/traceroute6
- /usr/sbin/arp
- /usr/sbin/tcpdump
- /usr/sbin/lsof
- /usr/sbin/ss

版权所有 2013-2017 Snow BV

Machine Translated by Google

- /usr/bin/nc
- /usr/bin/mtr
- /sbin/ip
- nmap
- wireshark

虚拟专用网络

什么是VPN

VPN（虚拟专用网络)允许您通过不安全的连接（例如通过公共互联网)安全地连接两个或多个远程网络。为此,创建了一个加密的安全隧道:所有数据在通过不安全的网络发送之前都将被加密。由此产生的网络连接在行为上和感觉上都像物理连接,但实际上可能会穿越许多物理网络和系统。因此它的名字:“虚拟”。

经常使用 VPN 来节省成本。在过去,物理连接必须从电信提供商那里租用,或者您必须使用 POTS 或 ISDN 线路。这是一项成本高昂的业务。如今互联网无处不在,而且几乎总是以低廉的每月固定价格提供。但是,互联网可能会被嗅探,入侵者可能会检查和/或拦截您的流量。 VPN 可以保护您免受大多数您可能遇到的问题。

一个用例可能是将 LAN 集成到多个办公室或分支机构中。因此,在洛杉矶办公室工作的用户可以访问纽约部门的网络服务,反之亦然。在大多数情况下,办公室已经有 Internet 连接,因此无需进行额外投资。

VPN 类型

实现 VPN 的方法有很多种,尽管大多数解决方案使用 IPSEC 或 SSL/TLS 作为加密的基础。
一些公司使用专有软件实现。许多路由器都内置了基于 IPSEC 的 VPN 支持。可用的 VPN 可以使用简单的 SSH 隧道或使用更复杂的专用解决方案来构建。一些 VPN 实施包括:

- IPSEC
- 虚拟专用网
- SSH
- 许多 Cisco 路由器（或其他专有实施）
- SSL/TLS

本书将概述 OpenVPN 和 IPSEC 的实现。 OpenVPN在系统安全章节中单独介绍。

安全协议

IPSEC 在网络协议栈的 IP（Internet 协议)层提供加密和认证服务。它取代/增强了常规的 3 级 IP 层,因此所有数据包都被加密,包括例如 UDP 数据包。 IPSEC 层已由 IETF 在 RFC 2401–2412 中标准化。实施 IPSEC 是 IPv4 的一个选项,但在 IPv6 堆栈中是强制性的。

在常规 IPv4 网络中,您可以设置专用的 IPSEC 网关机器以在需要时提供加密的 IP 网络连接。 IPSEC 可以在路由器、防火墙机器、各种应用程序服务器和最终用户台式机或笔记本电脑上运行　任何具有 IP 堆栈的系统。

使用 IPSEC 很简单,因为该协议内置于 IP 堆栈中。但与常规 IPv4 连接相比,还有其他任务,例如,在加密隧道可以连接之前,需要在端点之间交换加密密钥

版权所有 2013-2017 Snow BV

Machine Translated by Google

设置。决定通过更高级别的协议来处理此问题,即 Internet 密钥交换协议 (IKE)。 IKE 完成其工作后,IP 级服务 ESP 和 AH 知道使用哪些密钥来完成其工作。

IPSEC 实现中使用的三个协议的全名是:

ESP,封装安全有效负载加密和/或验证数据;

AH,Authentication Header 提供数据包认证服务;

IKE,Internet Key Exchange 为上述协议协商连接参数,包括密钥。 IKE 协议确保对等方的身份验证和它们的对称密钥的交换。 IKE 协议通常由使用端口 500/udp 的用户空间守护进程实现。

笔记

IPSEC 标准定义了所有三种协议,但在某些情况下,人们使用术语 IPSEC 来指代 AH 和 ESP。

OpenSwan,以前称为 FreeS/WAN,是 Linux 2.0 - 2.6 内核的完整 IPSEC 实现。 StrongSwan（也源自 FreeS/WAN)是另一种同样支持 3.x 内核的实现。 OpenSwan 和 FreeSwan 都实现了前面提到的所有三个协议。 Openswan 实现有几个主要部分:

- KLIPS (KerneL IPSec),它在内核级别为所有内核实现通用 IPSEC 数据包处理、AH 和 ESP 版本 2.5.47 之前。 KLIPS 已被本机 IPSEC 内核支持 (NETKEY) 取代。
- NETKEY 是包含在 2.6 内核中的内核 IPSec 实现。
- Pluto（IKE 守护程序)实现 IKE,协商与其他系统的连接。
- 各种脚本为机器提供管理员界面。

配置文件包含三个部分:

一个或多个连接规范 每个连接部分以conn ident 开头,其中 ident 是任意名称用于识别连接。

连接默认值 本节以conn %default 开头。对于其中的每个参数,任何没有同名参数的部分都会从%default部分获取该参数的副本。可能有多个 %default 部分,但只能为任何特定参数名称提供一个默认值,并且所有%default部分必须位于该类型的所有非 %default部分之前。

配置部分 配置部分以配置设置开始,包含启动软件时使用的信息。

示例配置文件如下所示:

```
# 基本配置配置设置

        interfaces=   ipsec0=eth0
        klipsdebug=none plutodebug=none
        plutoload=%search plutostart=%search
        # 当使用相同 ID 的新连接出现时关闭旧连
        接。 uniqueids=是



# 适用于所有连接描述的默认值
conn %默认值
        # 在（重新)键控谈判中坚持的程度（0 表示非常）。 keyingtries=0 # 如何验证网关 authby=rsasig
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
# 总部和分公司的VPN连接
康恩总支
        # 我们在身份验证交换中使用的身份 leftid=@head.example.com
        leftrsasigkey=0x175cffc641f... left=45.66.9.170


        leftnexthop=45.66.11.254
        leftsubnet=192.168.11.0/24 # right sg,它后
        面的子网,以及到达左侧的下一跳 rightid=@branch.example.com rightrsasigkey=0xfc641fd6d9a24...
        right=17.120.138.134 rightnexthop=17.120.138.1 rightsubnet=192.168.0.0/24 # 加载 ipsec 时自动启
        动




        自动=开始
```

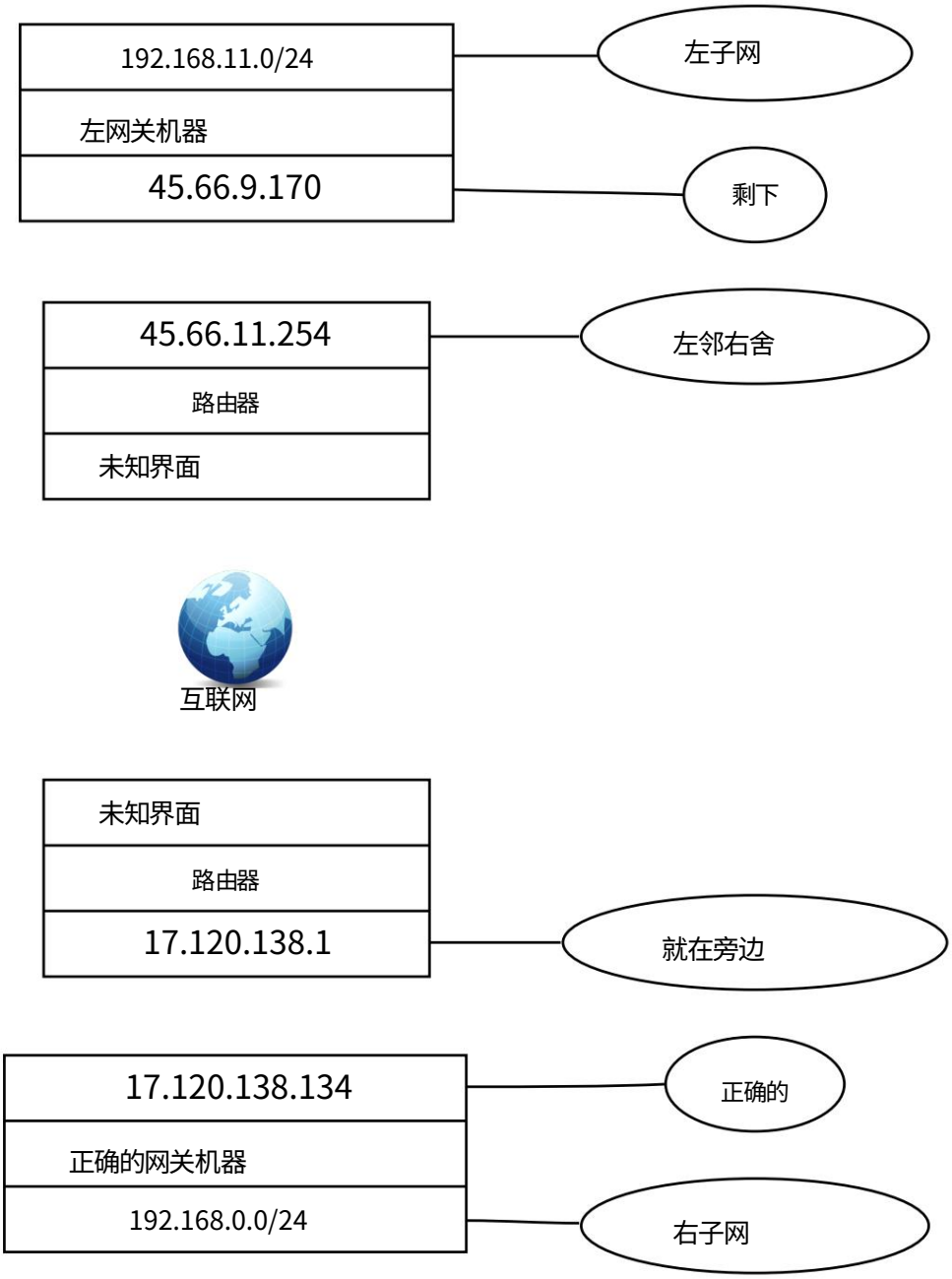
在典型的设置中,您有两个互连的网关,它们都运行 IPSEC 并路由数据包。这些网关之一可以看作是“左边的”,另一个可以看作是“右边的”。因此,规范是根据左右参与者编写的。这两个名称都没有特殊含义,它们只是标签 - 您可能已将“左”主机定义为“右”主机,反之亦然。

通常,您会在两侧使用完全相同的配置文件。该文件的解释是通过检查本地配置来完成的。例如,如果在配置文件中声明“left”的 IP 地址是 1.2.3.4,如果软件发现在其网络设备之一上配置了 IP 地址,则假定它在左节点上运行。同一个文件在另一个节点上的解释不同,因为主机配置不同。

left、 leftsubnet和leftnexthop （以及右...对应项)确定连接的布局：

版权所有 2013-2017 Snow BV

Machine Translated by Google



leftid和leftrsasigkey用于验证左参与者。 leftrsasigkey是左侧参与者的公钥（在上面的示例中,RSA 密钥被缩短以便于显示）。私钥存储在 /etc/ipsec.secrets 文件中,应妥善保管。

可以使用以下命令在客户端和服务器上生成密钥：

```
# ipsec rsasigkey --verbose 2048 > rsa.key
```

IPSEC 的 IKE 守护进程称为 pluto。它将验证和协商安全隧道。建立连接后,如果需要,IPSEC 的内核实现会通过隧道路由流量。

plutoload=%search 和 plutostart=%search 告诉 pluto 守护进程在配置文件中搜索auto=状态。所有带有auto=add的连接都将加载到 pluto 数据库中。使用auto=start的连接也将启动。

版权所有 2013-2017 Snow BV

Machine Translated by Google

故障排除

网络故障排除是一个非常广泛的主题,有数以千计的可用工具。有一些关于此主题的非常好的书籍,因此我们将讨论限制在对一些可用于解决或检测网络问题的工具的概述上。

配置文件

键入不带附加参数的 ifconfig 将显示系统上所有网络接口的配置。如果用户遇到连接问题,特别是当他们的系统刚刚（重新)配置时,您可以使用此命令来验证接口的配置。

当输入带有接口名称且没有其他参数的 ifconfig 时,它会显示分配给该特定接口的当前值。例如,检查接口eth0系统会给出以下报告:

```
$ ifconfig eth0 eth0
          Link encap:Ethernet HWaddr 00:10:60:58:05:36 inet addr:192.168.2.3 Bcast:192.168.2.255
          Mask:255.255.255.0
          向上广播运行多播 MTU:1500 指标:1
          RX 数据包:1398185 错误:0 丢弃:0 溢出:0 帧:0 TX 数据包:1411351 错误:0 丢弃:0 溢出:0 载波:0 冲突:829
          txqueuelen:100 RX 字节:903174205 (861.3 Mb) TX 字节:201042106 (191.7 兆字节)

          中断:11 基地址:0xa000
```

ifconfig 命令显示几行输出。显示的第三行显示了接口的特性。
检查这些特征:

UP 接口已启用。如果接口处于“关闭”状态,则使用 ifconfig 命令（例如 ifconfig eth0 起来）。

RUNNING 此接口可操作。如果接口未“运行”,则该接口的驱动程序可能未正确安装安装。

ifconfig 输出的第二行显示了 IP 地址、子网掩码和广播地址。检查这三个字段以确保网络接口配置正确。

两个常见的接口配置问题是配置错误的子网掩码和不正确的 IP 地址。当主机可以访问其本地子网上的某些主机但无法访问其他主机时,可能会出现错误的子网掩码,即使它们位于同一子网上。 ifconfig 快速显示是否设置了错误的子网掩码。

错误设置的 IP 地址可能是一个微妙的问题。如果地址的网络部分不正确,则每次 ping 都会失败,并出现“无应答”错误;因为网络上的其他主机不熟悉 IP 地址,返回的数据包将被定向到它们的默认网关（通常通向互联网）,甚至被丢弃。在这种情况下,使用 ifconfig 可能会显示不正确的地址。但是,如果地址的主机部分有误,问题就更难检测了。一个小型系统,例如只连接到其他系统并且从不接受传入连接的 PC,可以在错误的地址下运行很长时间而用户却没有注意到这个问题。此外,遭受不良影响的系统可能不是配置错误的系统。有人可能会不小心在他自己的系统上使用您的 IP 地址,并且他的错误可能会导致您的系统出现间歇性通信问题。 ifconfig 无法发现此类配置错误,因为错误发生在远程主机上。可以使用 arp 命令发现此类 IP 冲突,该命令将显示同一 IP 地址的两个交替 MAC 地址。

ifconfig 命令可用于设置多宿主网络设备。主机可以通过两种方式实现多宿主。

同一网络的两个或更多接口:出于性能和/或可靠性原因,服务器或高性能工作站等设备可能配备两个物理接口以连接同一网络。他们将在同一网络上拥有两个具有相同网络 ID 的 IP 地址。

两个或多个不同网络的接口:设备可能有多个不同网络的接口。 IP 地址通常包含不同的网络 ID。

版权所有 2013-2017 Snow BV

Machine Translated by Google

ping 和 ping6

Linux 系统上 ping 或 ping6 命令的基本格式是：

```
$ ping [-c 计数] 主机 $ ping6 [-c 计数] 主机
```

host 正在测试的远程主机的主机名或 IP 地址。请注意，您无法从 IPv4 主机 ping 到 IPv6 主机，反之亦然。两端需要使用相同的 IP 版本。

count 测试中要发送的数据包数。使用计数字段并将值设置得较低。否则，ping 命令将
继续发送测试数据包，直到您中断它，通常是按 CTRL-C (^C)。

要检查您的工作站是否可以访问 www.snow.nl，请使用以下命令发送四个数据包。

```
$ ping -c 4 www.snow.nl PING home.NL.net
(193.67.79.250): 56 数据字节 64 字节来自 193.67.79.250: icmp_seq=0 ttl=245 time=32.1 ms 64
字节来自 193.67.79.250: icmp_seq =1 ttl=245 time=32.1 ms 64 bytes from 193.67.79.250: icmp_seq=2 ttl=245 time=37.6
ms 64 bytes from 193.67.79.250: icmp_seq=3 ttl=245 time=34.1 ms



--- home.NL.net ping 统计数据 --- 发送 4 个数据包，接收 4 个数据包，0%
丢包往返 min/avg/max = 32.1/33.9/37.6 ms
```

此测试显示到 www.snow.nl 的广域网连接良好，无丢包且响应速度快。对于跨广域网建立的连接，小的数据包丢失和高出一个数量级的往返时间并不异常。

ping 命令显示的统计信息可以指示低级网络问题。关键统计数据是：

- 数据包到达的顺序，如每个数据包显示的ICMP 序列号(icmp_seq) 所示；
- 数据包完成往返所需的时间，以毫秒为单位显示在字符串time= 之后；
- 丢失数据包的百分比，显示在ping 输出末尾的摘要行中。

如果数据包丢失率高、响应时间非常长或数据包到达时乱序，则可能是网络硬件或链路问题。如果您在广域网上进行远距离通信时看到这些情况，则无需担心。 TCP/IP 旨在处理不可靠的网络，一些广域网遭受中等程度的数据包丢失。但是，如果在局域网上看到这些问题，则说明有问题。

在本地网络电缆段上，往返时间应接近于零；应该很少或没有数据包丢失，数据包应该按顺序到达。如果不满足这些条件，则网络硬件或连接它们的链路有问题。在以太网上，问题可能是电缆端接不当、电缆段损坏或“活动”硬件损坏，例如集线器、交换机或收发器。

简单 ping 测试的结果，即使 ping 成功，也可以帮助您进一步测试问题的最可能原因。但是需要其他诊断工具来更仔细地检查问题并找到根本原因。

路线

要检查 linux box 的路由，通常输入不带参数的 route 命令或使用 -n 关闭 ip-address 到名称解析。例如 route -n 可能会显示：

```
$路线-n
内核IP路由表
目的地          网关 0.0.0.0        基因掩码          标志公制参考          使用界面
192.168.11.0                        255.255.255.0 你      0      0      0 eth0
145.66.8.0      0.0.0.0             255.255.252.0 你      0      0      0 以太坊
0.0.0.0         145.66.11.254 0.0.0.0          优格      0      0      0 以太坊
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

该主机有两个接口,一个在子网 192.168.11.0/24 上,另一个在子网 145.66.8.0/22 上。在eth1上还有一个默认路由到 145.66.11.254（由“Flags”下的G和“Destination”和“Genmask”0.0.0.0 表示）。

为了能够排除此信息的故障,您需要知道路由应该是什么,也许可以通过在已知系统工作时保存路由信息来解决。

最常见的两个错误是：

- 接口没有网络条目。配置网络接口时,应自动添加路由条目。这个通知内核有关可以通过接口访问的网络。
- 没有默认路由（或两个默认路由）。应该只有一个默认路由。注意两个默认网关可以走很长一段时间未被发现,因为路由可能“意外地”使用了正确的网关。

一般来说,如果存在路由问题,最好首先定位问题起源的网络部分,例如使用 ping 或 traceroute,然后使用路由作为诊断的一部分。

追踪路线

traceroute 和 traceroute6 是用于发现路径沿线网关的工具。路径发现是诊断路由问题的重要步骤。请注意,traceroute6 等同于 traceroute -6 traceroute 命令基于对 IP 数据包标头中的生存时间 (TTL) 字段的巧妙使用。 TTL 字段用于限制数据包的生命周期。当路由器发生故障或配置错误时,可能会导致路由环路或循环路径。如果发生此类路由环路,TTL 字段可防止数据包无限期地保留在网络上。每次数据包在通过网络的途中穿过路由器时,数据包的 TTL 字段都会递减。当它的值达到 0 时,数据包被丢弃而不是转发。丢弃时,ICMP TIME_EXCEEDED 消息将发送回数据包的源,以通知源数据包已被丢弃。通过操纵原始数据包的 TTL 字段,程序 traceroute 使用来自这些 ICMP 消息的信息来发现通过网络的路径。 traceroute 发送一系列 UDP 数据包,其中包含您想要路径的设备的目标地址。默认情况下,traceroute 发送一组三个数据包来发现每一跳。 traceroute 将前三个数据包中的 TTL 字段设置为值 1,以便它们被路径上的第一个路由器丢弃。当该路由器返回 ICMP TIME_EXCEEDED 消息时,traceroute 会记录这些 ICMP 消息的源 IP 地址。这是到达目的地的路由中第一跳的 IP 地址。

接下来,发送三个数据包,其 TTL 字段设置为 2。这些将被路径上的第二个路由器丢弃。此路由器返回的 ICMP 消息显示路径上第二个路由器的 IP 地址。程序以这种方式进行,直到一组数据包最终具有足够大的 TTL 值,以便数据包到达目的地。 traceroute 的大多数实现默认在停止前仅尝试 30 跳。

linux 上的示例 traceroute 如下所示：

```
$ traceroute vhost2.cistron.net 到 vhost2.cistron.net
(195.64.68.36) 的跟踪路由,最多 30 跳,38 字节数据包
  1 gateway.kabel.netvisit.nl (145.66.11.254) 56.013 毫秒 19.120 毫秒 12.642 毫秒 2 145.66.3.45 (145.66.3.45) 138.138 毫秒 28.482 毫秒 28.788 毫秒
  3 asd-dc2.net-ias-19.np.19.19.19 .226.2) 102.338 毫秒 240.596 毫秒 253.462 毫秒 4 asd-dc2-ipc-dr03.nl.kpn.net (195.190.227.242) 95.325 毫秒
  76.738 毫秒 97.651 毫秒 5 asd-dc2-ipc-crpn.net.19.nl (195.190.227.242) .232.206) 61.378 毫秒 60.673 毫秒 75.867 毫秒 6 asd-sara-ipc-dr01.nl.kpn.net
  (195.190.233.74) 111.493 毫秒 96.631 毫秒 77.398 毫秒 7 asd-sara-ias-pr10.1net09 (19.kpn .227.6)78.721 MS 95.029 MS 82.613 MS 8 AMS-
  ICR-02.CARRIER1.NET（212.4.192.60)90.179 MS 80.634 MS 112.13.13.13.130 MS 9 212.4.192.21（212.4.192.21）




10 212.4.194.42 (212.4.194.42) 94.528 毫秒 60.698 毫秒 103.550 毫秒
11 vhost2.cistron.net (195.64.68.36) 102.057 毫秒 62.515 毫秒 66.637 毫秒
```

同样,了解通过网络的路由应该有助于定位问题。请注意,由于某些复杂因素,并非所有网络问题都可以使用跟踪路由检测到。首先,某跳的路由器可能不会返回 ICMP TIME_EXCEEDED 消息。其次,一些较旧的路由器可能会错误地转发数据包,即使 TTL 为 0。第三种可能性是 ICMP 消息可能被赋予低优先级并且可能无法及时返回。最后,在路径的某个点之外,ICMP 数据包可能会被防火墙过滤掉。

traceroute 命令是一个很好的工具,可以缩小网络问题的可能原因。

版权所有 2013-2017 Snow BV

Machine Translated by Google

arp 和 arpwatch

arp 用于操作内核的 ARP 缓存。主要选项是清除地址映射条目并手动设置一个。出于调试目的,arp 程序还允许完全转储 ARP 缓存。

如果您知道特定主机的 MAC 地址应该是什么,转储可用于确定重复的 IP 地址,但在中央系统上运行 arpwatch 可能更有效。

IP 地址冲突通常是配置错误的结果,包括:

- 由网络管理员分配相同的静态 IP 地址
- 在 DHCP 范围（动态范围)内分配静态 IP 地址,导致同一地址被自动分配
  由本地 DHCP 服务器分配
- DHCP 服务器错误
- 系统在长时间处于待机或休眠模式后恢复联机,并使用已重新分配的 IP 地址
  并且正在网络上使用。

即使使用 arpwatch,重复 IP 地址的检测也非常困难。当网络上的两台设备分配了相同的 IP 地址时,就会发生 IP 地址冲突,导致其中一台或两台设备被禁用并失去连接,直到冲突得到解决。

例如,如果流氓主机使用运行 arpwatch 程序的主机的 IP 地址或从不与其通信,则重复的 IP 地址将不会被注意到。尽管如此,arpwatch 仍然是检测网络问题的非常有用的工具。

arpwatch 跟踪以太网/IP 地址配对。通过系统日志和电子邮件报告更改。

当已知 IP 地址出现新的以太网地址时,arpwatch 将报告“更改的以太网地址”。当旧的以太网地址突然再次出现时,会报告一个“触发器”。

转储

tcpdump 是一个程序,它使网络管理员能够实时检查通过网络的每个数据包。此工具通常用于监视活动连接或解决偶尔出现的网络连接问题。然而,为了查看流量,运行 tcpdump 的主机必须位于交换流量的两个（或更多)主机之间的路径上。

这在完全交换的网络中可能被证明是困难的。一个简单的解决方案是在需要发送或接收流量的主机上运行 tcpdump。另一种选择是在其中一个交换机上配置一个端口,发送来自特定源端口和目标端口的流量副本;这称为 SPAN 端口。

tcpdump 可以生成大量输出,因此通过使用 -i 指定要侦听的接口来缩小捕获的数据包范围很有用。此外,如有必要,您可以通过布尔 AND 和 OR 语句指定要查看的流量的来源、目的地、协议类型和/或端口号。示例命令可以是:tcpdump -i eth0 src 10.10.0.1 and dst 10.10.0.254 and tcp port 80)。其他有用的选项是 -n 关闭名称解析和 -w 将捕获的数据包写入文件以供以后检查（例如在 Wireshark 中）。

地图

nmap 是用于网络探索和安全审计的多功能工具。它的主要用途是用作端口扫描器,它还可以识别正在运行的服务、正在运行的服务的版本和操作系统类型。

此命令和输出的示例是:

```
$ nmap -A 本地主机

在 2013-07-04 04:01 启动 Nmap 6.25 ( http://nmap.org ) 本地主机 (127.0.0.1) 的 CDT Nmap 扫描报告

主机启动（0.00092 秒延迟）。
本地主机的其他地址（未扫描）:127.0.0.1 未显示:993 已关闭端口
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
港口                国服版
22/tcp 打开 ssh | ssh 主机密钥：                OpenSSH 6.2p2 Debian 4（协议 2.0）
 1024 f6:bf:2d:57:41:1b:fe:fa:d2:10:e0:2d:c3:89:c1:80 (DSA) | 2048 0d:11:fc:60:83:69:19:76:d1:fd:01:e5:7f:36:19:00 (RSA) |_256
 0a:7b:0c:3a:86:c1:f7 :63:6e:fb:f0:3c:62:42:c1:58 (ECDSA) 25/tcp 打开 smtp | smtp-commands: linux.mailserver Hello localhost
 [127.0.0.1], SIZE 52428800, 8BITMIME, ←
                                                 进出口 smtpd 4.80

        流水线,帮助，
 |_ 支持的命令:AUTH HELO EHLO MAIL RCPT DATA NOOP QUIT RSET HELP 53/tcp open domain | dns-nsid:



 |_ bind.version: 9.8.4-rpz2+rl005.12-P1 80/tcp open http Apache httpd
2.2.22 ((Debian)) |_http-title: 站点没有标题（文本/html）。 111/tcp open rpcbind (R#100000) | rpc信息： |程序版本端
 口/原型服务 | 100000 2,3,4 111/tcp rpcbind | 100000 2,3,4 111/udp rpcbind | 100024 1 40817/udp
status |_ 100024 1 49956/tcp status 389/tcp open ldap OpenLDAP 2.2.X - 2.3.X 3000/tcp open
 ntop-http Ntop web interface 4.99.3 Service Info: Host: linux.mailserver;操作系统:Linux； CPE:
 cpe://o:linux:linux_kernel









执行服务检测。请在 http://nmap.org/ 上报告任何不正确的结果 ←。
        提交/ 。
Nmap 完成:在 12.46 秒内扫描了 1 个 IP 地址（1 个主机启动）
```

笔记

一些nmap命令选项需要 root 权限,请参阅NMAP(1)联机帮助页以获取更多信息

线鲨

wireshark（在 2006 年夏天的商标纠纷之前被称为 Ethereal)是一个开源网络协议分析器。
它允许您检查来自实时网络或磁盘上捕获文件的数据。您可以交互式浏览捕获数据,深入研究您需要的数据包详细信息级别。

Wireshark 有几个强大的功能,包括丰富的显示过滤器语言和查看 TCP 会话的重构流的能力。它还支持数百种协议和媒体类型。还包括一个名为 tethereal 的类似于 tcpdump 的控制台版本。需要注意的是,Ethereal 存在数十个可远程利用的安全漏洞,因此请保持最新状态并警惕在不受信任或敌对网络（例如安全会议)上以 root 权限运行它。

/usr/sbin/lsof

lsof 命令列出系统上所有打开的文件。由于 Linux 将所有内容都视为一个文件,因此它还会显示打开的网络套接字。它可用于查找系统上的开放端口以及确定网络连接的来源。

用于网络故障排除的 lsof 选项。

-i 列出 IP 套接字。 -i 选项还采用参数来限制列出的套接字,如 -i tcp或 -i 192.168.1.2。

-n 不解析主机名;可以使 lsof 运行得更快。

-P 不解析端口名;可以使 lsof 运行得更快。

+P 将端口名称解析为协议;默认行为。

版权所有 2013-2017 Snow BV

Machine Translated by Google

/usr/sbin/ss

ss 命令可用于调查系统上的网络套接字,类似于 netstat。

ss 用于网络故障排除的有用选项包括:

-a 列出所有套接字。这包括处于侦听状态的套接字。

-n 不解析端口名称。

-l 只显示监听套接字。

-p 显示使用套接字的进程

-t 限制输出到 TCP 套接字

-u 限制输出到 UDP 套接字

/bin/网络统计

与 lsof 一样,netstat 是一个列出系统上开放端口的程序。它会查找标准端口,但也会查找应用程序打开的自定义端口,例如 netcat。

经常使用的选项是:

-a 列出所有套接字。

-e 扩展模式。

-inet 仅列出 IP 连接

-l 只显示监听套接字

-n 显示 IP 号码而不是将它们解析为主机名。

-p 显示持有套接字的进程的 PID 号和名称。

-r 显示内核路由表。与路由命令相同。

/usr/bin/nc

nc 用于在任一端的任意端口之间建立 TCP 和 UDP 连接。打开端口后,它可以侦听输入,可以将输入传递给另一个命令以进行进一步处理。请注意,您需要对运行此命令的系统具有管理权限才能打开低于 1024 的侦听端口。该命令允许用户设置、分析和读取系统之间的连接。它是故障排除工具箱中非常有用的工具。 nc 可以用来打开任何端口。

因为 nc 能够连接和侦听任何端口,所以它可用于在缺少 Samba、FTP 或 SSH 等的系统之间传输文件。

如果 nmap 不可用,可以使用 nc 命令检查打开的端口:使用 -z 标志运行 nc 命令。您需要指定主机名/IP 以及端口范围以限制和加速操作。例如。

```
# nc -z 本地主机 1-1023
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

/usr/bin/mtr

mtr 对于解决网络问题非常有帮助,因为它结合了 ping 和 traceroute 的功能。
mtr 不是像 traceroute 那样提供流量在互联网上所采用的路线的简单轮廓,而是收集有关中间主机的状态、连接和响应能力的其他信息。

mtr 可以在没有任何选项的情况下使用。只需键入:mtr> host 即可直观显示您与主机之间的路径以及每跳统计信息。和 ping 一样,mtr 默认会无限期地继续发送 ICMP 数据包。有用的选项是:

-n 不要将 IP 地址反向解析为主机名

-c count 发送count个探测包,然后停止

/sbin/ip

ip 命令已经在前一章中讨论过了。

解决网络问题 (205.3)

考生应该能够识别和纠正常见的网络设置问题,包括了解基本配置文件和命令的位置。

关键知识领域

- 访问限制文件的位置和内容
- 配置和操作以太网网络接口的实用程序
- 管理路由表的工具
- 列出网络状态的实用程序
- 获取有关网络配置信息的实用程序
- 有关已识别和使用的硬件设备的信息方法
- 系统初始化文件及其内容（Systemd 和 SysV 初始化进程）
- 了解 NetworkManager 及其对网络配置的影响

条款和实用程序

- 知识产权
- ifconfig
- 路线
- SS
- 网络统计
- /etc/network/, /etc/sysconfig/network-scripts
- ping, ping6
- traceroute,traceroute6
- 地铁

版权所有 2013-2017 Snow BV

Machine Translated by Google

- 主机名
- 系统日志文件,例如 /var/log/syslog & /var/log/messages
- dmesg
- /etc/resolv.conf
- /etc/主机
- /etc/主机名、/etc/HOSTNAME
- /etc/hosts.allow, /etc/hosts.deny

网络故障排除介绍

如果您能够打开一本书并在其中找到所有可能的网络问题及其解决方案的索引,那就太好了。但实际上这是不可能的。有太多的场景无法描述,新技术不断涌现,带来新的选择和新的问题。然而,几乎所有问题都可以通过应用知识和逻辑来解决。在网络故障排除的情况下,这主要意味着您应该确定流量应该如何从源流向目的地并返回,并逐步检查您是否可以使用前面描述的工具看到流量通过。此外,如果可能,请确保检查两个方向的流量,因为许多网络问题源于这样一个事实,即从源到目的地的流量可能与相反方向的流量不遵循相同的路径。

关键文件、术语和实用程序已在第5章和其他章节中进行了描述。在本章中,我们将重点放在解决问题的过程上,并介绍一些额外的技术、实用程序和关键文件。

示例情况

您正在 PC 上浏览 Internet。它通过局域网和防火墙连接到 Internet。突然之间,您无法再访问自己喜欢的网页。可能是网络、防火墙、ISP,甚至是浏览器。找到问题并解决问题的合理方法是什么?

注意

本节的重点是对网络进行故障排除。因此,我们将专注于网络组件。在实际情况中,涉及的组件更多,例如浏览器、操作系统、本地防火墙设置等,我们将仅简要介绍这些组件。

第一步是汇集所有涉及的网络组件的列表。列表的长度会有所不同,具体取决于配置的复杂性和您的个人知识。一个简单的列表至少包含以下内容:

- PC 本身。它有一个连接到 LAN 的网络接口(大多数情况下是 eth0);
- 防火墙。它有两个接口:连接局域网的eth0接口和连接局域网的eth1接口到路由器,路由器又连接到提供互联网连接的 ISP;
- 您尝试访问的站点,已连接到 Internet。

然后想想一切如何协同工作。您在浏览器中输入 URL。您的机器使用 DNS 来查找您尝试访问的网站的 IP 地址等。

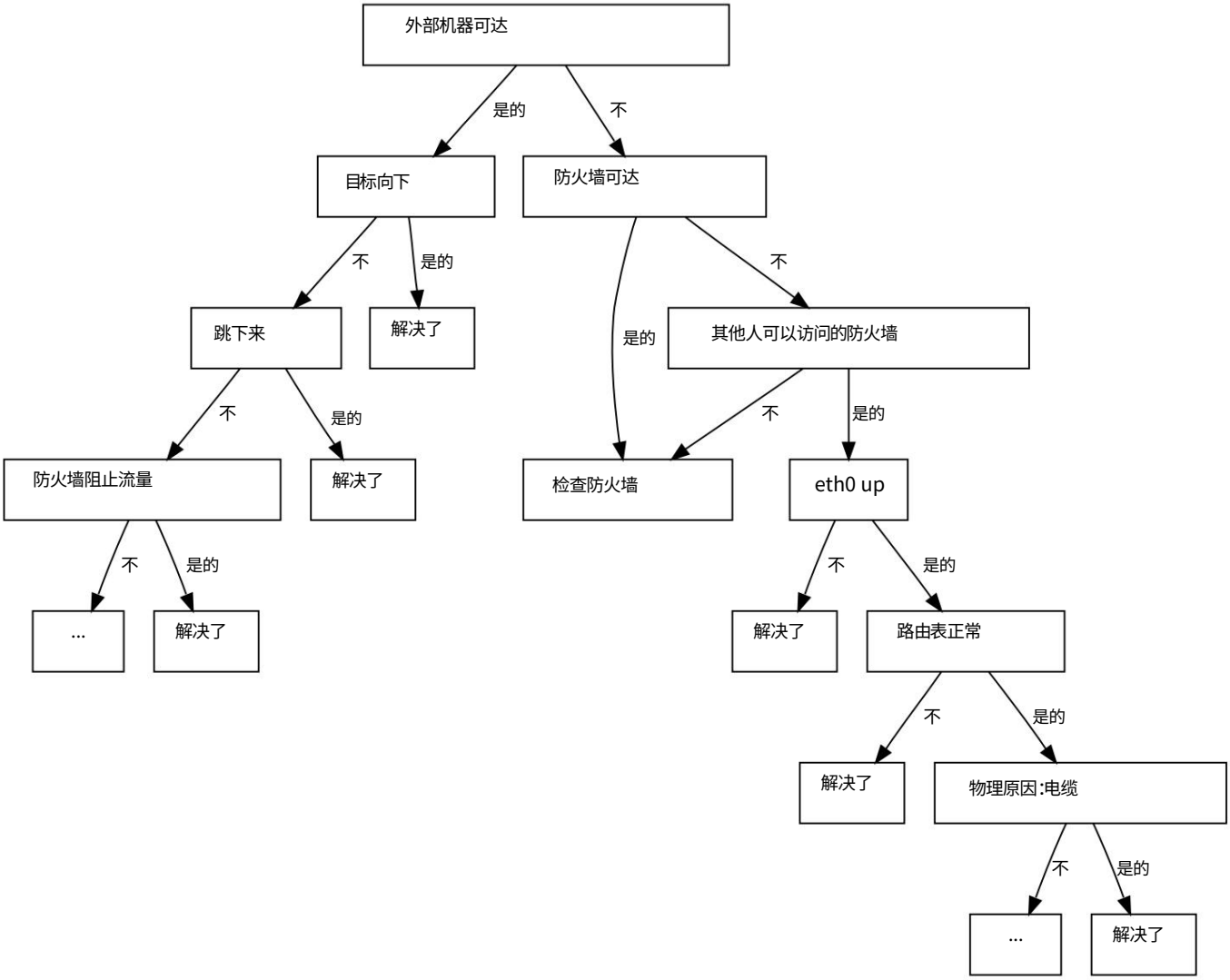
数据包通过 LAN 上的 eth0 接口传输到防火墙的 eth0 接口,然后通过防火墙的 eth1 接口传输到 ISP,并以某种方式从 ISP 传输到 Web 服务器。

现在您知道了不同组件如何相互作用,您可以采取措施来确定故障的根源。

下图给出了逐步故障排除的示例。

版权所有 2013-2017 Snow BV

Machine Translated by Google



S 问题的原因已经确定,可以解决（解决）。

1 我们可以访问互联网上的其他机器吗?尝试另一个 URL 或尝试 ping 互联网上的另一台机器。如果您的 ping 报告没有连接,请谨慎下结论 - 您的防火墙可能会阻止 ICMP 回显请求和回复。

2 我们试图到达的机器（目标)是否出现故障?如果例如可以到达其他站点,则这是一个可行的理论。尝试通过另一个网络连接到机器,联系朋友让他尝试连接到机器,打电话给机器负责人等。

3 我们能到达防火墙吗?尝试 ping 防火墙,登录它等。

4 路由（“跳”)上是否有路由器关闭?使用 traceroute 找出你和目标主机之间的跳数。例如,从一台机器到 LPI 的网络服务器的路由可以通过发出命令 traceroute -I www.lpi.org 来确定：

```
# traceroute -I www.lpi.org traceroute 到
www.lpi.org (209.167.177.93),最大 30 跳,38 字节数据包 1 fertuut.snowgroningen (192.168.2.1) 0.555 ms 0.480 ms 0.387 ms 2
wc-1.r -195-85-156.essentkabel.com (195.85.156.1) 30.910 ms 26.352 ms 19.406 ms 3 HgvL-WebConHgv.castel.nl
(195.85.153.145) 19.296 ms 28.656 ms 29.204 ms-lx.ngHMS-lx.ngHMS-AMS (195.85.155.2) 172.813 毫秒 199.017 毫秒 95.894 毫秒 5 f04-08.ams-
icr-03.carrier1.net (212.4.194.13) 118.879 毫秒 84.262 毫秒 130.855 毫秒 6 g02-00.amd-carrier1.net-01 (212.4.211.197) 30.790 毫秒 45.073 毫秒
28.631 毫秒 7 p08-00.lon-bbr-02.carrier1.net (212.4.193.165) 178.978 毫秒 211.696 毫秒 301.321 毫秒
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
8 p13-02.nyc-bbr-01.carrier1.net (212.4.200.89) 189.606 毫秒 413.708 毫秒 194.794 毫秒 9 g01-00.nyc-pni-02.carrier1.net (212.4.193.198) 134.624 毫秒 182.617.8
毫秒 4 ms6 10 500.POS2-1.GW14.NYC4.ALTER.NET (157.130.94.249) 199.503 毫秒 139.083 毫秒 158.804 ←。

        小姐
11 578.ATM3-0.XR2.NYC4.ALTER.NET (152.63.26.242) 122.309 毫秒 191.783 毫秒 297.066 毫秒
12 188.at-1-0-0.XR2.NYC8.ALTER.NET (152.63.18.90) 212.805 毫秒 193.841 毫秒 94.278 毫秒
13 0.so-2-2-0.XL2.NYC8.ALTER.NET (152.63.19.33) 131.535 毫秒 131.768 毫秒 152.717 毫秒
14 0.so-2-0-0.TL2.NYC8.ALTER.NET (152.63.0.185) 198.645 毫秒 136.199 毫秒 274.059 毫秒
15 0.so-3-0-0.TL2.TOR2.ALTER.NET (152.63.2.86) 232.886 毫秒 188.511 毫秒 166.256 毫秒
16 POS1-0.XR2.TOR2.ALTER.NET (152.63.2.78) 153.015 毫秒 157.076 毫秒 150.759 毫秒
17 POS7-0.GW4.TOR2.ALTER.NET (152.63.131.141) 143.956 毫秒 146.313 毫秒 141.405 毫秒
18 akainn-gw.customer.alter.net (209.167.167.118) 384.687 毫秒 310.406 毫秒 302.744 ←
        小姐
19 new.lpi.org (209.167.177.93) 348.981 毫秒 356.486 毫秒 328.069 毫秒
```

5 网络中的其他机器能否到达防火墙?使用 ping,或从该机器登录到防火墙,或尝试查看
从那台机器在互联网上的网页。

6 防火墙是否阻止到该特定机器的流量?也许有人阻止了进出该网站的流量。

7 检查防火墙。如果问题似乎出在防火墙上,请测试防火墙上的接口,检查防火墙规则,
检查布线等

8 我们的eth0接口up了吗?这可以通过发出命令 ifconfig eth0 来测试。

9 您的路线定义是否正确?想想默认网关之类的东西。路由表可以通过发出来查看
命令路由-n。

10 该问题是否有物理原因?检查问题是否出在布线中。这可能是电缆有缺陷或屏蔽不良。将电源电缆和数据电缆穿过同一根管子,而两者之间没有金属屏蔽,可能会导致数据传输中出现不可预测、难以重现的错误。

名称解析问题

连接失败的原因还有更多选择:

名称解析是将主机名转换为 IP 地址。如果用户尝试根据机器的主机名连接到机器,而主机名解析不正常,则根本不会建立连接。

文件 /etc/resolv.conf 包含名称服务器的 IP 地址。名称服务器是为外部网络进行名称解析的服务器。对于小型（本地)网络,可以使用 /etc/hosts 文件创建本地查找表。

此文件包含每个 IP 地址的别名列表或 FQDN（完全限定域名）（或两者）。

您可以使用命令 /usr/bin/dig（dig 是 Domain Information Groper 的首字母缩写)或 /us r/bin/host 检查名称解析。这两个命令都返回与主机名关联的 IP 地址。

```
$ 主机 ns12.zoneedit.com ns12.zoneedit.com
的地址为 209.62.64.46
```

```
$ 挖 zonetransfer.me

; <<> DiG 9.8.3-P1 <<>> zonetransfer.me ;;全局选项:+cmd

;;得到答案:
;; ->>HEADER<<- 操作码:QUERY,状态:NOERROR,id:6133 ;;标志:qr rd ra;问题:1,答案:1,权限:2,附加:1



;;问题部分: ;zonetransfer.me。在一个



;;答案部分:
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
zonetransfer.me. 7142 在 217.147.180.162

;;权限部分:zonetransfer.me。 7142 IN
NS ns12.zoneedit.com。
zonetransfer.me. 7142 IN NS ns16.zoneedit.com。

;;附加部分:
ns12.zoneedit.com。 7116 在 A 209.62.64.46

;;查询时间:1 毫秒; ;服务器:
213.154.248.156#53(213.154.248.156)
;;时间:2013 年 7 月 4 日星期四 10:59:55
;; MSG SIZE 收件人:115
```

dig 是名称解析的瑞士军刀,有很多选项。它提供精心制作的输出。 host 命令提供了一种快速方便的方法来查找已知主机的 IP 地址。

对于基于 Debian 的系统,机器本身的主机名存储在名为 /etc/hostname 或 /etc/HOSTNAME 的文件中。
在 Fedora 系统上,该名称存储在文件 /etc/sysconfig/network 中。对于所有系统,可以使用命令 /bin/hostname 找到主机名。如果没有给出任何参数,此命令将使用机器的主机名进行回复。如果参数与命令一起给出,机器的主机名将被更改。

系统初始化错误

$ 网络问题的另一个可能原因是系统初始化不正确。要查找任何初始化错误,请检查文件 /var/log/messages 或使用 /bin/dmesg 命令读取内核环形缓冲区。

安全设定

安全设置也可能是连接问题的根源。服务器可能使用 /etc/host.deny resp 阻止或允许某些客户端访问。 /etc/host.allow

网络配置

例如,如果固定网络设置是从另一个站点复制过来的,并且不适应本地情况。对于基于 Fedora 的系统,您可以在目录 /etc/sysconfig/network-scripts 中的文件中检查这些设置,对于基于 Debian 的系统,您可以在文件 /etc/network 中检查这些设置。

网络管理器

NetworkManager 是一个基于 GUI 的工具,用于管理您的网络连接。 NetworkManager 也是一种能够报告网络变化的服务。 NetworkManager 的目的是简化在 Linux 中配置网络的使用。

通常,用户设置将存储在: /home/$user/.gconf/system/networking/connections 系统设置存储在: /etc/Networkmanager/ /etc/NetworkManager/system-connections 请注意 NetworkManager将覆盖对网络设置所做的任何配置更改。

还有一个选项可以在命令行上配置您的 NetworManager。它称为 nmcli,您可以在 /us r/bin/nmcli 中找到它。

问题和解答

网络配置

1、没有联网的系统需要配置loopback接口吗?

是的,因为环回接口在正常操作期间被各种本地服务使用。使用环回接口[122]

版权所有 2013-2017 Snow BV

Machine Translated by Google

2、配置默认网关的目的是什么？

为了将网络流量路由到本地网络以外的目的地，使用了默认网关。定义默认网关[123]

3. 哪个命令用于扫描可用的无线网络？

iwlist 是扫描可用无线网络的命令。扫描无线网络

4. VPN 实施 - 如 IPSEC、VPND、SSH 和（许多)Cisco 路由器 - 有什么共同点？

它们都通过可能不安全的网络（如互联网)创建安全隧道。常见于 VPN [128]

5. 说出在 IPSEC 实现中使用的三个协议。

ESP（封装安全有效负载）、AH（身份验证标头)和 IKE（互联网密钥交换）。 IPSEC 使用的协议[128]

6. 最先使用哪个工具来检测您的 IP 地址在同一网络中的多次使用？

使用 arp 可以确定哪些 MAC 地址正在使用您的 IP 地址。检测 IP 地址[125]

7. 沿指定目的地的路径上的每个系统将把什么 ICMP 消息发送回 traceroute 客户端？

ICMP TIME_EXCEEDED 回复被发送回数据包的源地址。这是因为 traceroute 向目的地发送了一个数据包流，从 TTL 字段设置为 1 开始，每次递增 1，直到目的地回复。跟踪路由和 ICMP [134]

版权所有 2013-2017 Snow BV

Machine Translated by Google

第6章

系统维护 (206)

本主题总权重为 7 分,包含以下目标:

目标 206.1;从源代码制作和安装程序（4 分） 考生应该能够制作和安装 exe 来自源代码的可切割程序。此目标包括能够解压缩源文件。

目标 206.2;备份操作（3分） 考生应能使用系统工具备份重要系统数据。

目标 206.3;就系统相关问题通知用户（1 分） 考生应该能够就当前与系统相关的问题通知用户。

从源代码制作和安装程序 (206.1)

考生应该能够从源代码构建和安装可执行程序。此目标包括能够解压缩源文件。

资源:tar(1)、gzip(1)、bzip2(1)、xz(1)、make(1)、uname(1) 和 install(1) 的手册页。

关键知识领域

使用常见的压缩和归档实用程序解压源代码

了解调用 make 来编译程序的基础知识

将参数应用于配置脚本

了解默认情况下存储资源的位置

条款和实用程序

- /usr/源代码
- 压缩包
- gzip
- bzip2
- xz
- 柏油

版权所有 2013-2017 Snow BV

Machine Translated by Google

- 配置
- 制作
- 名称
- 安装
- 修补

解包源代码

大多数开源软件都以压缩包的形式分发,其中包含源代码和用于编译和安装软件的构建脚本。最好在 /usr/src/ 中提取和编译源代码,但任何位置都可以。

这些 tarball 通常使用 gzip、bzip2 或 xz 压缩。 GNU tar 支持这些压缩格式,并且可以轻松解压缩此类文件。例如,要解压一个 gzip 压缩的 tarball:

```
$ tar zxvf /path/to/tarball.tar.gz
```

z 选项告诉 tar 使用 gzip 算法,x 选项告诉 tar 提取文件。要提取 bzip2 压缩文件,请使用 GNU tar 的 j 选项:

```
$ tar jxvf /path/to/tarball.tar.bz2
```

要提取 xz 压缩文件,请使用 GNU tar 的 J 选项:

```
$ tar Jxvf /path/to/tarball.tar.xz
```

虽然 GNU tar 支持这些压缩算法,但其他几个 tar 实现不支持。要在具有此类 tar 实现的机器上提取压缩的 tarball,您首先需要解压缩文件,然后提取内容。

对于 gzip 压缩的 tarball:

```
$ gunzip /path/to/tarball.tar.gz
```

对于 bzip2 压缩的 tarball:

```
$ bunzip2 /path/to/tarball.tar.bz2
```

对于 xz 压缩包:

```
$ unxz /path/to/tarball.tar.xz
```

作为替代方案,您还可以对 gzip、bzip2 和 xz 命令使用“-d”(解压缩)参数。

解压缩后,您可以通过调用 tar 来提取内容,而无需提供压缩参数:

```
$ tar xvf /path/to/tarball.tar
```

从源头构建

通常构建脚本是使用 GNU autoconf 和 automake 生成的。 automake 用于生成符合 GNU 编码标准的 Makefile.in 文件。 autoconf 生成自包含的配置脚本,然后可以使用这些脚本在目标系统上调整、构建和安装软件。

从源代码构建和安装软件的常用方法如下所示:

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
$ tar zxvf snow-2.0.3.tar.gz $ cd snow-2.0.3 $ ./
configure $ make $ su # make install
```

./configure 命令将检查可选和强制依赖项。它还将使源代码适应目标系统（考虑系统架构、安装的库、编译器标志、安装目录，...）。如果缺少可选的依赖项，它将禁用对该依赖项的编译。如果缺少必需的依赖项，它将打印错误并退出。

根据 GNU 标准，上述命令将在 /usr/local 下安装“snow”应用程序。如果您想将应用程序安装在其他目录结构下，例如 /opt，则配置命令如下所示：

```
$ ./configure --prefix=/opt
```

尝试 ./configure --help查看所有可能的配置参数。

配置命令还会生成 Makefile，用于编译和安装软件。 Makefile 通常包含几个“构建目标”，您可以通过将它们作为 make 的参数来调用它们。经常使用的目标包括“all”（通常是默认操作）、“clean”（从源代码树中清除（删除）构建的目标文件）和“install”（在构建阶段后安装软件）。

可以将软件安装在构建目录以外的位置。如果您想在一台机器上构建软件并将其安装在另一台机器上，这将很有用：

```
$ tar zxvf snow-2.0.3.tar.gz $ cd snow-2.0.3 $ ./
configure --prefix=/opt/snow $ make DESTDIR=/
tmp/snow_tmp install
```

这种技术通常用于在一台机器上构建软件，并将其安装到其他多台机器上。在上述情况下，/tmp/snow_tmp 目录将仅包含由 snow-2.0.3 安装的文件。

如果不是针对另一个平台进行交叉编译，configure 将使用 uname 来猜测它在什么机器上运行。 Configure 通常可以猜测其运行的系统类型的规范名称。为此，它运行一个名为 config.guess 的脚本，该脚本使用 uname 命令或 C 预处理器预定义的符号来推断名称。

```
$ cd /tmp/snow_tmp $ tar
zcf ../snow_2.0.3_built.tgz opt/
```

tar 的 c 选项（与前面提到的 x 选项相反）告诉它创建一个新的存档。

如果将生成的 tarball 复制到目标机器中，比如说 /tmp 目录，则可以执行以下命令将 snow 软件安装到 /opt/snow 中：

```
# cd / #
tar zxvf /tmp/snow_2.0.3_built.tgz
```

作为使用 make 脚本的替代方法，您可以使用安装实用程序将二进制文件复制到所需位置。当您使用安装复制文件时，权限和所有者/组信息将被正确设置。如果目标文件已经存在，它们将被覆盖。但是您可以使用 -b 参数或使用 VERSION_CONTROL 环境变量让安装创建这些文件的备份。

修补

补丁是一个更新文件的程序。简而言之：在原始文件上应用差异文件。 patch 获取一个包含 diff 程序生成的差异列表的补丁文件，并将这些差异应用于一个或多个原始文件，从而生成补丁版本。

版权所有 2013-2017 Snow BV

Machine Translated by Google

通常,修补版本会取代原始版本。要打补丁的文件名通常取自补丁文件,但如果只有一个要打补丁的文件,可以在命令行中指定为originalfile。

以下是有关补丁如何工作的示例:

应用补丁

```
$ patch -p1 < /path/to/path-xyz
```

还原补丁

```
$ patch -R -p1 < /path/to/patch-xyz
```

注意修

补不是 LPIC-2 目标的一部分,但包含在此处是因为它在您从源代码构建时经常使用。

备份操作 (206.2)

考生应能使用系统工具备份重要的系统数据。

参考资料: LinuxRef07、 Wirzenius98。

关键知识领域

有关必须包含在备份中的目录的知识

了解 Amanda、Bacula、Bareos 和 BackupPC 等网络备份解决方案

了解磁带、CDR、磁盘或其他备份介质的优缺点

执行部分和手动备份

验证备份文件的完整性

部分或完全恢复备份

Bareos的意识

条款和实用程序

- /bin/sh
- 日
- 柏油
- /dev/st* 和 /dev/nst*
- 公吨
- 同步

版权所有 2013-2017 Snow BV

Machine Translated by Google

为什么?

进行备份是系统管理员 Pavlov 对至少要管理一个系统的反应。但是我们还需要备份吗?
毕竟,现在的数据往往存储在 RAID 机柜或“云端”。

这得看情况。首先,超冗余存储只会保护您免受数据丢失的一个原因,即硬件故障。但不是针对人为错误、软件错误和自然灾害。

毕竟,人是很不可靠的,他们可能会犯错误或恶意并故意破坏数据。现代软件甚至不假装可靠。坚如磐石的程序是例外,而不是规则。大自然可能并不邪恶,但有时也可能具有很强的破坏性。最可靠的东西是硬件,但它仍然看似自发地损坏,而且通常是在最糟糕的时间。

为了保护自己免受这些威胁中的任何一种,可以使用一种非常便宜且简单的控制方法:定期备份。数据丢失成本与备份成本之间的收支平衡点非常低。事实上,该控件过去非常便宜,以至于使用它成为了可靠的最佳实践。

但请注意,有些情况下不需要进行备份。一个例子是数学集群。其计算结果很重要,您可能会对这些结果进行备份。但集群节点本身是可丢弃的。如果某个节点出现故障,您只需将其替换为新安装的节点即可。

因此,总而言之:备份应该始终是正确风险分析的结果。但在绝大多数情况下,它们是确保数据可用性以及在某些情况下确保数据完整性的廉价方法。

什么?

您需要备份的应该是正确风险分析的结果。在过去,大多数系统管理员只是尽可能多地进行备份 这可能不再是一个好主意,因为备份和恢复大量数据需要花费大量时间。在所有情况下,您都不需要备份 Linux 文件系统的某些部分,例如 /proc 和 /sys 文件系统。它们只包含内核自动生成的数据,备份它绝不是一个好主意。 /proc/kcore 文件尤其没有必要,因为它只是您当前物理内存的映像;它也很大。

操作系统经常更改的一些特殊文件(例如 /etc/mtab)不应恢复,因此不应该备份。您的系统上可能还有其他人。

灰色区域包括新闻假脱机、日志文件和 /var 中的许多其他内容。您必须决定您认为重要的是什么 进行适当的风险分析。此外,请考虑如何处理 /dev 中的设备文件。大多数备份解决方案都可以备份和恢复这些特殊文件,但您可能希望使用脚本重新生成它们,或者它们可能在您重新安装系统时创建。

要备份的明显内容是用户文件 (/home) 和系统配置文件(/etc,但可能还有散布在整个文件系统中的其他内容)。

通常,备份发生故障后重建系统所需的一切是个好主意 仅此而已。
通常很难找出要备份什么不备份什么,因此系统管理员范例:有疑问时,备份它。

什么时候?

根据数据的变化率,这可能是从每天到几乎从不。后者发生在防火墙系统上,其中只有日志文件每天更改(但无论如何日志记录应该发生在不同的系统上)。因此,只有在更新系统或安全更新之后才需要备份。

要确定您的最低备份频率,请考虑您确信可以在不对您和/或您的公司造成严重后果的情况下丢失的数据量。然后考虑创建该数据量所需的最小时间跨度。

您应该在该时间段内至少进行一次备份。诸如制作备份所需的时间以及制作备份时您是否可以做任何工作等实际考虑因素将限制最大频率。过去系统主要在白天使用,周末根本不用。因此,许多公司在夜间进行增量备份,在周末进行完整备份。然而,如今许多系统需要 24 x 7 全天候使用,这就需要替代策略,例如在磁盘上创建快照,随后可以在生产系统继续在实时文件系统上工作时将其备份到磁带上。

备份可能需要几个小时才能完成,但是,对于成功的备份策略,人工交互应该最少。备份主要是在磁盘或磁带上进行的。在后一种情况下,可以通过使用磁带机器人来进一步减少人机交互

版权所有 2013-2017 Snow BV

Machine Translated by Google

给你换磁带。但是,请确保将磁带存储在异地,这样任何灾难都不会破坏您的备份。并确保保护异地存储的磁带上的敏感数据,例如对存储在磁带上的数据进行加密。

如何?

虽然用于备份的硬件和软件的品牌或技术并不重要,但在选择它们时仍有重要的考虑因素。例如,想象一下,恢复软件出现故障,出版商从此倒闭。

无论您如何创建备份,备份策略中最重要的两个部分是:

验证备份 最安全的方法是回读整个备份并将其与原始文件进行比较。这是非常耗时的,而且通常不是一种选择。一种更快且相对安全的方法是在备份期间创建一个目录(其中应包含每个文件的校验和)。然后,阅读磁带的内容并比较两者。

测试恢复过程 这意味着您必须有一个恢复过程。此还原过程必须指定如何将任何内容从单个文件还原到整个系统。每隔几个月,您应该通过执行还原来测试此过程。

在哪里?

如果在备份过程中出现故障,该介质将不会包含任何有用的信息。如果您在唯一的介质上进行了备份,您就会丢失数据。所以你应该至少有两套备份媒体。但是,如果您将两套设备存储在同一栋建筑物中,那么随之而来的第一场灾难将摧毁您所有宝贵的备份以及正在运行的系统。

因此,您应该至少将一组存储在远程站点。根据数据的性质,您可以远程存储每周或每天的数据集。

您将需要一个书面计划,即备份计划,它描述了备份的内容、如何恢复正在备份的内容以及与备份相关的任何其他程序。不要忘记将备份计划的副本与远程站点的备份一起存储。否则您无法保证系统会在尽可能短的时间内恢复运行。

需要注意的一件重要事情,如前几段所述,您需要能够根据需要尽快重建系统(或恢复某些文件)。在某些环境中,由于网络线路缓慢或其他原因,恢复时间可能长达数小时。
损失的时间可能太多,可能会破坏备份系统的目的。建议使用其他服务连续性解决方案,例如集群/故障转移系统。

有不同类型的备份介质,每种都有自己的优点和缺点。然而,通常会根据总成本来选择媒介。主要类型有:

胶带
磁盘
光媒体
远程/网络

胶带

磁带是企业环境中最常用的备份介质之一。它成本低,而且由于磁带是被动存储,因此它们发生故障的可能性很低,并且待机时消耗的电量也很少。磁带的一个缺点是它是一种流媒体,这意味着访问时间长,尤其是当磁带机器人用于访问多个磁带时。如果以连续流的形式提供/请求数据,带宽可能会很高。磁带特别适用于长期备份和归档。如果必须写入或恢复大量小文件,则需要停止和启动磁带,甚至可能需要经常部分倒带以允许恢复操作系统进行处理。这可能会消耗过多的时间。在那些情况下,磁带不是最好的媒介。

版权所有 2013-2017 Snow BV

Machine Translated by Google

磁盘

本地磁盘存储几乎不用于备份（尽管它用于网络存储,见下文）。与容量相比,优点是高带宽、低延迟和合理的价格。但它不适合异地备份（或者必须手动断开磁盘并运送到安全位置）。由于磁盘始终处于连接和运行状态,因此发生故障的可能性很高。虽然不适合异地备份,但有时用作备份系统和异地备份服务器之间的中间（缓冲)介质。优点是可以快速恢复最近的文件,并且生产系统不会被长时间的备份传输占用。另一种适用于较小系统的选择是使用廉价的基于 USB 总线的便携式磁盘。这些可以在不使用时停止旋转并关闭电源,同时仍保留其数据。它们可以包含数 TB 的数据,并且可以轻松地转移到异地。此外,现代磁盘采用快速 USB 协议来减少备份和恢复时间。

光媒体

像CDROM和DVDR磁盘这样的光学介质主要用于备份不会有太大变化的系统。通常会将系统的完整映像保存到磁盘以供快速恢复。光盘成本低,并且在正确存储时具有高可靠性。
它们可以很容易地运到异地。缺点是大多是一次写入,存储容量低。访问时间和带宽适中,尽管大多数情况下它们必须由操作员手动处理。

远程/网络存储

网络存储多为远程磁盘存储（NAS或SAN）。使用 RAID 等数据保护技术,可以降低磁盘的不可靠性。大多数现代网络存储系统使用压缩和重复数据删除来增加潜在容量。
此外,大多数系统都可以模拟磁带驱动器,这使得从磁带迁移变得容易。系统的成本可能很高,具体取决于功能、可靠性和容量。还应考虑电力成本,因为网络存储系统始终处于开启状态。因此,这种类型的介质不适合长期备份和存档。访问时间和带宽可能会有所不同并取决于基础设施,但大多都很高。

备份工具

同步

Rsync 是一种实用程序,用于将文件从一个位置复制/同步到另一个位置,同时保持所需的带宽较低。它将查看要复制的文件和已存在于目的地的文件,并使用时间戳、文件大小和高级算法来计算需要传输哪些（部分)文件。源和目标可以是本地或远程,如果是远程服务器,SSH 或 rsync 协议可用于网络传输。rsync 的调用很像 cp 命令。递归模式使用-r启用,存档使用-a。一个通过 SSH 将文件从本地目录复制到远程目录的简单示例：

```
rsync -av -e ssh /snow 远程：/snow
```

笔记

默认情况下， rsync从远程系统复制（相关部分)更改的文件和新文件。它不会删除在远程系统上删除的文件。如果您想要该行为,请指定选项--delete 。另请注意,权限将被正确复制,但如果您的本地 UID/GID 与远程设置不匹配,您最终可能会得到不正确的本地权限。

柏油

tar 实用程序用于将多个文件和目录组合成连续的字节流（并将流还原为文件/目录）。该流可以被压缩、通过网络连接传输、保存到文件或流式传输到磁带设备上。从流中读取文件时,可以恢复权限、模式、时间等信息。 Tar 是将文件和目录传输到磁带或从磁带传输文件和目录的最基本方法,无论是否进行压缩。

提取 gzipped tar 存档的示例,其中包含从文件读取的详细输出和输入数据：

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
焦油 xvzf snow.tgz
```

从 scsi 磁带驱动器中提取 tar 存档：

```
tar xvf /dev/st0
```

从目录 /snow 创建一个存档到文件：

```
裁谈会/; tar cvf /tmp/snow.tar 雪
```

默认情况下,tar 实用程序使用 (scsi) 磁带作为介质。从上面的示例中可以看出,可以在 /dev/st* 或 /dev/nst* 中找到 scsi 磁带设备。后一种是非倒带磁带,这意味着磁带不会在每次操作后自动倒带。这是备份的一个重要功能,因为否则当使用多个 tar 命令进行备份时,除最后一个备份之外的任何备份都将被下一个备份覆盖。

磁带可以通过 mt 命令控制（磁带）。该命令的语法为:mt [-h] [-f device] 命令 [count]。选项 -h（帮助)列出了所有可能的命令。如果设备未由 -f 选项指定,该命令将默认使用环境变量 TAPE。更多信息可以在手册页中找到。

dd

使用 dd 实用程序,可以将整个磁盘/分区从/转移到文件或其他磁盘/分区。使用 dd 可以立即备份整个文件系统。 dd 将在字节级别复制数据。 dd 的常见选项是：

if,输入文件:对于输入文件/磁盘/分区,输出文件：
bs,块大小:用于传输的块大小,可以根据使用的硬件
数量、要传输的块数进行优化（dd将读取直到结束否则为文件）

使用 dd 将 1GB 分区传输到文件的示例：

```
dd if=/dev/hda1 of=/tmp/disk.img bs=1024 count=1048576
```

cpio

cpio 实用程序用于将文件复制到存档或从存档复制文件。它可以读/写各种存档格式,包括 tar 和 zip。
虽然它早于 tar 但鲜为人知。 cpio 具有三种模式,输入模式(-i)读取存档文件并提取文件,输出模式(-o)读取文件列表并将它们压缩到存档文件中,传递模式(-p)读取文件文件列表并将这些复制到目标目录。文件列表从stdin读取,通常由 find 提供。将目录压缩到 cpio 存档的示例：

```
%cd /雪;寻找 。                    | cpio -o > snow.cpio
```

备份解决方案

完整的备份解决方案有助于简化大型环境中备份的管理和配置。
这些解决方案可以自动将多个服务器和/或客户端备份到多个备份介质。存在许多不同的解决方案,每个都有自己的优点和缺点。您将在下面找到这些解决方案的一些示例。

AMANDA,Advanced Maryland Automatic Network Disk Archiver,是一种备份解决方案,允许 IT 管理员设置单个主备份服务器,通过网络将多个主机备份到磁带驱动器/转换器或磁盘或光学介质。
Amanda 使用本机实用程序和格式（例如转储和/或 GNU tar)并且可以备份大量运行多个版本的 Linux 或 Unix 的服务器和工作站。

Bacula 是一组开源的、企业就绪的计算机程序,允许您（或系统管理员)管理跨不同类型计算机网络的计算机数据的备份、恢复和验证。 Bacula 相对易于使用且高效,同时提供许多高级存储管理功能,可以轻松查找和恢复丢失或

版权所有 2013-2017 Snow BV

Machine Translated by Google

损坏的文件。用技术术语来说,它是一个开源的、企业就绪的、基于网络的备份程序。根据 Source Forge 的统计数据（排名和下载量）,Bacula 是迄今为止最受欢迎的企业级开源程序。

Bareos 是 Bacula 5.2 版项目的一个分支,由于拒绝和忽视社区对 Bacula 项目的贡献而启动。 Bareos 备份程序是开源的,与 Bacula 几乎相同,但它确实具有一些附加功能,如 LTO 硬件加密、带宽限制和新的实用控制台命令。 Bareos 开发的重点之一是尽可能降低新手的障碍。因为新手通常对配置选项不知所措,所以 Bareos 项目为流行的 Linux 发行版和 Windows 提供了包存储库。

BackupPC 是一个高性能的企业级系统,用于将 Linux、WinXX 和 MacOSX PC 和笔记本电脑备份到服务器的磁盘。 BackupPC 具有高度可配置性并且易于安装和维护。

通知用户系统相关问题 (206.3)

考生应该能够将与系统相关的当前问题通知用户。

关键知识领域

通过登录消息自动与用户通信

通知活跃用户系统维护

条款和实用程序

- /etc/问题
- /etc/issue.net
- /etc/motd
- 墙
- /sbin/关机
- /bin/systemctl

资源:如何使用 systemctl;

/etc/issue、/etc/issue.net和/ etc / motd文件

/etc/issue、/etc/issue.net 和 /etc/motd 文件用于向登录系统的用户发送简单消息。 /etc/motd 用于在用户成功验证后显示消息。另一方面，/etc/issue 用于在用户登录之前向用户发送消息。此消息仅显示给使用控制台登录的用户,并且通常会包含一些有关系统的信息,例如内核版本和体系结构。 /etc/issue.net 文件与 /etc/issue 文件具有相同的用途,但用于使用 telnet 的不安全登录。也可以将 /etc/issue.net 文件用于 ssh。为此,您需要在 /etc/ssh/sshd_config 中添加或修改以下行:

```
旗帜/etc/issue.net
```

将 /etc/issue.net 与 ssh 一起使用时,您应该注意特殊序列可能不起作用。

版权所有 2013-2017 Snow BV

Machine Translated by Google

墙命令

wall 用于向所有交互终端广播最多 22 行的消息。默认情况下,任何用户都可以使用该命令,但通常会重新配置,因此只有 root 用户可以使用它。不希望接收广播消息的用户可以使用 mesg 来写禁用他们的终端。广播公司可以使用 finger 命令查看哪些终端被禁止写入。

您可以使用 wall 命令指定两个选项：-n 选项,它仅适用于 root 用户和消息本身。
-n 取消标准广播标语并将其替换为远程广播标语。仅当在 rpc.walld 守护程序上使用 wall 时,此选项才有意义。第二个参数,消息本身也可以在 stdin 上键入,在这种情况下,它必须以 EOF（文件结尾,在大多数情况下为 Ctrl-D)终止。

关机指令通讯。

顾名思义,shutdown 命令用于正常关闭服务器,通过运行级别的 kill 脚本退出,并可选择停止或重新启动服务器。这里不讨论 shutdown 命令本身,这一小节仅解释 shutdown 在系统关闭之前和期间执行的通信步骤。

关闭的最后一个参数可以选择用于广播一些自定义消息,解释关闭的目的,以及预计何时返回生产。例如：

```
# shutdown -H +10 服务器在 10 分钟内停止更改更改编号。预计在 ←
        00:00:00。
```

关机可以和-k一起使用。这使得关机做一个“空运行”:它模拟关机,但不关闭系统。

当您使用 -k 选项时,您也可以将广播消息附加为命令行的一部分,就像在“真正的”关机中一样。

注意
请注意， shutdown -k仍会暂时禁止用户登录,因为它会创建/etc/nologin文件。它将在“试运行”后被删除,但只要它存在,您的用户将无法登录系统。

如果需要取消正在运行的关闭,可以使用 -c 选项调用关闭,并再次向命令行添加广播消息以通知用户掉头。与所有形式的消息广播一样,接收终端必须启用写入。

systemctl命令通信,

systemctl 是 systemd init 系统的中央管理工具。 systemctl 可用于管理服务、系统状态（运行级别)和配置文件。

管理服务

启动和停止服务

在 sysVinit 中使用 service 命令的地方,您现在将使用 systemctl 命令来管理服务。如果您使用非 root 用户运行命令,则必须使用 sudo。以下示例显示使用 start 命令启动服务：

```
$ systemctl 启动应用程序服务
```

因为 systemd 知道您正在服务上运行系统管理命令,所以您也可以保留 .service 后缀。
为了清楚起见,我们将继续在命令中使用后缀。

```
$ systemctl 启动应用程序
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

对于某些程序,可以使用不同的配置文件多次启动应用程序。在这种情况下,您可以使用 @ 符号将配置文件的名称传递给命令。例如,要使用不同的配置文件两次启动 openvpn 服务,您可以使用以下命令:

```
$ systemctl 启动 openvpn@config1.service $ systemctl 启动
openvpn@config2.service
```

因为他们更改了 systemctl 命令的参数顺序,您还可以一次启动和停止多个服务:以下示例显示使用 stop 命令停止多个服务:

```
$ systemctl 停止 application1.service application2.service
```

重新启动和重新加载服务

要重新启动服务,您可以使用重新启动命令:

```
$ systemctl 重启应用程序服务
```

如果应用程序能够重新加载它的配置,您还可以使用 reload 命令:

```
$ systemctl 重新加载 application.server
```

如果您不确定服务是否可以重新加载其配置,您也可以使用 reload-or-restart 命令。如果可用,这将重新加载配置,否则将重新启动应用程序:

```
$systemctl 重新加载或重新启动 application.service
```

启用和禁用服务

前面的命令对于在当前会话中启动和停止服务很有用。如果你想让一个服务在启动时启动,为此 sysVinit 使用 chkconfig 命令,你必须使用 systemctl 启用它们:

```
$ systemctl 启用应用程序服务
```

此启用命令将创建一个符号链接,从该服务文件的系统副本（通常位于 /lib/sys temd/system 或 /etc/systemd/system 中)到 systemd 查找自动启动文件的目录（通常为 /etc /systemd/system/- some_target.target.wants）。要在启动时禁用服务,请使用 disable 命令:

```
$ systemctl 禁用应用程序服务
```

这将删除指示服务应在引导时启动的符号链接。

注意
请记住,启用服务不会在当前会话中启动它。要启动和启用服务,您需要同时发出启动和启用命令

检查服务状态

要检查服务的当前状态,您可以使用 status 命令:

```
$ systemctl status application.service
```

此命令将为您提供服务的当前状态、cgroup 层次结构和前几行日志。它可以让您很好地了解当前状态,并通知您任何问题。例如 sshd 服务的以下输出:

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
$ systemctl status sshd.service · sshd.service - OpenSSH 服
    务器守护进程
        已加载:已加载（/usr/lib/systemd/system/sshd.service;已启用;供应商预设:←。
            启用）
        活跃:自周五 2016-12-16 08:18:17 CET 以来活跃（运行）； 4 小时 58 分钟前
            文档:man:sshd(8)
                人:sshd_config(5)
      进程:1033 ExecStart=/usr/sbin/sshd $OPTIONS（代码=退出,状态=0/成功）
    主 PID:1067 (sshd)
        任务:1（限制:4915）
        CGroup: /system.slice/sshd.service    -1067 /usr/sbin/sshd


    12 月 16 日 08:18:17 localhost.localdomain systemd[1]:正在启动 OpenSSH 服务器守护程序...
    12 月 16 日 08:18:17 localhost.localdomain systemd[1]: sshd.service: PID 文件 /var/run/ ←。
        启动后 sshd.pid 不可读（还？）:没有这样的文件或目录
    Dec 16 08:18:17 localhost.localdomain sshd[1067]:服务器侦听 0.0.0.0 端口 ←。
        22.
    12 月 16 日 08:18:17 localhost.localdomain sshd[1067]:服务器侦听 :: 端口 22。
    12 月 16 日 08:18:17 localhost.localdomain systemd[1]:已启动 OpenSSH 服务器守护进程。
```

还有一些命令可用于检查特定状态,这对于在脚本中使用它们特别有用。例如,要检查服务当前是否处于活动状态,您可以使用 is-active 命令：

```
$ systemctl is-active application.service
```

此命令将显示应用程序是活动的还是非活动的。如果应用程序处于活动状态,它将返回退出代码“0”。要查看应用程序是否已启用,您可以使用 is-enabled 命令：

```
$ systemctl 已启用 application.service
```

无论应用程序是启用还是禁用,此命令都将返回,如果应用程序已启用,将返回退出代码“0”。要检查应用程序是否失败,您可以使用 is-failed 命令：

```
$ systemctl is-failed application.service
```

如果应用程序正在运行,此命令将返回活动状态,如果发生错误则失败,并且不活动或未知服务是否被有意停止。如果服务失败,它将返回退出代码“0”。

系统状态概览

到目前为止,这些命令对管理单个服务很有用。为了探索系统的当前状态,有许多 systemctl 命令可以提供更多信息。

列出当前单位

要获取 systemd 知道的所有活动单元的列表,您可以使用 list-units 命令：

```
$ systemctl 列表单位
```

此命令仅显示当前活动单位的列表。然后输出有以下列：

```
UNIT:systemd 单元名称 LOAD:该单元的配置是否已被
systemd 解析。加载单元的配置保存在内存中。

ACTIVE:有关单元是否处于活动状态的摘要状态。这通常是一个
        判断设备是否成功启动的基本方法。
SUB:这是一个较低级别的状态,表示更详细的信息
        关于单元 这通常因单元类型、状态和单元运行的实际方法而异。

DESCRIPTION: 对单元是什么/做什么/的简短文本描述
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

因为 list-units 命令默认只显示活动单位,所以上面的所有条目将在 LOAD 列中显示“loaded”,在 ACTIVE 列中显示“active”。这也是在没有附加命令的情况下调用 systemctl 时的默认行为,因此如果您在没有参数的情况下调用 systemctl,您将看到相同的输出。

```
$ 系统控制
```

您还可以通过添加额外的标志来告诉 systemctl 输出不同的信息。例如,要显示 systemd 已加载的所有单元,无论它们是否处于活动状态,您可以使用 --all 标志:

```
$ systemctl 列表单元 --all
```

这将显示 systemd 加载或尝试加载的所有单元,无论它在系统上的当前状态如何。也可以按当前状态过滤单元,为此您可以使用 --state= 标志。您必须保留 --all 标志,以便 systemctl 允许显示非活动单元。例如,如果您希望查看所有非活动单元,您可以发出以下命令:

```
$ systemctl --list-units --all --state=inactive
```

您可以使用的另一个过滤器是 --type= 标志。您可以告诉 systemctl 只显示您感兴趣的单元类型。例如,要只显示活动服务单元,您可以使用以下命令:

```
$ systemctl --list-units --type=服务
```

列出单元文件

我们刚刚使用的 list-units 命令仅显示 systemd 尝试加载到内存中的单位。 Systemd 只会读取它认为需要的单元,因此这不一定包括系统上所有可用的单元。要查看 systemd 路径中可用的每个单元文件,您可以改用 list-unit-files 命令:

```
$ systemctl 列表单元文件
```

单位是 systemd 知道的资源的表示。因为 systemd 不一定读取此视图中的所有单元定义,所以它只显示有关 hte 文件本身的信息。此命令的输出显示两列,UNIT FILE 和 STATE。 STATE 列通常是“enabled”、“disabled”、“static”或“masked”。对于此命令,静态意味着单元文件不包含启用服务所必需的“安装”部分。具有静态状态的单元无法启用将运行一次性操作或仅用作另一个单元的依赖项。稍后我们将介绍“屏蔽”的含义。

单位管理

到目前为止,我们一直在使用服务来显示有关 systemd 知道的单元文件的信息。通过一些额外的命令,我们可以获得有关单位的更多具体信息。

显示单元文件

要显示 systemd 已加载到其内存中的单元文件,您可以使用 cat 命令(在 systemd 版本 209 中添加)。要查看 atd 调度守护进程的单元文件,您可以使用以下命令:

```
$ systemctl 猫 atd.service

 [单元]
说明=ATD 守护进程

 [服务]
类型=分叉
ExecStart=/usr/bin/atd

 [安装]
WantedBy=多用户.target
```

此命令的输出是当前 systemd 进程已知的单元文件。了解您最近是否修改了单元文件或您是否覆盖了某些选项很重要。

版权所有 2013-2017 Snow BV

Machine Translated by Google

显示依赖关系

如果我们想查看服务的依赖树,我们可以使用 list-dependencies 命令:

```
$ systemctl 列表依赖项 sshd.service
```

此命令将显示必须处理以启动单元的依赖关系的层次视图。在此上下文中,依赖项是它上面的单元所需要或想要的单元。

```
服务
 |-system.slice    -
 basic.target |-
      microcode.service |-rhel-autorelabel-
      mark.service |-rhel-autorelabel.service |-rhel-
      configure.service |-rhel-dmesg.service |-rhel-
      loadmodules .service |-paths.target |-slices.target




. . .
```

递归相关性仅针对指示系统状态的目标单元显示。要列出所有依赖项,您可以包含 --all 标志。

要获得反向依赖关系,您可以将 --reverse 标志添加到命令中。其他有用的标志是 --before 和 --after 标志,它们可用于显示依赖于指定单元的单元分别在它们之前或之后开始。

检查单元属性

要获取单位的低级属性,您可以使用 show。这将以键=值格式显示属性列表:

```
$ systemctl 显示 sshd.service

id=sshd.service
名称=sshd.service
需要=basic.target Wants=system.slice
WantedBy=multi-user.target
Conflicts=shutdown.target Before=shutdown.target
multi-user.target After=syslog.target
network.target auditd.service systemd-journald.socket basic。目标←

        系统切片
Description=OpenSSH 服务器守护进程

. . .
```

要获得单个属性,您可以传递带有属性名称的 -p 标志。例如,要查看 sshd.service 单元的冲突,您可以使用以下命令:

```
$ systemctl show sshd.service -p 冲突

冲突=shutdown.target
```

掩蔽和解除掩蔽单位

在服务管理部分,我们展示了如何停止或禁用服务,但 systemd 也有能力将一个单元标记为完全无法启动。为此,它创建了一个指向 /dev/null 的符号链接,称为屏蔽单元。为此,您可以使用 mask 命令:

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
$ systemctl 掩码 nginx.service
```

这将防止 Nginx 服务在被屏蔽时被手动或自动启动。如果您使用 list-unit-files 命令检查,您将看到该服务现在被列为屏蔽:

```
$ systemctl 列表单元文件

. . .

kmod-static-nodes.service                                    静止的
ldconfig.service mandb.service                               静止的
                                                             静止的
messagebus.service nginx.service                             静止的
quotaon.service rc-local.service                             蒙面的
                                                             静止的
                                                             静止的
磁盘服务                                                     禁用
救援服务                                                     静止的

. . .
```

如果您尝试启动该服务,您将看到如下消息:

```
$ systemctl 启动 nginx.service

nginx.service 启动失败:nginx.service 单元被屏蔽。
```

要取消屏蔽服务,您可以使用 unmask 命令:

```
$ systemctl 取消屏蔽 nginx.service
```

这将使设备返回到它之前的启动状态,从而允许启动或启用服务。

编辑单元文件

如果您需要进行调整,systemctl 命令还可以编辑单元文件。此功能是在 systemd 版本 218 中添加的。

默认情况下,编辑将打开一个单元文件截图:

```
$ systemctl 编辑 nginx.service
```

这将是一个空白文件,可用于覆盖或向单元定义添加属性。将在 /etc/systemd/system 目录中创建一个目录,该目录将在单元名称后附加 .d。例如,对于 nginx.service,将创建一个目录 callend nginx.service.d。

如果你想编辑完整的单元文件,而不是添加一个片段,你可以传递 --full 标志:

```
$ systemctl edit --full nginx.service
```

这会将当前单元文件加载到编辑器中,您可以在其中修改它。当编辑器退出时,更改将写入 /etc/systemd/system。这个新文件将优先于系统单元定义(通常在 /lib/systemd/system 中找到)。

要删除任何添加的内容,您可以从 /etc/systemd/system 中删除 units .d 配置目录或修改后的服务文件。例如,要删除 snipper,您可以键入:

```
$ rm -r /etc/systemd/system/nginx.service.d
```

要删除完整修改的单元文件,您可以键入:

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
$ rm /etc/systemd/system/nginx.service
```

删除文件或目录后,您应该重新加载 systemd 进程,以便它不再尝试引用该文件并恢复使用系统副本。您可以使用 daemon-reload 命令执行此操作:

```
$ systemctl 守护进程重新加载
```

使用目标调整系统启动（运行级别）

目标是描述系统状态的特殊单元文件。与其他单元一样,定义目标的文件可以通过其后缀来识别,在本例中为 .target。目标本身并没有做太多事情,而是用来将其他单位组合在一起。

这些目标可用于使系统进入特定状态,就像其他 init 系统使用运行级别一样。它们用作某些功能何时可用的参考,允许您指定所需的状态,而不是产生相同状态所需的各个单元。

例如,有一个 swap.target 用于指示交换已准备好使用。作为此过程一部分的单元可以通过在其配置文件中指示 swap.target 需要或需要它们来与此目标同步。需要交换可用的单元可以通过使用 wants、requires 和 after 属性来指定此条件,以指示它们关系的性质。

获取和设置默认目标

Systemd 有一个在引导系统时使用的默认目标。满足该目标的级联依赖关系将使系统进入所需状态。要获取系统的默认目标,您可以使用 get-default 命令:

```
$ sytemctl 获取默认值

多用户.target
```

如果要将另一个目标设置为默认目标,可以使用 set-default 命令。例如,如果您想默认使用图形桌面,您可以使用以下命令更改它:

```
$ systemctl set-default graphical.target
```

列出可用目标

您可以结合使用 list-unit-files 命令和 --type=target 过滤器来获取可用目标的列表:

```
$ systemctl 列表单元文件 --type=target
```

与运行级别不同,可以同时激活多个目标。活动目标表示 systemd 已尝试启动与目标相关的所有单元,并且没有尝试再次将它们拆除。要查看所有活动目标,请使用以下命令:

```
$ systemctl list-units --type=目标
```

隔离目标

可以启动与目标关联的所有单元并停止不属于依赖树的所有单元。我们可以为此使用的命令是 isolate 命令。这类似于更改其他 init 系统中的运行级别。

例如,如果您在 graphical.target 处于活动状态的图形环境中工作,则可以通过隔离 multi-user.target 来关闭图形系统。由于 multi-user.target 是 graphical.target 的依赖项,而不是相反,隔离命令将停止所有图形单元。

您可能希望查看要隔离的目标的依赖项,以确保您不会停止任何对您至关重要的服务:

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
$ systemctl list-dependencies 多用户.target
```

当您对将保持存活的单位感到满意时,您可以隔离目标:

```
$ systemctl 隔离多用户.target
```

为重要事件使用快捷方式

一些目标是为重要事件定义的,例如关闭电源或重新启动。然而 systemctl 也有一些快捷方式可以添加一些额外的功能。

例如,要将系统置于救援（System V 初始化术语中的单用户)模式,您可以只使用救援而不是隔离救援。目标:

```
$ systemctl 救援
```

与 isolate 命令相比,此命令还将提供额外的功能,即向所有登录用户发出有关事件的警报。要停止系统,您可以使用 halt 命令:

```
$ systemctl 停止
```

要启动完全关机,您可以使用 poweroff 命令:

```
$ systemctl 关机
```

要重新启动系统,您可以使用 reboot 命令:

```
$ 系统重启
```

并不是说大多数系统会为这些操作链接更短、更传统的命令,以便它们可以与 systemd 一起正常工作。例如,要重新启动系统,您通常可以键入:

```
$ 重启
```

问题和解答

系统维护

1. 哪个 GNU tar 参数支持处理 bzip2 压缩文件?

   参数 j 用于处理 bzip2 压缩文件。解压缩 tarball [145]

2. 配置命令过早退出的最常见原因是什么?

   如果缺少必需的依赖项。但是,如果只缺少一个可选的依赖项,它只会禁用该特定依赖项的编译。检查可选和强制依赖项[146]

3. 根据GNU 标准,应用程序默认安装到哪个目录?

   如果没有将可选参数传递给 configure 或 make install 命令,应用程序将安装到 /usr/local/ 目录中。默认安装目录[146]

4. 一般来说,你宁愿多备份也不愿少备份。哪些文件系统是主要例外,可以全部跳过
   一起?

   /proc 和 /sys 文件系统包含有关内核状态的信息,因此可以跳过。什么不要备份[148]

版权所有 2013-2017 Snow BV

Machine Translated by Google

5. 你应该多久备份一次你的系统?

备份频率应与您在给定时间跨度内创建的新数据量同步。如果它很多,或者很难重新创建,请考虑备份之间的短暂间隔。什么时候备份? [148]

6. 为什么网络附加存储不是长期备份和归档的首选?

几乎从未使用过长期备份,即使使用过,也不希望立即恢复。了解这一点后,应考虑电力成本,因为这些系统始终处于运行状态。作为备份介质的网络存储[150]

7. 如果本地和远程目标之间的带宽很低,为什么 rsync 是要考虑的备份实用程序?

rsync 不仅通过压缩流量来优化带宽的使用,而且还考虑在不需要时不复制文件,并对大文件的部分执行此操作。保持所需的带宽较低[150]

8. 在备份大量运行不同版本的 Linux 和 Unix 的系统时,为什么要使用 AMANDA（Advanced Maryland Automatic Network Disk Archiver）?

AMANDA 使用本机实用程序和格式,很可能支持多个版本的 Linux 和 Unix,并能够允许跨系统移动备份文件。备份解决方案[151]

9. ping 一个系统是否提供了足够的信息来决定一个系统是否联机?

不,因为防火墙可能会阻止 ICMP 回应请求。同样也可能阻止回复。可以阻止 Ping [133]

10. 什么时候将名为 /etc/issue 的文件中的消息呈现给用户?

如果使用命令 login 进行登录,则在进行身份验证之前。大多数用于登录的客户端最终不会使用 login,但是 telnet 会。呈现 /etc/issue [152]

版权所有 2013-2017 Snow BV

Machine Translated by Google

第七章

域名服务器 (207)

本主题的权重为 8 分,包含以下目标:

目标 207.1;基本 DNS 服务器配置（3 分） 考生应该能够将 BIND 配置为用作仅缓存 DNS 服务器。此目标包括将旧的 BIND 配置文件转换为更新的格式、管理正在运行的服务器和配置日志记录的能力。

目标 207.2;创建和维护 DNS 区域（3 分） 考生应该能够为正向或反向区域或根级服务器创建区域文件。此目标包括为记录设置适当的值、在区域中添加主机以及将区域添加到 DNS。候选人还应该能够将区域委托给另一个 DNS 服务器。

目标 207.3;保护 DNS 服务器（2 分） 考生应该能够配置 DNS 服务器以非 root 用户身份运行并在 chroot 监狱中运行。此目标包括在 DNS 服务器之间安全地交换数据。

注意以

下有关 DNS 的章节仍然提到从 BIND 8 到 BIND 9 的升级。LPIC-2 考试侧重于 BIND 9。在历史的短暂时刻,有一个 BIND 10。但它被出售并重新命名。 BIND 9 是您在准备 LPIC-2 考试时应该重点关注的内容。

基本 DNS 服务器配置 (207.1)

资源和进一步阅读： Wikipedia.org 上的 Albitz01、 DNShowto、 BINDfaq、 BIND。成熟:绑定 10。

考生应该能够将 BIND 配置为充当权威和递归的、仅缓存的 DNS 服务器。此目标包括管理正在运行的服务器和配置日志记录的能力。

关键知识领域

- BIND 9.x 配置文件、术语和实用程序。
- 在BIND 配置文件中定义BIND 区域文件的位置。
- 重新加载修改后的配置和区域文件。
- 了解dnsmasq、djbdns 和PowerDNS 作为备用名称服务器。

版权所有 2013-2017 Snow BV

Machine Translated by Google

条款和实用程序

- /etc/named.conf
- /var/命名/
- /usr/sbin/rndc
- /usr/sbin/named-checkconf
- 杀死
- 挖掘
- 主持人

BIND 中的名称服务器组件

BIND（伯克利互联网名称域系统)等名称服务器是全球 DNS 系统的一部分,可将机器名称解析为 IP 地址。

在 Internet 的早期,主机名到 IP 地址的映射由网络信息中心 (NIC) 在一个名为 HOSTS.TXT 的文件中维护。然后通过 FTP 将此文件分发到其他连接到 Internet 的主机。

由于随着时间的推移有大量主机连接到 Internet,因此开发了另一种名称解析系统,称为域名。该系统结合了分布式更新和本地缓存等设计目标,以提高解析性能。由于这些特性,每个名称服务器都需要针对其用途进行专门配置。在配置 BIND 等名称服务器软件时,以下术语很明显:

区域等同于域。区域配置文件由主机名和 IP 地址信息组成。名称服务器软件响应端口 53 上的请求,并将 DNS（主机或域名)名称转换为 IP 地址。它还可以将 IP 地址转换为 DNS 名称,这称为“反向 DNS 查找”(rDNS)。为了使 rDNS 工作,被查询的主机必须存在所谓的指针 DNS 记录（PTR 记录）。

我们区分权威名称服务器、递归名称服务器和所谓的解析器。区域的权威名称服务器是管理区域配置的名称服务器。因此,它有时也被称为区域主机。递归名称服务器是解析区域的名称服务器,它在其他名称服务器上对其不具有权威性。解析器是执行实际查询的名称服务器和 DNS 客户端软件的一部分。通常,这些是作为 DNS 软件一部分的库。

表主要 BIND 组件列出了系统上 BIND 软件最相关的部分。请注意,目录可能因发行版而异。

组件程序 /usr/	描述实名服务器
sbin/named 程序 /usr/sbin/rndc 程序 /	名称守护程序控制程序检查
usr/sbin/named-checkconf 文件	named.conf文件错误的程序BIND配置
named.conf 程序 /etc/init.d/bind 目录 /var/named	文件分发指定startfile工作目录为named

表 7.1:主要的 BIND 组件

解析由第10 章中提到的文件 nsswitch.conf 控制。

BIND 组件将在下面讨论。

版权所有 2013-2017 Snow BV

Machine Translated by Google

named.conf文件_

named.conf文件是BIND的主要配置文件。它是 DNS 名称守护程序 named 读取的第一个配置文件。

注意

BIND 8 配置文件应该可以与 BIND 9 一起使用,尽管可能需要进行一些修改。某些选项不适用于 BIND 9。BIND 9 提供了新的配置区域,请在从 BIND 8 升级到 BIND 9 时查阅 BIND 文档和联机帮助页。

注意

可以使用名为named-bootconf.sh的脚本将 BIND 4 配置文件转换为 BIND 9 格式

---

named.conf的位置

根据 LPI,文件 named.conf 的位置在 /etc 目录中。但是,位置可能因分布而异。例如,在 Debian Linux 发行版中,named.conf 位于 /etc/bind 目录中。

仅缓存名称服务器

仅缓存名称服务器解析名称,这些名称也存储在缓存中,以便在要求名称服务器再次解析这些名称时可以更快地访问它们。但这是每个名称服务器所做的。不同之处在于,这是仅缓存名称服务器执行的唯一任务。除了一些内部区域外,它不提供区域服务。

这是一个仅用于缓存的 named.conf 文件的示例。下面的版本取自 Debian 绑定包（删除了一些注释）。

```
选项{ 目录“/var/
                named”；

                // 查询源地址 * 端口 53;

                // 转发器 { // // };
                        0.0.0.0;

};

// 在我们无法控制的问题上减少日志的冗长程度 logging { category lame-servers { null; };类别 cname
{ null; };


};

// 使用根服务器区域“.”的知识来启动服务器{

                输入提示;文件“/
                etc/bind/db.root”；
};

// 根据 RFC 1912 对本地主机正向和反向区域以及广播区域具有权威性


区域“本地主机”{
                类型大师;文件“/etc/
                bind/db.local”；
};
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
区域“127.in-addr.arpa”{
                类型大师;文件“/etc/
                bind/db.127”;
};

区域“0.in-addr.arpa”{ 类型 master;文件
                “/etc/bind/db.0”;

};

区域“255.in-addr.arpa”{
                类型大师;文件“/etc/
                bind/db.255”;
};

// 在此处添加其他区域的条目
```

包含此文件的 Debian 绑定包将提供功能齐全的仅缓存名称服务器。其他制造商的 BIND 包将提供相同的功能。

句法

named.conf 文件包含以关键字和左大括号“{”开头并以右大括号“}”结尾的语句。一个语句可以包含其他语句。 forwarders 声明就是一个例子。语句还可以包含 IP 地址或后跟文件名的文件字。这些简单的语句必须以分号 (;) 结束。

允许使用各种注释,例如，// 和# 作为行尾注释。有关详细信息,请参阅 named.conf(5) 手册页。

注意

“;”作为named.conf中的注释符号无效。但是,它是 BIND 区域文件中的注释符号,例如上面named.conf示例中的文件/etc/bind/db.local 。示例 BIND 区域文件可以在第7.2.3.1节中找到

选项声明_

在选项语句中的许多可能的条目（参见 named.conf (5))中,下面将只讨论目录、转发器、转发、版本和拨号。

笔记

named.conf文件中只能有一个选项语句。

名称DC ONF 中值得注意的关键字

directory指定名称守护程序的工作目录。一个常见的值是 /var/named。此外,区域文件没有
在此目录中查找目录部分。

最近的发行版将配置目录与工作目录分开。例如,在最近的 Debian Linux 发行版中,工作目录指定为 /var/cache/bind,但所有配置文件都可以在 /etc/bind 中找到。所有区域文件也可以在后一个目录中找到,并且必须用它们的目录部分指定,如上面的 named.conf 示例所示。

forwarders forwarders 语句包含一个或多个要查询的名称服务器的IP 地址。这些IP广告如何
使用的礼服由下面描述的前向声明指定。

默认是没有转发器。解析是通过全球（或公司本地)DNS 系统完成的。

通常指定的名称服务器与服务提供商使用的相同。

版权所有 2013-2017 Snow BV

Machine Translated by Google

forward仅当指定了转发器时转发才有效。

可以指定两个值:forward first; （默认)和仅转发; 。使用 forward first,查询首先发送到指定的名称服务器 IP 地址,如果失败,它应该在其他地方执行查找。对于仅转发,查询仅限于指定的名称服务器 IP 地址。

转发器和转发器的示例:

```
选项 { // ...

        转发器{
                123.12.134.2;
                123.12.134.3;
        }

        仅转发;

        // ...
};
```

在此示例中,bind 被告知仅查询名称服务器 123.12.134.2 和 123.12.134.3。

version可以从正在运行的名称服务器查询版本:

```
$ dig @ns12.zoneedit.com version.bind 混沌 txt

; <<>> DiG 9.8.3-P1 <<>> @ns12.zoneedit.com version.bind chaos txt ; （找到 1 个服务器）;;全局选项:+cmd ;;得到答案: ;;
->>HEADER<<- 操作码:QUERY,状态:NOERROR,id:59790 ;;标志:qr aa rd;查询:1,答案:1,权限:0,附加:0 ;;警告:已请求递归但不可用









;;问题部分: ;version.bind。
                                                中文文本文件

;;答案部分:
版本.绑定。 0 通道 TXT “8.4.X”

;;查询时间:169 毫秒; ;服务器:
209.62.64.46#53(209.62.64.46)
;;时间:2013 年 6 月 25 日星期二 11:38:48
;; MSG SIZE 收件人:60
```

请注意,BIND 版本显示在输出中。因为一些 BIND 版本有已知的漏洞,所以 BIND 版本有时会被隐藏起来。版本说明:

```
版本“未透露”;
```

或者

```
无版本;
```

options 语句内部导致不显示版本查询的响应。

拨号当名称服务器位于防火墙后,通过拨号连接与外界相连时,通常由名称服务器完成的一些维护可能是不需要的。不需要的操作示例包括:发送心跳数据包、使用防火墙另一端的名称服务器进行区域传输。

以下示例也在选项部分内停止外部区域维护:

```
心跳间隔 0;拨号是; // 注意:这实际上*停止
*拨号!
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

选项块中可以放置更多选项。有关详细信息,请参阅手册页。

根据所使用的发行版,可能会使用一个单独的 bind.conf.options 文件,其中包含 BIND 配置的所有选项。主配置文件 named.conf 必须包括这个单独的文件,这可以通过将以下行添加到 named.conf 来完成:

```
包括“/etc/bind/named.conf.options”;
```

其他单独的配置文件,如 named.conf.log 或 named.conf.default-zones 也可以这种方式嵌套。

记录语句_

BIND（版本 8 和 9)日志系统过于复杂,无法在此详细讨论。两者之间的一个重要区别与解析日志配置有关。 BIND 8 用于解析日志语句并立即启动日志配置。 BIND 9 仅在解析了整个配置文件后才建立日志记录配置。启动时,服务器将有关配置文件中语法错误的所有日志消息发送到默认通道。如果在启动期间提供了 -g 选项,这些错误可能会重定向到标准错误输出。

类别和通道之间的区别是日志记录的重要部分。

通道是输出规范。例如,空通道会忽略发送到通道的任何输出。

类别是一种数据类型。安全类别是众多类别之一。要将类型（类别)安全的消息记录到 default_syslog 通道,请使用以下命令:

```
记录 { 类别安全
        { default_syslog; }; // ...

};
```

要关闭某些类型数据的日志记录,请将其发送到空通道,如前面所示的示例 named.conf 中所做的那样:

```
日志记录 { 类别
                lame-servers { null; };类别 cname { null; };

};
```

这意味着类型为 lame-servers 和 cname 的消息将被丢弃。

日志记录有合理的默认值。这意味着没有日志语句的 named.conf 是可能的。

注意

named.conf文件中最多允许一个日志记录语句。

预定义区域语句

在 named.conf 中定义的区域可以在相应区域文件中使用“@”符号引用。 “@”称为当前原点。例如,

```
区域“127.in-addr.arpa”{
                类型大师;文件“/etc/
                bind/db.127”;
};
```

将导致 127.in-addr.arpa 的当前来源,在文件 /etc/bind/db.127 中作为“@”可用。

关于区域文件的详细信息,以及如何创建您自己的区域文件和语句将在第7.2 节中介绍。

版权所有 2013-2017 Snow BV

Machine Translated by Google

命名的名称服务器守护进程

命名名称服务器守护程序是与其他名称服务器通信以解析名称的程序。它接受查询,查看其缓存并在还没有答案时查询其他名称服务器。一旦它在自己的缓存或数据库中找到答案或从另一个名称服务器收到答案,它就会将答案发送回首先发送查询的名称服务器。

表7.2列出了控制命名名称服务器的方法。

方法	参见
rndc程序	第7.1.6节第7.1.8
发送信号	节第7.1.9节
使用启动/停止脚本	

表 7.2:控制命名

rndc程序_

rndc（远程名称守护进程控制)程序可用于在本地和远程控制命名名称服务器守护进程。它需要一个包含密钥的 /etc/rndc.key 文件。

```
密钥“rndc-key”{ 算法 hmac-
        md5;秘密
         “tyZqsLtPHCNna5SFBLT0Eg==”；
};

选项{默认密钥
         “rndc-key”;默认服务器 127.0.0.1;默认端
        口 953;

};
```

名称服务器配置文件 /etc/named.conf 需要包含相同的密钥以允许主机控制名称服务器。该文件的相关部分如下所示。

```
密钥“rndc-key”{ 算法 hmac-
        md5;秘密
         “tyZqsLtPHCNna5SFBLT0Eg==”；
};

控制{
        inet 127.0.0.1 端口 953
                允许 { 127.0.0.1; }键{“rndc键”； };
};
```

秘密本身永远不会通过网络传输。两端使用算法关键字后声明的算法计算哈希值并比较哈希值。

根据使用的发行版,配置文件也可能存储在 /etc/bind/*. rndc.key 文件应该归 root:bind 所有,模式为 640。主绑定配置文件 named.conf 应该有一行包含密钥文件。运行程序 rndc-confgen 将创建一个密钥,以防系统上没有可用密钥。

对名称服务器的命令可以作为参数提供给 rndc,例如:rndc reload。这将请求名称服务器重新加载其配置和区域文件。以这种方式指定的所有命令都被名称守护程序所理解。 help 命令提供名称服务器可以理解的命令列表。

虽然这里没有讨论,但 rndc 可用于远程管理多个名称服务器。有关 rndc 和 BIND 的更多信息,请参阅手册页和“BIND 9 管理员参考手册”。

版权所有 2013-2017 Snow BV

Machine Translated by Google

named-checkconf实用程序

named-checkconf 是一个非常有用的实用程序,可以检查 named.conf 文件是否有错误。如果 named.conf 文件位于您的发行版的常规 /etc/named.conf 位置,您只需键入命令来检查该文件。

```
# 命名检查配置
```

然而,named.conf 文件的位置可能不同（取决于您使用的发行版）。例如,在某些情况下,该文件位于 /etc/bind/named.conf。 named-checkconf 实用程序不会自动识别 /etc/named.conf 以外的位置,在这些情况下,您必须在命令后包含路径和文件名。

```
# named-checkconf /etc/bind/named.conf
```

当提示返回但没有给出任何消息时,这意味着 named-checkconf 没有发现任何错误。下面的示例将显示当它发现错误时会发生什么。在这种情况下,我犯了一个错误,忘记在包含语句中添加字母 i。

```
[root@localhost etc]# named-checkconf /etc/named.conf:56: 未知选
项 nclude
```

named-checkconf 实用程序将只检查 named.conf 文件。从命名的内部调用的其他配置文件。例如,不会自动检查使用 include 语句的 conf 文件。可以通过在执行 named.checkconf 实用程序时添加它们的路径和文件名来手动检查它们。

向named发送信号

可以向指定进程发送信号以控制其行为。可以在指定的联机帮助页中找到完整的信号列表。一个例子是 SIGHUP 信号,它导致 named 重新加载 named.conf 和数据库文件。

使用 kill 命令将信号发送到 named,例如,

```
杀死-HUP 217
```

这会将 SIGHUP 信号发送到进程 ID 为 217 的命名进程,从而触发重新加载。

使用启动/停止脚本控制命名

大多数发行版都会附带一个启动/停止脚本,允许您手动启动、停止或控制命名,例如,Debian 中的 /etc/init.d/bind 或 Red Hat 中的 /etc/init.d/named。

笔记

Red Hat（基于)系统具有可以替代使用的服务命令。服务使用同一组参数,因此您可以说：

```
# 名为重新加载的服务
```

表7.3列出了当前版本的 /etc/init.d/bind 接受的参数。

DNSmasq

dnsmasq 既是轻量级 DNS 转发器又是 DHCP 服务器。 dnsmasq 支持静态和动态 DHCP 租约,支持 BOOTP/TFTP/PXE 网络启动协议。

```
$ apt-cache search dnsmasq dnsmasq - 小型缓存
DNS 代理和 DHCP/TFTP 服务器 dnsmasq-base - 小型缓存 DNS 代理和 DHCP/TFTP 服务器 dnsmasq-utils
- 用于操作 DHCP 租约的实用程序
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

范围	描述 starts
开始	named stops
停止	named stops
重新开始	and restarts named
重新加载	reloads 配置
强制重新加载	和重启一样

表 7.3：/etc/init.d/bind 参数

djbdns

djbdns - Daniel J. Bernstein DNS - 由于对重复的 BIND 安全漏洞感到沮丧而构建。除了拥有 DNS 缓存之外,DNS 服务器和 DNS 客户端 djbdns 还包括几个 DNS 调试工具。源代码于 2007 年发布到公共领域。有几个分支,其中之一是 dbndns,Debian 项目的分支。

动力域名系统

PowerDNS 是一家荷兰 DNS 软件和服务供应商。 PowerDNS 软件是开源软件 (GPL),并与许多发行版打包在一起,如 pdns、powerdns-server 或 pdns-server。该系统允许多个后端允许访问 DNS 配置数据,包括接受 BIND 样式文件的简单后端。

```
$ apt-cache search pdns pdns-backend-
geo - PowerDNS 的地理后端 pdns-backend-ldap - PowerDNS 的 LDAP 后端
pdns-backend-lua - PowerDNS 的 lua 后端 pdns-backend-mysql - PowerDNS 的
通用 MySQL 后端 pdns-backend -pgsql - PowerDNS 的通用 PostgreSQL 后端 pdns-
backend-pipe - PowerDNS 的管道/协处理后端 pdns-backend-sqlite - PowerDNS 的 sqlite 后端 pdns-
backend-sqlite3 - PowerDNS 的 sqlite 后端 pdns-server - 极其强大和通用的名称服务器 pdns -server-dbg -
PowerDNS 的调试符号 pdns-recursor - PowerDNS 递归 pdns-recursor-dbg - PowerDNS 递归的调试符号 pdnsd
- 代理 DNS 服务器
```

挖掘和主机实用程序

Internet 系统联盟 (ICS) 已弃用 nslookup,取而代之的是 host 和 dig。然而,由于较旧的 Unix 版本的寿命较长,nslookup 仍被广泛使用。它仍然是大多数 Linux 发行版的一部分。

dig 和 host 命令都可以用来查询名称服务器,在什么情况下使用哪个是偏好问题。 dig 有更多的选项,默认情况下提供更详细的输出。这两个命令的帮助应该给出一些差异的见解：

```
$主机
用法:host [-aCdlriTwv] [-c class] [-N ndots] [-t type] [-W time]
                    [-R number] [-m flag] hostname [server] -a 等同于-v -t ANY -c 为非 IN
          数据指定查询类 -C 比较权威名称服务器上的 SOA 记录 -d 等同于 -v - l 列出域中的所
          有主机,使用 AXFR -i IP6.INT 反向查找 -N 更改完成根查找之前允许的点数 -r 禁用递
          归处理 -R 指定 UDP 数据包的重试次数
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
-sa SERVFAIL 响应应该停止查询 -t 指定查询类型

-T 启用 TCP/IP 模式 -v 启用详细输出 -w 指定
永远等待回复

-W 指定等待回复的时间 -4 仅使用 IPv4 查询传输 -6 仅使用 IPv6 查询传输 -m 设置内
存调试标志（trace|record|usage）
```

```
$ dig -h 用法：
dig [@global-server] [domain] [q-type] [q-class] {q-opt} {global-d-opt} host [@local-server] {local-d- opt} [主机[@local-
                    server] {local-d-opt} [...]

其中:域                    在域名系统中 q-class 是 (in,hs,ch,...) [默认值:in] q-type
          是 (a,any,mx,ns,soa,hinfo,axfr,txt) 之一,...) [默认值:a]

                    （使用 ixfr=version 类型 ixfr)是以下之一：
          q-选择
                    -x 点符号                    （反向查找的快捷方式）（使用 IP6.INT 进行 IPv6 反向查找）
                    -一世                        （批处理模式）（绑定到源地址/端口）（指定端口号）（指定查询名称）（指定查
                    -f 文件名                    询类型）（指定查询类）（指定tsig 密钥文件）
                    -b 地址[#端口] -p 端口

                    -q 名称 -t 类
                    型 -c 类别

                    -k keyfile -y
                    [hmac:]name:key（指定命名的 base64 tsig 密钥）-4（仅使用 IPv4 查询传输）（仅使用 IPv6 查询传输）
                    （启用内存使用调试）
                    -6
                    -m
          d-opt     形式为 +keyword[=value],其中 keyword 为:+[no]vc
                                                 (TCP模式)
                    +[no]tcp                     （TCP 模式,备用语法）
                    +time=###                    （设置查询超时)[5]
                    +tries=###                   （设置 UDP 尝试次数)[3]
                    +重试=### +域=###            （设置UDP重试次数)[2]
                                                 （设置默认域名）
                    +缓冲区大小=###              （设置 EDNS0 最大 UDP 数据包大小）
                    +ndots=###                   （设置 NDOTS 值）
                    +edns=###                    (设置EDNS版本)
                    +[no]search +                (设置是否使用searchlist)
                    [no]showsearch +             （搜索中间结果）
                    [no]defname +[no]recurse     （同上）
                    +[no]ignore +[no]fail +      （递归模式）
                    [no]besteffort +[no]aaonly   （不要为 TC 响应恢复到 TCP。）
                    +[no]adflag +[no]cdflag +    （不要在 SERVFAIL 上尝试下一个服务器）
                    [无]cl                       （尝试解析甚至是非法消息）
                                                 （在查询中设置 AA 标志 (+[no]aaflag)）
                                                 （在查询中设置 AD 标志）
                                                 （在查询中设置 CD 标志）
                                                 （控制类在记录中的显示）
                    +[无]命令                    （控制命令行的显示）
                    +[无]评论                    （控制注释行的显示）
                    +[无]问题 +[无]答案 +[无]     （控制问题的显示）
                    权限 +[无]附加               （控制答案显示）
                                                 （控制权限显示）
                                                 （控制显示附加）
                    +[无]统计                    （控制统计显示）
                    +[无]短                      （禁用除简短答案以外的所有内容）
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
                    +[no]ttlid +[no]all          （控制记录中ttls的显示）
                    +[no]qr +                    （设置或清除所有显示标志）
                    [no]nssearch +               （发送前打印问题）
                    [no]identify +[no]trace      （搜索所有权威名称服务器）
                                                 （简短回答中的 ID 响应者）
                                                 （从根向下跟踪委托）
                    +[无]dnssec                  （请求 DNSSEC 记录）
                    +[无]nsid                    （请求名称服务器 ID）
                    +[no]sigchase +trusted-      （追逐 DNSSEC 签名）
                    key=####（追逐 DNSSEC 信号时的可信密钥)+[no]topdown
                                                 （执行 DNSSEC 验证自上而下模式)+[no]multiline
                                                 （以扩展格式打印记录）
                    +[没有]onesoa                （AXFR只打印一条soa记录）
          全局 d-opts 和服务器（在主机名之前)影响所有查询。本地 d-opts 和服务器（在主机名之后)仅影响该查找。 （打印帮助并退出）-h
          （打印版本并退出）

          -v
```

正如演示的那样,dig 命令提供了更广泛的选项。虽然没有选项,但提供的信息非常相似：

```
$ 主机 zonetransfer.me zonetransfer.me
的地址为 217.147.180.162
zonetransfer.me 邮件由 20 ASPMX2.GOOGLEMAIL.COM 处理。 zonetransfer.me 邮件由 20 ASPMX3.GOOGLEMAIL.COM
处理。 zonetransfer.me 邮件由 20 ASPMX4.GOOGLEMAIL.COM 处理。 zonetransfer.me 邮件由 20
ASPMX5.GOOGLEMAIL.COM 处理。 zonetransfer.me 邮件由 0 ASPMX.L.GOOGLE.COM 处理。 zonetransfer.me 邮件
由 10 ALT1.ASPMX.L.GOOGLE.COM 处理。 zonetransfer.me 邮件由 10 ALT2.ASPMX.L.GOOGLE.COM 处理。
```

```
$ 挖 zonetransfer.me

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> zonetransfer.me ;;全局选项:+cmd ;;得到答案： ;; ->>HEADER<<-
操作码:QUERY,状态:NOERROR,id:31395 ;;标志:qr rd ra;问题:1,答案:1,权限:2,附加:1




;;问题部分： ;zonetransfer.me。在一个


;;答案部分:zonetransfer.me。
7193 在 A 217.147.180.162

;;权限部分:zonetransfer.me。 7193 IN
NS ns12.zoneedit.com。
zonetransfer.me. 7193 IN NS ns16.zoneedit.com。

;;附加部分:ns12.zoneedit.com。 1077 在
A 209.62.64.46

;;查询时间:6 毫秒； ;服务器：
213.154.248.156#53(213.154.248.156)
;;时间:2013 年 6 月 27 日星期四 07:30:36 ;; MSG SIZE 收件人：
115
```

$人挖

```
姓名
                dig - DNS 查找实用程序
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
概要
        dig [@server] [-b 地址] [-c 类] [-f 文件名] [-k 文件名] [-m] [-p 端口号]
            [-q name] [-t type] [-x addr] [-y [hmac:]name:key] [-4] [-6] [name] [type] [class] [queryopt...]
```

一个重要的选项是 -t,例如仅查询 MX 或 NS 记录。此选项适用于挖掘和主机。 （查看这两个命令的手册页以了解其他选项的说明。）

```
$挖雪.nl

; <<>> DiG 9.8.3-P1 <<>> snow.nl ;;全局选项:+cmd ;;得到答案：;;
->>HEADER<<- 操作码:QUERY,状态:NOERROR,id:60605 ;;标志:qr
rd ra;查询:1,答案:1,权限:0,附加:0



;;问题部分：;snow.nl。
                    在一个

;;答案部分:snow.nl。
                299 在 A 213.154.248.202

;;查询时间:52 毫秒；;服务器:8.8.8.8#53(8.8.8.8)

;;时间:2015 年 10 月 20 日星期二 09:55:13 ;; MSG SIZE 收件人：
41
```

```
$ dig -t NS 雪.nl

; <<>> DiG 9.8.3-P1 <<>> -t NS snow.nl ;;全局选项:+cmd ;;得到答案：;;
->>HEADER<<- 操作码:QUERY,状态:NOERROR,id:26099 ;;标志:qr rd ra;查询:1，
答案:3,权限:0,附加:0



;;问题部分：
;snow.nl。          在 NS

;;答案部分:snow.nl。
                20361 IN NS ns1.transip.nl。
snow.nl.        20361 IN NS ns2.transip.eu。
snow.nl.        20361 IN NS ns0.transip.net。

;;查询时间:59 毫秒；;服务器:8.8.8.8#53(8.8.8.8)

;;时间:2015 年 10 月 20 日星期二 10:00:24 ;; MSG SIZE 收件人：
108
```

```
$ dig -t MX 雪.nl

; <<>> DiG 9.8.3-P1 <<>> -t MX snow.nl ;;全局选项:+cmd ;;得到答案：;;
->>HEADER<<- 操作码:QUERY,状态:NOERROR,id:36671 ;;标志:qr rd ra;查询:1，
答案:2,权限:0,附加:0



;;问题部分：
;snow.nl。          在墨西哥

;;答案部分：
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
snow.nl.            299 IN MX 10 mc.snow.nl。
snow.nl.            299 IN MX 20 mx1.snow.nl。

;;查询时间:61 毫秒；;服务器：
8.8.8.8#53(8.8.8.8)
;;时间:2015 年 10 月 20 日星期二 10:02:01 ;; MSG SIZE 收件人：
64
```

创建和维护 DNS 区域 (207.2)

资源和进一步阅读： Albitz01、 DNShowto、 BINDfaq。

考生应该能够为正向或反向区域创建区域文件，并为根级服务器创建提示。此目标包括为记录设置适当的值、在区域中添加主机以及将区域添加到 DNS。候选人还应该能够将区域委托给另一个 DNS 服务器。

关键知识领域

- BIND 9 配置文件、术语和实用程序
- 从 DNS 服务器请求信息的实用程序
- BIND 区域文件的布局、内容和文件位置
- 在区域文件中添加新主机的多种方法，包括反向区域

条款和实用程序

- /var/命名/*
- 区域文件语法
- 资源记录格式
- 命名检查区
- 命名编译区
- 主文件格式
- 挖掘
- nslookup
- 主持人

区域和反向区域

有区域和反向区域。每个 named.conf 都将包含两者的定义。

区域的示例有 localhost（内部使用)和 example.com（现实中不一定存在的外部示例）。反向区域的示例是 127.in-addr.arpa（内部使用)和 240.123.224.in-addr.arpa（真实示例）。

区域与反向区域的关系如下所示。

版权所有 2013-2017 Snow BV

Machine Translated by Google

db.local文件_

在大多数情况下,将预定义一个特殊域 localhost。

这是从本章前面显示的示例 named.conf 调用的相应区域文件 /etc/bind/db.local。

```
;
;本地环回接口的 BIND 数据文件

; $TTL          604800
@IN SOA 本地主机。根目录。本地主机。 ( ; 串行
                                  1个
                604800            ;刷新
                 86400            ;重试
               2419200            ;到期
                604800)           ;负缓存 TTL
;
@IN NS                     本地主机。
@在一个                    127.0.0.1
```

@ 包含区域的名称。它被称为当前原点。 named.conf 中的 zone 语句定义了当前原点,正如我们之前看到的 named.conf 文件的这一部分所示:

```
区域“本地主机”{
                类型大师;文件“/etc/
                bind/db.local”;
};
```

所以在这种情况下,该区域称为 localhost,区域文件中的所有当前来源都将成为 localhost。

区域文件的其他部分将在下面的第7.2.3.4节中解释。

db.127文件_

对于区域文件中的每个 IP 范围,都有一个在反向区域文件中描述的相应反向区域。这是文件 /etc/bind/db.127:

```
;
;本地环回接口的 BIND 反向数据文件

; $TTL          604800
@IN SOA 本地主机。根目录。本地主机。 ( ; 序列号 ; 刷新
                                  1个
                604800
                 86400            ;重试
               2419200            ;到期
                604800)           ;负缓存 TTL
;
@              在 NS               本地主机。
1.0.0 在 PTR                       本地主机。
```

这是来自 named.conf 的调用部分:

```
区域“127.in-addr.arpa”{
                类型大师;文件“/etc/
                bind/db.127”;
};
```

可以看出,当前原点将是 127.in-addr.arpa,因此反向区域文件中的所有 @ 都将替换为 127.in-addr.arpa。

但还有更多:所有不以点结尾的主机名都会附加当前来源。这很重要,所以我重复:

版权所有 2013-2017 Snow BV

Machine Translated by Google

所有不以点结尾的主机名都会附加当前来源。

例如:1.0.0 将变为 1.0.0.127.in-addr.arpa。

普通 IP 地址（例如 127.0.0.1)不会附加当前来源。

同样,有关反向区域文件的详细信息将在下面讨论。

提示文件

localhost 和 127.in-addr.arpa 区域供系统内部使用。

在外部世界中,区域是按等级组织的。根区域（用点：. 表示)列在一个特殊的提示文件中。
named.conf 中的以下区域语句读取名为 /etc/bind/db.root 的根区域文件

```
区域“。”{
        输入提示;文件“/
        etc/bind/db.root”；
};
```

对了,注意type:hint!它是根区的一种特殊类型。此文件中没有存储任何其他内容,也不会动态更新。

这是 db.root 文件的一部分。

```
;以前的 NS.INTERNIC.NET
;
.                                   3600000 印度洋          A.ROOT-SERVERS.NET。
A.ROOT-SERVERS.NET。                360万          一个      198.41.0.4

; ;以前的 NS1.ISI.EDU
;
.                                   360万          NS        B.ROOT-SERVERS.NET。
B.ROOT-SERVERS.NET。                360万          一个      128.9.0.107
```

注意左边的点,这是根区域!

您的发行版或您个人必须保持根区文件最新。您可以查看 ftp.rs.internic.net 以获得新版本。或者,你可以运行类似

```
挖掘@a.root-servers.net。 ns > roothints
```

这将创建一个新文件。

你也可以运行

```
挖掘@a.root-servers.net。面向服务架构
```

您可以定期执行此操作以查看 SOA 版本号（见下文)是否已更改。

区域文件

根区域了解其直接下的所有顶级域,例如 edu 和 org 域以及特定国家/地区的域,如 uk 和 nl。

如果 example.org 域（此处用于说明目的)是真实域,则根名称服务器不会处理它。相反,org 域的名称服务器会知道所有相关信息。这称为委托:根名称服务器已将 org 下区域的权限委托给 org 区域的名称服务器。稍后将在第7.2.5.1 节中解释自己进行委托。

区域文件从 named.conf 文件中读取,区域语句如下

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
区域“example.org”在{
        类型大师;文件“/etc/bind/
        exampleorg.zone”;
};
```

这是区域 example.org 的示例区域文件。

```
$TTL 86400 @ IN
SOA lion.example.org dns.master.lion.example.org。（2001110700 ; Ser: yyyymmhhee (ee: ser/day start 00)

                                28800          ;刷新
                                 3600          ;重试
                               604800          ;过期
                                86400)         ;负缓存
                在 NS                  lion.example.org。猫.example.org。
                在 NS

                在 MX 0 lion.example.org。
                在 MX 10 cat.example.org 中。

狮子在一个                             224.123.240.1
                在 MX 0 lion.example.org。
                在 MX 10 cat.example.org 中。

小狗在一只猫                           224.123.240.2
                在一个                 224.123.240.3

;我们的笔记本电脑鸟在 A
                                       224.123.240.4
```

让我们详细检查一下这个文件。

$ TTL语句

$TTL 是该区域的默认生存时间。当名称服务器请求有关此区域的信息时,它还会获得 TTL。 TTL过期后,应该更新缓存中的数据。

```
$TTL 3小时
```

这会将默认 TTL 设置为 3 小时,并且

```
$TTL 86400
```

这一到 86400 秒（与 1d（1 天)相同）。

注意

自 BIND 版本 8.2 起,每个区域文件都应以默认 TTL 开头。如果省略 TTL,则会替换默认值并生成警告。此更改在 RFC2308 中定义。

同时,最后一个 SOA 时间字段的含义发生了变化。现在是负缓存值,下面会解释。

SOA资源记录

首字母缩略词 SOA 的意思是 Start Of Authority。告诉外界这个域名服务器是查询这个域名的权威域名服务器。 SOA 记录应包含管理员联系地址以及从名称服务器的超时设置。声明一个 SOA 记录有两个方面的作用:首先,父区域组织已将 example.org 域的使用委托（授予)给我们,其次是我们声称是该区域的权限。

版权所有 2013-2017 Snow BV

Machine Translated by Google

SOA 记录对于每个 DNS 区域文件都是强制性的,并且也应该是区域文件的第一个指定资源记录 (RR)。

早些时候我们看到了以下 SOA 记录:

```
@ 在 SOA lion.example.org 中。 dnsmaster.lion.example.org。 (
        2001110700          ; Ser: yyyymmhhee (ee: ser/day start 00)
             28800          ;刷新
              3600          ;重试
            604800          ;过期
              3600)         ;负缓存
```

这些 SOA 系列的元素是

@当前来源,它扩展到 example.org（请参阅 named.conf 文件）。

在Internet 数据类中。来自 rfc4343:“如 [STD13] 和 [RFC2929] 中所述,DNS 有一个附加的数据轴
位置称为 CLASS。目前唯一在全球使用的 CLASS 是“IN”（互联网)CLASS。

SOA授权开始 - 这就是本节的内容。

lion.example.org。拥有该域的主控（参见第7.2.4 节)名称服务器的机器的名称。

dnsmaster.lion.example.org。遇到麻烦时要邮寄的人的电子邮件地址,电子邮件地址中的商业符号 (@) 通常用点代替。揭开其原因留给读者作为练习（具有当前起源的东西.....？）

（五个数字） · 第一个数字是序列号。对于永不更改的区域定义（例如,localhost 区域）
一个1就够了。

然而,对于确实发生变化的区域,另一种格式相当常见:yyyymmddee。也就是说,年份的 4 位数字,月份的两位数字,日期的两位数字,加上以 00 开头的两位数字,每次更改时都会递增。例如序列号

```
2001110701
```

对应于 2001 年 11 月 7 日的第二个（！)更改。第二天,第一个更改将获得序列号 2001110800。

注意每

次区域定义发生变化时,序列号都必须增加（至少增加一个）。如果序列号没有增长,则不会注意到区域中的更改。

- 第二个数字是刷新率。这是从属服务器（见下文)应该检查的频率
数据已更改。

  rfc1537 中的建议值:24h

- 第三个数字是重试值。这是从属服务器在刷新或失败后重试之前必须等待的时间,
或重试之间。

  rfc1537 中的建议值:2h

- 第四个数字是过期值。如果从服务器无法联系主服务器刷新信息,
区域信息过期,从服务器将停止为该区域提供数据。

  rfc1537 中的建议值:30d

- 第五个数字是负缓存值TTL。负缓存意味着 DNS 服务器记住它无法解析特定域。数字是该内存保留的时间。

  合理值:1h（3600s）

版权所有 2013-2017 Snow BV

Machine Translated by Google

A资源记录

A记录是地址记录。它将 IP 地址连接到主机名。一个示例记录是

```
狮子在一个                    224.123.240.1
```

这会将名称 lion.example.org（请记住,当前源 example.org 添加到任何不以点结尾的名称)到 IP 地址 224.123.240.1。

> 注意
> 每个A记录都应该有一个对应的PTR记录。这在第7.2.3.5 节中有描述。

A 记录由 IP 协议的当前版本 IPv4 使用。下一代协议 IPv6 具有 A6 记录类型。此处不讨论 IPv6。

CNAME资源记录

CNAME（规范名称)记录为具有 A 记录的主机指定另一个名称。 BIND 8 过去通过接受一个名为 multiple-cnames 的选项来允许多个 CNAME 记录。不过,从 BIND 9.2 开始,CNAME 规则严格遵守 DNS 标准。 A 和 CNAME 组合记录的示例是

```
猫在一个                    224.123.240.3
www 在 CNAME cat.example.org 中。
```

这使得 www.example.org 指向 cat.example.org。

NS资源记录

指定区域的名称服务器。例如

```
@       在 SOA 中.....

        在 NS           lion.example.org。猫.example.org。
        在 NS
```

首先引起注意的是 NS 行中的 IN 标记之前没有任何内容。在这种情况下,先前指定的当前原点（此处使用 SOA)仍然有效作为当前原点。

> 注意
> 每个域应该至少有两个独立的名称服务器。独立意味着连接到单独的网络、单独的电源线等。请参阅下面的第7.2.4 节。

MX资源记录

MX（邮件交换器)记录系统由邮件传输系统使用,例如,sendmail 和 postfix 守护进程。
可以为每个域提供多个 MX 记录。 MX 标签后面的数字是优先级。优先级 0 是最高优先级。数字越大,优先级越低。优先级 0 将用于邮件发往的主机。如果该主机宕机,另一台优先级较低（因此编号较高)的主机将临时存储邮件。

示例条目:

```
狮子在一个                    224.123.240.1
            在 MX 0 lion.example.org。
            在 MX 10 cat.example.org 中。
```

因此,此示例指定 lion.example.org 的邮件首先发送到 lion.example.org。如果那个主机挂了,猫。改为使用 example.org。

要在两台主机之间平均分配邮件,请给它们相同的优先级。也就是说,如果添加了另一个优先级为 10 的主机,那么当 lion.example.org 关闭时,它们都会收到一份邮件。

版权所有 2013-2017 Snow BV

Machine Translated by Google

MX 域

邮件可以传送到主机,如上一节所示。但是邮件传输代理 (MTA) 和 DNS 可以配置为主机接受域的邮件。

考虑我们的示例,主机 lion.example.org 可以配置为接受来自 example.org 的邮件。

要实现这一点,请将 example.org 的 MX 记录放在 example.org 区域文件中,例如:

```
;也接受域 example.org 的邮件。在 MX 0 lion.example.org。 example.org。
在 MX 10 cat.example.org 中。
```

如果为此配置了 MTA,则发送到 example.org 的邮件将仅被 lion.example.org 主机上的 MTA 接受。

反向区域文件

每个 IP 范围都有一个反向区域,它由反向顺序的部分 IP 编号和 in-addr.arpa 组成。该系统除其他外用于检查主机名是否属于特定地址。

我们的 example.org 域使用 IP 范围 224.123.240.x。相应的反向区域称为 240.123.224。输入地址.arpa。在 named.conf 中,这可能是相应的条目:

```
区域    240.123.224.in-addr.arpa    IN {
          类型大师;文件“/etc/bind/
          exampleorg.rev”;
};
```

一个示例 /etc/bind/exampleorg.rev（对应于我们之前看到的 example.org 域)是:

```
$TTL 86400 @ IN
SOA lion.example.org.  dns.master.lion.example.org。 ( 2001110700 ; Ser: yyyymmhhee (ee: ser/day start 00)

                                28800          ;刷新
                                  3600         ;重试
                              604800           ;过期
                                 3600)         ;负缓存
                在 NS                lion.example.org。猫.example.org。
                在 NS

1个             在 PTR 中             lion.example.org。
2个             在 PTR 中             doggy.example.org。猫.example.org。
3个             在 PTR 中             bird.example.org。
4个             在 PTR 中
```

当前来源是 240.123.224.in-addr.arpa,所以 bird 的入口实际上是

```
4.240.123.224.in-addr.arpa 在 PTR bird.example.org 中。
```

PTR记录_

PTR 记录将反向名称 (4.240.123.224.in-addr.arpa) 连接到 A 记录给出的名称 (bird.example.org)。

版权所有 2013-2017 Snow BV

Machine Translated by Google

IPv6记录

IPv6地址格式

IPv6 地址是 128 位地址,而不是 IPv4 的 32 位地址。它们被标记为 8 组 16 位值,每部分以 4 个十六进制数写入,并用冒号分隔。为了便于阅读,可以省略每个部分中的前导零,并且可以完全省略仅由零组成的部分。后者可能每个地址只执行一次,因为多次出现会产生歧义表示。

例如:2001:0db8:0000:0000:0000:ff00:0042:8329可以改写为2001:db8::ff00:42:8329。

本地主机（环回)地址 0:0:0:0:0:0:0:1 可以减少为 ::1

AAAA记录

A 记录用于 IPv4,而 AAAA 记录用于 IPv6。它以与 A 记录解析 IPv4 相同的方式将主机名解析为 IPv6 地址。

```
AAAA 中的狮子                    2001:db8::ff00:42:8329
```

笔记

注意:用于解析 IPv6 地址记录的另一种格式是 A6 记录。虽然当前的 BIND 版本支持它,但它被 RFC6563 认为是“历史性的”

PTR记录

对于反向解析,IPv6 地址以另一种格式表示,每个十六进制数字由点分隔。我们上面的例子变成:2.0.0.1.0.db8.0.0.0.0.0.0.0.0.0.0.0.0.ff0.0.0.0.4.2.8.3.2.9

IPv6 PTR 记录是该地址以完全相反的顺序附加了域 ip6.arpa。上面地址变成:9.2.3.8.2.4.0.0.0.0.ff0.0.0.0.0.0.0.0.0.0.0.0.8.bd0.1.0.0.2.ip6.arpa

该域的一部分可能会被委托,就像 IPv4 地址一样。在 named.conf 中,这可能是相应的条目:

```
zone    0.0.0.0.0.0.0.0.8.bd0.1.0.0.2.ip6.arpa    IN { type master;文件“/etc/bind/exampleorg-ip6.rev”;



};
```

相应的文件应类似于以下文件:

```
$TTL 86400 @
IN SOA lion.example.org。  dnsmaster.lion.example.org。 ( 2001110700 ; Ser: yyyymmhhee (ee: ser/day
                    start 00)
                        28800          ;刷新
                         3600          ;重试
                       604800          ;过期
                        86400)         ;负缓存
            在 NS                lion.example.org。
            在 NS                猫.example.org。

9.2.3.8.2.4.0.0.0.0.ff0.0.0.0                    在 PTR 中          lion.example.org。
```

由于 IPv4 和 IPv6 PTR 记录位于不同的区域（即 in-addr.arpa 和 ip6.arpa）,它们应该在单独的区域文件中定义。

版权所有 2013-2017 Snow BV

Machine Translated by Google

主从服务器

每个区域（机器本地区域除外,例如 localhost)必须至少有一个主名称服务器。它可以由一个或多个从名称服务器支持。

每个区域应该有两个独立的名称服务器。独立意味着连接到不同的网络和其他电源。如果一个发电厂或网络出现故障,来自该区域的解析名称必须仍然是可能的。一个很好的解决方案是在不同位置使用一台主服务器和一台从服务器。

主名称服务器和从属名称服务器都是该区域的权威（如果该区域被正确委派,请参阅第7.2.5.1 节）。也就是说,他们都给出了关于区域的相同答案。

区域的数据源自该区域的主名称服务器。从名称服务器从主服务器复制数据。
其他名称服务器可以联系主服务器或从服务器来解析名称。

这意味着配置必须处理

- 主服务器上的从名称服务器访问控制
- 主从名称服务器上的其他名称服务器访问控制

配置主站

区域定义被定义为主通过使用

```
类型大师;
```

区域定义内的语句。例如,参见定义 example.org master 的区域语句（在 named.conf 中）。

```
区域 example.org IN { type master;
文件“/etc/bind/exampleorg.zone”； };
```

当然,exampleorg.zone 文件必须如前所述创建。

同一区域可以有多个独立的主名称服务器。

notify 语句控制 master 是否在 master 区域更改时发送 DNS NOTIFY 消息。 notify 语句可以放在 zone 语句中,也可以放在 options 语句中。如果区域和选项语句中都存在一个,则前者优先。当一个slave接收到这样的NOTIFY请求（并支持它）,它会立即向master发起一个zone transfer来更新zone数据。默认是 notify yes;。要关闭它（例如,当一个区域仅由 masters 提供服务时)设置 notify no;。

笔记
旧版本的 BIND 使用术语 primary 而不是 master。

配置从站

区域定义被定义为从属通过使用

```
类型奴隶;
```

区域定义内的语句,附有主服务器的 IP 地址。例如,这里是 zone 语句（在 named.conf 中）,它定义了一个 example.org 从站：

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
区域    example.org    IN { type slave;大师
        { 224.123.240.1; }; // 狮子文件
          db.example.org   ;

};
```

文件 db.example.org 是由从名称服务器自己创建的。从站没有 example.org 的数据。相反,从服务器从主名称服务器接收数据并将其存储在指定的文件中。

请注意,文件名没有目录部分。因此它将被写入 named.conf 中目录选项给出的 BIND 工作目录中。例如，/var/cache/bind 或 /var/named。名称服务器必须对该文件具有写入权限。

注意旧

版本的 BIND 使用术语辅助而不是从属。

存根名称服务器

存根区类似于从区,除了它只复制主区的 NS 记录而不是整个区。换句话说,托管存根区域的 DNS 服务器只是权威名称服务器的信息来源。此服务器必须具有对远程 DNS 服务器的网络访问权限,以便复制权威名称服务器信息。

存根区域的目的有两个：

- 使授权区域信息保持最新。通过定期更新其中一个子区域的存根区域,托管存根区域的 DNS 服务器将为该子区域维护一份最新的权威 DNS 服务器列表。
- 改进名称解析。存根区域使 DNS 服务器可以使用存根区域的列表执行名称解析的名称服务器,而不必使用转发或根提示。

创建子域

有两种创建子域的方法:在区域内或作为委托区域。

第一种是将子域放入普通区域文件中。子域不会有自己的 SOA 和 NS 记录。不推荐使用此方法 - 它更难维护并且可能会产生管理问题,例如签署区域。

另一种方法是将子域委托给一个单独的区域。这将在下一节中描述。

委派 DNS 区域

通过将子域配置为独立区域（拥有自己的 SOA 和 NS 记录)并从父域中委派该域,可以创建一个真正的、独立的子域。

仅当父区域已将其权限委派给区域时,区域才具有权威性。例如,example.org 域由 org 域委托。

同样,example.org 域可以将权限委托给独立的 scripts.example.org 域。后者将独立于前者并拥有自己的 SOA 和 NS 记录。

让我们看一下 scripts.example.org 区域的示例文件：

```
$ORIGIN scripts.example.org。
ctl IN A 224.123.240.16
        在 MX 0 控制
        在 MX 10 lion.example.org 中。
www IN CNAME ctl
perl IN A 224.123.240.17
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
                在 MX 0 中
                在 MX 10 ctl
                在 MX 20 lion.example.org 中。
bash 在 A 224.123.240.18
                在 MX 0 狂欢中
                在 MX 10 ctl
                在 MX 20 lion.example.org 中。
sh 在 CNAME 中 bash
```

没什么新的,只是一个完整的区域文件。

但是,要使其具有权威性,父域（在本例中为 example.org)必须将其权限委托给脚本。 example.org 区域。这是在 example.org 区域文件中委托的方式：

```
NS ctl.scripts.example.org 中的 2d 脚本。
                NS bash.scripts.example.org 中的 2d。
ctl.scripts.example.org。 2d IN A 224.123.240.16 bash.scripts.example.org。 2d 在 A
224.123.240.18
```

就这样！

脚本的 NS 记录进行实际委托。 A 记录必须存在,否则无法定位脚本的名称服务器。

注意建

议插入一个 TTL 字段（如示例中的2d ）。

检查区域文件的语法错误

创建或更改区域文件后,最好检查它们是否存在语法错误。这可以通过 named-checkzone 命令来完成。执行此操作的方法是在命令后输入域名,然后输入区域文件的名称。请参阅下面的示例。

```
# named-checkzone test.com /var/named/test.com.zone 区域 test.com/IN: loaded serial 0

好的
```

在此示例中,我没有在域名末尾放置一个点,但您可以选择这样做。两种方式都是正确的,并且会给出相同的答案。

正如您在上面的示例中看到的,该实用程序给出“OK”作为响应,这意味着该文件没有任何语法错误。您可能还想考虑查看 named-checkzone 的手册页以了解其他选项。

从 BIND 软件的 9.9 版本开始,辅助服务器区域文件默认以原始二进制格式而不是文本格式保存。因此,阅读和检查这些文件可能会更具挑战性。幸运的是,有一个名为 compilezone 的工具。该工具与 named-checkzone 工具非常相似,但可以将原始区域文件更改为文本格式（反之亦然） ,以便普通人可读。

可以更改以更改此行为并将默认格式更改回文本格式。这可以通过将 masterfile-format 选项添加到您管理的辅助名称服务器上的 named.conf 文件中的区域语句来完成。
下面的示例显示了正确的语法。

```
主文件格式的文本；
```

DNS 实用程序

四个 DNS 实用程序可以帮助解析名称,甚至可以调试 DNS 区域。它们被称为 dig、host、nslookup 和 dnswalk。
前三个是 BIND 源代码分发的一部分。

版权所有 2013-2017 Snow BV

Machine Translated by Google

挖掘计划

dig 命令允许您以接近区域文件设置的方式解析名称。例如,你可以做

```
挖 bird.example.org A
```

这将查找 bird.example.org 的 A 记录。部分输出如下所示:

```
;;答案部分:bird.example.org。 1D
在 A 224.123.240.4

;;权威部分:example.org。
                                        1D IN NS lion.example.org。

;;附加部分:
lion.example.org。 1D 在 A 224.123.240.1
```

如果 dig 显示的是 SOA 记录而不是请求的 A 记录,那么域是可以的,但请求的主机不存在。

甚至可以查询另一个名称服务器而不是 /etc/resolv.conf 中指定的名称服务器:

```
挖@cat.example.org bird.example.org A
```

这将查询名称服务器 cat.example.org 以获取主机 bird 的 A 记录。为查询联系的名称服务器将列在输出的尾部。

dig 命令可用于测试或调试您的反向区域定义。例如,

```
挖 4.240.123.224.in-addr.arpa PTR
```

将测试狮子主机的反向入口。你应该期待这样的事情:

```
;;答案部分:4.240.123.224.in-
addr.arpa。 PTR lion.example.org 中的 1D。

;;授权部分:240.123.224.in-addr.arpa。
1D IN NS lion.example.org。

;;附加部分:
lion.example.org。                    一维输入                    224.123.240.4
```

如果你得到类似的东西

```
;;答案部分:4.240.123.224.in-
addr.arpa。 1D IN PTR lion.example.org.240.123.224.in-addr.arpa。
```

你在你的区域文件中犯了一个错误。给定 240.123.224.in-addr.arpa. 的当前来源,考虑以下行:

```
4 在 PTR lion.example.org 中;错误的!
```

末尾的点被省略,因此自动附加当前原点。要更正此问题,请添加尾随点:

```
4 在 PTR lion.example.org 中。 ;正确的!
```

笔记

当指定像bird.example.org或4.240.123.224.in-addr.arpa这样的主机名来挖掘时,可能会添加一个尾随点。

版权所有 2013-2017 Snow BV

Machine Translated by Google

主机程序

主机程序以简单的格式报告解析器信息。

当指定主机名作为参数时,将显示相应的 A 记录。例如:

```
主机 bird.example.org
```

将导致输出像

```
bird.example.org 网站 一个 224.123.240.4
```

当需要主机名并给出 IP 地址时,主机程序特别有用。例如:

```
主机 224.123.240.4
```

从我们的示例主机显示输出如下:

```
名称:bird.example.org 地址:224.123.240.4
```

也可以使用以下命令:

```
主机 4.240.123.224.in-addr.arpa
```

解析 PTR 记录:

```
4.240.123.224.in-addr.arpa PTR bird.example.org
```

笔记

与dig 的情况一样,当指定像bird.example.org或4.240.123.224.in-addr.arpa这样的主机名时,可能会添加一个尾随点。

---

nslookup 程序

nslookup 程序是另一种解析名称的方法。如前所述,不推荐使用 nslookup,而应该使用命令 host 和 dig。尽管有此建议,您应该对其用法有一些了解。

例如,通过输入 nslookup 并键入以下内容来启动交互模式:

```
ls -d example.org。
```

结果,输出将以类似区域文件的格式显示(开始显示):

```
[lion.example.org]
$ORIGIN example.org。 @1D IN SOA
lion postmaster.lion (2001110800 ;serial, refresh, retry, expiry 1D);最低限度
                                                    8小时
                                                    1小时
                                                    1D IN NS 狮子
                                                    1D IN MX 0 狮子
```

方括号中的第一行包含发送应答的名称服务器的名称。

版权所有 2013-2017 Snow BV

Machine Translated by Google

注意所

示示例需要从连接的名称服务器进行区域传输。如果此名称服务器拒绝区域传输（将在下一节中讨论），您当然不会看到此输出。

可以在交互模式下向 nslookup 提供很多命令：help 命令将显示一个列表。

DNS漫游

DNSwalk 是一个 DNS 调试器。请谨慎使用，因为它会在检查 DNS 数据库的一致性和准确性的同时尝试执行区域传输。例子：

```
$ dnswalk zoneedit.com。
检查 zoneedit.com。
获取 zoneedit.com 的区域转移。来自 ns2.zoneedit.com...完成。
SOA=ns2.zoneedit.com contact=soacontact.zoneedit.com
警告：zoneedit.com A 64.85.73.107：无 PTR 记录
警告：golf.zoneedit.com A 69.72.176.186：无 PTR 记录 警告：zoneapp1.zoneedit.com A 64.85.73.104：无
PTR 记录 警告：dynamic1.zoneedit.com A 64.85.73.40：无 PTR 记录 警告：zoneapp2.zoneedit .com A 64.85.73.107：
无 PTR 记录 WARN：ezzi.zoneedit.com A 207.41.71.242：无 PTR 记录


警告：dynamic2.zoneedit.com A 64.85.73.40：无 PTR 记录 警告：legacyddns.zoneedit.com A 64.85.73.40：无
PTR 记录 警告：api2.zoneedit.com A 64.85.73.104：无 PTR 记录 警告：wfb.zoneedit .com A 69.72.142.98：无PTR记录


警告：new.zoneedit.com A 64.85.73.107：无 PTR 记录
警告：zebra.zoneedit.com A 69.72.240.114：无 PTR 记录
警告：api.zoneedit.com A 64.85.73.40：无 PTR 记录 警告：www.zoneedit.com A 64.85.73.107：无 PTR
记录
警告：newapi.zoneedit.com A 64.85.73.104：无 PTR 记录 0 次失败，15 次警告，0 次错误。
```

保护 DNS 服务器 (207.3)

资源和进一步阅读： Albitz01、 BINDfaq、 DNShowto、 Lindgreen01、 ChrootBind9 Nijssen99、选择正确的 DNS 配置、 DANE 和 DNSSEC 教程。 LPIC2sybex2nd，让我们加密 TLSA。 Wikipedia.org 上的 DANE。 Wikipedia.org 上的 DigiNotar。 RFC6698。

考生应该能够将 DNS 服务器配置为以非 root 用户身份运行并在 chroot 监狱中运行。此目标包括在 DNS 服务器之间安全地交换数据。

关键知识领域

- BIND 9 配置文件
- 配置 BIND 在 chroot jail 中运行
- 使用 forwarders 语句拆分 BIND 配置
- 配置和使用事务签名 (TSIG)
- 了解 DNSSEC 和基本工具
- 了解 DANE 和相关记录

版权所有 2013-2017 Snow BV

Machine Translated by Google

条款和实用程序

- /etc/named.conf
- /etc/密码
- DNSSEC
- dnssec 密钥生成器
- dnssec-signzone

DNS 安全策略

有几种策略可以使 DNS 更安全。

当在 BIND 名称服务器中发现安全漏洞时,它通常会在几个小时内得到修复。这会产生一个新的 BIND 版本。因此,旧版本的 BIND 可能存在与安全相关的错误。 BIND 用户应经常访问BIND 源站点 (http://www.isc.org/downloads/bind/)以检查新的更新。您可以考虑订阅安全公告列表;大多数 Linux 发行版都有一个。例如,Debian debian-security-announce。要订阅该列表,请访问https://lists.debian.org。

让信息更难获取

与所有其他程序一样,BIND 名称服务器具有可能被滥用的功能和安全漏洞。犯罪黑客（黑客)很清楚这一点,并跟踪版本及其安全相关问题。明智的做法是不要公布您的漏洞,因此您可能希望在 BIND 中禁用版本号报告。

此外,批量名称服务器数据可用于披露有关您的基础架构的详细信息。因此,可以设置限制以防止用户获取大量名称服务器数据或至少在短时间内获取这些数据。这样做的另一个原因是名称服务器旨在在短时间内回答大量短查询。不得不意外地提供大量数据可能会中断服务。

重要的是要记住,隐藏信息是通过默默无闻来保证安全。这样做,获取信息的简单方法已被阻止,但可能还有其他获取信息的方法。通过默默无闻的安全性使获取信息变得更加困难,但并非不可能。然而,它确实引入了更长的时间范围,在该时间范围内您可以检测到入侵企图并采取相应措施,例如通过在防火墙中阻止请求者的 IP 地址。

隐藏版本号

命令

```
挖掘@target chaos version.bind txt
```

将在主机目标上显示 BIND 名称服务器的版本。

可以通过在 named.conf 的选项语句中输入版本语句来隐藏 BIND 版本。在以下示例中,版本将设置为隐藏的字符串。

```
选项 { // ...

        // 隐藏绑定版本 version    hidden   ;

};
```

CERT 文章(Nijssen99)展示了一种限制对绑定区域访问的方法。这是替换 BIND 版本的替代方法。

版权所有 2013-2017 Snow BV

Machine Translated by Google

限制访问

有几种方法可以限制对名称服务器数据的访问。首先,必须定义访问控制列表。这是通过 acl 语句完成的。

```
acl“受信任”{ 本地主机;
        192.168.1.0/24;

};
```

这个 acl 定义了一个标签为 trusted 的访问控制列表。 ACL 标签用于简化管理:您只需要在一个地方更新您的 ACL,之后每个对标签的引用都会自动指向更新后的数据。

名称服务器支持解析器的查询和其他名称服务器的区域传输。

正常查询 当解析器只需要非常有限的数据子集上的数据时,就会发生正常查询。例如,它想知道哪个 IP 地址与主机名相关联。 allow-query 语句控制从哪些主机接受这些查询。

区域传输 Internet 名称空间是建立在域层次结构的基础上的。要查找主机的 IP 地址,您需要其完全限定的域名 (FQDN)。它由一系列由点分隔的子域组成,例如:www.example.com。第一部分是主机名,即“www”。接下来是“示例”域,它又是“com”域的细分。单个域上的数据称为“区域”。必须允许从（备份)名称服务器从同一区域的主服务器获取信息。重要的是将这些“区域传输”限制到仅需要数据的服务器 - 不再需要。

区域传输由 allow-transfer 语句控制。该语句也可以在 zone 语句中指定,从而覆盖全局选项 allow-transfer 语句。传统上,区域传输可以由任何人发起。

allow-query 语句也可以在区域语句或选项语句中使用。它可以包含 acl 标签（如受信任）、无,或者一个或多个 IP 地址或 IP 范围。尽可能使用标签:以这种方式跟踪权限要容易得多,并且它使您不易受到可能导致安全漏洞的意外疏忽的影响。

限制区转移

dig 和 host 都可以发起区域传输。默认情况下,主从服务器都允许发送所有区域信息。
使用 dig 命令的示例:

```
$ dig axfr @ns12.zoneedit.com zonetransfer.me

; <<> DiG 9.8.3-P1 <<>> axfr @ns12.zoneedit.com zonetransfer.me ; （找到 1 个服务器）;;全局选项:+cmd zonetransfer.me。
7200 在 SOA ns16.zoneedit.com。 soacontact.zoneedit.com。 \


    2013064418 2400 360 1209600 300
zonetransfer.me. 7200 IN NS ns16.zoneedit.com。
zonetransfer.me. 7200 IN NS ns12.zoneedit.com。
zonetransfer.me. 7200 在一个 217.147.180.162
zonetransfer.me. 7200 IN MX 0 ASPMX.L.GOOGLE.COM。
zonetransfer.me. 7200 IN MX 10 ALT1.ASPMX.L.GOOGLE.COM。
zonetransfer.me. 7200 IN MX 10 ALT2.ASPMX.L.GOOGLE.COM。
zonetransfer.me. 7200 IN MX 20 ASPMX2.GOOGLEMAIL.COM。
zonetransfer.me. 7200 IN MX 20 ASPMX3.GOOGLEMAIL.COM。
zonetransfer.me. 7200 IN MX 20 ASPMX4.GOOGLEMAIL.COM。
zonetransfer.me. 7200 IN MX 20 ASPMX5.GOOGLEMAIL.COM。
zonetransfer.me. 301 IN TXT“记得给 Pippa 打电话或发邮件 +44 123 4567890 \
    或 pippa@zonetransfer.me 在进行 DNS 更改时“zonetransfer.me.301 IN TXT”google-site-
verification=←  tyP28J7JAUHA9fw2sHXMgcCC0I6XBmmoVi04VlMewxA“
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
testing.zonetransfer.me. 301 CNAME www.zonetransfer.me。 164.180.147.217.in-addr.arpa.zonetransfer.me. 7200
IN PTR www.zonetransfer.me。 ipv6actnow.org.zonetransfer.me。 7200 在 AAAA 2001:67c:2e8:11::c100:1332 asfdbauthdns.zonetransfer.me。 7900 IN
AFSDB 1 asfdbbox.zonetransfer.me。

office.zonetransfer.me. 7200 在 4.23.39.254
owa.zonetransfer.me. 7200 在 A 207.46.197.32
info.zonetransfer.me. 7200 IN TXT     ZoneTransfer.me 服务由 Robin 提供\
      伍德 - robin@digininja.org。有关更多信息,请参见 www.digininja.org/projects/zonetransferme.php ←。
                信息。”
asfdbbox.zonetransfer.me. 7200 在 127.0.0.1
canberra_office.zonetransfer.me. 7200 在 A 202.14.81.230
asfdbvolume.zonetransfer.me。 7800 IN AFSDB 1 asfdbbox.zonetransfer.me。
电子邮件.zonetransfer.me。 2222 IN NAPTR 1 1            E2U+email            email.zoneedit.com。 ←
          zonetransfer.me.
dzc.zonetransfer.me. 7200 IN TXT “AbCdEfG”
zonetransfer.me 博士300 英寸 LOC 53 20 56.558 N 1 38 33.526 W 0.00m 1m 10000m 10m
rp.zonetransfer.me. 321 IN RP robin.zonetransfer.me.zonetransfer.me.罗宾伍德。 ←
          zonetransfer.me.
sip.zonetransfer.me. 3333 IN NAPTR 2 3     au        E2U+sip        !^.*$!sip:customer- ←。
          service@zonetransfer.me！”。
alltcpportsopen.firewall.test.zonetransfer.me。 301 IN A 127.0.0.1 www.zonetransfer.me。 7200 在一个 217.147.180.162

staging.zonetransfer.me. 7200 CNAME www.sydneyoperahouse.com。 deadbeef.zonetransfer.me. 7201 在 AAAA 死了 :beaf::

robinwood.zonetransfer.me. 302 IN TXT “罗宾·伍德”
vpn.zonetransfer.me. 4000 IN A 174.36.59.154 _sip._tcp.zonetransfer.me。 14000 IN SRV 0
0 5060 www.zonetransfer.me。 dc_office.zonetransfer.me. 7200 IN A 143.228.181.132 zonetransfer.me。 7200 在 SOA
ns16.zoneedit.com。 soacontact.zoneedit.com。 \

      2013064418 2400 360 1209600 300
;;查询时间:334 毫秒； ;服务器：
209.62.64.46#53(209.62.64.46)
;;时间:2013 年 6 月 28 日星期五 12:01:16 ;; XFR 大小:37 条记录（消
息 37,字节 2673）
```

前面的示例显示了有关 zonetransfer.me 的完整信息。命令 host -l 的作用相同,但其输出格式不同：

```
$ host -l zonetransfer.me ns16.zoneedit.com 使用域服务器： 名称:ns16.zoneedit.com



地址:69.64.68.41#53
别名：

zonetransfer.me 名称服务器 ns16.zoneedit.com。
zonetransfer.me 名称服务器 ns12.zoneedit.com。
zonetransfer.me 的地址为 217.147.180.162
164.180.147.217.in-addr.arpa.zonetransfer.me 域名指针 www.zonetransfer.me。 ipv6actnow.org.zonetransfer.me 的 IPv6 地址为 2001:67c:2e8:11::c100:1332
office.zonetransfer.me 的地址为 4.23.39.254

owa.zonetransfer.me 的地址为 207.46.197.32
asfdbbox.zonetransfer.me 有地址 127.0.0.1
canberra_office.zonetransfer.me 的地址为 202.14.81.230 alltcpportsopen.firewall.test.zonetransfer.me 的地址
为 127.0.0.1
www.zonetransfer.me 的地址是 217.147.180.162
deadbeef.zonetransfer.me 有 IPv6 地址 dead:beaf::
vpn.zonetransfer.me 的地址为 174.36.59.154 dc_office.zonetransfer.me 的地址为
143.228.181.132
```

只应允许经过验证的从属名称服务器发出区域传输请求。这可以通过调整特定区域的主服务器和所有从服务器的配置文件 (named.conf) 来完成。例如：

版权所有 2013-2017 Snow BV

Machine Translated by Google

在主名称服务器上 在主名称服务器上,您应该使用 allow-transfer 语句来限制区域传输到已知从属服务器列表。

```
acl    my_slave_servers    { 224.123.240.3; //
        猫.example.org
};

// ...

区域“example.org”在{
        类型大师;  // ....


        允许转移 { my_slave_servers;

        };
};
```

现在只有从服务器（示例中只有主机 cat)可以请求来自该主名称服务器的区域传输。

在从属名称服务器上 在从属名称服务器上,您永远不应允许任何区域传输。这可以通过将 allow-transfer 子句设置为“无”来实现：

```
区域    example.org    IN { type slave; // ....



        允许转移{
                没有任何;
        };
};
```

注意
重要提示:不要忘记保护反向区域!

限制查询

虽然严格来说任何客户端都应该被允许查询名称服务器,但实际上您可能希望将查询限制在一定范围内的主机,例如您自己网络中的主机。这可以按如下方式完成：

```
acl“我的主机”
        { 224.123.240.0/24;
};

// ...

区域    example.org    IN { // ...


        允许查询{ myhosts;

        };
};
```

这将查询限制为 IP 地址以 224.123.240 开头的主机。

版权所有 2013-2017 Snow BV

Machine Translated by Google

控制请求

可以使用更多的安全控制。

欺骗是指恶意名称服务器使用虚假数据回答客户端请求。由于出于性能原因对名称服务器数据进行了大量缓存,因此缓存数据可能会被伪造数据感染。欺骗是很难实现的,但明智的做法是对其采取预防措施。许多较旧的欺骗方法已被软件中的补丁消除,但可以配置 BIND 以进一步增强欺骗保护。

限制入侵的影响

即使您准备好在 DNS 修复出现时立即实施,也可能为时已晚 - 即使只有几个小时:您的系统无论如何都可能受到损害。因此,建议采取预防措施将影响降至最低。

以较少的权限运行 BIND

在某些发行版中,BIND 以 root 身份运行。如果 BIND 受到威胁,攻击者可能拥有 root 访问权限。这可以通过在非特权用户和组下运行 BIND 来防止。

使用用户 nobody 和组 nogroup 可能很诱人。但是,由于许多服务已经这样做了,因此出现了另一个安全问题:这些服务可能能够以一种或另一种方式进行通信。

最佳实践是创建一个特殊用户,例如 named 和一个相应的组（也可以称为 named）,并在该用户/组组合下运行名称服务器。

使用 -u 和 -g 选项运行 named as named/named：

```
命名-u 命名-g 命名
```

在 Debian 系统上，/etc/init.d/bind 中的起始行是：

```
启动停止守护进程 ... --exec /usr/sbin/named -- -u named -g named
```

（为清楚起见,其他选项已用点代替）。额外的 -- 告诉 start-stop-daemon 它的选项在哪里结束,所有其他选项都将传递给指定的程序。

在 Red Hat（或兼容)系统上,可以在 /etc/rc.d/init.d/named 中找到启动文件。在 RHEL 系统上,该服务已配置为在用户“named”下运行。您可以在 /etc/sysconfig/named 中设置其他选项,例如,您可以在其中添加 -u named。要在 Red Hat（兼容)系统上启动服务,请使用服务命令：

```
# 名为启动的服务
生成 /etc/rndc.key:开始命名：                                    [ 好的 ]
                                                                 [ 好的 ]
```

要使 named 在 Red Hat（兼容)系统上以正确的运行级别启动,请运行 chkconfig named on。

> 注意
> 确保名称服务器在named.conf中的目录选项指定的目录中具有写入权限。

在 chroot 监狱中运行 BIND

另一种防止名称服务器进程受损的方法是将该进程置于 chroot 监狱中。在启动时,进程将设置一个目录（例如/var/cache/bind)作为它的根目录。由于没有进程能够访问其设置的根以上的任何内容,因此这会阻止指定进程访问 /var/cache/bind 之外的任何内容。

版权所有 2013-2017 Snow BV

Machine Translated by Google

准备一个 chroot 监狱

BIND 需要的所有文件都必须复制到新的根目录。完整的细节可以在这里找到： ChrootBind9。

简而言之：

- 您需要将/etc、/dev、/lib、/sbin 或/usr/sbin 和/var/run 目录复制到chroot。
- 您可能需要新根目录下的/dev/null 设备。下一个命令将创建它：

```
mknod -m 666 /var/cache/bind/dev/null c 1 3
```

- 您将需要一个密码和一个位于新 /etc 中的组文件：

```
cd /var/cache/bind/etc cp -p /etc/
{passwd,group} 。
```

这将复制文字密码和组文件。另一种方法是创建特殊的 passwd 和组文件，如第7.3.6.3 节所示。

您可能还需要复制 ld.so.cache 和本地时间：

```
cd /var/缓存/绑定/等
cp /etc/ld.so.cache 。 cp /etc/本地时
间。
```

- BIND 配置也应该放在新的根目录下，因为当您重新加载服务器时可能需要它。鉴于您的新根设置为 /var/cache/bind，您应该将文件复制到 /var/cache/bind/etc/bind。

  请注意，新的 named.conf 中的所有目录（由 directory 选项指定)都是相对于新的根目录的，因此您不需要（或不应该)在其中包含前缀 /var/cache/bind。

- 新的lib 目录必须至少包含BIND 使用的所有共享库。您可以使用 ldd 命令列出这些。
  例如，假设命令：

```
ldd /usr/sbin/命名
```

产量

```
libc.so.6 => /lib/libc.so.6 (0x40015000) /lib/ld-linux.so.2 => /lib/ld-
linux.so.2 (0x40000000)
```

在（真实的）/lib 目录中，libc.so.6 是指向 libc-2.1.3.so 的符号链接，而 ld-linux.so.2 是指向 ld-2 的符号链接。 1.3.所以。符号链接和它们指向的文件都必须复制到新根目录下的 lib 目录：

```
cd /var/缓存/绑定/lib
cp -pd /lib/{libc.so.6,libc-2.1.3.so,ld-linux.so.2,ld-2.1.3.so} 。
```

- named 和named-xfer 程序都必须存在于新根下。 rndc 程序可能会有用
  也。例如：

```
cp -p /usr/sbin/named{,-xfer} /var/cache/bind/usr/sbin cp -p /usr/sbin/rndc /var/cache/bind/usr/
sbin
```

运行 BIND chroot

要在 chroot 监狱中启动 BIND 名称服务器，只需在命令行、启动脚本或 sysconfig 文件中添加 -t 选项后跟目录名称。

版权所有 2013-2017 Snow BV

Machine Translated by Google

注意-
t选项实际上运行chroot命令来启动名称服务器。阅读有关chroot命令的手册页以了解更多信息。

在某些系统中,您可能需要将选项添加到启动文件中。例如:

```
启动停止守护进程 ... --exec /usr/sbin/named -- -t /var/cache/bind
```

或者

```
守护进程 ... /sbin/named -t /var/cache/bind
```

重要BIND
名称服务器在命令行参数解析后立即切换到 chroot,因此在读取配置文件之前。配置文件中的所有路径都应该相对于新的根目录。

chrooted BIND 的配置

作为 chroot 监狱的结果,所有配置文件（例如区域文件和 named.conf)必须存在于 chroot 目录下的目录（例如 /etc/bind)中。

登录到 syslog 也不起作用,因为 chrooted 进程无法访问位于监狱外的 Unix 套接字（无法访问 /dev/log）。有多种解决方法,但最简单的是在 chrooted 环境中使用替代日志文件。
请参阅您可以使用的 named.conf 文件的片段:

```
logging
        { channel some_log { 文件
                "bind.log"版本 3;严重性信息;

        };

        类别默认{ some_log; };

        // ...
};
```

这会将日志记录信息写入文件 /var/cache/bind/var/cache/bind/bind.log。

区域文件（如果有的话)也应该放在 chroot 环境中。例如,如果 named.conf 包含这样一个主区域的定义:

```
区域"example.com"在{
        类型大师;文件"/etc/
        bind/example.com.zone";
};
```

然后 example.com.zone 文件需要位于 /var/cache/bind/etc/bind 中,再次假设您的 chroot 目录是 /var/cache/bind。

结合特殊用户和 chroot

通过结合使用 chroot 环境和在特殊的非特权用户和组下运行名称服务器,可以获得更多的技术安全性。请注意,这些用户必须在 chrooted 环境中创建的 /etc/passwd 和 /etc/group 副本中可用。最后,设置对 chrooted 区域和配置文件的权限,使它们对指定用户和组只读。

版权所有 2013-2017 Snow BV

Machine Translated by Google

笔记

检查 BIND 的源代码可以揭示事物的顺序：

- 解析命令行选项
- chrooting
- 成为后台进程并写入pid 文件
- 更改为新用户和组

pid 文件仍然由root用户写入，之后将使用特权较低的“命名”用户。

chroot 环境中的 /etc/passwd、/etc/shadow 和 /etc/group 文件可能与系统其余部分使用的文件不同。名称服务器的特殊用户和组甚至可能不存在于监狱之外。

保护名称服务器连接

可以使用 named.conf 中的 allow-transfer 和 allow-query 语句来限制对受信任主机的访问，这可能有助于限制风险。由于限制是基于 IP 地址的，因此仍然存在恶棍使用欺骗技术的残余风险。这仍然可能导致您的系统被滥用。

签署数据可以帮助防止这种情况。当服务器和客户端使用共享密钥对它们之间的数据进行签名时，我们可以非常确定双方都是他们所说的人。

为了进一步保护名称服务器连接，可以使用一组称为 DNSSEC（域名系统安全扩展)的协议作为额外的安全层。

密码学速成班

要了解 DNSSEC 的工作原理，您需要了解消息身份验证代码、哈希和非对称加密的概念。

您可能知道对称加密的概念。使用密钥和某种算法，将明文消息加扰成难以理解的内容。要恢复消息，您需要相同的密钥。这可以很好地保护消息免受窥探，但无法证明消息是由给定方发送的。任何一方都可以写一条消息，用私钥加密，然后说是另一方写了这条消息。你无法判断这是不是真的。

非对称加密的相关概念还允许对消息进行加密和解密。但不是一个共享密钥，而是两个独立但相关的密钥：一个用于加密消息，另一个用于解密消息。在生成这样一个密钥对之后，所有者只将这些密钥中的一个分发给另一方或多方，即公钥。密钥对的另一半保密（私钥）。任何有权访问公钥的人都可以使用它来加密消息。相应私钥的所有者将是唯一可以破译它的人。因此，发件人可以确定只有他一个人可以阅读该消息。此外，他可以发送加密消息，使用私钥对其进行加密。每个有权访问公钥的人都可以破译它，并且可以确定发送消息的确实是相应私钥的所有者。

私钥、公钥或对称密钥均可用于计算 MAC（消息验证码）。计算 MAC 的算法使用两个输入：密钥和消息。输出是固定长度的十六进制数字字符串。使用密钥（或密钥对的一半)计算 MAC，随后发送 MAC 和消息。接收方还将使用该消息与共享密钥或密钥对的另一半计算 MAC。如果两个哈希都匹配，则可以非常确定消息没有被更改并且它来自发件人。请注意，当您使用 MAC 时，消息本身不会被加密，任何人都可以阅读它。

DNSSEC 使用 MAC 对其消息进行签名。它使用只有 DNSSEC 知道的私钥。相应的公钥发布在下一级区域的名称服务器上。它可以通过查询父域名服务器（“信任锚”)来获得。因此，例如 sidn.nl 域的公钥将在“nl”名称服务器上可用。公钥本身将使用私钥进行签名，私钥的公钥将在上一层可用。所以对于域 abc

版权所有 2013-2017 Snow BV

Machine Translated by Google

“a”的公钥可以在“b”的区域服务器上找到。公钥用私钥签名,私钥的公钥发布在“c”的名称服务器上（你可能想再读一遍那句话;-)）。

最终,验证链在根服务器处结束。要获得用于签署顶级域的密钥,可以使用挖掘,但当然,查询也可能被欺骗。因此,严肃的验证解析器通过其他方式获取密钥。

例如,通过从签名的软件包中检索它们,类似于浏览器中的内置 SSL 根证书。

使用 dnssec-keygen 命令

根据 lpic2 目标,您必须能够使用 BIND 9 的 dnssec-keygen。

笔记

dnssec-keygen命令是 DNS 的 DNSSEC 安全扩展的一部分（在 Albitz01 中有完整描述）。
低于 9.2 的 BIND 版本未完全实现 DNSSEC。
13 个权威根服务器自 2010 年 5 月 6 日起使用 DNSSEC。

---

dnssec-keygen 可用于生成公钥/私钥和对称密钥。请参阅CrashCryptoCourse。密钥对旨在用于验证区域数据。对称密钥用于请求/交易签名。

生成密钥所需的参数：

一种算法（-a选项)RSA | RSAMD5 |卫生署 |动态服务代理 | RSASHA1 | HMAC-MD5

密钥大小（-b选项)取决于所选算法:RSAMD5:[512..4096]

RSASHA1:[512..4096]
卫生署:[128..4096]
DSA:[512..1024] 和 64 的倍数 HMAC-MD5：
[1..512]

名称类型（-n选项)主机 |实体 |用户 |其他

密钥文件的格式

dnssec-keygen 创建两个文件:Kname+ algorithm+ footprint.private 和 Kname+ algorithm+ footprint.key。

例如,假设我们输入了这个命令：

```
$ dnssec-keygen -a DSA -b 768 -n ZONE example.com。
```

完成后会有两个文件,名字类似Kexample.com.+003+58649.key和Kexample。 com.+003+58649.私人。

私钥文件 Kkey.example.com.+003+58649.private 的内容类似于：

```
私钥格式:v1.2 算法:3 (DSA)

质数（p）:ww4 次质数 ..
（q）:qHwn ..
基础（g）:ncWqWFJ ...
Private_value(x): YuO ...
公共值（y）:JKvGZ ...
```

请注意,为了更好的可读性而被截断的行以省略号结尾。

Kkey.example.com.+003+58649.key 的内容类似于：

```
example.com。在 DNSKEY 256 3 3 BKh8J+a ...
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

请注意,生成 HMAC-MD5 也会生成两个文件,但两者都将包含完全相同的“密钥”:

```
$ dnssec-keygen -a HMAC-MD5 -b 512 -n HOST peek.a.boo Kpeek.a.boo.+157+39506


$ ls Kpeek.a.boo.+157+39506.* Kpeek.a.boo.
+157+39506.key Kpeek.a.boo.+157+39506.private

$ cat Kpeek.a.boo.+157+39506.key peek.a.boo.输入键 512
3 157 HUcmGO7VZ ...

$ cat Kpeek.a.boo.+157+39506.private 私钥格式:v1.2 算法:157
(HMAC_MD5)

键:HUcmGO7VZ ...
```

使用钥匙

您可以通过在 named.conf 中添加密钥语句来引用密钥,然后复制并粘贴您在该部分中生成的密钥:

```
密钥 key.example.com。 {
        算法“hmac-md5”;秘密
         “5HUcmGO7VZ ...”;
};
```

任何有权访问 named.conf 文件的人都可以读取(并可能滥用)该密钥。确保除了名称服务器用户之外,任何人都无法读取它。

您还可以将 key 语句放在一个单独的文件中,并使用 include 语句包含该文件。在这种情况下,同样的警告适用于权限。

为了允许客户端使用密钥连接到我们,我们需要在我们的配置中添加一个服务器子句。假设我们在 IP 地址为 224.123.400.2 的服务器上,并希望允许访问 IP 地址为 224.123.400.1 的服务器,我们可以使用:

```
服务器 224.123.400.1 {
        键 key.example.com.;
};
```

在另一台服务器 (224.123.400.1) 上,我们需要对服务器 224.123.400.2 执行相同的操作:添加密钥语句并复制密钥,并允许另一台主机连接到我们。

您现在可以配置拥有密钥时允许的访问类型。例如,要将区域传输限制到安装了正确密钥的主机,请使用:

```
区域“example.com”{
        类型大师;文件
         “example.com.zone”;允许传输 { key
        key.example.com.; };
};
```

有关其他用例,请参阅 named.conf(5) 手册页。

dnssec-signzone

与 dnssec-keygen 命令类似,BIND 9 包括 dnssec-signzone 实用程序。顾名思义,dnssec-signzone 可用于对区域进行签名。我们已经看到了如何使用密钥来授权服务器。对区域进行签名有助于客户端验证数据的完整性。

使用 DNSSEC 时,所谓的 RRSIG 记录存储在使用 dnskey 为域的资源记录签名时创建的实际数字签名。

版权所有 2013-2017 Snow BV

Machine Translated by Google

dnssec-signzone 实用程序可用于将 RRSIG 记录添加到区域。

NSEC 记录 (Next SECure) 用于指定一系列不存在的域。发明 NSEC 记录是为了解决一个问题:如果你查询一个安全的名称服务器（一个对查询结果进行签名的服务器)并且你会要求一个不存在的名称,它只会返回 - 什么都没有。由于您不能签署“无”,因此您永远不会知道您得到的答案 - 无 - 是否真的是您应该得到的。因此 NSEC 记录。 NSEC 记录 (Next SECure) 列出了一系列不存在的名称。由于可以签署 NSEC 记录,因此可以再次信任答案。

dnssec-tools.org 域的 NSEC 记录示例:

adonis.zonetransfer.me.	10800 英寸	NSEC helena.zonetransfer.me.

该记录指定 adonis.zonetransfer.me 之间不存在 DNS 记录。和 helena.zonetransfer。 me.,这将排除 brutus.zonetransfer.me。从一个现有的,因此有效的领域。

使用 NSEC 记录解决了一个问题,但引入了下一个问题:它允许通过简单地询问一个随机名称来轻松检索所有区域数据。这是有效的,因为 NSEC 记录假定域名按字母顺序排列。举个例子:如果你请求“a.example.com”,而答案是“没有,但下一个有效记录是 joe.example.com”,你就知道 joe.example.com 存在,之前什么都没有确实如此。因此,您可以尝试 jof.example.com,因为“f”是字母表中的下一个字母等。解决此问题的方法是 NSEC3 记录。 NSEC3 记录实际上也是一个 NSEC 记录,但它不会返回下一个名称,而是下一个名称的哈希值。如果您请求 a.example.com,您将得到一条记录,上面写着“不在这里,这是下一个有效域名的哈希值”。除了蛮力之外,没有已知的方法可以找到与该散列匹配的名称。如果您已经知道下一个有效域名,则可以计算其哈希值并验证答案。如果你不这样做,数据将无法帮助你找出答案。

丹麦人

为某个域设置 DNSSEC 后,可以使用 DNS 身份验证命名实体:DANE。
为了了解 DANE 的优势,我们需要看看 DANE 试图解决的问题。这个问题涉及证书颁发机构,简称 CA。

依赖 CA 的加密解决方案的问题在于实现。通过 HTTPS 访问 SSL/TLS 加密网站时,浏览器软件将很乐意接受任何使用匹配 CN 值并被视为由有效证书颁发机构颁发的证书。 CN 值取决于 DNS,幸运的是我们刚刚设置了 DNSSEC 来对此有一些保证。但是,现代浏览器带有 1000 多个受信任的根证书。网络服务器向浏览器提供的每个证书都将使用这些根证书进行验证。如果提供的证书可以与根证书之一相关联,则科学匹配并且浏览器不会抱怨。这是一个问题。

在一个完美的世界里,这个系统可以提供一些保证。不幸的是,每条链条的强度都取决于它最薄弱的环节。
许多已知的攻击向量存在于 HTTPS 会话的完整性和机密性方面。证书颁发机构虽然属于自己的联盟。正如每个 CA 都需要在 100% 的时间内控制其安全性一样,攻击者只需要一个 CA 不需要的时刻。如果攻击者获得了用于签署证书的密钥之一,则游戏结束 [tm]。这在过去发生过,而且不能保证将来不会再发生。

问题包括浏览器接受任何域的任何有效证书。并且缺乏对哪些证书对哪些域有效的确切控制。有一些解决方法,例如证书透明度和证书固定。但就目前而言,这些仍然是有缺陷的系统的变通办法。毕竟,问题不仅限于您的桌面。当 CA 不再受信任时,对 CA 颁发的证书的信任就会成为问题。这已经发生了。

臭名昭著的“Black Tulip”案例描述了一个名为 DigiNotar 的荷兰 CA 发生的违规行为。攻击者获得了对系统的访问权限并为主要网站生成了伪造的证书。除此之外,这些证书不会被识别为伪造的,因为它们是由受信任的 CA 生成的。证书链可以追溯到 DigiNotar 并且看起来是有效的。尽管 DigiNotar 从未收到为受影响的域生成证书的命令或许可。在发现攻击者颁发了多个 DigiNotar 证书后,无法维持对 DigiNotar 作为 CA 的信任。许多软件供应商已将受影响的 CA 根证书从受信任证书列表中删除。 DigiNotar 在数周内申请破产。

如果这听起来很糟糕,请考虑这次数据泄露的总体影响。由于 DigiNotar 颁发的证书因此被列入黑名单,依赖有效证书的软件无法再正常运行。这产生了现实世界的后果,因为 DigiNotar 还向政府和交易相关客户颁发了证书。郁金香运不出去变黑了,

版权所有 2013-2017 Snow BV

Machine Translated by Google

可以这么说。出于这个原因,一些依赖于在线业务开展核心业务的大公司从多个 CA 获得证书。这样,如果一个 CA 受到威胁并且所有相关证书都必须被吊销,那将不会成为业务的单点故障。对业务有利,但不利于证书的透明度。

这就是 DANE 的用武之地;在依赖于 DNSSEC 提供的保证的同时,为 DNS 记录提供了证书关联信息。这减轻了对某些类型的 SSL/TLS 连接的静态根证书的依赖性。这些记录后来被称为 TLSA 记录。与 HTTPS 一样,DANE 应该被正确或更好地实施,但根本没有。正确实施后,DANE 将提供有关用于加密的证书的额外完整性。通过这样做,会话的机密性方面也得到了提升。

RFC 6698 第 2 节和第 7 节中描述了 TLSA 资源记录语法。从 RFC 6698 中获取的 PKIX CA 证书的 SHA-256 散列关联示例如下所示:

```
_443._tcp.www.example.com。在 TLSA (
        0 0 1 d2abde240d7cd3ee6b4b28c54df034b9
                7983a1d16e8a410e4561cb106618e971)
```

每个 TLSA 资源记录 (RR) 指定以下字段以创建证书关联:证书使用、选择器、匹配类型和证书关联数据。上例中的证书关联数据字段由 SHA-256 哈希字符串值表示。该数据字段的内容取决于前面三个 RR 字段之前的值。这三个字段在上面的示例中由“0 0 1”表示。我们将以相反的顺序解释这些字段。如果您查看散列值,然后从右到左读取值为“1 0 0”,这是有道理的。

上例中第三个字段中的值“1”表示匹配类型字段。该字段的值可以介于“0”和“2”之间。它指定 Certification Association Data 字段内容是否未散列（值 0）、使用 SHA-256 散列（值 1)或使用 SHA-512 散列（值 2）。在上面的示例中,证书关联数据字段的内容表示 SHA-256 哈希字符串。

由“0”表示的第二个字段表示选择器字段。 TLSA 选择器由“0”或“1”表示。 “0”字段值表示证书关联数据字段内容基于完整证书。值“1”表示证书关联数据字段的内容基于证书的公钥。在上面的示例中,选择器字段指示来自证书关联数据字段的 SHA-256 哈希字符串基于完整证书。

上例中由“0”表示的第一个字段表示证书使用字段。该字段可以包含“0”和“3”之间的值。 “0”值 (PKIX-TA) 指定证书关联数据字段值与 X.509 树中的公共证书颁发机构相关。值“1”(PKIX-EE) 指定证书关联数据字段值与您正在连接的端点上的证书相关,使用 X.509 验证。值“2”(DANE-TA) 指定证书关联数据字段值与 X.509 树中的私有 CA 相关。值“3”(DANE-EE) 指定证书关联数据字段值与您正在连接的端点上的证书相关。在上面的示例中,Certificate Usage 字段指示 SHA-256 字符串所基于的证书属于 X.509 树中的公共证书颁发机构。这些是您的浏览器使用的相同证书颁发机构。 “0”、“1”或“2”的字段值仍然取决于这些 CA 根证书。当正确配置 DNSSEC 并使用值“3”作为证书使用字段的值时,TLSA 记录的真正优势得到释放。

回顾上面 TLSA 资源记录示例的开头,语法显然遵循 _port._proto col.subject 语法。 TLSA 资源记录始终以下划线“_”开头。然后定义服务端口（HTTPS 为 443)并指定传输协议。根据 RFC 6698,这可以是 tcp、udp 或 sctp。生成 TLSA 记录时,也可以将 dccp 指定为传输协议。尽管 RFC 6698 没有明确提及 dccp。主题字段通常等于服务器名并且应该与证书的 CN 值相匹配。

LPIC-2 考试并不强制要求记住所有这些细节。但是,最好了解不同的选项及其影响。正如将值“3”用作证书用途可以带来一些安全优势一样,将值“0”用作匹配类型字段可以在完整证书最终出现在 TLSA 资源记录中时导致碎片化的 DNS 回复。安全性的成功来自于可用性。

要生成您自己的 TLSA 记录,可以使用如下页面,并且应该提供对使用的参数和值的一些了解： www.huque.com/bin/gen_tlsa

尽管前面提到的网站适合偶尔创建 TLSA 记录,但在某些情况下自定义工具更合适。 hashslinger 命令行工具的开发是为了简化 TLSA 记录的创建。

版权所有 2013-2017 Snow BV

Machine Translated by Google

如果正确实施并得到广泛支持,DANE 有可能为各种服务提供基于证书的加密。

内部域名系统

如果您的组织在其内部网络上使用 TCP/IP（如今这种情况非常普遍）,他们还需要一种方法在本地将主机名解析为 IP 地址,反之亦然。在基础设施通常不会随时间发生太大变化的小型组织中,他们可能会使用静态文件（主机文件）,但考虑到维护此类基础设施的成本与安装和维护您自己的 DNS 的成本之间的收支平衡点较低,大多数组织运行内部 DNS 系统。

在更简单的设置中,只有一个名称服务器可能就足够了,尽管考虑到所涉及的名称服务器的成本较低,通常会成对安装。如果主服务器没有响应,则解析器知道多个名称服务器将查询另一台服务器。如果您有多个部门和各种互连网络,您的设置将开始类似于 Internet。但是有一些重要的区别。

假设您的部门位于公司主楼外。工作组将拥有自己的名称服务器。在内部,hosts 将是 exworks（exampleworks 的缩写)域的成员,即直接在根 (.) 域下。为防止 Internet 上的意外路由,公司选择使用不可路由（专用)IP 网络之一:192.168.72。

这意味着我们需要维护域区域和反向查找区域 72.168.192.in-addr.arpa。

我们不能使用 Internet 基础设施,因为他们的根名称服务器对 exworks 和 72.168.192 一无所知。 in-addr.arpa 区域,它们也不应该:毕竟我们选择使用专用网络。如果我们还希望能够解析真实的 Internet 主机名和地址,我们可以使用所谓的分层 DNS 设置。将显示两个分层 DNS 设置。

限制谈判

我们可以使用在拨号连接时代引入 BIND 的功能。拨号连接背后的 BIND 可能很麻烦:每次 DNS 想要与根名称服务器通信时,自动拨号程序都会建立一个连接。这很贵。您可以通过放置来阻止 BIND 执行此操作

```
// 防止拨号访问 heartbeat-interval 0;拨
号是； // 实际上禁用拨号！
```

在 named.conf 文件的选项语句中。必须将 dialup yes 置于实际上旨在防止拨号的配置中有点违反直觉,但这就是实现它的方式。

这个技巧也可以用来限制来自内部区域的通信,比如 exworks。出厂价和 72.168.192。 in-addr.arpa 区域作为本章前面所述的常规主区域实现。这样做的根本问题是真正的根服务器仍然不委托 exworks（和相应的反向)域。

因此,我们应该将内部区域与根名称服务器定义一起拉入一个单独的 DNS 实现中。
这需要至少两个独立的名称服务器,它们可以用两种不同的方式设置,如下所述。

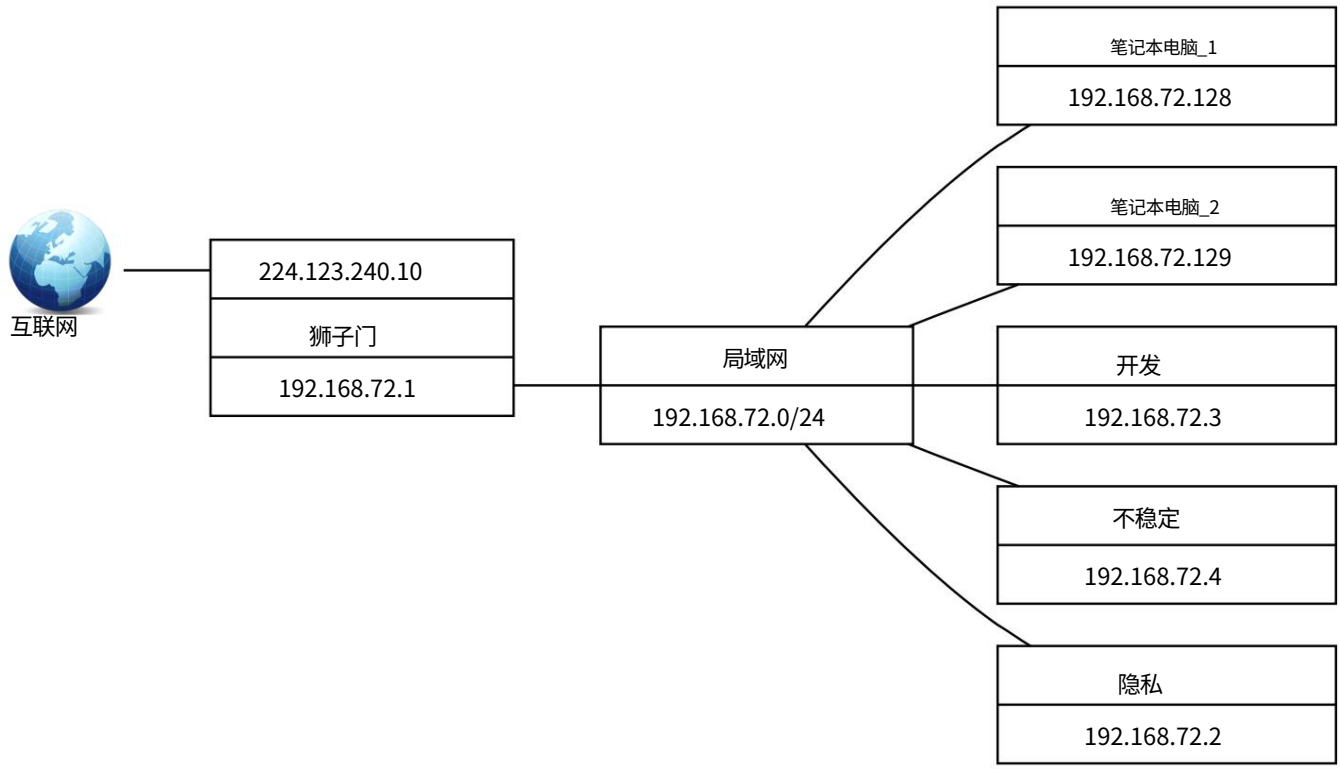
拆分 DNS:独立的内部主控

第一种方法包括一个独立的名称服务器（即内部区域的主服务器)和另一台主机上的另一个名称服务器,可用于解析来自外部世界和内部区域的名称。

下图显示了用作示例的出厂域。

版权所有 2013-2017 Snow BV

Machine Translated by Google



出厂网络。

此示例中的两个名称服务器将位于不同的主机上。第一个名称服务器在 privdns 上运行,并将提供内部区域信息。另一个名称服务器在 liongate 上,它将成为内部区域和外部世界的转发名称服务器。

主机 privdns（用于私有 DNS)将包含 exworks 和 72.168.192.in-addr.arpa 区域的 DNS。
该名称服务器将具有以下特征：

- 它将成为根域的主控者,但除了它自己之外没有任何 DNS 能够访问该信息
- 它将成为exworks 和72.168.192.in-addr.arpa 域的主人。只有内部的其他名称服务器建筑物将能够访问此信息

另一方面,liongate 将执行以下操作：

- 转发内部区域（这应该首先完成）
- 为外界做转发

在privdns上配置 master

首先,为内部根区域准备一个主文件。例如：

```
SOA
·     privdns.exworks 中的 $TTL 25h。 postmaster.privdns.exworks。 ( ; yyyymmhhee (ee == ser/day start 00)
                                2001121000
                                        28800
                                          3600
                                      604800
                                        86400)
                        在 NS privdns.exworks。 privdns.exworks.
在 192.168.72.2
```

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
;胶水唱片出厂。
                                                    在 NS privdns.exworks。 72.168.192.in-
addr.arpa。在 NS privdns.exworks。
```

请注意将 exworks 和 72.168.192.in-addr.arpa 区域委托为托管在名称服务器 192.168.72.2 上的粘合记录。

接下来,在 named.conf 中添加一个区域语句,指向根区域的主文件:

```
// 内部 DNS 服务器的 ROOT MASTER 区域
区域“。”在 {
        类型大师;文件“/etc/
        bind/internal.root.zone”;
        允许转移{无; };允许查询{无; };

};
```

使用类型 master 告诉 DNS 这确实是根区域的主服务器。根区域不应该为任何其他名称服务器所知,因此 allow-transfer 和 allow-query 设置为 none。

注意根

区域定义类型提示,如存在于默认的仅缓存服务器中,应该关闭(通过不将其包含在named.conf文件中)。

现在,为 exworks 和 72.168.192.in-addr.arpa 区域准备区域文件,并在 named.conf 文件中添加相应的区域条目。出厂条目应如下所示:

```
区域“exworks” IN { type master;
        文件“/etc/bind/
        exworks.zone”;允许转移{无; };允许查询

                                        { 192.168.72.1; }; //狮门
};
```

liongate 上的名称服务器是为其他主机提供名称的服务器,因此 liongate 的 IP 地址必须列在允许查询字段中。

必须对相应的反向区域执行相同的操作。

确保选项语句包含

```
递归没有;取胶没有;
```

这些语句告诉名称服务器它不应该接受对它所知道的区域以外的区域的查询。

exworks 区域中的主机应将其 resolv.conf 指向 liongate。也就是说,该文件应包含以下行:

```
域名服务器 192.168.72.1
```

其中 192.168.72.1 是 liongate 的 IP 地址。然而,在 liongate 本身,

```
名称服务器 127.0.0.1
```

确保本地查询在本地循环接口上运行,这比使用外部 IP 地址更有效。

笔记

privdns上的resolv.conf中的条目不应指向privdns本身上的名称服务器。如果主机用于需要名称解析的目的,最好将条目指向liongate 上的名称服务器。

版权所有 2013-2017 Snow BV

Machine Translated by Google

在liongate上配置 DNS

DNS 的功能有两个:首先它应该解析 exworks 区域（和相应的反向区域)的名称,其次它应该解析来自外部世界的名称。

使用以下转发器语句,对该名称服务器的查询将转发到列出的 IP 地址上的一个（请记住转发器是选项语句的一部分）：

```
转发器{
        192.168.72.2； // 私有域名 224.121.121.99; // ISP
        的 DNS
};
```

名称服务器按照列出的顺序进行尝试。请求首先转发到 privdns。如果那个人没有答案,则会联系 ISP 的 DNS。

请注意,应首先提及 privdns 的 IP 地址:我们不希望将内部名称请求发送到 ISP。

或者,可以将 liongate 上的名称服务器配置为 exworks 域和相应反向域的从属服务器。我们将在下一节讨论这个问题。

备择方案

此设置有两种选择。两者都需要两个独立的名称服务器。

第一种选择是在 resolv.conf 中使用两个名称服务器语句：

```
域名服务器 192.168.72.2
域名服务器 192.168.72.1
```

第一个 IP 地址指向 privdns,第二个指向 liongate。在这种情况下,privdns 也应该接受来自 192.168.72.0/24 的查询,liongate 上的 forwarders 语句不应包含 192.168.72.2（privdns 的 IP 地址）。这样做的缺点是没有单一的名称服务器入口点。

第二种选择是为出厂（加上相应的反向)域使用从属设置。当两个名称服务器在同一台主机上运行时需要这种情况。下文将对此进行详细说明,此处不再赘述。为此，liongate 上的转发器声明不应包含 privdns 的 IP 地址。

拆分 DNS:一台机器上有两个 DNS 服务器

我们再次假设同一个网络,但有一个重要的区别:主机 privdns 不可用。这意味着内部名称服务器现在必须在其他主机之一上运行,该主机也需要访问外部世界。

两个名称服务器将在同一主机上运行。至少其中一个名称服务器必须以 chroot 方式运行。

一台主机上的两个名称服务器

必须进行以下设置：

- 内部名称服务器 有一个主名称服务器为出厂和相应的反向域提供服务。它在自己的 uid/gid 下以 chroot 模式运行,在一个特殊的端口上监听。它不会做任何其他解析器工作。该名称服务器将被称为内部名称服务器。

- 可见的名称服务器 另一个名称服务器不需要运行chroot,但它有自己的uid/gid。它充当 exworks 和 72.168.192.in-addr.arpa 域的从站。它还将其他请求转发给 ISP。该名称服务器将被称为可见名称服务器,因为这是对其他主机可见的名称服务器。

可见名称服务器的 resolv.conf 文件中的名称服务器行指向 127.0.0.1； exwork 域中的其他主机将其解析器指向 192.168.72.1（liongate 的 IP 地址）。

版权所有 2013-2017 Snow BV

Machine Translated by Google

配置内部名称服务器

名称服务器被放入一个带有他自己的用户和组 ID 的 chroot 监狱中。名称服务器将 chroot 到 /var/cache/bindInternal。例如,它将以 UID 5301 命名的用户和具有相同名称和 GID 5301 的组运行。

这个 chrooted 名称服务器应该能够写入某些目录。为此,这些目录应由已命名的 UID 和 GID 拥有。需要它的目录（在 chroot 监狱内)是 /var/run（因为它需要写入 named.pid)和 /var/cache/bind（因为它需要写入例如日志文件）。

配置目录将为 /var/cache/bindInternal/etc/bind,因此该目录中 named.conf 文件的引用将指向 /etc/bind。该目录中的文件包括根区域的主文件以及 exworks 和 72.168.192.in-addr.arpa 域的区域文件。

以下是三个主区域定义：

```
选项{ 目录“/var/
        cache/bind”;监听端口 5353 { 127.0.0.1; };递归没有;
        取胶没有;



};

// 此处未显示,其他地方有描述： // - 特殊 chroot 日志记录,见上文 // - 区域
“localhost”、“127.in-addr.arpa”、“0.in-addr.arpa”和 // “255” .in-
addr.arpa”像往常一样


// 内部 DNS 服务器区域“.”的 ROOT MASTER IN { 输入 master;文件“/
etc/bind/internal.root.zone”;允许转移{无; };允许查询{无; };





}; // 没有根区域,类型提示定义！

// 工厂区
区域“出厂”在{
        类型大师;文件“/etc/
        bind/exworks.zone”;允许转移{ 127.0.0.1; };允许查询
        { 127.0.0.1; };

};

区域“72.168.192.in-addr.arpa”在{
        类型大师;文件“/etc/
        bind/exworks.rev”;允许转移{ 127.0.0.1; };允许查询
        { 127.0.0.1; };

};
```

该目录指的是 chrooted 树中的一个位置。 listen-on 指定此名称服务器应仅侦听内部地址 127.0.0.1 上的端口 5353。另外两个,递归和获取粘附,阻止解析除此处定义的区域之外的任何内容。

根区域不应允许查询和传输,因为它仅供本地使用。

exworks 和 72.168.192.in-addr.arpa 区域定义允许来自其他名称服务器（可见的服务器)的区域传输和查询。这些区域不允许进行其他传输和查询。

配置可见名称服务器

本节将描述如何配置可见名称服务器（如果有疑问,请再次阅读第7.3.10.3.1节）。请记住,它不运行 chrooted,也不使用备用端口。不过,它确实在自己的 UID/GID 下运行。

版权所有 2013-2017 Snow BV

Machine Translated by Google

可见名称服务器应该能够回答有关内部和外部世界的查询。

笔记

liongate上的可见名称服务器可以使用转发连接到内部名称服务器,前提是该软件允许您指定端口。

可见名称服务器的 named.conf 省略了公共部分,如下所示:

```
// /etc/bind/named.conf // 在 liongate
上绑定可见配置

选项{ 目录“/
        var/cache/bindVisible”;

        // 指向外部世界转发器的 ISP 名称服务器 {

                224.121.121.99; // ISP 的 DNS
        };
};

// 此处未显示,在别处描述: // - 照常记录 // - 照常显示根区域类型提示 //
- 区域“localhost”、“127.in-addr.arpa”、“0.in-addr.arpa”      和 //
   255.in-addr.arpa    像往常一样



// 指向这些区域的端口 5353,作为两个区域“exworks”的从属 IN { type slave;掌握端口 5353 { 127.0.0.1; };
文件“exworks.slave”;允许转移{无; };允许查询




                                        { 127.0.0.1; 192.168.72.0/24; };
};

zone    72.168.192.in-addr.arpa    IN { type slave;掌握端口
        5353 { 127.0.0.1; };文件“72.168.192.in-addr.arpa.slave”;
        允许转移{无; };


        允许查询                        { 127.0.0.1; 192.168.72.0/24; };
};
```

这实现了:

- 一个名称服务器为exworks 和72.168.192.in-addr.arpa 区域执行从属解析。
- 转发,解析来自外界的名称。

directory 语句指定的目录必须可由正在运行的名称服务器写入。也就是说,如果名称服务器以用户 vnamed 和组 vnamed 身份运行,目录 /var/cache/bindVisible 必须由用户 vnamed 和组 vnamed 所有。从属文件 exworks.slave 和 72.168.192.in-addr.arpa.slave 将放在那里。

让我们看一下从属区域定义。两者都指的是同一主机上端口 5353 的主机。这两个定义都不允许全区传输。唯一的区域传输是从主站（在端口 5353)到从站（主站的配置允许这样做,参见第7.3.10.2.1 节）。允许对 localhost（127.0.0.1,即 liongate 上的内部连接)以及 exworks 区域 (192.168.72.x) 中的所有主机进行正常查询。

在 liongate 上,resolv.conf 文件将包含

版权所有 2013-2017 Snow BV

Machine Translated by Google


```
名称服务器 127.0.0.1
```

也就是说,它指向监听端口 53 的可见名称服务器。exworks 域中的其他主机将具有

```
域名服务器 192.168.72.1
```

(liongate 的 IP 地址)在他们的 resolv.conf 中。

注意
无法在resolv.conf 中指定端口号。将在端口53 联系指定的名称服务器。

一个问题

每当区域定义更改时,主名称服务器将向从属名称服务器发送通知消息。这会导致从设备启动区域传输。但由于两台服务器都侦听相同的 IP 地址,唯一可以区分它们的是它们的端口号。
唉,你不能在 also-notify 语句中指定一个端口号,所以如果主数据改变了,就没有办法通知从机。

这个问题有一个简单的解决方案:如果主区域发生变化,并且在内部名称服务器重新加载区域数据后,也重新启动可见名称服务器。

Linux 支持将多个 IP 地址绑定到一个网络接口。另一种方法是为两台服务器提供唯一的本地 IP 地址。这涉及其他配置更改,现在您应该能够管理这些更改。因此,有关如何执行此操作的练习留给读者。

TSIG

TSIG（交易签名)为区域传输提供名称服务器之间的安全通信通道。它是一个资源友好的附加组件,在 BIND 8.2 及更高版本中可用。

TSIG 使用共享密钥和单向哈希来提供真实性和完整性。有关这些概念的简短介绍,请参阅CrashCryptoCourse 。

为 BIND 配置 TSIG

我们需要一个 HMAC-MD55 密钥对,它可以使用现在熟悉的 dnssec-keygen 命令生成:

```
# dnssec-keygen -a HMAC-MD5 -b 512 -n HOST rndc-key Krndc-key.+157+24876
```

该实用程序将生成两个输出文件。请注意,这两个文件中的密钥将始终重复,因为我们需要 HMAC-MD5。

```
# ls -l Krndc-key.+157+24876* -rw------- 1 root bind 117
Jul 5 05:28 Krndc-key.+157+24876.key -rw------- 1 root bind 229 Jul 5 05:28 Krndc-key.+157+24876.private
```

私钥文件的内容可能与此类似:

```
# cat Krndc-key.+157+24876.private 私钥格式:v1.3 算法:157
(HMAC_MD5)

密钥:XIQDYlGaIbWfyopYHS1vtFrlfJiiIkiEbiNj4kN8Ke+F0hEqA7KVwcJMR/6 ←。
        URez32XBEmKZf2W1GUzjpbI2KJQ==
位:AAA=
创建时间:20130705102726
发布:20130705102726
激活:20130705102726
```

版权所有 2013-2017 Snow BV

Machine Translated by Google

创建一个文件来保存您刚刚生成的秘密和从属名称服务器的 IP 地址。文件的名称可以自由选择,在此示例中将使用 tsig.key。这个文件的语法可以从这个例子中学习:

```
关键“传输”{算法hmac-md5;秘
        密
        “XIQDYlGaIbWfyopYHS1vtFrlfJiiIkiEbiNj4kN8Ke+F0hEqA7KVwcJMR/6 ←
            URez32XBEmKZf2W1GUzjpbI2KJQ==”;
};

# nameserver 2 (slave) server 10.0.1.2
{ keys { TRANSFER; };

};

# nameserver 3 (slave) server 10.0.1.3
{ keys { TRANSFER; };

};
```

注意:服务器选项指向从名称服务器。

编辑 BIND 主配置文件 named.conf 并包含如下文件:

```
包括“/etc/bind/tsig.key”;
```

现在,重新加载 BIND 配置:

```
# rndc 重新加载
服务器重新加载成功
```

使用 rndc tsig-list 列出当前活动的 TSIG 密钥。这样就完成了 master 上的配置。接下来我们需要配置从站。

要设置从名称服务器,请在那里创建一个类似的 tsig.key 文件,其中包含与主名称服务器上使用的相同的秘密。但是,现在让服务器选项指向主名称服务器:

```
关键“传输”{算法hmac-sha1;秘
        密
        “XIQDYlGaIbWfyopYHS1vtFrlfJiiIkiEbiNj4kN8Ke+F0hEqA7KVwcJMR/6 ←
            URez32XBEmKZf2W1GUzjpbI2KJQ==”;
};

# nameserver 1 (master) server 10.0.1.1
{ keys { TRANSFER; };

};
```

和以前一样,将 tsig.key 文件的路径包含到 BIND 主配置文件中并重新加载配置。 TSIG 现已启用。

问题和解答

域名服务器

1. 仅缓存名称服务器和普通名称服务器有什么区别?

   仅缓存名称服务器不为区域提供服务,少数内部区域除外。仅缓存名称服务器[164]

2. named.conf文件中的forward选项是不是不用配置就可以运行?

   否。要使其正常工作,还需要填写转发器选项。使用转发器列表[166]

版权所有 2013-2017 Snow BV

Machine Translated by Google

3. bind 中使用哪种语法将各种类型的数据路由到日志输出通道?

   类别类型命令用于日志记录语句。它后面是选择的输出通道,例如: { default_syslog; }.类别命令[167]

4. 当在区域文件中使用时@叫什么?

   @ 指的是在 named.conf 中定义的当前原点。 @ 在区域文件中的使用[167]

5. 您将以何种方式提示名称服务器重新加载其配置和区域文件?

   要使用的命令是 rndc reload。重新加载名称服务器配置[168]

6. 哪种类型的 DNS 服务器是使用类型提示定义的特殊区域?

   对于根服务器。这些列在根区域 (zone .) 中并静态定义为提示 (type hint;)。这些是查找地址的起点。提示文件[176]

7. 区域文件中的 SOA 记录包含 - 除其他外 - 序列号。这个序列号有要求的格式吗?

   是的,但只是在它是自然数的意义上。每次更改某些内容时,序列号都必须递增(至少增加一个),以便在区域中实施此更改。对于永不改变的区域,单个 1 就足够了。但是,格式 yyyymmddee 很常见。序列号[178]

8、CNAME资源记录有什么作用?

   它为具有 A(地址)记录的主机指定一个命名别名。 CNAME 记录[179]

9. 当主机命令使用时,命名反向区域的用途。

   将 IP 地址映射到相应的主机名。反向区域文件[180]

10. zone 是如何定义为 master 的?

    通过在区域定义中使用 type master 语句将区域定义为主区域。定义一个区域为主[182]

11. 除了明显的隐蔽措施安全之外,还有哪两种 DNS 安全策略可以应用?

    一种是通过以较低权限运行 BIND 或在 chroot 环境或监狱中运行它来限制入侵的影响。另一个是通过签署名称服务器发送的响应来保护名称服务器连接。以较少的权限或在 chroot 监狱中运行 BIND [192]

12. 至少命名两个类别,它们在日志记录语句中由category 命令区分。

    security、lame-servers、cname 都是有效的类别。一些日志记录类别[167]

13. 如果 dig 命令在用于测试主机名的反向条目时追加当前来源,可能会出现什么问题到名字?

    区域文件可能有错误,在 PTR 记录定义中没有尾随点附加到主机名。
    确保 PTR 中的尾随点[185]

版权所有 2013-2017 Snow BV